Does Every Reasoning Step Count? A Mechanistic Analysis of Hierarchical Reasoning Models

Zirui Ren
Department of Physics
Tsinghua University
Beijing, China
rzr23@mails.tsinghua.edu.cn

Ziming Liu
College of AI
Tsinghua University
Beijing, China
lzmsldmjxm@gmail.com

Abstract

Hierarchical reasoning model (HRM) achieves extraordinary performance on various reasoning tasks, significantly outperforming large language model-based reasoners. To understand the strengths and potential failure modes of HRM, we conduct a mechanistic study on its reasoning patterns and find three surprising facts: (a) **Failure of extremely simple puzzles**, e.g., HRM can fail on *a puzzle with only one unknown cell*. We attribute this failure to *the violation of the fixed point property*, a fundamental assumption of HRM. (b) **"Grokking" dynamics in reasoning steps**, i.e., the answer is not improved uniformly, but instead there is a *critical reasoning step* that suddenly makes the answer correct; (c) **Existence of multiple fixed points**. HRM "guesses" the first fixed point, which could be incorrect, and gets trapped there for a while or forever. All facts imply that **HRM appears to be "guessing" instead of "reasoning"**. Leveraging this "guessing" picture, we propose three strategies to scale HRM's guesses: data augmentation (scaling the quality of guesses), input perturbation (scaling the number of guesses by leveraging inference randomness), and model bootstrapping (scaling the number of guesses by leveraging training randomness). On the practical side, by combining all methods, we develop **Augmented HRM**, **boosting accuracy on *Sudoku-Extreme* from 55.0% to 96.9%**. On the scientific side, our analysis provides new insights into how reasoning models "reason".

1 Introduction

Current large language models (LLMs) are based on the transformer architecture [Vaswani et al., 2017]. Despite their success, they still struggle in reasoning-intensive tasks. For example, Wang et al. [2025a] showed that even the best language models achieve 0% success rate in solving hard Sudokus and mazes. In an effort to build models competent for these reasoning tasks, which generally require systematic System-2 thinking for humans, massive work has been done to prolong the reasoning process of LLMs, i.e., the intermediate outputs before reaching the final answer. It either relies on chain-of-thought (CoT) prompting [Wei et al., 2022, Chen et al., 2025a] or fine-tuning with reinforcement learning [Guo et al., 2025, Wen et al., 2025, Yue et al., 2025], both of which perform reasoning at the token level, limiting the potential of deep networks [Turpin et al., 2023, Helwe et al., 2021].

Alternatively, latent-space reasoning models [Hao et al., 2024, Wang et al., 2025a] rise as a new paradigm of reasoning depth scaling. Among them, the hierarchical reasoning model (HRM) [Wang et al., 2025a] achieves extraordinary accuracy on various reasoning-intensive tasks, outperforming LLM reasoners by a significant margin.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

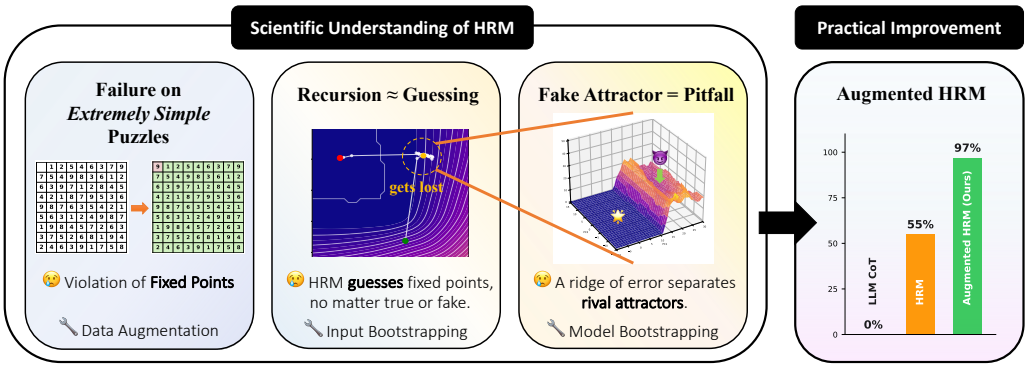

Figure 1: A lone unknown cell exposes the fixed-point violation. HRM secretly guesses fixed points, no matter they are true or not. Multiple fixed points exist in the latent space; escaping them via data augmentation, input and model bootstrapping boosts accuracy from 55.0% to 96.9%.

To understand the secret sauce of the success of HRM and reveal its potential failure modes, we closely inspect the reasoning patterns of HRM, mainly focusing on the *Sudoku-Extreme* dataset [Wang et al., 2025a]. To our surprise, we identify three counterintuitive facts:

- **Failure on *extremely simple* puzzles, due to violation of fixed points (Section 3)**: HRM could fail on *a puzzle with only one unknown cell*, due to a theory-practice mismatch. HRM theory assumes the fixed point property, i.e., the ability to maintain stability after finding the solution; however, we find that this property breaks down in practice. Luckily, we find that a simple fix suffices – data augmentation.
- **"Grokking" dynamics in recursion, due to HRM "guessing" instead of "reasoning" (Section 4)**: When approaching a puzzle, HRM does not incrementally refine the answer at each recursive step. Instead, it typically gets completely perplexed (error remains high and flat for many steps), and then "groks" (error drops to zero in one step). We hypothesize that the recursion (outermost loop) of HRM serves as a way of scaling "guessing" attempts for a plausible latent state, challenging the common belief that recursive reasoning boosts performance by incremental refinement.
- **Reasoning "gets lost" in the latent space, due to multi-stability of the reasoning landscape (Section 5)**: Closely inspecting the reasoning trajectory in the latent space, we are able to classify reasoning modes of HRM, among which the most interesting failure mode is when it "gets lost", i.e., lingering around some misleading attractive point. We show that these false attractors can be interpreted as local optima of a heuristic error metric measuring the number of conflicts. This trap discourages HRM from further exploring the latent space, postponing or precluding the encounter of the "true" fixed point. It turns out to be the central factor that caps HRM at suboptimal accuracy.

All insights above imply that HRM appears to be "guessing" instead of "reasoning". In order to get better performance, the "guessing" picture points to a new scaling axis – guess attempts, in addition to model and data. We thus propose three methods to scale guessing attempts: data augmentation (scaling the quality of guesses), input perturbation (scaling the number of guesses by leveraging inference randomness), and model bootstrapping (scaling the number of guesses by leveraging training randomness). Combining all methods, our **Augmented HRM** is able to boost the accuracy from 55.0% to 96.9% for *Sudoku-Extreme*, surpassing vanilla HRM [Wang et al., 2025a] and its existing variants such as the Tiny Recursive Model [Jolicoeur-Martineau, 2025]. Scientifically, our findings provide new insights into how reasoning models could "reason".

2 Background on HRM

HRM is a recursive latent-space model in the sense that in each forward pass, the inputs and hidden states are recurrently passed through the same module several times. Each call of this module is named a *segment*, and the recursive loop of segments is named the *outer loop*.

2.1 Forward Pass

We formalize the (simplified)[1] forward pass of HRM as follows. The input sequence is mapped to its embedding $\tilde{x}$ by the input network f_I:

$$\tilde{x} = f_I(x; \theta_I) \tag{1}$$

In this paper, we mainly focus on the *Sudoku-Extreme* dataset. The input samples x's are sudoku puzzles, formatted as 9×9 sequences containing integer tokens ranging from 1 to 9, together with a special `<blank>` token representing masked cells.

As a latent-space model, HRM maintains a latent state z^i, deterministically initialized as z^0. The HRM segment $\mathcal{F}$ takes the input embedding and the current latent state as input, and computes the next latent state:

$$z^{i+1} = \mathcal{F}(z^i, \tilde{x}; \theta) \tag{2}$$

After all M segments, the prediction vector is extracted from the terminal latent state:

$$\hat{y} = f_O(z^M; \theta_O) \tag{3}$$

In practice, when HRM has already reached a plausible solution, the remaining segments are essentially redundant. Accordingly, an adaptive computation time (ACT) mechanism [Graves, 2016] is introduced to decide whether to halt the computation after each segment.

2.2 Deep Supervision & One-step Gradient

2.2.1 Reasoning Depth Scaling

The core training technique to achieve reasoning depth scaling is *deep supervision* [Wang et al., 2025a]. Each forward pass corresponds to only one ground truth label, while the number of segments can be arbitrarily scaled. This mismatch makes the loss signal sparse compared to reasoning depth.

Deep supervision addresses this issue by computing the loss for the latent state z^i and the associated output $\hat{y}^i = f_O(z^i, \tilde{x}; \theta_O)$ of *each* segment. Formally,

$$L^i = l(\hat{y}^i, y) \tag{4}$$

where l is the loss function. However, the computation required for a full back propagation through time (BPTT) [Rumelhart et al., 1986, Werbos, 1990] of all these segment losses scales at $\Theta(T)$, and deep supervision at each segment would cost $\Theta(T^2)$ in total.

To overcome this, a dual technique called *one-step gradient* [Wang et al., 2025a] is used: during optimization, the gradient of L^i is only computed with respect to the i-th segment. In other words, z^i is detached from the computational graph each time it is updated via Equation (2). It guarantees that training costs increase at the same rate as reasoning depth.

2.2.2 The Role of Fixed Point Property

The one-step gradient alters the standard approach of training RNN-like models, and thus needs further theoretical grounding. Wang et al. [2025a] gave their justification, based on a few assumptions. One is that $\mathcal{F}$ is continuously differentiable. More importantly, they made the plausible presumption that if HRM finds z^* whose associated output is correct, it does not make further updates to the latent state. Thus, z^* should be the fixed point of $\mathcal{F}$, satisfying

$$z^* = \mathcal{F}(z^*, \tilde{x}; \theta) \tag{5}$$

Differentiating Equation (5) gives ($J_{\mathcal{F}} \equiv \frac{\partial \mathcal{F}}{\partial z}$):

$$\frac{\partial z^*}{\partial \theta} = (I - J_{\mathcal{F}}|_{z^*})^{-1} \left.\frac{\partial \mathcal{F}}{\partial \theta}\right|_{z^*} \approx \left.\frac{\partial \mathcal{F}}{\partial \theta}\right|_{z^*} \tag{6}$$

[1]The original HRM architecture uses separate H-module and L-module within one segment. However, ablation studies have shown that this structure is not the core factor of superior performance [Ge et al., 2025, Jolicoeur-Martineau, 2025]. It is also not relevant to our analysis of reasoning trajectories in following sections. Regarding these facts, in this paper we abstract away this inner structure of segments. This is merely a simplification of notation and does not alter the model architecture; see Appendix C for details of this simplification.

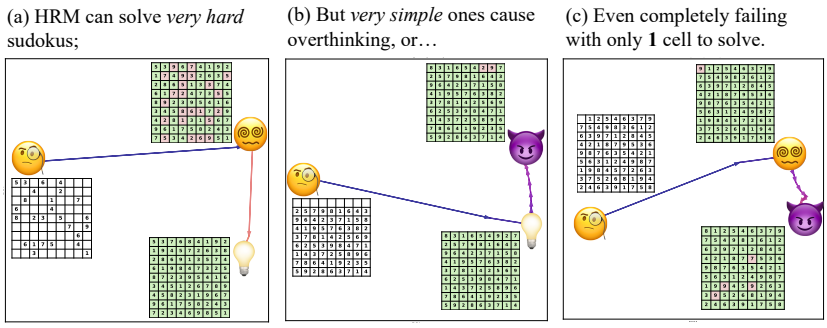

Figure 2: Reasoning trajectories in latent space (projected onto first two principal components) for different sudoku puzzles, and associated answers (red cell: wrong, green cell: correct). (a) For a difficult sudoku from the validation set of *Sudoku-Extreme*, the answer is correct; (b) For a simple puzzle with only one row masked, the answer is correct for the first two segments, but the model continues updating to wrong answers, violating the fixed point assumption; (c) Complete failure on an *extremely simple* puzzle with only one masked token.

which shows that as long as $I - J_{\mathcal{F}}|_{z^*} \approx I$ (a common approximation for implicit models), we can substitute the full BPTT gradient with the one-step gradient [Geng et al., 2021].

This succinct argument clearly relies strongly on the fixed point assumption. However, despite being intuitive and plausible, this assumption does not hold trivially. Our experiments demonstrate that this violation has profound consequences, to be discussed in Section 3.

2.3 Evaluation of HRM

The target benchmark is *Sudoku-Extreme* [Wang et al., 2025a], consisting of extremely difficult Sudoku puzzles. HRM is trained on the augmented versions of 1000 samples from the training data set, which are of the same difficulty level as the test set. HRM achieves a remarkable 55% accuracy on *Sudoku-Extreme*, demonstrating strong generalization abilities, while o3-mini-high, Claude 3.7 8K and Deepseek R1 all fail completely on this extremely complex task, achieving 0.0% success rate [Wang et al., 2025a].

3 Failure on Extremely Simple Puzzles: Violation of Fixed Points

The fixed point property, which states that the latent state is no longer updated after the true answer is found, is the fundamental assumption of HRM. It helps establish Equation (6), which justifies the one-step gradient approximation, in turn enabling reasoning depth scaling. Due to its simplicity and importance, it is a natural question whether HRM indeed demonstrates the fixed point property.

To our surprise, when dealing with *extremely simple* puzzles (e.g. with only one cell to fill), HRM does not retain the correct solutions after solving these puzzles in very early segments. In extreme cases, this instability manifests itself even earlier, resulting in a complete failure on such samples. This phenomenon is uncharacteristic of such a strong model, and causes tangible loss to performance.

In this section, we discuss this counterintuitive phenomenon of fixed point violation. We also present our explanation. A straightforward fix is introduced in Section 6.1.

3.1 Violation of Fixed Point Assumption

Indeed, for the test samples from *Sudoku-Extreme*, if HRM was able to reach the solution at any segment, the remaining segments do not alter the output anymore (Figure 2.a). Nevertheless, this single observation does not prove that HRM has learned the fixed point property, since these samples are as difficult as the ones HRM has been trained on.

Fixed points emerge at the final stage of HRM reasoning, where the model is very close to the solution. Thus we present a few extremely simple Sudoku puzzles to the same model to probe this property. These puzzles are constructed by masking out only one row, or even only one cell, from a complete sudoku. Shockingly, HRM frequently corrupts its answer by making unnecessary updates to its latent

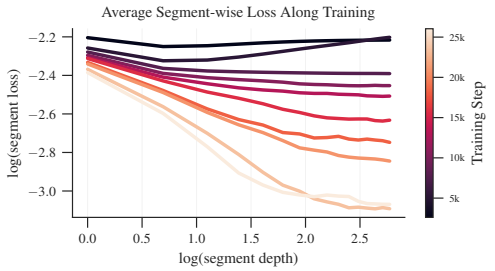

Figure 3: Segment-wise loss reduction improves over training. When averaging across the test samples, more segments lead to smaller losses in a smooth and incremental way, contrasting the "grokking" curves for per-sample analysis in Figure 4.

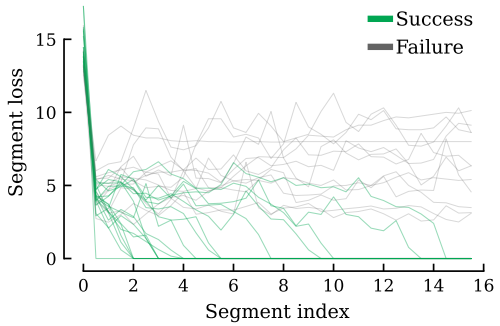

Figure 4: Per-sample analysis shows "grokking" dynamics along segments. For success samples, the loss value hovers above some threshold value before suddenly dropping to zero.

state, even after having reached the correct answer at very early segments (Figure 2.b). In fact, HRM is only able to maintain stability on such puzzles about 75% of the time.

In a slightly rarer case, instability shows up earlier, taking over even before HRM is able to find the solution. The consequence is a peculiar phenomenon: HRM sometimes gets a sudoku completely wrong throughout the segments, even if only one token is missing (Figure 2.c).[2]

3.2 One-step Gradient Postpones Acquisition of Stability

The ability to maintain fixed points, though proposed as a prior assumption, is by no means automatically obtained and has to be acquired from training. Due to one-step gradients, segments are disentangled and all contribute to the same goal (despite their sequential structure) – solving extremely hard Sudokus in one step. As a result, HRM is not explicitly trained to complete easier Sudokus, which is the key to ensuring stability around fixed points. Internally, HRM may implicitly create easier problems after some training, when a segment reasonably maps a hard problem into a partial solution (which itself can be viewed as an easier problem for later segments). However, this implicit curriculum is ineffective and not well-controlled. Consequently, the acquisition of stability is postponed to the terminal phase of training, which is far from sufficient.

4 Reasoning Modes of HRM: "Guessing" instead of Reasoning

In this section, we examine the way HRM approaches extremely hard puzzles, specifically by looking at the trajectory of its latent states.

4.1 Mean-field Analysis: Scaling Laws of Loss Curves

When presented with the formalization of HRM, or any recursive reasoning model, one would expect that scaling reasoning depth benefits the accuracy of the output monotonically. In other words, intuitively L^i should *gradually* decrease with i.

We start by pointing out that this is indeed true if we average the loss across all test samples (see Figure 3). Also, the loss reduction rate steadily increases as training progresses. This observation justifies the repeated utilization of HRM segment: it does improve the latent state iteratively on average.

[2]Importantly, HRM architecture does not enforce the preservation of unmasked input tokens - it merely maps one sequence to another. In extreme cases, unmasked tokens can be corrupted, rendering the segments completely ineffective.

4.2 Single-Sample: "Grokking" Along Segments

The surprise comes when we examine how the loss decreases with segments in a single sample. As shown in Figure 4, the error reduction process is by no means "gradual". After being quickly reduced to a relatively low value, the loss enters a lengthy plateau before suddenly dropping to zero. It seems that HRM gets perplexed in most of its segments, hovering around a state with relatively low loss, yet still far from correct. After that, it either remains so for the rest of the time and fails, or suddenly "groks" the correct answer in very few segments.

This is in sharp contrast to the intuitive picture of gradual loss reduction, where the segments are viewed as individual refining steps. A similar phenomenon is also reported in CoT-based models [Wang et al., 2025b] and the quanta model [Michaud et al., 2023].

4.3 Four Reasoning Modes of HRM

We visually analyze latent reasoning trajectories by projecting them onto the plane defined by their first two principal components. Remarkably, we can classify the reasoning patterns of HRM into the following modes:

1. **Trivial Success** The model finds the solution in the first few segments (Figure 5.a);
2. **Non-trivial Success** The latent state first lingers around some point for quite a few segments, then takes a sudden leap in an orthogonal direction and then immediately finds the solution (Figure 5.b);
3. **Trivial Failure** The latent state wanders about or oscillates in the latent space, encountering nothing special, and error remains high (Figure 5.c);
4. **Non-trivial Failure** The latent state converges to a fixed point. However, this fixed point does *not* correspond to the solution, and the error is in fact still high (Figure 5.d).

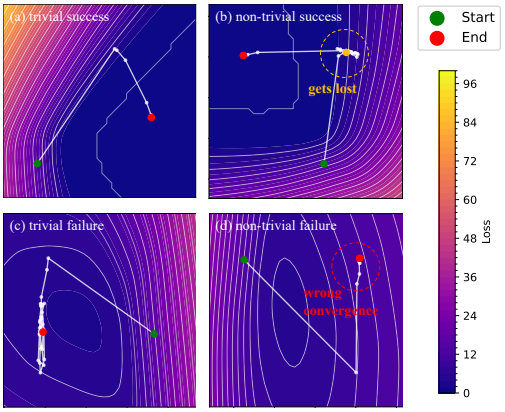

Figure 5: HRM has four reasoning modes on different samples. Trajectories of latent states projected onto the principal plane, with background showing loss values. The four samples demonstrate four modes observed: (a) trivial success; (b) non-trivial success; (c) trivial failure; (d) non-trivial failure.

Both trivial success and trivial failure modes are expected: if HRM gets a good latent state early (e.g. initialized close to the solution), it easily reaches a correct fixed point and stays there ever since. On the contrary, if it fails to make any improvement, its inner state wanders about or oscillates in the latent space.

4.4 Spurious Fixed Points as Misleading Attractors

Of real interest are the non-trivial cases: they seem to imply that there exists a kind of "false" fixed point associated with a wrong output. When entering the neighborhood of such points, it becomes challenging for HRM to leave the region. Such attractors sometimes cause HRM to linger around it for many segments before finally leaping out, leading to the non-trivial success mode; alternatively, if they trap the model forever, the model would converge to a false fixed point (non-trivial failure mode).

4.5 Reasoning or Guessing?

The discoveries in Section 4.2 have already showed that the segments of HRM do not serve as a cumulative way of refining the output. Further, for the samples on which HRM achieved success,

the contribution of each segment are not equal: most of the intermediate steps clustered around the spurious fixed point, making no substantial progress. This shows that HRM is perfectly capable of getting the solution in very few steps, if only already being around the correct latent state. However, it does not really strategize its search in the latent space.

We conclude that HRM does not "reason" in the commonsense way of approaching the solution gradually, despite using recursive architecture to mimic human reasoning behavior. If one insists on making an analogy to human intelligence, it resembles "guessing" more than "reasoning".

Another corroboration of this claim is the fact that the multiple pass techniques in Section 6.2 benefits performance. Given multiple chances to try approaching the puzzle with different perspectives, HRM is able to achieve much better accuracy; likewise, a human is more likely to be correct with multiple guessing attempts. However, if one is to approach a complex problem through deliberative reasoning, the number of attempts typically matters less.

5 Spurious Fixed Points

Section 4.4 confirms the existence of spurious fixed points which mislead or trap HRM from exploring more of the latent space. In this section, we attempt to characterize the nature of spurious fixed points more concretely.

5.1 Rival Attractor

In Figure 6, both the true and spurious fixed points for a sample from the **nontrivial success** class are shown. The model first finds the false one, lingers for several segments before vaulting towards the true one. Both fixed points are indeed attractive points, in the sense that when the latent state is initialized close to either of them, it quickly gets updated to it.

The attractive effect of these two points competes with each other, creating a clear line of separation in the middle. The latent state is consistently attracted to the fixed point closer to it. Whenever z^0 is initialized closer to the true attractor, the model converges to it with the correct output in 1 or 2 steps. However, when initialized closer to the false one, it gets trapped there for large and varying numbers of segments, or even forever. Such phenomena are ubiquitous for the non-trivial success mode discussed in Section 4.3.

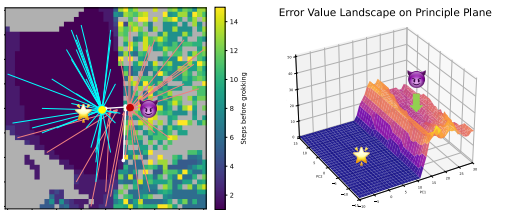

Figure 6: An example exhibiting rival attractors. **Left**: Arrows show the first update to the latent state at the starting point. Darker color in the background means that the model reaches the correct solution faster starting from there, while gray means a failure. There appear to be two attractors (yellow: correct; red: misleading), the interaction between which create a clear boundary of two regions. **Right**: The landscape of $\mathcal{E}$ value on the PCA plane. The ridge shows that this metric is non-monotonic (has an "energy barrier") between the rival attractors.

5.2 Plausible Local Minima

In nonconvex optimization, gradient methods are susceptible to being trapped by local minima. This bears a strong resemblance to spurious fixed points being malignant attractors.

We hypothesize that the segment dynamics of HRM are implicitly doing minimization on some *label-agnostic* energy function, measuring "how good the current output is" by counting conflicts. Here is one possible definition: for an output sudoku $\hat{y}$, we count the number $\mathrm{count}(d, u)$ of each token d in the rows $r(\hat{y})$, columns $c(\hat{y})$ and boxes $b(\hat{y})$. If any of these counts exceeds 1, we penalize this violation of sudoku rule by $\mathrm{count}(d, u) - 1$. Formally, we define an error metric as:

$$\mathcal{E}(\hat{y}) = \sum_{u \in \{r(\hat{y}), c(\hat{y}), b(\hat{y})\}} \sum_{d=1}^{9} \mathrm{ReLU}\big(\mathrm{count}(d, u) - 1\big) \tag{7}$$

This metric helps partially explain the reason HRM lacks the incentive to escape spurious fixed points automatically. By evaluating $\mathcal{E}(f_O(z; \theta_O))$ with z sampled around the rival attractors on the PCA

plane, we find that the misleading attractor does appear to be a shallow local minimum of this metric (Figure 6). In particular, along the segment from the spurious fixed point to its true counterpart, $\mathcal{E}$ first slightly increases before dropping to 0.

We point out that it would be premature to say that HRM actually implements an optimization algorithm on this quantity. Nevertheless, this opens up a new perspective for understanding HRMs, worthy of investigation in future work.

6 Augmented HRM: Mechanism-Inspired Remedies

The preceding analysis motivates three complementary interventions. Data mixing targets stability on near-solved states (Sections 3.1 and 3.2), while input relabeling and checkpoint bootstrapping diversify trajectories that may otherwise approach spurious attractors (Sections 4.4 and 5.2). These interventions are mechanism-inspired: they improve the opportunities to preserve a correct answer or reach a different attractor, without directly changing the architecture to control attractors.

6.1 Restoring Fixed Points via Data Augmentation

In order to restore the fixed point property of HRM for arbitrary inputs, we seek a direct way of enforcing the acquisition of stability. According to the discussion in Section 3.2, we are faced with the question: how do we provide HRM with more opportunities to train its stability?

We can explicitly advance such chances by exposing the model to nearly-solved puzzles directly. This inspires a simple data augmentation: for each puzzle in the training set, we make one simplified replicate. We reveal a random portion of the originally hidden tokens in the replicate, obtaining a simpler version of the puzzle, which is still valid as a training sample.

By training an HRM model with this augmentation technique, the overall test accuracy is increased from 55.0% to 59.9% (see Table 1). More importantly, the failure on simple puzzles (discussed in Section 3.1) is eradicated. Similarly, the unfavored drift after finding the solution is also eliminated (see Figure 7). We conclude that by mixing samples of different complexity levels into training data, the fixed point property of HRM is restored.

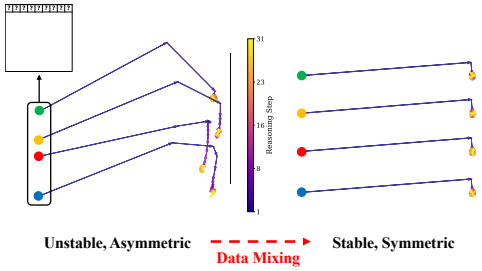

Figure 7: Data mixing restores **stability** and **symmetry** of latent reasoning trajectories (projected onto the first two principal components). It eliminates unfavored drifts after getting the correct answer. Furthermore, when dealing with distinct samples simplified via the same format, all latent trajectories now show perfect symmetry.

Other nice properties emerged as a by-product of fixed point restoration. For example, for puzzles generated via the same simplification formats (e.g., masking the first row), the latent-state trajectories now become symmetric: although these puzzles are different in tokens, they demonstrate nearly identical reasoning trajectories, suggesting the emergence of more general rules instead of overfitting to subtle details. This has not been the case for vanilla HRM (see Figure 7).

6.2 Escaping the Trap

The aforementioned behavior (Section 4.4) inspires us to incentivize HRM to escape or avoid such "traps" proactively, by adding perturbation to its reasoning trace. The idea is that perturbation increases the chance of encountering a true fixed point, or escaping a false one. The implementation of this idea, however, is slightly subtler. Due to the high dimensionality of the latent space, perturbing z in an arbitrary direction typically destructs the coherence of hidden states, leading to unstable behavior.[3]

[3]We do observe that in some cases, such techniques help HRM solve puzzles that it could not solve before, though. However, no consistent path to success was found.

To circumvent this difficulty, we point out that there naturally exist *indirect* ways to exert perturbation in alternative spaces. The input space can be manipulated to create semantically equivalent input sequences with distinct formats. Besides, the adjacent training checkpoints are naturally perturbed versions of the model. Thus, perturbation in the model parameter space is possible, although the space itself is not directly manipulable.

We propose two *indirect* inference-time scaling techniques to unleash HRM from the trap: **input perturbation** and **model bootstrapping**. Both these techniques requires multiple forward passes; each pass reports failure when the ACT mechanism fails to halt within the reasoning depth limit. A majority vote is done among those that successfully halted. Combining these with data augmentation introduced in Section 6.1, we propose **Augmented HRM**, reaching the state-of-the-art performance on *Sudoku-Extreme*.[4]

6.2.1 Perturbation in the Input Space

It is well known that a sudoku puzzle can be transformed into alternative versions by applying certain types of equivalent transformations. For example, if we have a valid[5] puzzle, we can swap the first and the second row, resulting in another valid puzzle. If we knew the solution of this transformed version, an inverse transformation would give the solution of the primal puzzle. Such transformations include relabeling the tokens, swapping bands, swapping rows or columns within a band, and reflecting/rotating the entire puzzle.

Importantly, tokens are treated by HRM as independent features, each assigned with a unique embedding vector by the mapping f_I. Consequently, vanilla HRM is intrinsically oblivious to the symmetry under transformations. In other words, HRM views the transformed puzzle as a fresh input. This fact allows us to perturb the input by applying any of the aforementioned transformations to it.

One would expect HRM to perform equally well on these equivalent puzzles, since transformations had been used as data augmentation during training. However, much to our surprise, transforming the input *does* improve the performance. Specifically, we choose the relabeling transformation[6]; by creating as few as 9 transformed puzzles via relabeling, we achieve a remarkable 18.2% improvement in exact accuracy (see Table 1).

6.2.2 Perturbation in the Model Parameter Space

Table 1: Evaluation results of techniques in the paper: data mixing for fixed point property restoration, input transformation by token relabeling, and model bootstrapping.

METHOD	ACCURACY
BASELINE	55.0%
+BOOTSTRAP	64.2%
+RELABEL	73.2%
+BOOTSTRAP+RELABEL	85.4%
+DATA MIXING	59.9%
+DATA MIXING+BOOTSTRAP	79.3%
+DATA MIXING+RELABEL	78.4%
+ALL (**AUGMENTED HRM**)	**96.9%**

Another natural space to create perturbed variants is the model parameter space of θ. Altering θ by force does not work, because such arbitrary perturbation severely imperils model capability. However, the training process offers an indirect way of doing this: the adjacent model checkpoints are approximately equally strong, while the optimizer steps between them diversifies their parameters. Inspired by this, we pick 10 of the checkpoints from the later half of training phase separated by approximately 1000 steps as an ensemble.

As we are testing the ensemble of checkpoints in the same training run, which should be strongly correlated, one might expect that the final strongest checkpoint (receiving the most training) should cover the capability of all its predecessors. However, we do observe a 9.2% improvement in accuracy with this model bootstrapping method. This strongly contradicts intuition, demonstrating that these mutually dependent models actually differ significantly, in terms of which of the test samples they get right.

[4]It is worth pointing out that these methods are essentially different from *pass@k* [Chen, 2021] used in LLM evaluation. The next-token prediction of an LLM is *randomized*, which is why multiple passes generate different outputs; these passes, however, are not guaranteed to be intrinsically distinct. Our method leverages perturbation in various working spaces of HRM, creating *intrinsic* diversity in its *deterministic* forward passes.

[5]By "valid" we mean that the sudoku puzzle can be uniquely solved, given the revealed tokens.

[6]This resonates with the color permuting augmentation used by [Franzen et al., 2025] when approaching ARC [Chollet, 2019]

7 Conclusion and Discussion

The goal of this paper is to develop a mechanistic account of how hierarchical reasoning models "reason". Starting with the examination of fixed point property, we detect a violation of this fundamental assumption, which limits performance and causes several peculiar behaviors.

Fixed points are important not only because they ground the architecture of HRM mathematically, but also in that they are directly related to the correctness of HRM output. Via a series of qualitative experiments to understand HRM's seek for fixed points, we conclude that multiple fixed points exist in the latent space, but some correspond to false solutions. HRM "guesses" the solution by sticking to the first fixed point it encounters, which usually is the one closest to its initialization. We also develop a heuristic discrimination between true and false fixed points.

Utilizing the acquired understanding, we fix existing flaws of HRM, develop several simple but effective techniques to exploit its potential, and achieve significant performance enhancement with Augmented HRM.

Our analysis is primarily based on the recursive mechanics of HRM. Thus, while our experiments focus on HRM, the lens naturally extends to the emerging class of recursive reasoners.

Acknowledgments and Disclosure of Funding

This work is supported in part by computational resources provided by the Shanghai Qi Zhi Institute. The authors declare no competing interests.

References

Sohyun An, Ruochen Wang, Tianyi Zhou, and Cho-Jui Hsieh. Don't think longer, think wisely: Optimizing thinking dynamics for large reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=nxnBaaRLnz`.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025a.

Xi Chen, Aske Plaat, and Niki van Stein. How does chain of thought think? mechanistic interpretability of chain-of-thought reasoning with sparse autoencoding. *arXiv preprint arXiv:2507.22928*, 2025b.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*, 2025.

Daniel Franzen, Jan Disselhoff, and David Hartmann. Product of experts with llms: Boosting performance on arc is a matter of perspective. *arXiv preprint arXiv:2505.07859*, 2025.

Renee Ge, Qianli Liao, and Tomaso Poggio. Hierarchical reasoning models: Perspectives and misconceptions. *arXiv preprint arXiv:2510.00355*, 2025.

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

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

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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.

Chadi Helwe, Chloé Clavel, and Fabian M Suchanek. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd conference on automated knowledge base construction*, 2021.

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

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.

Kyubyong Park. Can convolutional neural networks crack sudoku puzzles? `https://github.com/Kyubyong/sudoku`, 2018.

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

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986. URL `https://api.semanticscholar.org/CorpusID:62245742`.

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

Justin Theodorus, V Swaytha, Shivani Gautam, Adam Ward, Mahir Shah, Cole Blondin, and Kevin Zhu. Finding sparse autoencoder representations of errors in cot prompting. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL `https://openreview.net/forum?id=oCprwPRqwW`.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Martina G Vilas, Safoora Yousefi, Besmira Nushi, Eric Horvitz, and Vidhisha Balachandran. Tracing the traces: Latent temporal signals for efficient and accurate reasoning. *arXiv preprint arXiv:2510.10494*, 2025.

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*, 2025a.

Xi Wang, James McInerney, Lequn Wang, and Nathan Kallus. Entropy after ⟨`/Think`⟩ for reasoning model early exiting. *arXiv preprint arXiv:2509.26522*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

Paul Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1550 – 1560, 11 1990. doi: 10.1109/5.58337.

Kevin Xu and Issei Sato. A formal comparison between chain-of-thought and latent thought. *arXiv preprint arXiv:2509.25239*, 2025.

Xinhao Yao, Ruifeng Ren, Yun Liao, and Yong Liu. Unveiling the mechanisms of explicit cot training: How cot enhances reasoning generalization. *arXiv preprint arXiv:2502.04667*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=4OsgYD7em5`.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *arXiv preprint arXiv:2512.21711*, 2025b.

Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru R Zhang. The geometry of reasoning: Flowing logics in representation space. *arXiv preprint arXiv:2510.09782*, 2025.

A Related Work

Latent-space and Recursive Reasoning Models Rather than generating explicit chain-of-thought tokens, latent-reasoning models embed the intermediate computation in a continuous hidden trajectory. Coconut [Hao et al., 2024] chains hidden states to enable latent space planning; Heima [Shen et al., 2025] compresses each step into one "thinking token", cutting output length while maintaining precision. These models, featuring recursive latent-state updates, are emerging as a new paradigm for modeling reasoning [Jolicoeur-Martineau, 2025, Zhang et al., 2025a, Guan et al., 2025]. Recursion decouples reasoning depth from parameter count, yielding sizable savings in memory and compute [Dehghani et al., 2018]. HRM [Wang et al., 2025a] – one of the latest entries in this line – introduces hierarchical modules trained with deep supervision to achieve reasoning depth scaling, and is the main subject of our work.

Hierarchical Reasoning Model and Variants HRM is proposed by Wang et al. [2025a], achieving 55% accuracy on *Sudoku-Extreme*, surpassing latest LLM-based reasoners. Ge et al. [2025] showed via ablation study that the hierarchical architecture does not significantly contribute to overall performance. They verified the fixed point property on *Sudoku-Extreme*, but did not notice the violation on other samples. CGAR [Qasim and Zhang, 2025] refines the HRM architecture and speeds up training. Another variant of HRM named Tiny Recursive Model [Jolicoeur-Martineau, 2025] emphasizes the role of recursive outer loop, achieved 87.4% accuracy, at the cost of non-scalable reasoning depth. Our work achieves even better performance without altering the HRM architecture.

Nature of Reasoning Despite the performance boost achieved by both CoT and latent-space reasoning, there remain debates on what type of computation truly counts as 'reasoning' [Xu and Sato, 2025]. Various methods have been developed to gain a mechanistic understanding of CoT [Bogdan et al., 2025, Yao et al., 2025, An et al., 2025] and of latent reasoning [Zhang et al., 2025b], mostly through the features of reasoning traces. Such features include monosemantic features extracted by sparse autoencoding [Chen et al., 2025b, Theodorus et al., 2025] and geometric properties of the trace itself [Zhou et al., 2025, Vilas et al., 2025, Du et al., 2025].

Table 2: Evaluation results of augmentation methods proposed in Section 6.2, on the *Maze-Hard* dataset. Model bootstrapping and relabeling both improve accuracy as for *Sudoku-Extreme*.

METHOD	EXACT ACCURACY
BASELINE	74.5%
+BOOTSTRAP	92.1%
+RELABEL	76.7%
+ALL (**AUGMENTED HRM**)	**93.3%**

B Results on Other Datasets

The set of tools that we used for analyzing reasoning patterns is mostly independent of the specific task. Thus, most results are easily transferable to other datasets. In the original paper of Wang et al. [2025a], besides *Sudoku-Extreme*, *Maze-Hard* data set is used to probe reasoning capabilities. This task involves finding the optimal path in a 30×30 maze, with exactly one starting and one terminal point specified.

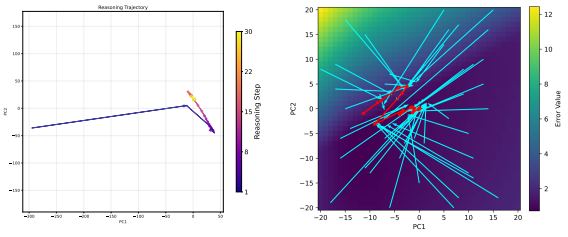

Figure 8: Reasoning trajectories for *Maze-Hard* dataset. **Left**: The final-stage instability during inference, due to the violation of fixed-point assumption, is still present in most failure cases. **Right**: The multiple spurious attractors, despite being rarer, still exist; their rival effect on the reasoning trajectory is also similar.

The augmentation techniques devised in Section 6.2 have natural counterparts for *Maze-Hard*. Model bootstrapping is nearly identical (10 checkpoints from the second half of training, with interval of 1000 steps). We implement the token relabeling technique by swapping the starting and terminal points of a maze puzzle, creating an equivalent variant. The evaluation results are shown in Table 2.

The HRM reasoning trace of the maze task proves to be simpler than that of *Sudoku-Extreme*. Successful cases typically have very short reasoning trajectories – the solution is nearly always found in very early segments. In other words, the non-trivial success case in Section 4.3 is hardly present. Meanwhile, fixed point violation (Section 3.1) is the dominant factor of failures. This might be because maze puzzles are intrinsically "easier" than sudoku puzzles, the latter involving more combinatorial structures that neural networks struggle to model [Park, 2018].

Although spurious attractors do not constitute (Section 4.4) the main cause of failure in maze puzzles, we still identify their existence in some rarer cases. Their misleading effect on reasoning trajectories is evident as well (see Figure 8).

C Implementation Details of HRM

To facilitate exposition, the main paper condenses one full reasoning cycle into the single recurrence

$$z^{i+1} = \mathcal{F}(z^i, \tilde{x}; \theta).$$

This appendix exposes the sub-segment structure that was abstracted away, ensuring full reproducibility. Notation is inherited from the main text; additional sub-scripts L (low-level) and H (high-level) distinguish the two internal modules.

C.1 Sub-segment Unfolding

Each segment is implemented as $N \times T$ elementary time-steps (default $N = 2$, $T = 2$). Two latent trajectories are maintained:

$$z_L^t \in \mathbb{R}^d, \quad z_H^k \in \mathbb{R}^d, \qquad t = 1 \ldots NT,\ k = 0 \ldots N.$$

The low-level module updates at every step; the high-level module updates once every T steps, yielding a *hierarchical convergence* schedule:

cycle $k = 0$

$$\begin{aligned} t = 1 \ldots T \quad & z_L^t = f_L(z_L^{t-1}, z_H^0, \tilde{x}; \theta_L) \\ t = T \quad & z_H^1 = f_H(z_H^0, z_L^T; \theta_H) \end{aligned}$$

cycle $k = 1$

$$\begin{aligned} t = T{+}1 \ldots 2T \quad & z_L^t = f_L(z_L^{t-1}, z_H^1, \tilde{x}; \theta_L) \\ t = 2T \quad & z_H^2 = f_H(z_H^1, z_L^{2T}; \theta_H) \\ & \vdots \end{aligned}$$

final output

$$t = NT \quad z^{i+1} \triangleq z_H^N.$$

Both f_L and f_H are 4-layer transformer blocks ($d{=}512$, 8 heads, Post-Norm with RMSNorm, with RoPE position encoding).

z_L^0 is always initialized as a deterministic tensor (trainable) inside a segment; z_H^0 is inherited from the previous segment's z_H^N and detached to block gradient flow. This hard reset is claimed to force the low-level module to re-converge every segment, preventing premature saturation [Wang et al., 2025a].

C.2 ACT Implementation

After obtaining z_H^N, a linear Q-head produces scalars

$$Q_{\text{halt}},\ Q_{\text{continue}} = \text{Linear}(z_H^N)\ \in \mathbb{R}^2.$$

A greedy rule decides whether to halt; if continuation is chosen, $z^{i+1} = z_H^N$ is fed into the next segment without extra parameters.

C.3 Mapping to the Main-text Notation

The abstract operator $\mathcal{F}(\cdot)$ in Equation (2) of the main paper realizes the entire $N \times T$ hierarchy described above; z^i corresponds to z_H^0 entering the segment, and z^{i+1} corresponds to z_H^N exiting it. The operator $\mathcal{F}(\cdot)$ is applied M (default $M = 16$) times as the outer loop (described in Section 2.1). All low-level dynamics are encapsulated inside $\mathcal{F}$, so the simplified description remains functionally identical to the original HRM.

D Additional Quantitative Results

D.1 Data Mixing under a Fixed Training-Data Budget

To control the number of training examples, we replace half of the training samples with simplified versions of themselves. This differs from the added-replicate setting in Section 6.1. Hard accuracy is exact accuracy on the full Sudoku-Extreme test set. Stability is measured among near-solved probes that are ever correct, as the fraction that remain correct after one-cell corruption. Table 3 reports both metrics throughout training.

Table 3: Fixed-budget data mixing. All stability and accuracy values are fractions.

	Vanilla HRM		Data-mixed HRM	
Step	Stability	Hard accuracy	Stability	Hard accuracy
1302	0.00	0.000	0.82	0.000
3906	0.00	0.022	1.00	0.030
6510	0.00	0.083	1.00	0.040
9114	0.00	0.122	1.00	0.030
11718	0.93	0.159	1.00	0.170
14322	0.99	0.290	1.00	0.200
16926	1.00	0.370	1.00	0.290
19530	0.83	0.446	1.00	0.355
22134	0.00	0.450	0.81	0.395
24738	0.00	0.533	1.00	0.460
27342	0.55	0.515	0.97	0.525
29946	0.00	0.180	1.00	0.510
32550	0.00	0.214	1.00	0.575

D.2 Measurements around the First Exact-Match Segment

We evaluate the released paper checkpoint on a Sudoku-Extreme test subsample of 400 puzzles and focus on 88 nontrivial successes: trajectories that remain incorrect for several segments before reaching an exact solution. The jump is the first exact-match segment. Table 4 reports decoded-output statistics and latent update norms, where $\|\Delta z\|$ is the Euclidean distance between consecutive latent states. The conflict energy is defined in Section 5.2.

Table 4: Pre-jump and transition statistics for 88 nontrivial successes.

Metric	Value
Pre-jump exact match	0%
Mean number of wrong cells before jump	9.39 cells
Mean conflict energy before jump	16.58
Mean segments before jump	7.4 / 16
Mean $\|\Delta z\|$ on plateau vs. at jump	68.1 vs. 121.1

Pre-jump exact-match accuracy is zero by the definition of the first exact-match segment; it is not an independent test of the information encoded in the latent state.

D.3 Trajectory-Mode Frequencies across Training

We apply the four-mode taxonomy in Section 4.3 to checkpoints from one vanilla HRM training run. Each early checkpoint uses a test subsample of 200 puzzles, and the final checkpoint uses 400. Table 5 reports exact accuracy and all four mode frequencies. The final 55.8% accuracy is a subsample statistic, distinct from the full-test-set baseline in Table 1.

Table 5: Trajectory-mode frequencies across training. All entries except training step are percentages. Totals can differ from 100% due to rounding; "stuck" denotes nontrivial failure.

		Success		Failure	
Step	Exact accuracy	Trivial	Nontrivial	Trivial	Stuck
3906	3.0	3.0	0.0	1.5	95.5
6510	10.0	3.0	7.0	36.0	54.0
9114	15.5	4.0	11.5	10.5	74.0
13020	29.5	8.0	21.5	45.5	25.0
16926	41.0	15.5	26.0	51.0	7.5
Final	55.8	33.8	22.0	41.5	2.8

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarizes the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations throughout the paper as they were encountered, e.g. in Section 5.2 we revealed the partially speculative nature of our designed error metric.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical derivations or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix C describes the implementation details of HRM, and other experimental details are distributed across the paper. We also provide our code for maximal reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code as supplemental material, which fully reproduces the results in the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All settings are described in the paper and specified in the code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Repeated training experiments would be too computationally expensive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details are covered in the code base.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is in accordance with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: HRM is a preliminary model on toy tasks. Thus the authors believe that there is no substantial societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no safety risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The assets (mainly the datasets) used are mentioned and properly cited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The experimental code base is submitted alongside the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components. LLMs are used only for editing purposes.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.