# INVARIANT-MEASURE REASONERS: STABLE REPRESENTATIONS FOR LATENT REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Latent reasoning models repeatedly update a latent state using the same recurrent block. As the recurrent depth increases, the sequence of latent states may converge to a compact subset of the state space without necessarily converging to a fixed point. Existing models typically predict by applying a prediction head to a single latent state. However, the latent state can continue to change even after many updates, potentially making predictions unstable across recurrent depths. To address this instability, we introduce invariant-measure reasoners (ImR), a framework that uses an invariant measure as a stable representation. This measure describes the long-run distribution of latent states on the compact subset and is invariant under updates by the recurrent block. ImR predicts from the expectation of the prediction head's output under this measure. We use ImR in two ways: fine-tuning only the prediction head of existing models and training models from scratch. Both approaches reduce prediction instability and improve accuracy in many settings on maze and Sudoku tasks. In some settings, models trained with ImR exhibit non-fixed-point behavior more frequently than existing models yet achieve high accuracy even with such behavior, unlike existing models. These results suggest that ImR can leverage otherwise destabilizing dynamics for latent reasoning.

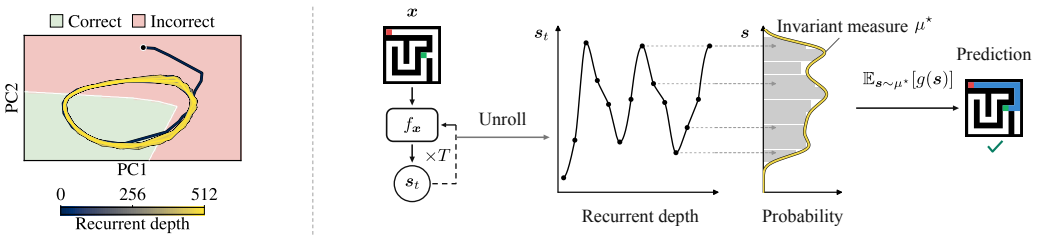
## 1 INTRODUCTION

Reasoning has attracted increasing attention in artificial intelligence as a capability to derive solutions through a sequence of logical steps rather than just recalling memorized answers (Guo et al., 2025). Latent reasoning offers an alternative to chain of thought (Wei et al., 2022) by performing intermediate computations in latent space without expressing each step in natural language (Hao et al., 2025). In particular, latent reasoning models that repeatedly apply the same recurrent block to a latent state have shown promise in complex structured reasoning tasks such as solving mazes and Sudoku puzzles (Wang et al., 2025; Jolicoeur-Martineau, 2025). Increasing recurrent depth, defined as the number of recurrent updates, allows these models to devote more computation to challenging problems at test time (Graves, 2017; Schwarzschild et al., 2021; Geiping et al., 2025). However, predictive performance can deteriorate beyond the recurrent depth used during training, a phenomenon known as overthinking (Bansal et al., 2022). Recent advances, including input injection (Bai et al., 2019; Bansal et al., 2022) and deep supervision (Wang et al., 2025), have mitigated this degradation and enabled effective test-time scaling (Huang et al., 2026; Knutson et al., 2026).

Despite these advances, latent reasoning models can still exhibit prediction instability across recurrent depths. For a fixed input, recurrent updates define a discrete-time dynamical system. In recent latent reasoning models, the resulting trajectories have been observed to approach a compact subset of the state space without converging to a single fixed point (Anil et al., 2022; Geiping et al., 2025; Huang et al., 2026; Knutson et al., 2026). These models typically predict by applying a prediction head to the final latent state, so continued changes in that state can lead to changes in the predicted solution. Figure 1(a) illustrates this issue with a latent trajectory from equilibrium reasoners (EqR) (Huang et al., 2026). The trajectory remains within a bounded region and repeatedly traces a loop reminiscent of a limit cycle. Along this trajectory, predictions alternate between correct and incorrect. One way to obtain stable predictions is to predict from a fixed point, a latent state invariant under recurrent updates (Bai et al., 2019; Movahedi et al., 2026). Yet the continued changes in latent states observed in recent models raise the question of whether predictions can be stabilized without requiring convergence to a fixed point.



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




(a) Prediction instability

(b) Invariant-measure reasoners (ours)

Figure 1: (a) An example of prediction instability in equilibrium reasoners (EqR) (Huang et al., 2026) on a maze problem with the latent trajectory projected onto the first two principal components. Trajectory color indicates recurrent depth, and pale green and red regions approximately indicate correct and incorrect predictions, respectively. (b) Schematic of ImR. For an input $\boldsymbol{x}$, repeated applications of the recurrent block $f_{\boldsymbol{x}}$ to the latent state $\boldsymbol{s}_t$ generate a latent trajectory. ImR predicts from the expectation $\mathbb{E}_{\boldsymbol{s}\sim\mu^\star}[g(\boldsymbol{s})]$ under an invariant measure $\mu^\star$.

We introduce invariant-measure reasoners (ImR), a framework that uses an invariant measure as a stable representation of latent dynamics. An invariant measure is a probability distribution over latent states that is preserved by recurrent updates (Eckmann & Ruelle, 1985). Bounded dynamics can admit such a measure even when individual states do not converge to a fixed point. As illustrated in Figure 1(b), ImR predicts from the expectation of the prediction head's output under this measure. Because recurrent updates preserve the measure, they also leave this expected output unchanged. Prediction from a fixed point is recovered as the special case in which all probability is concentrated at that point. In practice, ImR approximates this ideal prediction by averaging the prediction head's outputs over a finite latent trajectory, using the corresponding empirical measure as an approximation to an invariant measure. ImR retains the existing model architecture, including the recurrent block and prediction head, and introduces no additional trainable parameters.

Our main contributions are as follows:

- We introduce ImR, a framework for latent reasoning that uses invariant measures as a stable representation and includes prediction from a fixed point as a special case.
- We present two ways to use ImR: fine-tuning only the prediction head of existing models and training models from scratch. Both approaches reduce prediction instability and improve accuracy in many settings on maze and Sudoku tasks.
- We show that, in some settings, models trained with ImR exhibit non-fixed-point behavior more frequently than models trained to predict from the final state, yet form stable predictions and achieve high accuracy even with such behavior.

## 2 Preliminaries

We first overview latent reasoning models, which can be seen as discrete-time dynamical systems. To make the formulation concrete, we use structured reasoning tasks such as Sudoku and maze solving as running examples, but our formulation applies more broadly.

Structured reasoning tasks can typically be formulated as learning a mapping from an input configuration to a solution, both represented over a finite set of grid cells. For a task with $n$ grid cells indexed in a fixed order, let $\boldsymbol{x} \in \{1, \ldots, I\}^n$ denote the input configuration, where each cell takes one of the $I$ possible values. We represent the solution as $\boldsymbol{y} \in \{1, \ldots, C\}^n$, where $C$ is the number of output classes per cell. For mazes, $\boldsymbol{x}$ specifies the walls and the start and goal cells, while $\boldsymbol{y}$ encodes the solution path. For Sudoku, $\boldsymbol{x}$ is a partially filled $9 \times 9$ grid, and $\boldsymbol{y}$ represents a completed grid in which each row, column, and $3 \times 3$ box contains the digits 1 through 9 exactly once.

The latent reasoning models considered here repeatedly update a latent state that encodes an intermediate representation of the entire grid (Bansal et al., 2022; Huang et al., 2026). For $t \in \mathbb{N}_0 = \{0, 1, 2, \ldots\}$, we denote the latent state after $t$ recurrent updates by $\boldsymbol{s}_t \in \mathcal{S} \subseteq \mathbb{R}^d$, where $\mathcal{S}$ is the

state space and $d \in \mathbb{N} = \{1, 2, \ldots\}$ is the state dimension. We refer to $t$ as the recurrent depth. For a fixed input $\boldsymbol{x}$, these updates are performed by a recurrent block, which we represent as a continuous deterministic map $f_{\boldsymbol{x}} : \mathcal{S} \to \mathcal{S}$. This defines a discrete-time dynamical system:

$$\boldsymbol{s}_{t+1} = f_{\boldsymbol{x}}(\boldsymbol{s}_t). \tag{1}$$

The recurrent block $f_{\boldsymbol{x}}$ is implemented by a neural network, such as a transformer or a convolutional neural network (CNN), with parameters shared across recurrent steps and $\boldsymbol{x}$ supplied as input at each step. The initial state $\boldsymbol{s}_0$ may be computed from $\boldsymbol{x}$ (Schwarzschild et al., 2021; Bansal et al., 2022) or sampled from an initialization distribution $\rho_0$ on $\mathcal{S}$ that does not depend on $\boldsymbol{x}$ (Wang et al., 2025; Huang et al., 2026). For brevity, we write $f$ for $f_{\boldsymbol{x}}$ when the input is clear from context. Some models introduce noise into their recurrent updates, yielding stochastic latent dynamics (Huang et al., 2026; Baek et al., 2026; Sghaier et al., 2026). Appendix A.1 presents the corresponding stochastic formulation.

To produce a solution, we stop the recurrent updates at depth $T \in \mathbb{N}$, which we refer to as the stopping depth. Starting from $\boldsymbol{s}_0$, these updates generate a finite sequence of latent states $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T$, which we call a finite latent trajectory and denote by $(\boldsymbol{s}_t)_{t=1}^T$. A prediction head $g : \mathcal{S} \to \mathbb{R}^{n \times C}$, also implemented by a neural network, maps a latent state $\boldsymbol{s}_t$ to a matrix of logits $\boldsymbol{z}_t \in \mathbb{R}^{n \times C}$ over $C$ classes for each of the $n$ grid cells:

$$\boldsymbol{z}_t = g(\boldsymbol{s}_t). \tag{2}$$

Prediction from the logits $\boldsymbol{z}_T = g(\boldsymbol{s}_T)$ at stopping depth $T$ yields the solution $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ with $y_i = \arg\max_{c \in 1,\ldots,C} z_{T,i,c}$, where $z_{T,i,c} \in \mathbb{R}$ denotes the logit for class $c$ at cell $i$ in $\boldsymbol{z}_T$. A notable feature of latent reasoning models is that the stopping depth $T$ can vary during training and testing, but the resulting solution can depend on its choice.

## 3 Invariant-Measure Reasoners

As described in Section 2, prediction from the logits $\boldsymbol{z}_T = g(\boldsymbol{s}_T)$ at stopping depth $T$ can depend on the choice of $T$. Supporting different stopping depths is an important requirement, particularly for test-time scaling, where models perform more recurrent updates at test time than during training to improve generalization (Schwarzschild et al., 2021; Geiping et al., 2025; Huang et al., 2026; Knutson et al., 2026). However, latent trajectories in advanced models may converge to a compact subset of the state space without converging to a fixed point (Anil et al., 2022; Geiping et al., 2025; Huang et al., 2026; Knutson et al., 2026). Such dynamics may support reasoning beyond fixed-point convergence, but can also destabilize predictions: Figure 1(a) shows a trajectory repeatedly crossing between regions associated with correct and incorrect solutions.

To address this prediction instability, we propose invariant-measure reasoners (ImR), which use an invariant measure as a stable representation of latent dynamics and predict from the expected output of the prediction head under this measure. We first introduce invariant measures, then describe how ImR forms predictions and approximates them using finite latent trajectories. We then present procedures for fine-tuning only the prediction head and training models from scratch.

### 3.1 Invariant measures of latent dynamics

We characterize the long-term behavior of latent trajectories in a latent reasoning model from a distributional perspective. For a fixed input $\boldsymbol{x}$, we consider the empirical measure $\hat{\mu}_T$ of the finite latent trajectory $(\boldsymbol{s}_t)_{t=1}^T$ and assume that this measure converges weakly to a probability measure $\mu^\star$ on $\mathcal{S}$ as $T \to \infty$:

$$\hat{\mu}_T = \frac{1}{T} \sum_{t=1}^{T} \delta_{\boldsymbol{s}_t} \Rightarrow \mu^\star, \tag{3}$$

where $\delta_{\boldsymbol{s}_t}$ is the Dirac measure assigning probability one to $\boldsymbol{s}_t$, and $\Rightarrow$ denotes weak convergence. The empirical measure assigns equal weight to each visit along the finite latent trajectory and describes a distribution of latent states even when the recurrent updates are deterministic.

The probability measure $\mu^\star$ is invariant under recurrent updates. To express this property, we use the pushforward operator $f_\#$, also called the Perron–Frobenius operator (Lasota & Mackey, 1994). For a probability measure $\mu$, $f_\#\mu$ is the distribution obtained by applying $f$ to states distributed

according to $\mu$. See Appendix A.2 for the formal definition of the pushforward operator. Because $f$ is continuous, the weak convergence of the empirical measures along the trajectory implies

$$f_{\#}\mu^\star = \mu^\star. \tag{4}$$

Such a measure is called an invariant measure (Eckmann & Ruelle, 1985), a fixed point of $f_{\#}$ in the space of probability measures. It is preserved by recurrent updates even when individual latent states continue to evolve.

The support of $\mu^\star$ reflects the long-term behavior of the latent trajectory. If the trajectory converges to a compact set $\mathcal{B} \subseteq \mathcal{S}$, in the sense that $\inf_{\boldsymbol{s}\in\mathcal{B}} \|\boldsymbol{s}_t - \boldsymbol{s}\|_2 \to 0$ as $t \to \infty$, then $\mu^\star$ is supported on $\mathcal{B}$, meaning that $\mu^\star(\mathcal{B}) = 1$. If the trajectory converges to a fixed point $\boldsymbol{s}^\star$, we can take $\mathcal{B} = \{\boldsymbol{s}^\star\}$, with $\mu^\star = \delta_{\boldsymbol{s}^\star}$. If it converges to a periodic orbit with $P$ distinct states, we can take $\mathcal{B} = \{\boldsymbol{s}_1^\star, \ldots, \boldsymbol{s}_P^\star\}$, with $\mu^\star = P^{-1}\sum_{j=1}^{P} \delta_{\boldsymbol{s}_j^\star}$. More complex long-term behavior is also covered, provided that the empirical measure converges.

Note that the invariant measure $\mu^\star$ may depend on the initial state $\boldsymbol{s}_0$. Convergence to the same measure from different initial states is an additional desirable property, as in path-independent equilibrium models (Anil et al., 2022), but is not required by our convergence assumption. Appendix A.3 discusses this distinction.

### 3.2 Prediction from an invariant measure

Building on the preceding formulation, we define ImR as a framework for prediction based on the invariant measure $\mu^\star$ of latent dynamics. ImR predicts from the expectation of the logits under $\mu^\star$:

$$\boldsymbol{z}^\star = \mathbb{E}_{\boldsymbol{s}\sim\mu^\star}[g(\boldsymbol{s})] = \int_{\mathcal{S}} g(\boldsymbol{s})\, d\mu^\star(\boldsymbol{s}). \tag{5}$$

Here, we assume that the prediction head $g$ is continuous on $\mathcal{S}$ and that the logits $g(\boldsymbol{s}_t)$ remain bounded along the latent trajectory. Together with the weak convergence assumed in Section 3.1, these conditions ensure that the expected logits $\boldsymbol{z}^\star$ are finite. By equation 4, they are invariant under recurrent updates, since $\mathbb{E}_{\boldsymbol{s}\sim\mu^\star}[g(f(\boldsymbol{s}))] = \mathbb{E}_{\boldsymbol{s}\sim f_{\#}\mu^\star}[g(\boldsymbol{s})] = \boldsymbol{z}^\star$. This does not require the prediction head itself to be invariant: $g(f(\boldsymbol{s}))$ may differ from $g(\boldsymbol{s})$.

The invariant measure $\mu^\star$ describes limiting behavior and is generally not available explicitly. We approximate the expected logits $\boldsymbol{z}^\star$ using the empirical measure $\hat{\mu}_T$ at a finite stopping depth $T$:

$$\hat{\boldsymbol{z}}_T = \mathbb{E}_{\boldsymbol{s}\sim\hat{\mu}_T}[g(\boldsymbol{s})] = \frac{1}{T}\sum_{t=1}^{T} g(\boldsymbol{s}_t). \tag{6}$$

As in Section 2, we select the class with the largest averaged logit at each grid cell to obtain the solution. The weak convergence in equation 3, together with the conditions on $g$, implies $\hat{\boldsymbol{z}}_T \to \boldsymbol{z}^\star$ as $T \to \infty$. If $\boldsymbol{z}^\star$ has a unique maximum at each cell, the solution is constant for all sufficiently large $T$.

ImR requires only a simple change in how latent reasoning models compute predictions. Standard prediction from the final state uses $\boldsymbol{z}_T = g(\boldsymbol{s}_T)$, whereas ImR uses the averaged logits $\hat{\boldsymbol{z}}_T$. The former can continue to fluctuate as latent states evolve, while the latter converge under the stated assumptions, even when individual states do not. At finite depths, $\hat{\boldsymbol{z}}_T$ can still depend on the stopping depth $T$, but averaging can reduce this sensitivity and thereby improve prediction robustness across recurrent depths. ImR also retains the existing recurrent block and prediction head and introduces no additional trainable parameters.

At finite recurrent depths, states from the initial transient can affect the empirical approximation of $\mu^\star$. To reduce their influence, one can discard the states from the first $B$ recurrent updates, with $0 \le B < T$, and average the logits over steps $B+1, \ldots, T$. For fixed $B$, the resulting empirical measure converges to the same invariant measure $\mu^\star$ under the assumption in Section 3.1. Appendix B.3 defines this estimator and gives the choices used in our experiments.

ImR also applies when the prediction head uses only part of the latent state. In this case, the expected logits depend only on the corresponding marginal of $\mu^\star$, which is preserved when the full latent state is updated by $f$. Appendix B.2 specifies the latent state and the part used by the prediction head for each model in our experiments.

### 3.3 Training with Prediction from an Invariant Measure

We propose two approaches to training with ImR: fine-tuning a pretrained model and training a model from scratch. For fine-tuning, we retain the latent dynamics of the pretrained recurrent block $f$ and update only the prediction head $g$. We compute a loss from the averaged logits in equation 6 and the target solution, treating the latent states as constants so that gradients pass only through $g$. This approach aims to reduce prediction instability in an existing model without backpropagating through recurrent updates. In contrast, training from scratch jointly optimizes $f$ and $g$ using a loss computed from the averaged logits, allowing the model to learn latent dynamics suited to prediction with ImR.

For training from scratch, we combine ImR with a deep supervision procedure (Wang et al., 2025; Huang et al., 2026) that repeatedly supervises predictions as recurrent computation continues. The procedure consists of an inner loop and an outer loop. Each inner loop performs $L$ recurrent updates with fixed model parameters and computes the averaged logits $\hat{\boldsymbol{z}}_L$ over the resulting $L$ latent states. At each outer step, we compute the loss from these averaged logits and the target solution, then update the model parameters. We retain the original model's loss function, replacing prediction from the final state with prediction based on the averaged logits. The input remains fixed across outer steps, and each subsequent inner loop starts from the preceding inner loop's final latent state, detached from its computation graph.

To limit the cost of gradient computation, we use truncated backpropagation through time (TBPTT) (Williams & Peng, 1990), retaining gradients through the recurrent block $f$ only for the final $K \leq L$ updates of each inner loop. All $L$ states contribute to training the prediction head $g$, while earlier states are treated as constants when computing gradients. We construct a new empirical measure in each inner loop and do not carry it over to the next. Each empirical measure therefore contains only states generated with the same model parameters. Algorithm 1 summarizes the procedure, with further training details in Appendix B.3.

**Algorithm 1** Deep supervision for ImR

**Require:** input $\boldsymbol{x}$, target $\boldsymbol{y}$
Initialize the latent state $\boldsymbol{s}$
**for** each outer step ▷ outer loop
  $\boldsymbol{z}_{\text{sum}} \leftarrow 0$
  **for** $j = 1, \ldots, L$ ▷ inner loop
    **if** $j \leq L - K$ **then**
      $\boldsymbol{s} \leftarrow \text{NoGrad}[f_{\boldsymbol{x}}(\boldsymbol{s})]$
    **else**
      $\boldsymbol{s} \leftarrow f_{\boldsymbol{x}}(\boldsymbol{s})$
    **end if**
    $\boldsymbol{z}_{\text{sum}} \leftarrow \boldsymbol{z}_{\text{sum}} + g(\boldsymbol{s})$
  **end for**
  $\hat{\boldsymbol{z}}_L \leftarrow \boldsymbol{z}_{\text{sum}}/L$ ▷ empirical expectation
  Compute the loss from $\hat{\boldsymbol{z}}_L$ and $\boldsymbol{y}$
  Backpropagate the loss
  Update the model parameters
  Detach $\boldsymbol{s}$ for the next outer step
**end for**

## 4 Related Work

**Equilibrium models and stability.** Deep equilibrium models define latent representations as fixed points of recurrent transformations (Bai et al., 2019). To encourage convergence to a fixed point, Jacobian regularization has been used in equilibrium and looped models (Bai et al., 2021; Yang et al., 2026). However, such regularization can restrict the model class and does not guarantee convergence to a fixed point (Bai et al., 2021). Recent studies have also reported long-term behavior beyond convergence to a fixed point in latent reasoning models (Anil et al., 2022; Geiping et al., 2025; Huang et al., 2026; Knutson et al., 2026). In this paper, we consider latent trajectories whose empirical measures converge weakly, encompassing both convergence to a fixed point and more general long-term behavior. ImR generalizes the formulation based on convergence to a fixed point in the state space to one based on weak convergence of empirical measures, where the representation satisfies $f_{\#}\mu^{\star} = \mu^{\star}$.

**Adaptive computation and halting.** Adaptive computation time (ACT) learns a halting signal that determines the stopping depth and weights for averaging intermediate states and outputs (Graves, 2017). PonderNet learns a distribution over stopping depths and samples a stopping depth for prediction at inference (Banino et al., 2021). ImR predicts from the expectation of the prediction head's output under the empirical measure of a latent trajectory. Its weights reflect state visit frequencies rather than learned halting probabilities, and it introduces no additional trainable parameters. ImR

remains complementary to halting mechanisms, which can determine when to stop collecting states to estimate the invariant measure.

# 5 EXPERIMENTS

We conducted numerical experiments to evaluate whether ImR reduces prediction instability and improves accuracy across different architectures. We used EqR (Huang et al., 2026) and the deep thinking network with progressive training and recall (Bansal et al., 2022), which we refer to as DT. For both architectures, we fine-tuned only the prediction head. With EqR, we also trained models from scratch and assessed prediction stability separately for trajectories classified as fixed and non-fixed. With DT, we examined prediction instability on out-of-distribution (OOD) mazes.

**Model architectures.** EqR uses transformer blocks on Maze-Unique and multilayer perceptrons (MLPs) to mix information across positions and channels on Sudoku-Extreme, while DT uses a CNN. ImR models use the EqR architecture on Maze-Unique and Sudoku-Extreme and the DT architecture on OOD mazes.

EqR adopts hierarchical iteration, which uses a high-level latent state $\boldsymbol{s}^{(H)}$ and a low-level latent state $\boldsymbol{s}^{(L)}$. Together, these states form the full latent state $\boldsymbol{s}$. At each recurrent step, the same module updates $\boldsymbol{s}^{(L)}$ several times and then $\boldsymbol{s}^{(H)}$ once. EqR also uses a puzzle prefix to extend the input representation and both latent states with additional positions beyond those corresponding to grid cells. In the input representation, the prefix consists of a learned vector shared across problems and zero vectors. The corresponding positions in both latent states are updated along with the grid positions, providing additional dimensions for recurrent computation. The prediction head uses only the grid positions of $\boldsymbol{s}^{(H)}$. Appendix B.2 gives architectural details.

**Baselines.** We included the released EqR and DT models as baselines without fine-tuning. To distinguish the effect of ImR on prediction stability from that of fine-tuning the prediction head, we also included fine-tuned baselines, EqR (FT) and DT (FT). These baselines were trained using each model's original task loss applied to the logits $g(\boldsymbol{s}_T)$ from the final latent state. We obtained ImR (FT) by fine-tuning with the same loss applied to the expectation of $g$ under the empirical measure of the latent trajectory. For each architecture, both fine-tuning methods started from the same released weights.

For the EqR experiments without the puzzle prefix, we also trained EqR baselines from scratch using prediction from the final latent state. In the EqR experiments, ImR without the suffix "(FT)" denotes models trained from scratch. We used 3 training seeds for each configuration trained or fine-tuned in this study. Results for each released baseline are from a single pretrained model.

**Evaluation.** At each recurrent depth $t$, EqR, DT, and their fine-tuned versions predict from the current latent state. All ImR models predict from the expectation of $g$ under the empirical measure $\hat{\mu}_t$, which uses all states from depths $1, \ldots, t$. We plot accuracy as a function of recurrent depth. Accuracy alone does not fully capture prediction instability: changes between correct and incorrect predictions can offset each other across instances, and changes between different incorrect predictions do not affect accuracy.

We also assess prediction stability for each instance over an interval specified in each experiment. An instance is stable correct if the model outputs the same correct prediction throughout the interval. It is stable incorrect if the model outputs the same incorrect prediction throughout the interval. It is unstable if its prediction changes at least once, including changes between different incorrect predictions.

## 5.1 EQR ON MAZE-UNIQUE AND SUDOKU-EXTREME

We evaluated EqR and ImR on Maze-Unique (Huang et al., 2026) and Sudoku-Extreme (Wang et al., 2025). Maze-Unique requires marking the unique path from the start cell to the goal cell on a $30 \times 30$ grid without crossing walls. Sudoku-Extreme requires completing a partially filled $9 \times 9$ Sudoku grid. For both tasks, a prediction is correct only if the entire predicted grid matches the target. We used the released datasets and constructed validation sets for hyperparameter selection.

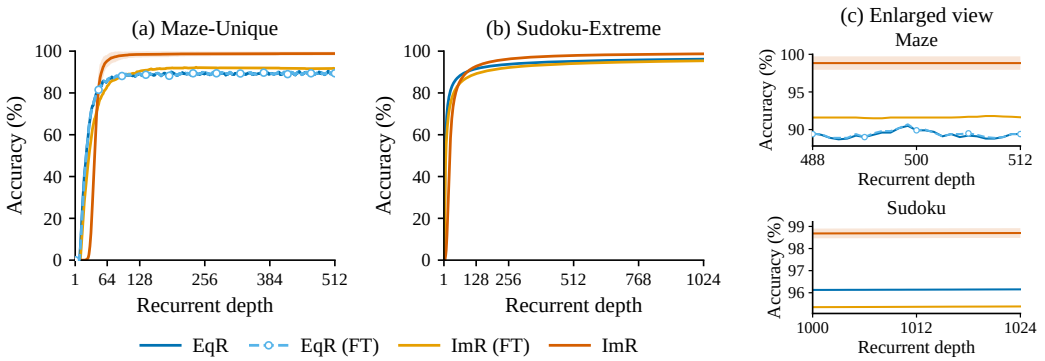



Figure 2: Accuracy across recurrent depth on Maze-Unique (a) and Sudoku-Extreme (b), with enlarged views (c) over the final 25 recurrent depths. For both fine-tuned models and ImR, curves show means and shaded regions show standard deviation across 3 training seeds.

We largely followed the model configurations, datasets, and hyperparameters of EqR (Huang et al., 2026). Accuracy and prediction stability were evaluated up to recurrent depths of $T = 512$ on Maze-Unique and $T = 1024$ on Sudoku-Extreme, with transition noise set to zero during evaluation. Appendix B provides the data splits and detailed training and evaluation settings.

We compared models with and without the puzzle prefix to examine how latent state dimensionality relates to learned dynamics and prediction stability. All models without the prefix were trained from scratch. For each model, the prefix configuration remained unchanged throughout training and evaluation.

To assess prediction stability under different latent dynamics, we classified trajectories as fixed or non-fixed using the relative $L_\infty$ residual between successive values of $\boldsymbol{s}^{(H)}$ at the grid positions. A trajectory was classified as *fixed* if this residual remained below $0.05$ throughout the final 64 recurrent updates, and as *non-fixed* otherwise. These labels describe state changes over the evaluated interval and do not guarantee convergence of the full latent state to a fixed point. Within each group, we evaluated prediction stability over the final 64 recurrent depths. For configurations with multiple training seeds, we computed the proportions separately for each trained model and reported their mean and standard deviation. Appendix B.4 provides the residual definition and calculation details.

**Results.** With the puzzle prefix, ImR trained from scratch achieves higher accuracy than the released EqR model on both tasks at large recurrent depths, as shown in Figure 2. At these depths, ImR (FT) achieves higher accuracy than EqR (FT) on Maze-Unique but lower accuracy than the released EqR model on Sudoku-Extreme. For Sudoku-Extreme, EqR (FT) is not shown separately because fine-tuning did not improve validation accuracy and the initial prediction head was retained. Table 7 in Appendix C.1 reports the corresponding accuracies at the final recurrent depth. The larger gains from training from scratch suggest that jointly training the recurrent block and prediction head with ImR can be more effective than fine-tuning the prediction head alone.

Table 1 shows that ImR reduces prediction instability in the non-fixed group. Predictions in the fixed group are predominantly stable correct across all models, whereas EqR predictions in the non-fixed group are mostly unstable. ImR (FT) reduces this instability on both tasks while leaving the recurrent block unchanged, showing that prediction stability can improve without altering the latent dynamics. On Sudoku-Extreme, however, most non-fixed predictions under ImR (FT) are stable incorrect, so this stabilization does not translate into higher accuracy. Among ImR models trained from scratch, those without the prefix have higher mean non-fixed rates than those with it, reaching about 50% on both tasks, with substantial variation across seeds. Nevertheless, over 97% of predictions in their non-fixed group are stable correct on average. These results show that ImR can improve prediction stability and accuracy even for trajectories classified as non-fixed.

Figure 6 in Appendix C.2 shows principal component analysis projections of additional latent trajectories.

Table 1: Trajectory groups and prediction stability. Entries with $\pm$ report mean and standard deviation across 3 training seeds. EqR$^\dagger$ reports results for both EqR and EqR (FT), which match on all reported metrics. For each dataset and state dimension of $\boldsymbol{s}^{(H)}$, bold marks the highest stable correct proportion and the lowest unstable proportion in the non-fixed group.

| | | Fixed | | | Non-fixed | | |
|---|---|---|---|---|---|---|---|
| Model | Non-fixed rate | Stable correct | Stable incorrect | Unstable | Stable correct | Stable incorrect | Unstable |
| *Sudoku-Extreme*: dimension of $\boldsymbol{s}^{(H)}$: 49.7K; parameters 5.03M | | | | | | | |
| EqR$^\dagger$ | 4.1 | 100.0 | 0.0 | 0.0 | 3.6 | 0.0 | 96.4 |
| ImR (FT) | $4.1 \pm 0.0$ | $99.2 \pm 0.0$ | $0.0 \pm 0.0$ | $0.8 \pm 0.0$ | $3.4 \pm 0.0$ | $79.7 \pm 0.1$ | $\mathbf{16.9 \pm 0.1}$ |
| ImR | $1.2 \pm 0.2$ | $99.7 \pm 0.1$ | $0.0 \pm 0.0$ | $0.2 \pm 0.1$ | $\mathbf{8.0 \pm 12.2}$ | $44.5 \pm 5.8$ | $47.5 \pm 8.0$ |
| *Sudoku-Extreme*: dimension of $\boldsymbol{s}^{(H)}$: 41.5K; parameters 4.86M | | | | | | | |
| EqR | $4.5 \pm 0.9$ | $100.0 \pm 0.0$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $11.6 \pm 18.9$ | $0.0 \pm 0.0$ | $88.4 \pm 18.9$ |
| ImR | $50.3 \pm 33.7$ | $99.7 \pm 0.1$ | $0.0 \pm 0.0$ | $0.3 \pm 0.1$ | $\mathbf{97.9 \pm 0.8}$ | $0.8 \pm 0.3$ | $\mathbf{1.4 \pm 0.5}$ |
| *Maze-Unique*: dimension of $\boldsymbol{s}^{(H)}$: 117.2K; parameters 264.1K | | | | | | | |
| EqR$^\dagger$ | 14.7 | 98.5 | 1.5 | 0.0 | 16.3 | 0.7 | 83.0 |
| ImR (FT) | $14.7 \pm 0.0$ | $98.4 \pm 0.0$ | $1.1 \pm 0.0$ | $0.6 \pm 0.0$ | $51.7 \pm 0.0$ | $23.8 \pm 1.8$ | $24.5 \pm 1.8$ |
| ImR | $3.6 \pm 3.2$ | $99.8 \pm 0.2$ | $0.2 \pm 0.1$ | $0.1 \pm 0.1$ | $\mathbf{71.9 \pm 4.5}$ | $24.9 \pm 8.4$ | $\mathbf{3.2 \pm 5.5}$ |
| *Maze-Unique*: dimension of $\boldsymbol{s}^{(H)}$: 115.2K; parameters 263.9K | | | | | | | |
| EqR | $18.6 \pm 21.2$ | $99.9 \pm 0.1$ | $0.1 \pm 0.1$ | $0.0 \pm 0.0$ | $11.7 \pm 17.6$ | $0.2 \pm 0.4$ | $88.1 \pm 17.4$ |
| ImR | $50.8 \pm 28.4$ | $100.0 \pm 0.1$ | $0.0 \pm 0.1$ | $0.0 \pm 0.0$ | $\mathbf{97.5 \pm 3.9}$ | $2.1 \pm 3.3$ | $\mathbf{0.5 \pm 0.6}$ |

### 5.2 DT ON OOD MAZES

Prediction instability may increase when a pretrained recurrent model is applied to more challenging problems. We examined whether fine-tuning with ImR can reduce this instability on OOD mazes. We characterized maze structure by the degree of the start cell, defined as its number of traversable neighbors. The released DT model was pretrained on $9 \times 9$ mazes whose start cell had degree 1. We refer to mazes satisfying these conditions as in-distribution (ID) mazes.

Following the OOD maze setting of Knutson et al. (2026), we evaluated mazes whose start cells had degrees from 1 to 4. The test set contained 100 mazes at each of 10 sizes from $9 \times 9$ to $99 \times 99$, for 1,000 mazes in total. Mazes with a start cell of degree greater than 1 are OOD relative to the pretraining conditions, and larger mazes are also OOD in size. We analyzed the results separately by the degree of the start cell. Figure 4 in Appendix B.1 shows examples of ID and OOD inputs.

We compared the released DT model with DT (FT) and ImR (FT). Fine-tuning and hyperparameter selection used only ID mazes. We evaluated predictions at every recurrent depth from 1 through 5,000 and assessed prediction stability over depths 4,900–5,000. Appendix B describes the dataset construction and gives detailed training and evaluation settings.

**Results.** The accuracy gains of ImR (FT) emerge at larger recurrent depths, as shown in Figure 3. For mazes whose start cell has degree 2 or 3, its accuracy initially increases more slowly than that of the released DT model, but eventually surpasses it and remains higher through the end of evaluation. At the final depth, ImR (FT) also achieves higher accuracy than DT (FT) in both groups and across the full test set. Table 7 in Appendix C.1 reports the final-depth accuracies for the degree-2 and degree-3 groups. These results indicate that additional recurrent computation can improve predictions through the empirical expectation even after accuracy from the final state has largely plateaued. Figure 7 in Appendix C.3 provides results separately for each maze size.

Table 2 shows that these accuracy gains are accompanied by more stable predictions for individual instances. Relative to DT (FT), ImR (FT) further reduces the proportion of unstable predictions and increases the stable correct proportion in both groups. Although the stable incorrect proportion also increases, particularly for degree 3, the gains over DT (FT) show that ImR can obtain more stable and accurate predictions from the same pretrained dynamics. These gains are achieved under the

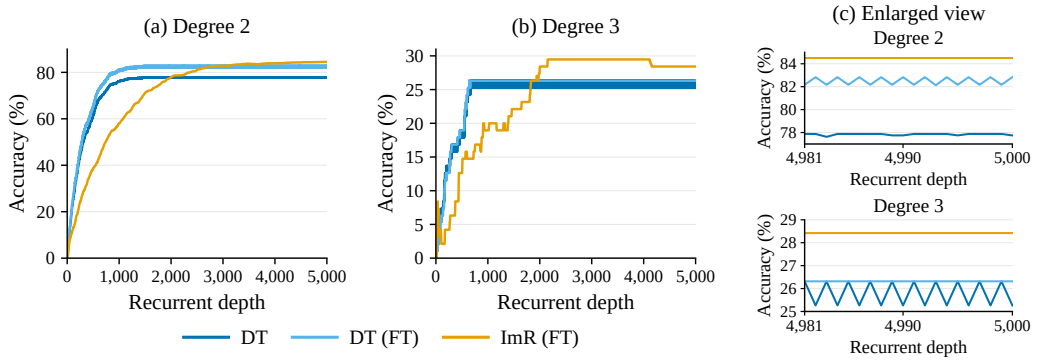



Figure 3: Accuracy across recurrent depth on OOD mazes whose start cell has degree 2 (a) or 3 (b), with enlarged views (c) over the final 20 recurrent depths. For both fine-tuned models, curves show means and shaded regions show standard deviation across 3 training seeds.

evaluated OOD conditions using only ID mazes for fine-tuning and hyperparameter selection, with the recurrent block unchanged.

For mazes whose start cell has degree 1, all models solve every test maze at the final depth, including mazes larger than those seen during pretraining. Thus, the gains on degree-2 and degree-3 mazes are obtained while preserving DT's performance on degree-1 mazes that are OOD in size. The test set contains only one maze whose start cell has degree 4, and none of the evaluated models produces a correct solution for this instance.

Table 2: Prediction stability on OOD mazes over depths 4,900–5,000. Fine-tuned models report mean ± standard deviation across 3 training seeds. For each degree of the start cell, bold marks the highest stable correct proportion and the lowest unstable proportion.

| Model | Stable correct (%) | Stable incorrect (%) | Unstable (%) |
|---|---|---|---|
| *Degree of the start cell: 2* | | | |
| DT | 76.6 | 17.3 | 6.1 |
| DT (FT) | $81.5 \pm 0.1$ | $13.9 \pm 0.1$ | $4.5 \pm 0.1$ |
| ImR (FT) | $\mathbf{84.5 \pm 0.0}$ | $15.1 \pm 0.1$ | $\mathbf{0.4 \pm 0.1}$ |
| *Degree of the start cell: 3* | | | |
| DT | 25.3 | 57.9 | 16.8 |
| DT (FT) | $26.3 \pm 0.0$ | $62.8 \pm 0.6$ | $10.9 \pm 0.6$ |
| ImR (FT) | $\mathbf{28.4 \pm 0.0}$ | $70.2 \pm 0.6$ | $\mathbf{1.4 \pm 0.6}$ |

## 6 Discussion

The results without the puzzle prefix suggest that ImR can support accurate reasoning in lower-dimensional latent spaces. Compared with ImR models trained with the prefix, those trained without it used fewer latent state dimensions and had a higher mean non-fixed rate. Nevertheless, their predictions in the non-fixed group were predominantly stable correct. Understanding how latent trajectories and their induced measures represent solutions as latent state dimensionality varies could guide the design of more efficient recurrent models.

Our experiments used fixed stopping depths. Developing adaptive stopping rules that balance predictive accuracy and computation remains an important direction for future work.

## 7 Conclusion

We introduced ImR to address prediction instability at large recurrent depths in latent reasoning models. ImR uses an invariant measure as a stable representation of latent dynamics and predicts from the expectation of the prediction head's output under this measure. Experiments on maze and Sudoku tasks showed reduced prediction instability and improved accuracy in many settings. In particular, models trained with ImR produced stable and correct predictions even when their latent states continued to evolve. These findings support a broader formulation of latent reasoning based on convergence of empirical measures, allowing stable prediction without requiring individual states to converge to a fixed point.

### AI USE STATEMENT

We used generative AI tools to assist with drafting and revising the manuscript, searching for relevant literature, and discussing mathematical formulations and their assumptions. We also used these tools to write code for the numerical experiments and to discuss the interpretation and presentation of experimental results. The authors reviewed and revised the AI-assisted material, including the generated code, and take full responsibility for the final content of this work.

### REPRODUCIBILITY STATEMENT

Section 3 describes the prediction rule and training procedures for ImR, with an overview of training from scratch in Algorithm 1. Appendix B.1 describes the datasets, their preparation, and the training, validation, and test splits. Appendix B.2 specifies the model architectures, while Appendix B.3 details the training and evaluation settings and the procedures for selecting hyperparameters and checkpoints. Appendix B.4 provides the criteria for classifying latent trajectories and the procedures for calculating prediction stability proportions within each group. Appendix B.5 describes how the PCA projection and prediction regions in Figure 1(a) were constructed and reports the corresponding predictions from the actual latent states.

## REFERENCES

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

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning, 2026. arXiv preprint arXiv:2605.19376.

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

Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing equilibrium models by Jacobian regularization. In *International Conference on Machine Learning*, 2021.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, 2022.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, 2018.

J.-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57(3):617–656, 1985.

Katie E. Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning*, 2024.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025.

Alex Graves. Adaptive computation time for recurrent neural networks, 2017. arXiv preprint arXiv:1603.08983.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Brandon Knutson, Amandin Chyba Rabeendran, Michael I. Ivanitskiy, Jordan Pettyjohn, Cecilia G. Diniz Behn, Samy Wu Fung, and Daniel McKenzie. On logical extrapolation for mazes with recurrent and implicit networks. In *Fortieth AAAI Conference on Artificial Intelligence*, 2026.

Andrzej Lasota and Michael C. Mackey. *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*. Applied Mathematical Sciences. Springer-Verlag, 1994.

A. Mauroy and I. Mezić. On the use of Fourier averages to compute the global isochrons of (quasi)periodic dynamics. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(3): 033112, 2012.

Igor Mezić and Andrzej Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1):101–133, 2004.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. In *4th ICML Workshop on High-dimensional Learning Dynamics (HiLD)*, 2026.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, 2021.

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

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems*, 2021.

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

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

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

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.

Xiao-Wen Yang, Ziyu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing recurrent dynamics for test-time scalable latent reasoning in looped language models. In *International Conference on Machine Learning*, 2026.

# A An Operator-Theoretic Perspective on Latent Reasoning

This appendix describes how recurrent updates act on probability measures and on functions of latent states. Appendix A.1 introduces stochastic latent dynamics. Appendix A.2 presents the pushforward and Koopman operators and relates prediction under an invariant measure to the invariant component of the prediction head. Appendix A.3 distinguishes convergence along a trajectory from convergence to a common measure across initializations.

## A.1 Stochastic Latent Dynamics

Some latent reasoning models, including EqR, sample noise during their recurrent updates. We collect all noise samples used within one complete recurrent update into $\epsilon_t$. For EqR, the full latent state contains both $\boldsymbol{s}^{(H)}$ and $\boldsymbol{s}^{(L)}$, and one recurrent update consists of $J$ internal updates of $\boldsymbol{s}^{(L)}$ followed by one update of $\boldsymbol{s}^{(H)}$. Thus, we write $\epsilon_t = (\epsilon_{t,1}, \ldots, \epsilon_{t,J}, \epsilon_{t,H})$, where each entry is an independently sampled Gaussian noise vector added during the corresponding internal update. The stochastic extension of equation 1 is

$$\boldsymbol{s}_{t+1} = \tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}_t, \epsilon_t), \qquad \epsilon_t \overset{\text{i.i.d.}}{\sim} \nu = \mathcal{N}(0, \sigma^2 I). \tag{7}$$

Given these noise samples, $\tilde{f}_{\boldsymbol{x}}$ performs the internal updates in sequence to produce the next latent state. Appendix B.2 gives the explicit update equations. Setting all noise samples to zero recovers the deterministic map:

$$\tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}, 0) = f_{\boldsymbol{x}}(\boldsymbol{s}). \tag{8}$$

The input, model parameters, and noise distribution remain fixed throughout the trajectory, and the noise sequence is independent of the initial state.

### A.2 Pushforward and Koopman Operators

To describe how recurrent updates transform state distributions, we use the pushforward introduced in the main text. For a probability measure $\mu$ on $\mathcal{S}$, $(f_{\boldsymbol{x}})_{\#}\mu$ is the distribution obtained by applying $f_{\boldsymbol{x}}$ to a state sampled from $\mu$. For every measurable set $A \subseteq \mathcal{S}$,

$$\big((f_{\boldsymbol{x}})_{\#}\mu\big)(A) = \mu\big(f_{\boldsymbol{x}}^{-1}(A)\big) = \mu(\{\boldsymbol{s} \in \mathcal{S} : f_{\boldsymbol{x}}(\boldsymbol{s}) \in A\}). \tag{9}$$

Here, $f_{\boldsymbol{x}}^{-1}(A)$ is the preimage of $A$, the set of latent states that $f_{\boldsymbol{x}}$ maps into $A$.

For fixed noise samples $\epsilon$, the same definition applies to $\tilde{f}_{\boldsymbol{x}}(\cdot, \epsilon)$. The expectation of this pushforward under the noise distribution $\nu$ defines the operator $\mathcal{P}$, also called the Perron–Frobenius or transfer operator. Its dual, the Koopman operator $\mathcal{K}$, acts on scalar functions $\varphi$ of latent states, called observables (Mezić & Banaszuk, 2004):

$$\begin{aligned} \mathcal{P}\mu &= \int \big(\tilde{f}_{\boldsymbol{x}}(\cdot, \epsilon)\big)_{\#}\mu \, d\nu(\epsilon), \\ (\mathcal{K}\varphi)(\boldsymbol{s}) &= \int \varphi\Big(\tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}, \epsilon)\Big) \, d\nu(\epsilon). \end{aligned} \tag{10}$$

The deterministic dynamics correspond to $\nu = \delta_0$, which gives

$$\mathcal{P}\mu = (f_{\boldsymbol{x}})_{\#}\mu, \qquad (\mathcal{K}\varphi)(\boldsymbol{s}) = (\varphi \circ f_{\boldsymbol{x}})(\boldsymbol{s}) = \varphi(f_{\boldsymbol{x}}(\boldsymbol{s})).$$

The prediction head $g$ in the main text defines a scalar observable $\boldsymbol{s} \mapsto (g(\boldsymbol{s}))_{i,c}$ for each grid cell $i$ and class $c$.

We write $\langle \mu, \varphi \rangle = \int_{\mathcal{S}} \varphi(\boldsymbol{s}) \, d\mu(\boldsymbol{s})$ for the pairing between a measure and an observable. For bounded measurable observables, the definitions above give

$$\begin{aligned} \langle \mathcal{P}\mu, \varphi \rangle &= \int_{\mathcal{S}} \varphi(\boldsymbol{s}) \, d(\mathcal{P}\mu)(\boldsymbol{s}) \\ &= \int \int_{\mathcal{S}} \varphi\Big(\tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}, \epsilon)\Big) \, d\mu(\boldsymbol{s}) \, d\nu(\epsilon) \\ &= \int_{\mathcal{S}} \left[ \int \varphi\Big(\tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}, \epsilon)\Big) \, d\nu(\epsilon) \right] d\mu(\boldsymbol{s}) \\ &= \int_{\mathcal{S}} (\mathcal{K}\varphi)(\boldsymbol{s}) \, d\mu(\boldsymbol{s}) = \langle \mu, \mathcal{K}\varphi \rangle. \end{aligned} \tag{11}$$

If $\mu_t$ denotes the distribution of latent states at depth $t$, then this distribution evolves according to $\mu_{t+1} = \mathcal{P}\mu_t$. The same recurrent computation therefore admits three descriptions: the recurrent map evolves latent states, $\mathcal{K}$ describes the expected evolution of functions of those states, such as individual logits, and $\mathcal{P}$ evolves the distribution of latent states. Duality connects the latter two descriptions. The expectation of an observable under the evolved distribution equals the expectation of the evolved observable under the original distribution.

An invariant measure $\mu$ satisfies $\mathcal{P}\mu = \mu$. For stochastic dynamics, such a measure is also called a stationary measure. Duality then gives

$$\langle \mu, \mathcal{K}\varphi \rangle = \langle \mathcal{P}\mu, \varphi \rangle = \langle \mu, \varphi \rangle. \tag{12}$$

Thus, recurrent updates preserve expectations under an invariant measure, even though individual latent states may change.

**Eigenfunctions and Fourier averages.** We now consider the deterministic dynamics in the main text. An invariant measure is an eigenmeasure of $\mathcal{P}$ with eigenvalue 1. Likewise, a nonzero observable satisfying $\mathcal{K}\varphi = \varphi$ is a Koopman eigenfunction with eigenvalue 1. This equation expresses invariance of the observable under recurrent updates, since $\varphi \circ f_{\boldsymbol{x}} = \varphi$.

Applying $\mathcal{K}$ coordinatewise to the prediction head gives $(\mathcal{K}^t g)(\boldsymbol{s}_0) = g(\boldsymbol{s}_t)$. The empirical expectation in equation 6 is therefore the finite Fourier average at frequency zero, evaluated at $\boldsymbol{s}_0$ (Mauroy & Mezić, 2012). The corresponding limiting function is

$$\bar{g}(\boldsymbol{s}) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} (\mathcal{K}^t g)(\boldsymbol{s}), \tag{13}$$

wherever this limit exists. Shifting the starting state by one recurrent update leaves the limit unchanged, so $\bar{g}(f_{\boldsymbol{x}}(\boldsymbol{s})) = \bar{g}(\boldsymbol{s})$, or equivalently $\mathcal{K}\bar{g} = \bar{g}$. Each nonzero coordinate function of $\bar{g}$ is therefore a Koopman eigenfunction with eigenvalue 1.

Under the empirical convergence assumption and the conditions on $g$ in the main text, the prediction from the invariant measure satisfies

$$\boldsymbol{z}^\star = \int_{\mathcal{S}} g(\boldsymbol{s})\, d\mu^\star(\boldsymbol{s}) = \bar{g}(\boldsymbol{s}_0). \tag{14}$$

This perspective clarifies that ImR does not assume $g \circ f_{\boldsymbol{x}} = g$ or enforce this identity through training or the construction of the prediction head. It predicts using the invariant component $\bar{g}$ extracted from $g$ by the Fourier average, even when $g(f_{\boldsymbol{x}}(\boldsymbol{s})) \neq g(\boldsymbol{s})$.

### A.3 Dependence on Initialization

For the deterministic dynamics in the main text, different initial states can lead to different long-term behavior even when the input $\boldsymbol{x}$ is fixed and the trajectories remain bounded. The empirical convergence in equation 3 does not require the invariant measure $\mu^\star$ to be the same across initial states.

We group initial states according to the invariant measure to which their empirical measures converge. Let $\{\mu^\star_\alpha\}_{\alpha\in\mathcal{I}}$ denote these distinct invariant measures, where $\mathcal{I}$ is an index set. For each $\alpha \in \mathcal{I}$, we define the corresponding class of initial states as

$$\mathcal{S}_\alpha = \{\boldsymbol{s}_0 \in \mathcal{S} : \hat{\mu}_T \to \mu^\star_\alpha \quad (T \to \infty)\}. \tag{15}$$

Here, $\hat{\mu}_T$ is the empirical measure in equation 3 for the trajectory initialized at $\boldsymbol{s}_0$, and convergence is weak. The sets $\mathcal{S}_\alpha$ are pairwise disjoint and partition the initial states for which the empirical convergence holds. When each limiting invariant measure $\mu^\star_\alpha$ is ergodic, the partition $\{\mathcal{S}_\alpha\}_{\alpha\in\mathcal{I}}$ corresponds to an ergodic partition (Mezić & Banaszuk, 2004), although our formulation does not assume ergodicity.

We do not assume that $\mu^\star$ is independent of initialization, and the notation in the main text suppresses this dependence. Independence from initialization is a desirable property that may emerge through training. However, for some inputs and initial states, the learned dynamics may still lead trajectories toward subsets associated with incorrect predictions, as illustrated by the analysis of EqR (Huang et al., 2026).

## B Experimental and Training Details

We first describe the datasets and model architectures used in our experiments. We then present the training and evaluation settings. Finally, we describe how we classify latent trajectories and visualize latent dynamics.

### B.1 Datasets and Validation

**Datasets for EqR.** We used the Maze-Unique and Sudoku-Extreme training and test sets released with EqR (Huang et al., 2026).[1] Maze-Unique contains $30 \times 30$ mazes with unique solution paths of 100–140 moves. The target marks the solution path on the full grid. For Sudoku-Extreme, the training set contains $1{,}000$ base puzzles on $9 \times 9$ grids, each with $1{,}000$ augmented versions. Augmentation used digit permutations, transposition, and row and column permutations that preserve Sudoku constraints. Including the original puzzles gives $1{,}001{,}000$ training examples.

We constructed validation sets for hyperparameter selection. For Maze-Unique, we generated $1{,}000$ mazes under the same conditions as the training set released with EqR and excluded exact matches to mazes in that set. For Sudoku-Extreme, we selected $10{,}000$ unaugmented puzzles from the official Sudoku-Extreme training pool. We excluded exact matches to puzzles in EqR's released Sudoku-Extreme training set and potential duplicates of those puzzles under Sudoku symmetries. For a fair comparison with EqR, the validation examples were not added to the training sets. The test sets were used only for final evaluation.

[1] https://huggingface.co/datasets/locuslab/EqR-data

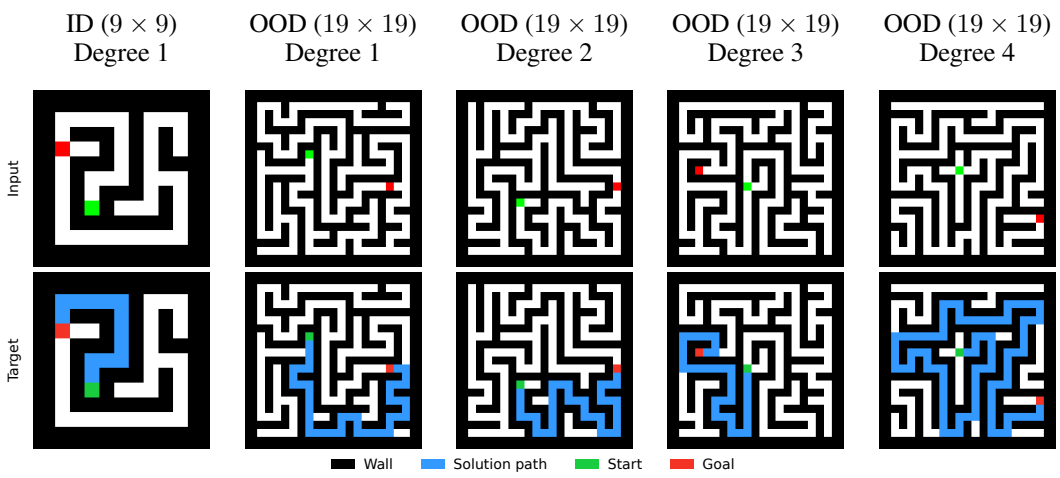



Figure 4: Examples of ID and OOD mazes used for DT. The first column shows a $9 \times 9$ ID validation maze. The remaining columns show $19 \times 19$ OOD mazes whose start cells have degrees 1 through 4. The top row contains the inputs, and the bottom row marks the target solution path in blue, the start cell in green, and the goal cell in red.

**Datasets for DT.** For fine-tuning and validation, we used the same 50,000 ID mazes of size $9 \times 9$ as Schwarzschild et al. (2021) and Bansal et al. (2022). We randomly split these into 40,000 mazes for fine-tuning and 10,000 for validation. The 80/20 proportions match those in the original DT implementation.

We generated 100 test mazes at each size $m \times m$, with $m = 9, 19, \ldots, 99$, using the public code of Knutson et al. (2026). Each maze was generated using depth-first search, with no additional edges added to the resulting tree. Start cells were not restricted to dead ends and could have degrees 1 through 4. The target was the unique path from the start cell to the goal cell.

Following the input and target format used in prior work (Schwarzschild et al., 2021; Bansal et al., 2022), we converted the generated mazes and solution paths into input images and target masks. For a maze of size $m \times m$, the generator used a $\frac{m+1}{2} \times \frac{m+1}{2}$ grid. Rendering the passages and walls, including the outer walls, produced an image of $(m + 2) \times (m + 2)$ pixels. We expanded each pixel to $2 \times 2$ pixels and added padding of 1 pixel on each side using the pixel value for walls. The resulting input dimensions were $H = W = 2m + 6$. For example, a $9 \times 9$ maze produced a $24 \times 24$ input image. Figure 4 shows examples of the ID validation mazes and OOD test mazes.

Table 3 summarizes the dataset sizes and the counts of generated test mazes by the degree of the start cell.

Table 3: Dataset sizes (top) and counts of generated test mazes by the degree of the start cell (bottom). The degree counts are pooled across all maze sizes. The Sudoku-Extreme training count includes augmentation.

| Dataset | Train | Validation | Test |
|---|---|---|---|
| Maze-Unique | 1,000 | 1,000 | 1,000 |
| Sudoku-Extreme | 1,001,000 | 10,000 | 422,786 |
| Maze | 40,000 | 10,000 | 1,000 |

| Degree of the start cell | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Test mazes | 104 | 800 | 95 | 1 |

### B.2 Model Architectures and the Number of Parameters

We use the model architectures of EqR (Huang et al., 2026) and DT (Bansal et al., 2022) unchanged when applying ImR. Table 4 summarizes the model architectures and the number of parameters.

Table 4: Model architectures and the number of parameters. For EqR, parameters used to embed the input are included under $f_{\boldsymbol{x}}$. The totals also include the halting head in EqR and the neural network used to initialize the latent state in DT.

| Dataset | Position mixing | Puzzle prefix | $f_{\boldsymbol{x}}$ | $g$ | Total |
|---|---|---|---|---|---|
| Maze-Unique | Self-attention | With | 263,040 | 768 | 264,066 |
| | | Without | 262,912 | 768 | 263,938 |
| Sudoku-Extreme | MLP | With | 5,022,720 | 5,632 | 5,029,378 |
| | | Without | 4,848,640 | 5,632 | 4,855,298 |
| Maze | Convolution | Without | 740,736 | 39,312 | 783,504 |

**EqR.** The latent state consists of 2 tensors, $\boldsymbol{s}_t^{(H)}, \boldsymbol{s}_t^{(L)} \in \mathbb{R}^{(n+p)\times w}$, where $n$ is the number of grid cells and $w$ is the hidden width. Here, $p = 16$ with the puzzle prefix and $p = 0$ without it. We identify the pair $(\boldsymbol{s}_t^{(H)}, \boldsymbol{s}_t^{(L)})$ with the flattened state $\boldsymbol{s}_t$ in the main text, so $d = 2(n+p)w$. The hidden width is $w = 128$ for Maze-Unique and $w = 512$ for Sudoku-Extreme.

Let $\boldsymbol{E}_{\boldsymbol{x}}$ denote the input embedding. With the puzzle prefix, one learned vector shared across problems and 15 zero vectors are prepended to the grid embeddings. The recurrent block $f_{\boldsymbol{x}}$ transforms both latent tensors, including their prefix positions. Removing the prefix also reduces the input, output, and intermediate dimensions of the token-mixing MLPs on Sudoku-Extreme.

Let $B_\theta$ denote the shared network. It contains 1 transformer block (Vaswani et al., 2017) for Maze-Unique and 2 MLP-Mixer blocks (Tolstikhin et al., 2021) for Sudoku-Extreme. An internal update of a state $h$ with injected input $a$ is

$$F_\theta(h, a) = (1-\lambda)h + \lambda B_\theta(h + a), \qquad \lambda = 0.95. \tag{16}$$

The recurrent block first updates $\boldsymbol{s}^{(L)}$ $J$ times while holding $\boldsymbol{s}^{(H)}$ fixed, and then updates $\boldsymbol{s}^{(H)}$ once:

$$\begin{aligned}
\boldsymbol{u}^{(0)} &= \boldsymbol{s}_t^{(L)}, \\
\boldsymbol{u}^{(j)} &= F_\theta\left(\boldsymbol{u}^{(j-1)}, \boldsymbol{s}_t^{(H)} + \boldsymbol{E}_{\boldsymbol{x}}\right), \qquad j = 1, \ldots, J, \\
\boldsymbol{s}_{t+1}^{(L)} &= \boldsymbol{u}^{(J)}, \\
\boldsymbol{s}_{t+1}^{(H)} &= F_\theta\left(\boldsymbol{s}_t^{(H)}, \boldsymbol{s}_{t+1}^{(L)}\right), \\
f_{\boldsymbol{x}}(\boldsymbol{s}_t) &= \left(\boldsymbol{s}_{t+1}^{(H)}, \boldsymbol{s}_{t+1}^{(L)}\right).
\end{aligned} \tag{17}$$

These $J + 1$ internal updates together constitute one application of $f_{\boldsymbol{x}}$ and increase recurrent depth by one. We use $J = 4$ for Maze-Unique and $J = 6$ for Sudoku-Extreme, corresponding to 5 and 14 block applications, respectively. The parameters of $B_\theta$ are shared across all internal updates and recurrent depths. Adding an independent noise sample after each internal update gives the stochastic map $\tilde{f}_{\boldsymbol{x}}(\boldsymbol{s}_t, \epsilon_t)$ described in Appendix A.1.

The prediction head reads the grid positions of $\boldsymbol{s}_t^{(H)}$. Let $\boldsymbol{s}_{t,\mathrm{grid}}^{(H)} \in \mathbb{R}^{n\times w}$ denote positions $p+1, \ldots, p+n$ of $\boldsymbol{s}_t^{(H)}$. The prediction head is

$$g(\boldsymbol{s}_t) = \boldsymbol{s}_{t,\mathrm{grid}}^{(H)} \boldsymbol{W}_g^\top \in \mathbb{R}^{n\times C}, \qquad \boldsymbol{W}_g \in \mathbb{R}^{C\times w}. \tag{18}$$

The same linear map is applied at every grid position, with no bias term. The halting head reads the first position of $\boldsymbol{s}_t^{(H)}$ and uses a weight matrix and a bias to produce 2 outputs. The first output is used as the halting logit.

The prediction head $g$ defined in equation 18 is linear, so

$$\mathbb{E}_{\boldsymbol{s}\sim\hat{\mu}_T}[g(\boldsymbol{s})] = g(\mathbb{E}_{\boldsymbol{s}\sim\hat{\mu}_T}[\boldsymbol{s}])\,. \tag{19}$$

The halting head also satisfies this identity. For computational efficiency, we first compute the mean of $\boldsymbol{s}^{(H)}$ over the states in the empirical measure and then apply each head once. The prediction head uses the grid positions of this mean, and the halting head uses its first position.

**DT.** The architecture of DT uses convolutional residual blocks (He et al., 2016). As in the main text, the input consists of discrete values at grid cells. For the CNN, these discrete values are converted into pixel values to form an input image. To make the convolutional operations explicit, we write this image as $\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$ here. The latent state is represented as a tensor $\boldsymbol{s}_t \in \mathbb{R}^{128\times H\times W}$. Flattening this tensor gives the state in the main text, with $d = 128HW$. A convolution followed by a rectified linear unit (ReLU) activation initializes the latent state from the input. The recurrent block then updates this state using input concatenation, another convolution, and 2 residual blocks:

$$\begin{aligned} \boldsymbol{s}_0 &= \mathrm{ReLU}(C_{\mathrm{in}}(\boldsymbol{x}))\,, \\ f_{\boldsymbol{x}}(\boldsymbol{s}_t) &= B_2(B_1(C_{\mathrm{rec}}([\boldsymbol{s}_t, \boldsymbol{x}])))\,, \end{aligned} \tag{20}$$

where $[\boldsymbol{s}_t, \boldsymbol{x}]$ denotes concatenation along the channel dimension. The convolution $C_{\mathrm{in}}$ is used only for initialization, whereas $C_{\mathrm{rec}}$ is applied at every recurrent depth. Each residual block has the form

$$B_j(\boldsymbol{u}) = \mathrm{ReLU}(\boldsymbol{u} + C_{j,2}(\mathrm{ReLU}(C_{j,1}(\boldsymbol{u}))))\,, \qquad j \in \{1, 2\}.$$

The 2 residual blocks have distinct parameters. One application of $f_{\boldsymbol{x}}$ comprises $C_{\mathrm{rec}}$ and both residual blocks, for a total of 5 convolutions, and increases recurrent depth by one. The entire recurrent block shares its parameters across depths.

In tensor form, the prediction head is

$$g(\boldsymbol{s}_t) = C_{g,3}(\mathrm{ReLU}(C_{g,2}(\mathrm{ReLU}(C_{g,1}(\boldsymbol{s}_t))))) \in \mathbb{R}^{2\times H\times W}. \tag{21}$$

The convolutions $C_{g,1}$, $C_{g,2}$, and $C_{g,3}$ have channel dimensions $128 \to 32$, $32 \to 8$, and $8 \to 2$, respectively. The 2 output channels contain logits for whether each pixel belongs to the solution path. Arranging the pixels as rows and the output channels as columns gives the logit matrix in the main text, with $n = HW$ and $C = 2$ for this image representation. Since this head is nonlinear, ImR applies it to each state before computing the empirical expectation of its outputs.

All convolutions use $3 \times 3$ kernels, stride 1, padding 1, and no bias. They preserve the spatial resolution, so the same parameters can process all evaluated maze sizes.

### B.3 Training and Evaluation Settings

We largely followed the training and evaluation settings of EqR (Huang et al., 2026) and DT (Bansal et al., 2022).

To reduce the influence of the initial transient, we can exclude the first $B$ states and construct the empirical measure

$$\hat{\mu}_{T,B} = \frac{1}{T-B}\sum_{t=B+1}^{T} \delta_{\boldsymbol{s}_t}, \qquad 0 \le B < T. \tag{22}$$

The corresponding prediction uses the expectation $\mathbb{E}_{\boldsymbol{s}\sim\hat{\mu}_{T,B}}[g(\boldsymbol{s})]$.

Fine-tuning used trajectories of $T = 384$ recurrent updates with both architectures. The baseline predicted from $\boldsymbol{s}_{384}$, and ImR used $\hat{\mu}_{384,84}$, which contains the states at depths 85–384. In the main evaluations, ImR used $B = 0$ and therefore included all states from depth 1 through the stopping depth $T$. The evaluation depths are given in Sections 5.1 and 5.2. The validation settings are described below.

Tables 5 and 6 summarize the hyperparameters for fine-tuning and training from scratch, respectively. For training from scratch, hyperparameters were selected separately for each configuration using validation accuracy. For EqR trained from scratch on Sudoku-Extreme without the puzzle prefix, $(L, K) = (3, 1)$ was selected based on validation accuracy in a hyperparameter search that

also included $(L, K) = (36, 4)$. The optimizer-update counts in both tables give the full training budgets.

For all fine-tuning experiments, checkpoint selection included the initial prediction head before any fine-tuning updates. Based on validation metrics, fine-tuned heads were selected for all ImR (FT) experiments and for EqR (FT) on Maze-Unique, whereas the initial head was retained for EqR (FT) on Sudoku-Extreme.

Table 5: Hyperparameters for fine-tuning the prediction head.

| Architecture | Dataset | Batch size | Optimizer updates |
|---|---|---|---|
| EqR | Maze-Unique | 768 | 1,000 |
| EqR | Sudoku-Extreme | 768 | 1,000 |
| DT | Maze | 50 | 1,600 |

Table 6: Hyperparameters selected for training from scratch with the architecture of EqR. Here, $L$ is the number of recurrent updates per inner loop, and gradients through the recurrent block are retained for the final $K$ updates.

| Model | Dataset | Puzzle prefix | $L$ | $K$ | Batch size | Optimizer updates |
|---|---|---|---|---|---|---|
| ImR | Maze-Unique | With / Without | 36 | 4 | 768 | 150,000 |
| ImR | Sudoku-Extreme | With / Without | 36 | 4 | 768 | 50,000 |
| EqR | Maze-Unique | Without | 36 | 4 | 768 | 150,000 |
| EqR | Sudoku-Extreme | Without | 3 | 1 | 768 | 50,000 |

**EqR initialization and noise.** When a new example was loaded, the entries of $\boldsymbol{s}^{(H)}$ and $\boldsymbol{s}^{(L)}$ were sampled independently from a standard normal distribution truncated to $[-2, 2]$ and rescaled so that the resulting distribution had standard deviation 1.

During training and validation, each internal update $F_\theta$ added an independent Gaussian noise sample with standard deviation 0.01. The noise was added after the weighted combination in equation 16, including every update of $\boldsymbol{s}^{(L)}$ and $\boldsymbol{s}^{(H)}$. For the main evaluations, we set this noise to zero.

**EqR training from scratch.** Training followed the deep supervision procedure in Section 3.3. We used AdamATan2 (Everett et al., 2024) with $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 1.0 for all parameters except the shared prefix vector. The learning rate increased linearly to $10^{-4}$ over the first 2,000 optimizer updates and remained constant thereafter. The shared prefix vector was updated separately using signSGD (Bernstein et al., 2018) with learning rate $10^{-4}$ and weight decay 1.0.

Following EqR, we maintained an exponential moving average (EMA) (Song & Ermon, 2020), with coefficient 0.999, of the parameters updated by AdamATan2. We used these EMA parameters for validation and evaluation. The prefix vector was used without EMA.

We used EqR's original task loss, averaged over valid tokens within each example. We added the binary cross-entropy loss for the halting prediction with weight 0.5 and averaged the combined loss over the batch. The binary target for halting indicated whether the current prediction matched the entire target grid.

Each example was processed for at most 16 outer steps. The outer loop stopped for an example when its halting logit was positive, except when the following rule required computation to continue. At each outer step, with probability 0.1, we sampled a minimum outer-step count uniformly from 2 through 16. For that step, we ignored a positive halting logit if the current count was below the sampled minimum. Completed examples were replaced by new examples with newly sampled latent states.

Validation used 16 outer steps for each example. Predictions were computed from the final outer step: ImR used the expectation under the empirical measure of all $L$ states in its inner loop, and EqR predicted from the final state of that loop. For evaluation, we used the model weights from the point in training with the highest accuracy on the validation set.

**EqR fine-tuning.** For EqR with the puzzle prefix, we used the released weights for evaluation and to initialize fine-tuning. We initialized a new AdamATan2 optimizer for the prediction head, with learning rate $10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.95)$, and weight decay 1.0. The learning rate remained constant throughout fine-tuning. Each optimizer update used a new batch and newly sampled latent initial states.

Validation used trajectories of 384 recurrent updates. EqR (FT) predicted from $\boldsymbol{s}_{384}$, and ImR (FT) used the expectation under $\hat{\mu}_{384,0}$, which contains all states at depths 1–384. We used accuracy on the validation set to choose how many fine-tuning updates to use for the prediction head. For Sudoku-Extreme, EqR fine-tuning did not improve validation accuracy, so the original prediction head was retained. We therefore do not show a separate fine-tuned EqR curve in Figure 2.

Fine-tuned prediction heads were used without EMA. All other model parameters remained at their released values.

**DT fine-tuning.** Both fine-tuning methods started from the same released model weights and the corresponding Adam optimizer state. We used Adam (Kingma & Ba, 2015) with a constant learning rate of $10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay $2 \times 10^{-4}$. The task loss was cross-entropy computed only on traversable pixels.

Validation ran to recurrent depth 384 for both methods. DT (FT) used the logits $g(\boldsymbol{s}_{384})$. For ImR (FT), the validation logits were

$$\mathbb{E}_{\boldsymbol{s}\sim\hat{\mu}_{384,0}}[g(\boldsymbol{s})] = \frac{1}{384}\sum_{t=1}^{384} g(\boldsymbol{s}_t).$$

The prediction head was applied to each state before computing this expectation, and the initial state $\boldsymbol{s}_0$ was not included. For evaluation, we used the prediction-head weights from the point in fine-tuning with the lowest loss on the ID validation set.

### B.4 Classification of Latent Trajectories

We classified the latent trajectories reported in Table 1 using changes in $\boldsymbol{s}^{(H)}_{t,\mathrm{grid}}$, the grid positions of $\boldsymbol{s}^{(H)}_t$ used by the prediction head, as described in Appendix B.2. After each recurrent update, we computed the relative $L_\infty$ residual

$$r_t = \frac{\|\boldsymbol{s}^{(H)}_{t+1,\mathrm{grid}} - \boldsymbol{s}^{(H)}_{t,\mathrm{grid}}\|_\infty}{\|\boldsymbol{s}^{(H)}_{t+1,\mathrm{grid}}\|_\infty + 10^{-8}}. \tag{23}$$

Here, $\|\cdot\|_\infty$ denotes the largest absolute entry across all grid positions and latent dimensions.

We evaluated trajectories to depth $T = 512$ on Maze-Unique and $T = 1024$ on Sudoku-Extreme. A trajectory was classified as fixed if

$$\max_{t=T-64,\ldots,T-1} r_t < 0.05, \tag{24}$$

and as non-fixed otherwise. Thus, the fixed label requires small relative changes over all of the final 64 recurrent updates. This diagnostic does not guarantee convergence of the full latent state to a fixed point.

We computed the proportion of non-fixed trajectories among all evaluated examples. For prediction stability, we applied the categories defined in Section 5 over depths $T-63, \ldots, T$. The proportion of each prediction category was computed separately within the fixed and non-fixed groups, with the number of examples in that group as the denominator.

For each configuration trained or fine-tuned in this study, we computed the proportions separately for each of the 3 training seeds and reported their mean and standard deviation. For EqR with the puzzle

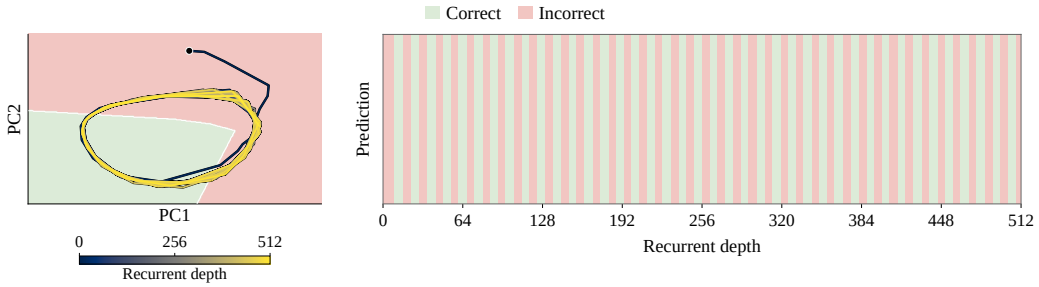



Figure 5: The EqR trajectory in Figure 1(a) and its actual predictions. Left: the trajectory over depths 0–512 and prediction regions on the reconstructed PCA plane. Trajectory color indicates recurrent depth; the circle marks the initial state. Right: whole-maze correctness from the original head applied to each actual latent state. Pale green and pale red indicate correct and incorrect predictions, respectively.

prefix, we reported results from the single released model for each dataset. Fine-tuning updates only the prediction head, so the proportions of fixed and non-fixed trajectories remain unchanged.

### B.5 Visualization of Latent Trajectories and Predictions

For Figure 1(a), we used the publicly released EqR model for Maze-Unique with its EMA parameters, original prediction head, and puzzle prefix. For one test maze, we ran 512 recurrent updates from the model's standard random initialization with zero transition noise. Figure 5 reproduces this trajectory alongside the predictions from its actual latent states.

We fitted PCA to the same 513 states shown in the figure, including the initial state at depth 0. Let $\boldsymbol{u}_t = \mathrm{vec}(\boldsymbol{s}^{(H)}_{t,\mathrm{grid}})$ denote the flattened high-level state excluding the puzzle prefix. With mean $\bar{\boldsymbol{u}}$ and the first two principal directions $\boldsymbol{V}$, the displayed trajectory is

$$\boldsymbol{q}_t = \boldsymbol{V}^\top(\boldsymbol{u}_t - \bar{\boldsymbol{u}}), \qquad t = 0, \ldots, 512.$$

These components explain 77.0% of the variance. The trajectory was generated by the full recurrent model; PCA was applied only for visualization.

For the background, we evaluated the original prediction head on a $400 \times 260$ grid of reconstructed states

$$\boldsymbol{u}(\boldsymbol{q}) = \bar{\boldsymbol{u}} + \boldsymbol{V}\boldsymbol{q},$$

with all omitted PCA components fixed at their mean. A point is correct only if all 900 predicted maze cells match the target. The white line separates the correct and incorrect regions. This boundary is defined on a two-dimensional slice and only approximately represents the prediction regions visited by the full trajectory: omitted components can change the prediction at the same projected location.

The right panel of Figure 5 reports the original head's predictions at the actual latent states, without PCA reconstruction. These predictions repeatedly alternate between correct and incorrect. Thus, the original high-dimensional trajectory also repeatedly visits both prediction regions, supporting the qualitative interpretation of Figure 1(a). The precise geometry of the high-dimensional boundaries, however, cannot be inferred from the two-dimensional visualization alone.

## C Additional Results

This appendix presents accuracy at the final recurrent depth, PCA projections of latent trajectories, and DT results by maze size.

### C.1 Accuracy at the Final Recurrent Depth

Table 7 reports accuracy at the final recurrent depths used in the EqR and DT experiments. For DT, results are grouped by the degree of the start cell, as in Section 5.2.

For both prefix settings, ImR trained from scratch achieves higher accuracy than EqR on both datasets. Fine-tuning with ImR yields higher accuracy than the fine-tuned baseline on Maze-Unique and both groups of DT mazes, but lower accuracy on Sudoku-Extreme.

Table 7: Accuracy at the final recurrent depth. Entries with $\pm$ report the mean and standard deviation across 3 training seeds. The remaining entries use a single released model.

| Model | Puzzle prefix | Accuracy (%) |
|---|---|---|
| *Maze-Unique*, $T = 512$ | | |
| EqR | With | 89.4 |
| EqR (FT) | With | $89.4 \pm 0.0$ |
| ImR (FT) | With | $91.6 \pm 0.1$ |
| ImR | With | $98.8 \pm 0.9$ |
| EqR | Without | $82.9 \pm 20.9$ |
| ImR | Without | $99.0 \pm 1.4$ |
| *Sudoku-Extreme*, $T = 1024$ | | |
| EqR | With | 96.2 |
| EqR (FT) | With | $96.2 \pm 0.0$ |
| ImR (FT) | With | $95.4 \pm 0.0$ |
| ImR | With | $98.7 \pm 0.2$ |
| EqR | Without | $96.2 \pm 0.3$ |
| ImR | Without | $99.0 \pm 0.2$ |
| *OOD mazes, degree of the start cell: 2*, $T = 5{,}000$ | | |
| DT | – | 77.75 |
| DT (FT) | – | $82.83 \pm 0.14$ |
| ImR (FT) | – | $84.50 \pm 0.00$ |
| *OOD mazes, degree of the start cell: 3*, $T = 5{,}000$ | | |
| DT | – | 25.26 |
| DT (FT) | – | $26.32 \pm 0.00$ |
| ImR (FT) | – | $28.42 \pm 0.00$ |

### C.2 Projections of Latent Trajectories

Figure 6 shows PCA projections of $\boldsymbol{s}_{t,\mathrm{grid}}^{(H)}$, the part of the latent state used by the prediction head. For these visualizations, we display depths 897–1024 for both Maze-Unique and Sudoku-Extreme. The trajectories show continued state changes over this interval with noise set to zero.

### C.3 DT Results by Maze Size

Figure 7 compares accuracy across recurrent depth separately for each maze size. The enlarged views illustrate the reduction in accuracy fluctuations at large recurrent depths with ImR (FT).

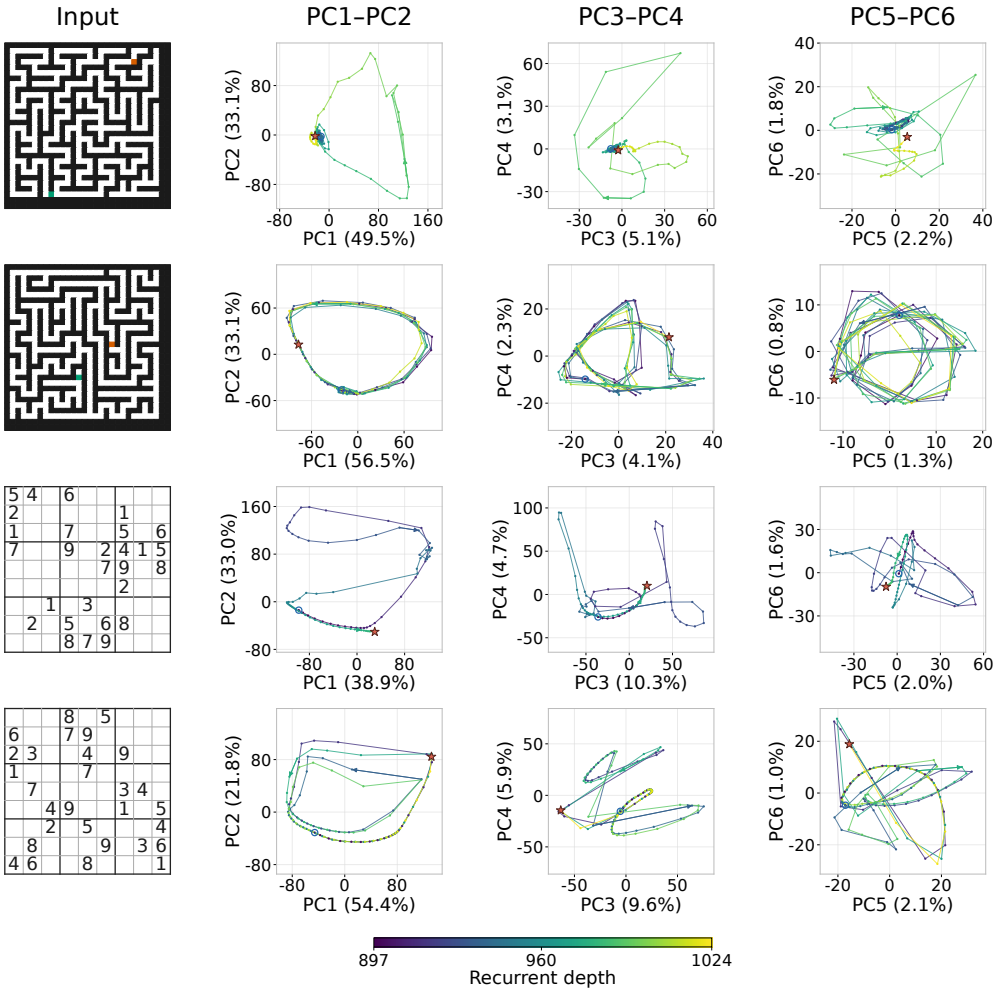



Figure 6: PCA projections of the part of the latent state used by the prediction head for ImR models trained without the puzzle prefix. Each row shows an input and the corresponding latent trajectory projected onto PC1–PC2, PC3–PC4, and PC5–PC6. The first two rows show Maze-Unique examples, and the last two show Sudoku-Extreme examples. Axis labels give the percentage of variance explained by each principal component. Colors indicate recurrent depths 897–1024; circles and stars mark the first and last states in this interval, respectively. All trajectories were generated with noise set to zero.

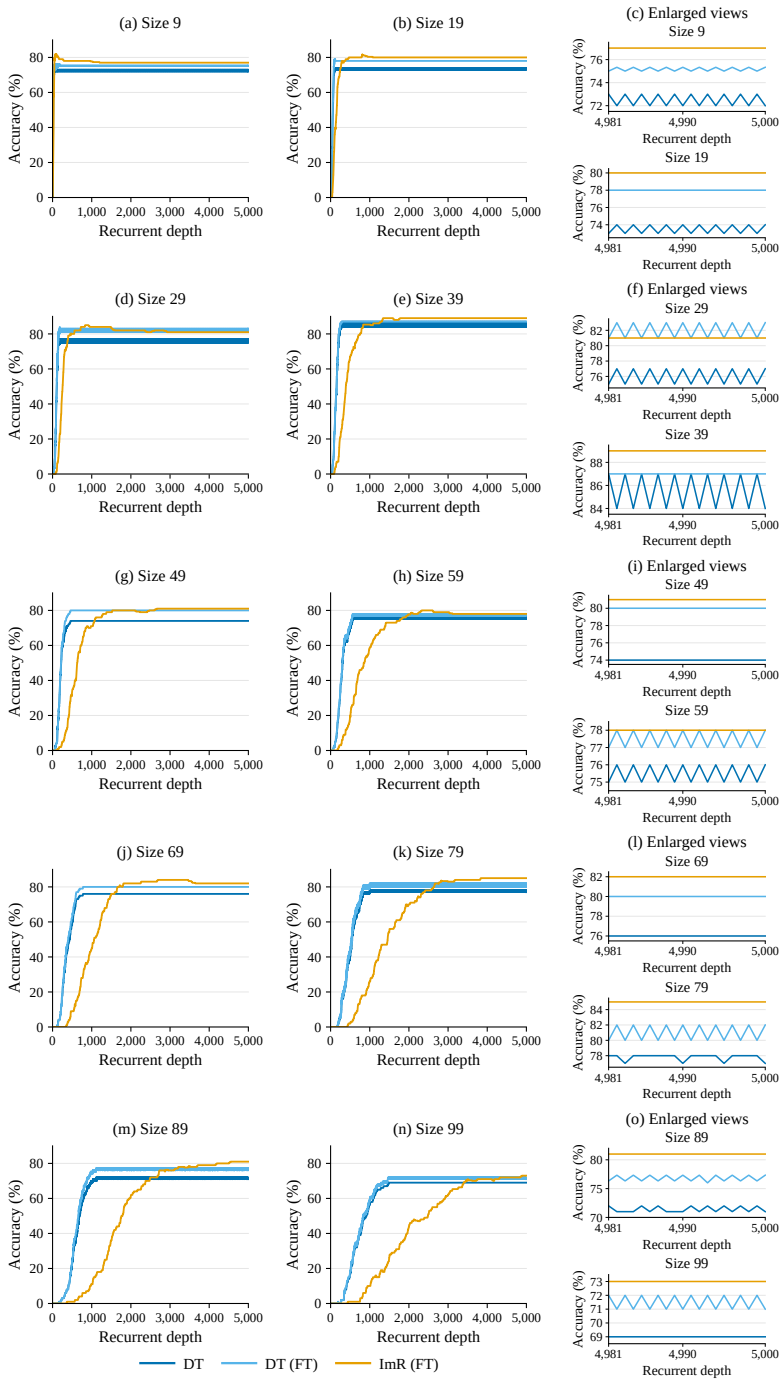



Figure 7: Accuracy across recurrent depth for each maze size. The first two columns show sizes $9 \times 9$ through $99 \times 99$, with 100 test mazes per size, combining all degrees of the start cell. The right column shows enlarged views over depths $4{,}981$–$5{,}000$. Curves for DT (FT) and ImR (FT) report means across 3 training seeds.