Fixed-Point Reasoning: Stable and Adaptive Deep Looped Models

Anonymous Authors[1]

Abstract

Looped-in-depth architectures provide an inductive bias toward learning step-by-step procedures for tasks that require compositional reasoning. The effective depth reached by looping determines the quality of the solution. Similar to deep architectures, looped architectures are prone to signal propagation issues as the halting decision is postponed. In this paper, we address these signal propagation issues by using pre-norm layers and residual scaling. Furthermore, we propose ***FPRM***: a Fixed-Point Reasoning Model that uses fixed-point convergence as an end-to-end halting mechanism in a looped architecture. We show that fixed-point halting allows FPRM to adapt its compute to the difficulty of the task. FPRM proves effective on common reasoning benchmarks, namely sudoku, maze, and state tracking.

1. Introduction

Reasoning in neural networks has increasingly been framed as a problem of scaling test-time compute: a model should be able to spend more computation on inputs it finds harder [25, 29]. However, doing so requires two ingredients: **(1) flexibility:** a mechanism for spending a variable amount of compute on the problem, and **(2) adaptivity:** a way to decide how much compute to spend on the problem. One way to achieve this is through Chain-of-Thought (CoT) mechanism [33]. In CoT, the model scales compute through verbalization, and makes halting decisions based on predicting a specialized halting token. However, this emerging behavior requires a special training regime [15].

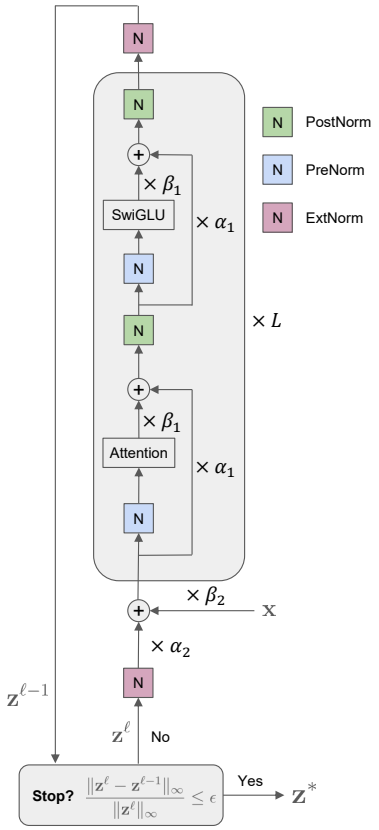


Figure 1. FPRM: our fixed-point looped transformer with better signal propagation due to pre-norm and residual scalings.

An alternative approach towards an end-to-end training method for reasoning models is emerging in the form of looped architectures [7, 4, 28]. Whereas in CoT, the compute scales along the sequence dimension, in looped architectures it scales along the depth dimension:

$$\mathbf{z}^{\ell+1} = f_\theta(\mathbf{z}^\ell; \mathbf{x}), \tag{1}$$

where $f_\theta(\mathbf{z}; \mathbf{x})$ is a neural network. Since, in looped models, compute scales independently of parameter count, it can be increased naturally, satisfying the first requirement. The halting decision, however, is no longer trivial. Most approaches either fix or randomly sample the number of loops [12], which eliminates adaptivity, or use external modules trained to make halting decisions [32, 16]. The latter is associated

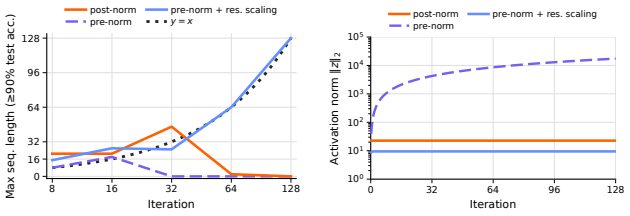
[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

with adaptive computation time (ACT) modules [14], which introduce non-differentiable halting objectives that can only be trained via continuous relaxation.

One of the recent examples of this paradigm are Hierarchical Reasoning Model (HRM) [32] and Tiny Reasoning Model (TRM) [16], which achieve strong performance on reasoning benchmarks such as Sudoku, maze, and ARC-AGI [6]. Using small networks with bounded iteration counts, these models have been able to outperform prominent LLM-based reasoning models. At the core of these methods is the idea of the need for a hierarchical looping mechanism, wherein the compute is distributed between a fast-looping component (the L-level) and a slow-looping component (the H-level). However, paired with the ACT mechanism, they ultimately fail on the adaptivity test as the compute during test-time is fixed and pre-determined.

Alternative approaches such as Deep Equilibrium Models [2] and energy-based reasoning models [9] provide a more natural criterion for halting, namely either reaching a fixed-point or a local minimum in an energy landscape. However unrolling these models can reveal that they are akin to a deep, weight-tied neural network, which is accompanied by a plethora of issues such as vanishing and exploding gradients [5, 26], rank-collapse [8], and signal propagation problems [24]. As such, these models have been proven to be difficult to optimize [1, 9, 10]. Other works have demonstrated that looped architectures can also be particularly prone to overfitting [4], making it clear that despite the demonstrable success, looped architectures also inherit the complications associated with training deep neural networks.



(a) Trainability at different scales

(b) Activation norm at initialization

Figure 2. **The blessing and the curse of depth in Universal Transformers. (a)** Trainability of Universal Transformers with post-norm and pre-norm on A_5, **(b)** the norm of the final activations in Universal Transformers at initialization as a function of depth. While the pre-norm model without residual scaling (see Section 2.1) diverges in activation norm, only the pre-norm model with residual scaling can fully utilize the effective depth. Thus, a large effective depth, if achieved when needed, unlocks expressivity – the blessing, but at the same time poses a challenge to stabilize (in a case of pre-norm) or utilize the signal (in a case of post-norm) – the curse.

Motivated by these challenges, the **core insight** of this paper is the following: The ability to reason in a latent space with long traces, requires effective signal propagation through the network, which is confounded with the fact that looped architectures are notoriously difficult to optimize [12, 27, 19]. Addressing the signal propagation issues in these networks is essential for enabling long-horizon reasoning. To this end, we adapt architectural techniques originally developed to optimize traditional transformers at large effective depths, carefully modifying them for the looped transformer setting. Our philosophy is that *looped transformers are, in part, very deep transformers*.

One of the striking differences between looped and fixed-depth transformers is the common use of a post-norm in looped transformers [7, 12, 32, 16], which is considered to cause unstable training in deep non-looped architectures compared to prenorm [34, 24]. However, in a looped model, post-norm blocks ensure that the magnitude of the activations stay bounded, which looped architectures rely on to ensure the hidden states do not diverge as the model iterates [19]. This begs the question: *can we switch the post-norm to pre-norm, while ensuring the activations stay bounded in a different way?* If so, this could make the training of looped models stable at large depths.

We give an affirmative answer to this question; we do so by introducing scaling parameters on the skip connections and residuals. With an additional set of modifications, we are able to train an adaptive reasoning model that loops until it converges to a fixed point. For literature review, please refer to Section A. We summarize our contributions as follows:

1. We modify a transformer block such that it is trainable over larger depths by switching post-norm to pre-norm and adding residual scaling parameters. To the best of our knowledge, we are the first to train a looped transformer with pre-norm.

2. To further enable stable training, we propose two novel and theoretically grounded modifications that are specific to the fixed-point halting mechanism: First, an algorithm to ensure that the model halts and does not get stuck in a limit cycle. Second, we show that unrolling the backward pass k times at the fixed-point enables an estimation of the gradients that is exponentially accurate with k.

3. Combining these changes, we propose FPRM: Fixed-Point Reasoning Model. We show that we outperform TRM and HRM on Sudoku, Maze and the state-tracking $A5$ benchmark. To the best of our knowledge, we are the first to show out-of-distribution generalization on state tracking with a transformer-based architecture. Notably, we achieve our results without the hierarchical structure of HRM and TRM.

2. Method

As looped architectures need bounded activations across iterations, current methods mostly employ post-norm to satisfy the boundedness condition [16]. However, post-norm introduces serious signal propagation issues, often associated with unstable training and low effective depth [8, 24, 30]. Consequently, we resolve this issue by decoupling the two requirements: we switch to pre-norm to recover trainability at depth, and recover boundedness through a small set of additional architectural changes which together yield a block that is both stable under looping and trainable at large depth. On top of this block, we introduce a fixed-point halting mechanism that decides the iteration count adaptively per input.

2.1. Improving signal propagation with pre-norm

We start by introducing pre-norm and post-norm in our architecture. A transformer block consists of two residual sub-layers — multi-head attention (MHA) and a feed-forward network (FFN) — interleaved with layer normalization (LN). Two canonical placements of the normalization define two variants of the block. The original *post-norm* formulation [31] applies LN after the residual addition,

$$\mathbf{h} = \mathrm{LN}(\mathbf{x} + \mathrm{MHA}(\mathbf{x})), \quad \mathbf{y} = \mathrm{LN}(\mathbf{h} + \mathrm{FFN}(\mathbf{h})),$$

while the *pre-norm* variant [34] applies LN to the input of each sub-layer and leaves the residual stream untouched,

$$\mathbf{h} = \mathbf{x} + \mathrm{MHA}(\mathrm{LN}(\mathbf{x})), \quad \mathbf{y} = \mathbf{h} + \mathrm{FFN}(\mathrm{LN}(\mathbf{h})).$$

The two are not interchangeable. Post-norm bounds activation magnitudes at every layer, but attenuates the residual signal with depth and is well known to be unstable to train in deep networks [24]. Pre-norm preserves a clean residual path, yielding well-behaved gradient flow at depth, at the cost of activations that can grow without bound.

In Figure 2a, we observe that increasing the iteration count of a post-norm Universal Transformer [7] does not translate into improved accuracy, while a pre-norm variant continues to improve with depth. Motivated by this, our first change is to switch to pre-norm in our reasoning architecture.

2.2. Recovering boundedness via residual scaling

While pre-norm mitigates trainability problems in looped architectures, it removes the necessary boundedness condition that motivated the use of post-norm in them. This effect can be observed in Figure 2b, where the activations of the pre-norm model grow with more iterations. Consequently, we propose to restore boundedness by introducing scaling parameters applied at two different scales: one within each layer of the network, and one over the looping.

Layer-wise residual scaling. Within a single application of $f_\theta(\mathbf{z}, \mathbf{x})$, the residual stream and sub-layer output $f^\ell_{\theta_\ell}(\mathbf{z}^{\ell-1})$ (an MHA or FFN sub-block with layer normalization, as in Section 2.1) are weighted by tied scalars (α_1, β_1) shared across all L layers:

$$\mathbf{z}^\ell = \alpha_1 \, \mathbf{z}^{\ell-1} + \beta_1 \, f^\ell_{\theta_\ell}\big(\mathbf{z}^{\ell-1}\big), \qquad \ell = 1, \ldots, L. \tag{2}$$

Iteration-wise input mixing. Between consecutive applications of $f_\theta(\mathbf{z}, \mathbf{x})$, we re-inject the input $\mathbf{x}$ with tied scalars (α_2, β_2) shared across all iterations:

$$\mathbf{z}^0_{i+1} = \alpha_2 \, \mathbf{z}^L_i + \beta_2 \, \mathbf{x}. \tag{3}$$

The two scaling schemes are not independent. With an appropriate coupling between them, the resulting recurrence is bounded for any input, resulting in a stable looping. In the following statement, we formalize this claim:

Theorem 1 (Boundedness of FPRM iterates)**.** *Consider the model defined by Equations 2 and 3, and assume each layer map satisfies $\|f^\ell_{\theta_\ell}(\mathbf{u})\| \le c_f$ for all ℓ and $\mathbf{u}$. Let $0 \le \alpha_1, \alpha_2 < 1$, and set*

$$\beta_2 = 1 - \alpha_2 \alpha_1^L, \qquad \beta_1 = \frac{\beta_2 \, (1 - \alpha_1)}{1 - \alpha_1^L}.$$

Then the fixed-point iterates $\{\mathbf{z}^0_i\}_{i \ge 0}$ are bounded, and if $\mathbf{z}^0_i \to \mathbf{z}^0_\infty$, then

$$\|\mathbf{z}^0_\infty\| \le \|\mathbf{x}\| + \alpha_2 \, c_f.$$

The proof is in Section B.1. Note that the boundedness condition of the sequence model in Theorem 1 is satisfied when using pre-norm [18]. However, boundedness still does not guarantee the looping to converge to a fixed-point, which is a condition we need to satisfy the adaptivity ingredient. In the following, we show that there exists a choice of α_2 that satisfies this requirement. Consequently, as empirically shown by [4], the model may become locally contractive during training.

Theorem 2 (Small α_2 implies convergence)**.** *Let $\tilde{f}_\theta(\mathbf{u})$ denote one full pass through the L layers defined by Equation 2, and define the iteration map*

$$T_{\alpha_2}(\mathbf{z}; \mathbf{x}) \;:=\; \tilde{f}_\theta(\alpha_2 \, \mathbf{z} + \beta_2 \, \mathbf{x}).$$

If $\tilde{f}_\theta$ is Lipschitz with constant λ_f and $\alpha_2 \lambda_f < 1$, then $T_{\alpha_2}(\,\cdot\,; \mathbf{x})$ is a contraction, and the fixed-point iteration $\mathbf{z}_{i+1} = T_{\alpha_2}(\mathbf{z}_i; \mathbf{x})$ converges to a unique $\mathbf{z}^\star$ at a linear rate:

$$\|T_{\alpha_2}(\mathbf{z}_i; \mathbf{x}) - \mathbf{z}_i\| \le (\alpha_2 \lambda_f)^i \, \|T_{\alpha_2}(\mathbf{z}_0; \mathbf{x}) - \mathbf{z}_0\|.$$

For $\mathbf{z}_0 = 0$, the right-hand side specialises to $(\alpha_2 \lambda_f)^i \, \big\|\tilde{f}_\theta(\beta_2 \mathbf{x})\big\|$. The proof is in Section B.2

Table 1. Sensitivity of FPRM to residual scaling initialization on Sudoku Extreme dataset. Each cell reports best test sequence accuracy (%) for a given pair of initial values.

	α_2 init		
α_1 init	0.25	0.50	0.75
0.25	67.04	67.41	67.72
0.50	73.55	70.99	70.88
0.75	**74.91**	72.51	71.60

As pointed out by previous works [2, 1] such an approach would severely limit the expressivity of the model, and is thus undesirable. We avoid this by making α_1 and α_2 learnable. In practice, we find that initializing the network to be more contractive by setting α_2 to be small yields better performance (Table 1). Intriguingly, this observation is also in line with the common solutions to signal-propagation and rank-collapse problems in deep neural networks [24, 30], suggesting a connection between rank-collapse and signal propagation problems in looped architectures. Together, these modifications yield a pre-norm looped transformer that maintains performance over longer looping horizons before saturating, compared to the post-norm variant (see Figure 2).

2.3. Oscillation around the fixed-point

So far, we have been able to guarantee that, given a small enough α_2, the model introduced in Equation (3) becomes locally contractive and converges to a fixed-point. However, in practice the contraction factor introduced in Theorem 2 is only as small as the training makes it. For some inputs, we observe that the model often descends into an oscillatory behavior, causing the iteration to stay in a small region of latent space without converging. This non-convergent behavior is not in tension with Theorem 2, as the theorem gives a sufficient condition for convergence, not a complete characterisation of the iteration's behaviour. In fact, oscillation around the fixed-point can happen when the Jacobian satisfies certain conditions.

Linearizing the iteration near a fixed point $\mathbf{z}^\star$ gives $\mathbf{z}_{i+1} - \mathbf{z}^\star \approx \mathbf{J}\,(\mathbf{z}_i - \mathbf{z}^\star)$, where $\mathbf{J} = \partial f_\theta / \partial \mathbf{z}\,|_{\mathbf{z}^\star}$. Oscillation around $\mathbf{z}^\star$ arises when $\mathbf{J}$ has an eigenvalue with $\Re(\lambda_i) < 1$ but $|\lambda_i| \geq 1$, in which case the iteration spirals around $\mathbf{z}^\star$ rather than contracting toward it. The half-plane condition $\Re(\lambda_i) < 1$ is exactly what licenses a runtime fix that does not require modifying f_θ:

Theorem 3 (Damping stabilizes oscillatory fixed-point dynamics)**.** *Suppose $f_\theta(\,\cdot\,;\mathbf{x})$ is continuously differentiable in a neighbourhood of a fixed point $\mathbf{z}^\star$, and that every eigenvalue λ_i of the Jacobian $\mathbf{J}$ at $\mathbf{z}^\star$ satisfies $\Re(\lambda_i) < 1$. Define the damped iteration map*

$$g_{\eta,\theta}(\mathbf{z};\mathbf{x}) \;:=\; \eta\, f_\theta(\mathbf{z};\mathbf{x}) + (1-\eta)\,\mathbf{z}.$$

Then there exists $\eta_0 \in (0,1)$ such that, for every $\eta \in (0,\eta_0)$, the iteration $\mathbf{z}_{i+1} = g_{\eta,\theta}(\mathbf{z}_i;\mathbf{x})$ converges locally to $\mathbf{z}^\star$. Moreover, $g_{\eta,\theta}(\,\cdot\,;\mathbf{x})$ and $f_\theta(\,\cdot\,;\mathbf{x})$ have the same fixed points.

The proof is in Section C.1.

Theorem 3 shows that a suitable damping factor η eliminates the oscillations while preserving the fixed points of $f_\theta(\mathbf{z};\mathbf{x})$. To choose η adaptively at inference time, we use a patience mechanism that decreases η whenever the residual stops improving. We track the smallest relative residual observed so far,

$$r^\star \;=\; \min_i \frac{\|\mathbf{z}_i - f_\theta(\mathbf{z}_i;\mathbf{x})\|_\infty}{\|f_\theta(\mathbf{z}_i;\mathbf{x})\|_\infty + \epsilon},$$

and when it fails to improve for P consecutive iterations, we apply a geometric decay $\eta \leftarrow \gamma\,\eta$ with $\gamma \in (0,1)$. The full procedure is given in Algorithm 1. In Section D, we provide further details about the optimization of a fixed-point model.

2.4. Fixed-point reasoning model

An overview of the algorithm is available in Algorithm 2. Since we observed no improvements in our experiments by using the hierarchical structure introduced in [32], we opt for a classic looped architecture [7] instead. The model performs looping in windows of K iterations, which determines the truncated BPTT value. At each forward pass, the fixed-point optimizer introduced in Algorithm 1 is called to dampen the fixed-point iteration steps. Then, we get a prediction from the current state $\mathbf{z}$ of the model and perform a deep supervision step, following [32, 16]. Finally, we detach the state $\mathbf{z}$, and stop when the optimizer detects fixed-points. This happens when the residual falls below the tolerance, or the patience budget is exhausted.

3. Experiments

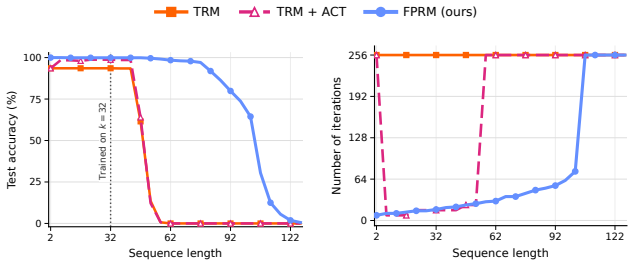
3.1. Results

Sudoku and maze navigation. We first evaluate FPRM on the Sudoku-Extreme and Maze-Hard benchmarks introduced by HRM [32]. Sudoku-Extreme consists of challenging 9×9 Sudoku puzzles in a small-sample learning setting, while Maze-Hard evaluates optimal path finding in hard 30×30 mazes. These benchmarks were designed to test whether latent recurrent reasoning models can solve symbolic search problems from limited supervision. Table 2 shows that FPRM achieves the highest accuracy on both tasks, with the largest improvement on Sudoku-Extreme.

State tracking. We evaluate FPRM on state tracking, where a model must compose a sequence of updates to recover a latent state. This task is a useful proxy for stateful reasoning problems such as entity tracking, code execution, and game-state tracking [20]. It is also a diagnostic for

Table 2. Test accuracy on Sudoku-Extreme and Maze-Hard. FPRM outperforms the baselines across the tasks.

Model	Sudoku-Extreme	Maze-Hard
HRM	55.0%	74.5%
TRM	74.7%	85.3%
FPRM	**81.3%**	**86.4%**



(a) Test accuracy. *(b)* Iterations used.

Figure 3. State-tracking generalization as a function of sequence length on $A5$. Models are trained on sequence length $k = 32$; the region with $k > 32$ marks out-of-distribution evaluation. FPRM maintains high test accuracy substantially beyond the training regime while using far fewer iterations than the fixed-depth TRM baseline over much of the evaluation range.

our setting: prior work shows that longer state-tracking instances require greater effective depth [20, 22], and length generalization requires learning a composable update rule rather than memorizing training-length computations. We train sequences of lenght $k = 32$ and test up to $k = 128$. We compare to TRM [16], which by default disables its ACT head at test time and runs 16 outer (deep supervision) iterations, and to TRM+ACT, which leaves the ACT head enabled for adaptive halting. For fairness, FPRM is capped at 256 iterations, matching TRM's maximum possible number of iterations. As shown in Figure 3, *FPRM maintains high accuracy beyond the training regime, while both TRM variants degrade with length.* Moreover, FPRM uses few iterations on short sequences and increases its iterations smoothly as sequence length grows, showing that it allocates more effective depth to harder inputs. In contrast, TRM's halting mechanism does not show the same length-sensitive behavior, as its iteration count does not increase monotonically with sequence length.

3.2. Analyzing depth-induced signal propagation issues in looped transformers

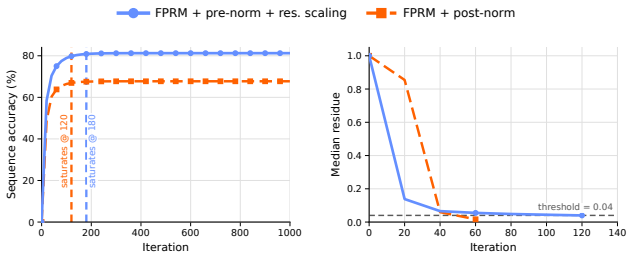
Pre-norm looped models quickly diverge. As introduced in Section 2.1, recurrent-in-depth models often achieve large effective depth, at the cost of more unstable signal propagation. To isolate and analyze this instability, we adopt the Universal Transformer framework with a fixed number of iterations, controlling for the confounding effects of dynamic halting mechanisms of looping. Similarly to very deep networks without weight-sharing, looped models also suffer from instabilities coming from inadequate signal propagation handling, which are observable both at initialization and during training. In Figure 2b we show the norm of the final activations of the residual branch of Universal Transformer at initialization. As expected, pre-norm model diverges due to unbounded activations. Conversely, having post-norm architecture or pre-norm with residual scaling ensures boundedness of the activations. Thus, models with post-norm or pre-norm with residual scaling can achieve much bigger effective depths without diverging compared to pre-norm-only models.

Residual scaling enables trainability at large effective depth; post-norm collapses; pre-norm without scaling explodes. However, the boundedness-at-initialization condition from the previous section is not sufficient to ensure feature learning at large depth. We demonstrate this in Figure 2a using a state-tracking task, where increasing difficulty, corresponding to longer sequences, requires training at greater depths. To evaluate trainability, we plot the maximum sequence length solved with $> 90\%$ accuracy against the number of loops. Only the pre-norm UT variant with residual scaling is capable of solving the same task length the model was trained on, in the regime where the iteration budget is matched to the task length. It is the only architecture whose maximum solvable length grows monotonically with the trained iteration count. Post-norm trains at small sequence lengths but collapses on higher ones. Pre-norm without residual scaling fails immediately at all sequence lengths except the smallest. We interpret Figure 2a as direct evidence that norm-controlled, non-attenuating residual propagation is a necessary condition for a looped transformer to perform sequential composition that requires larger effective depth.

3.3. Loop-utilization

In the previous section, we showed that post-norm and pre-norm transformer architectures suffer from training instability as the model reaches greater effective depth, which translates into an inability to solve tasks that require scaling effective depth. In this section, we switch to a class of models that can adapt their compute budget at inference time. Specifically, we train FPRM with pre-norm and residual scaling, and FPRM with post-norm, both on the Sudoku task until their fixed-point halting mechanisms stop them. In Figure 4a, we demonstrate that test-time compute scaling improves performance for both types of normalization. However, once the fixed point is reached, additional compute no longer improves performance. We further show that the gap between the pre-norm model with residual scaling

and the post-norm model in Figure 4a could arise from underutilization of effective depth in the post-norm FPRM. This manifests as faster performance saturation at a lower value. The results in Figure 4 further support our hypothesis about signal-propagation problems, justifying the modifications introduced in this paper.



(a) Test accuracy *(b)* Fixed-point residues

Figure 4. **Loop-utilization of FPRM on Sudoku. (a)** test accuracy of FPRM with pre-norm with residual scales vs. post-norm. **(b)** median residue. The pre-norm model is better at loop utilization, while both have similar residues. This indicates similar latent-space convergence, with more meaningful updates in the pre-norm variant, resulting in improved performance.

4. Discussion

Looped fixed-point models are adaptive. In contrast to the ACT mechanism of TRM, FPRM adapts to the difficulty of the problem, even outside of the training distribution (Figure 3). To the best of our knowledge, FPRM is the first purely transformer-based architecture that length-generalizes on state-tracking, which is noteworthy given that previous successful attempts to do so were done with architectures with an inductive bias for this task, such as state-space models or RNNs [23, 20].

The role of hierarchy in HRM and TRM. HRM [32] motivates the use of the hierarchy by analogy to the brain, while TRM [16] suggests that the fast module f_L provides a scratch pad to refine the solution that f_H encodes. In light of the fact that by improving signal propagation FPRM improves upon these models in Sudoku, Maze, and state tracking, we hypothesize that there might be a simpler explanation: *the hierarchy facilitates optimization*. We identify the theoretical explanation of the role of hierarchy from the lens of optimization as an interesting direction for future work.

Scaling behavior of the loops. Previous theoretical research found that looped models can express algorithms with better scaling behavior than CoT [22, 36]. For solving A5, CoT would require a superlogarithmic number of iterations [21], while an optimal algorithm could solve it in $O(\log(n))$. This suboptimal scaling has also been observed in recurrent-in-depth state-space models [23]. We propose it as an open challenge to find a latent reasoning architecture that achieves an $O(\log(n))$ solution to A5 while remaining Turing-complete [7].

Limitations. In a similar spirit to previous literature on end-to-end reasoning [37, 11, 32, 16, 9, 10], we test our model only on algorithmic tasks and not on natural language. It is an open challenge to demonstrate that the compositional reasoning behavior that latent models exhibit on algorithmic tasks translates to other domains. In addition, even though the base architecture of FPRM could adopt any model (e.g. CNN, MLP, state-space models), we limit our experiments to transformers.

5. Conclusion

We present architectural modifications for looped fixed-point transformers that enable the use of pre-norm, improving the model's ability to exploit the larger effective depth provided by looping. These modifications allow FPRM to outperform its closest baselines, HRM and TRM, on common symbolic reasoning benchmarks. We show that on state tracking it generalizes beyond its training distribution, which is to our knowledge, the first transformer-based architecture to length-generalize on this task. This capability stems from dynamically scaling depth through fixed-point iterations and improving signal propagation. We hope these architectural modifications and the accompanying insights will support further progress on latent reasoning models.

References

[1] 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 *NeurIPS*, 2022.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *NeurIPS*, pages 688–699, 2019.

[3] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder, 2021.

[4] 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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[5] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.

[6] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report, 2025.

[7] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019.

[8] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 2793–2803. PMLR, 2021.

[9] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization, 2022.

[10] Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning iterative reasoning through energy diffusion, 2024.

[11] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization, 2025.

[12] Jonas Geiping, Sean 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, 2025.

[13] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *NeurIPS*, pages 24247–24260, 2021.

[14] Alex Graves. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983, 2016.

[15] 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, Hao 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, Tao 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. *Nat.*, 645(8081):633–638, 2025.

[16] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025.

[17] Ferdinand Kapl, Emmanouil Angelis, Kaitlin Maile, Johannes von Oswald, and Stefan Bauer. From growing to looping: A unified view of iterative computation in llms, 2026.

[18] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 5562–5571. PMLR, 2021.

[19] Asher Labovich. Stability and generalization in looped transformers, 2026.

[20] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models, 2025.

[21] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought, 2024.

[22] William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers, 2025.

[23] Sajad Movahedi, Felix Sarnthein, Nicola Muca Cirone, and Antonio Orvieto. Fixed-point rnns: Interpolating from diagonal to dense, 2025.

[24] Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurélien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[25] OpenAI. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, September 2024. OpenAI blog post, accompanying the o1 release.

[26] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, JMLR Workshop and Conference Proceedings, pages 1310–1318. JMLR.org, 2013.

[27] Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models, 2026.

[28] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers, 2025.

[29] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024.

[30] Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *CoRR*, abs/2502.05795, 2025.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[32] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi-Yadkori. Hierarchical reasoning model. *CoRR*, abs/2506.21734, 2025.

[33] 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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[34] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*,

Proceedings of Machine Learning Research, pages 10524–10533. PMLR, 2020.

[35] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, 2024.

[36] Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought, 2025.

[37] Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms, 2016.

A. Background and Related Works

Looped models. Looped architectures address the flexibility ingredient of reasoning models by decoupling effective depth from parameter count, providing variable computation per input without scaling the number of parameters. This makes looping a natural inductive bias for tasks that can be solved by repeatedly applying local or compositional subroutines. Early examples include Neural GPUs [37] and Universal Transformers [7]; more recent work shows that looped transformers can improve length generalization, learn algorithmic structure, and approximate much deeper untied models on reasoning tasks [35, 11, 28, 17, 12]. Recent recursive reasoning models such as HRM [32] and TRM [16] instantiate this principle in compact architectures. However, they leave open a central design question: *how should a looped model decide how many iterations to run on each input at test-time?* FPRM addresses this through fixed-point halting.

Adaptive computation. Classical ACT methods answer the adaptivity question by learning an explicit halting rule, such as halting probabilities, per-token stopping decisions, or learned distributions over computation depth [7, 3]. Such mechanisms can allocate more computation to harder instances, but they introduce an additional learned decision process on top of the recurrent computation itself, with an inherently non-differentiable objective.

Deep equilibrium and fixed-point models. An alternative is to use the convergence of the latent dynamics as the halting criterion. In this view, computation stops when consecutive iterates become sufficiently close $\|\mathbf{z}_{i+1} - \mathbf{z}_i\| \leq \epsilon$, which corresponds to convergence toward a fixed point $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x})$. This view is most extensively developed in Deep Equilibrium Models [2], which replace a finite stack of layers by the equilibrium point of a weight-tied transformation. It has been shown that the same fixed-point view extends to looped recurrent architectures. This provides an input-dependent notion of computation: *easy instances may reach a stable representation after few iterations, while harder instances can continue refining their state until convergence* [23]. Therefore, for looped reasoning models fixed-point convergence is not only a representation-learning device; it is also the mechanism that determines how much computation is performed. FPRM exploits this view to obtain a parameter-free halting criterion that inherits the convergence guarantees of the iteration map.

Signal propagation in looped models. As unrolled looped architectures can be viewed as deep networks, they are exposed to the same signal-propagation difficulties that arise in very deep transformers. In deep sequence models, increasing depth can make optimization harder and can prevent later layers from being effectively used, a phenomenon often discussed as the curse of depth [8, 24, 30]. Pre-norm is the standard remedy for these issues, but in a looped setting it removes a property the architecture relies on: **bounded activations**. Bounded activations, important for stable training, are typically enforced through post-norm [12, 32, 16]. FPRM combines pre-norm with residual scaling to recover bounded and stable dynamics while still allowing gradients and representations to propagate through many iterations.

B. Proofs

B.1. Fixed-point iterations bound

Proof. We start by unrolling the computation across the L layers within a single fixed-point iteration. By recursively applying the layer update, we obtain

$$\begin{aligned}
\mathbf{z}_i^L &= \alpha_1 \cdot \mathbf{z}_i^{L-1} + \beta_1 \cdot f_{\theta_L}^L\left(\mathbf{z}_i^{L-1}\right) && (4)\\
&= \alpha_1^2 \cdot \mathbf{z}_i^{L-2} + \alpha_1\beta_1 \cdot f_{\theta_{L-1}}^{L-1}\left(\mathbf{z}_i^{L-2}\right) + \beta_1 \cdot f_{\theta_L}^L\left(\mathbf{z}_i^{L-1}\right) && (5)\\
&= \alpha_1^3 \cdot \mathbf{z}_i^{L-3} + \alpha_1^2\beta_1 \cdot f_{\theta_{L-2}}^{L-2}\left(\mathbf{z}_i^{L-3}\right) + \alpha_1\beta_1 \cdot f_{\theta_{L-1}}^{L-1}\left(\mathbf{z}_i^{L-2}\right) + \beta_1 \cdot f_{\theta_L}^L\left(\mathbf{z}_i^{L-1}\right) && (6)\\
&\ \ \vdots && (7)\\
&= \alpha_1^L \cdot \mathbf{z}_i^0 + \beta_1 \cdot \left(\sum_{j=0}^{L-1} \alpha_1^j \cdot f_{\theta_{L-j}}^{L-j}\left(\mathbf{z}_i^{L-j-1}\right)\right). && (8)
\end{aligned}$$

Now, substituting this expression into the fixed-point update gives

$$\mathbf{z}_{i+1}^0 = \alpha_2 \cdot \mathbf{z}_i^L + \beta_2 \cdot \mathbf{x} \tag{9}$$

$$= \alpha_2 \cdot \left(\alpha_1^L \cdot \mathbf{z}_i^0 + \beta_1 \cdot \sum_{j=0}^{L-1} \alpha_1^j \cdot f_{\theta_{L-j}}^{L-j} \left(\mathbf{z}_i^{L-j-1} \right) \right) + \beta_2 \cdot \mathbf{x} \tag{10}$$

$$= \alpha_2 \alpha_1^L \cdot \mathbf{z}_i^0 + \alpha_2 \beta_1 \cdot \left(\sum_{j=0}^{L-1} \alpha_1^j \cdot f_{\theta_{L-j}}^{L-j} \left(\mathbf{z}_i^{L-j-1} \right) \right) + \beta_2 \cdot \mathbf{x}. \tag{11}$$

For compactness, define $\rho = \alpha_2 \alpha_1^L$ and $\mathbf{s}_i = \sum_{j=0}^{L-1} \alpha_1^j \cdot f_{\theta_{L-j}}^{L-j} \left(\mathbf{z}_i^{L-j-1} \right)$. Then the fixed-point iteration can be written as $\mathbf{z}_{i+1}^0 = \rho \cdot \mathbf{z}_i^0 + \alpha_2 \beta_1 \cdot \mathbf{s}_i + \beta_2 \cdot \mathbf{x}$. Unrolling this recursion over fixed-point iterations gives

$$\mathbf{z}_{i+1}^0 = \rho^{i+1} \cdot \mathbf{z}_0^0 + \beta_2 \left(\sum_{k=0}^{i} \rho^k \right) \cdot \mathbf{x} + \alpha_2 \beta_1 \left(\sum_{k=0}^{i} \rho^k \cdot \mathbf{s}_{i-k} \right). \tag{12}$$

Since $0 \leq \alpha_1, \alpha_2 < 1$, we have $0 \leq \rho < 1$. Therefore, the geometric series is convergent. Taking norms and using the boundedness of each layer map, we get

$$\left\| \mathbf{z}_{i+1}^0 \right\| \leq \rho^{i+1} \left\| \mathbf{z}_0^0 \right\| + \beta_2 \left(\sum_{k=0}^{i} \rho^k \right) \|\mathbf{x}\| + \alpha_2 \beta_1 \left(\sum_{k=0}^{i} \rho^k \right) \left(\sum_{j=0}^{L-1} \alpha_1^j \right) c_f. \tag{13}$$

Letting $i \to \infty$, the first term vanishes and the two geometric sums converge, which gives

$$\limsup_{i \to \infty} \left\| \mathbf{z}_i^0 \right\| \leq \frac{\beta_2}{1-\rho} \|\mathbf{x}\| + \frac{\alpha_2 \beta_1}{1-\rho} \left(\frac{1-\alpha_1^L}{1-\alpha_1} \right) c_f. \tag{14}$$

Substituting back $\rho = \alpha_2 \alpha_1^L$, we obtain

$$\limsup_{i \to \infty} \left\| \mathbf{z}_i^0 \right\| \leq \frac{\beta_2}{1-\alpha_2 \alpha_1^L} \|\mathbf{x}\| + \frac{\alpha_2 \beta_1 \left(1-\alpha_1^L\right)}{\left(1-\alpha_2 \alpha_1^L\right)\left(1-\alpha_1\right)} c_f. \tag{15}$$

We now set $\beta_2 = 1 - \alpha_2 \alpha_1^L$. This makes the coefficient of $\|\mathbf{x}\|$ equal to 1. Furthermore, setting $\beta_1 = \frac{\beta_2 (1-\alpha_1)}{\left(1-\alpha_1^L\right)}$ makes the coefficient of c_f equal to α_2. Therefore,

$$\limsup_{i \to \infty} \left\| \mathbf{z}_i^0 \right\| \leq \|\mathbf{x}\| + \alpha_2 \cdot c_f. \tag{16}$$

In particular, if the fixed-point iteration converges to $\mathbf{z}_\infty^0$, then

$$\left\| \mathbf{z}_\infty^0 \right\| \leq \|\mathbf{x}\| + \alpha_2 \cdot c_f. \tag{17}$$

This completes the proof. □

B.2. Contractive mapping

Proof. We first show that $T_{\alpha_2}(.; \mathbf{x})$ is a contraction with respect to $\mathbf{z}$. For any $\mathbf{z}, \mathbf{z}'$, using the Lipschitzness of $\tilde{f}_\theta(.)$, we have

$$\begin{aligned} \|T_{\alpha_2}(\mathbf{z}; \mathbf{x}) - T_{\alpha_2}(\mathbf{z}'; \mathbf{x})\| &= \left\| \tilde{f}_\theta \left(\alpha_2 \cdot \mathbf{z} + \beta_2 \cdot \mathbf{x} \right) - \tilde{f}_\theta \left(\alpha_2 \cdot \mathbf{z}' + \beta_2 \cdot \mathbf{x} \right) \right\| \\ &\leq \lambda_f \left\| \alpha_2 \cdot \mathbf{z} + \beta_2 \cdot \mathbf{x} - \alpha_2 \cdot \mathbf{z}' - \beta_2 \cdot \mathbf{x} \right\| \\ &= \alpha_2 \lambda_f \left\| \mathbf{z} - \mathbf{z}' \right\|. \end{aligned}$$

Therefore, if $0 \leq \alpha_2 \lambda_f < 1$, the map $T_{\alpha_2}(.; \mathbf{x})$ is strictly contractive. By the Banach fixed-point theorem, it has a unique fixed point $\mathbf{z}^\star$, and the iteration $\mathbf{z}_{i+1} = T_{\alpha_2}(\mathbf{z}_i; \mathbf{x})$ converges to $\mathbf{z}^\star$.

We now prove the residual bound. Since $\mathbf{z}_{i+1} = T_{\alpha_2}(\mathbf{z}_i; \mathbf{x})$, the residual at iteration i can be written as

$$\|T_{\alpha_2}(\mathbf{z}_i; \mathbf{x}) - \mathbf{z}_i\| = \|\mathbf{z}_{i+1} - \mathbf{z}_i\| .$$

Using the contraction property of $T_{\alpha_2}(.; \mathbf{x})$, we get

$$\begin{aligned} \|\mathbf{z}_{i+1} - \mathbf{z}_i\| &= \|T_{\alpha_2}(\mathbf{z}_i; \mathbf{x}) - T_{\alpha_2}(\mathbf{z}_{i-1}; \mathbf{x})\| \\ &\leq \alpha_2 \lambda_f \|\mathbf{z}_i - \mathbf{z}_{i-1}\| . \end{aligned}$$

Applying this inequality recursively gives

$$\|\mathbf{z}_{i+1} - \mathbf{z}_i\| \leq (\alpha_2 \lambda_f)^i \|\mathbf{z}_1 - \mathbf{z}_0\| .$$

Since $\mathbf{z}_1 = T_{\alpha_2}(\mathbf{z}_0; \mathbf{x})$, we obtain

$$\|T_{\alpha_2}(\mathbf{z}_i; \mathbf{x}) - \mathbf{z}_i\| \leq (\alpha_2 \lambda_f)^i \|T_{\alpha_2}(\mathbf{z}_0; \mathbf{x}) - \mathbf{z}_0\| .$$

Finally, if $\mathbf{z}_0 = 0$, then $T_{\alpha_2}(\mathbf{z}_0; \mathbf{x}) = \tilde{f}_\theta(\beta_2 \cdot \mathbf{x})$, and therefore

$$\|T_{\alpha_2}(\mathbf{z}_i; \mathbf{x}) - \mathbf{z}_i\| \leq (\alpha_2 \lambda_f)^i \left\| \tilde{f}_\theta(\beta_2 \cdot \mathbf{x}) \right\| .$$

This completes the proof. □

C. Proof of Proposition 1

Proof. Since $\|\mathbf{J}\|_2 = \sigma < 1$, the Neumann series is convergent, and we have $(\mathbf{I} - \mathbf{J})^{-1} = \sum_{j=0}^{\infty} \mathbf{J}^j$. Therefore, the error of the k-term truncated approximation is

$$\begin{aligned} \left\| (\mathbf{I} - \mathbf{J})^{-1} - \sum_{j=0}^{k-1} \mathbf{J}^j \right\|_F &= \left\| \sum_{j=k}^{\infty} \mathbf{J}^j \right\|_F \\ &\leq \sum_{j=k}^{\infty} \left\| \mathbf{J}^j \right\|_F . \end{aligned}$$

Using the relation $\|\mathbf{A}\|_F \leq \sqrt{D}\|\mathbf{A}\|_2$ for $\mathbf{A} \in \mathbb{R}^{D \times D}$, together with submultiplicativity of the spectral norm, we get

$$\begin{aligned} \left\| \mathbf{J}^j \right\|_F &\leq \sqrt{D} \left\| \mathbf{J}^j \right\|_2 \\ &\leq \sqrt{D} \|\mathbf{J}\|_2^j \\ &= \sqrt{D} \cdot \sigma^j . \end{aligned}$$

Substituting this into the previous inequality gives

$$\begin{aligned} \left\| (\mathbf{I} - \mathbf{J})^{-1} - \sum_{j=0}^{k-1} \mathbf{J}^j \right\|_F &\leq \sqrt{D} \sum_{j=k}^{\infty} \sigma^j \\ &= \sqrt{D} \cdot \frac{\sigma^k}{1 - \sigma} . \end{aligned}$$

Thus, the approximation error decays as $\mathcal{O}(\sigma^k)$.

To make the corresponding gradient statement explicit, let $\mathbf{P} = \frac{\partial f_\theta}{\partial \theta}(\mathbf{z}^\star; \mathbf{x})$ and $\boldsymbol{\delta} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^\star}$. The exact implicit gradient is

$\nabla_\theta \mathcal{L} = \mathbf{P}^\top (\mathbf{I} - \mathbf{J})^{-\top} \boldsymbol{\delta}$, whereas the k-step truncated BPTT gradient is $\widehat{\nabla}_\theta^{(k)} \mathcal{L} = \mathbf{P}^\top \left(\sum_{j=0}^{k-1} \left(\mathbf{J}^\top\right)^j \right) \boldsymbol{\delta}$. Therefore,

$$\begin{aligned}
\left\| \nabla_\theta \mathcal{L} - \widehat{\nabla}_\theta^{(k)} \mathcal{L} \right\|_2 &\leq \|\mathbf{P}\|_2 \left\| (\mathbf{I} - \mathbf{J})^{-\top} - \sum_{j=0}^{k-1} \left(\mathbf{J}^\top\right)^j \right\|_2 \|\boldsymbol{\delta}\|_2 \\
&\leq \|\mathbf{P}\|_2 \left(\sum_{j=k}^{\infty} \|\mathbf{J}\|_2^j \right) \|\boldsymbol{\delta}\|_2 \\
&= \|\mathbf{P}\|_2 \frac{\sigma^k}{1-\sigma} \|\boldsymbol{\delta}\|_2 .
\end{aligned}$$

Hence, the truncated BPTT gradient error also decays exponentially with the number of backward passes k. This completes the proof. □

C.1. Damping of fixed-point iterations

Proof. We first show that the fixed points of $g_{\eta,\theta}(\,\cdot\,;\mathbf{x})$ and $f_\theta(\,\cdot\,;\mathbf{x})$ coincide. Since

$$g_{\eta,\theta}(\mathbf{z};\mathbf{x}) - \mathbf{z} \;=\; \eta\big(f_\theta(\mathbf{z};\mathbf{x}) - \mathbf{z}\big),$$

and $\eta > 0$, we have $g_{\eta,\theta}(\mathbf{z};\mathbf{x}) = \mathbf{z}$ if and only if $f_\theta(\mathbf{z};\mathbf{x}) = \mathbf{z}$.

We now study the local stability of the damped iteration around $\mathbf{z}^\star$. The Jacobian of $g_{\eta,\theta}(\,\cdot\,;\mathbf{x})$ at $\mathbf{z}^\star$ is

$$\frac{\partial g_{\eta,\theta}}{\partial \mathbf{z}}(\mathbf{z}^\star;\mathbf{x}) \;=\; (1-\eta)\mathbf{I} + \eta\mathbf{J},$$

so for every eigenvalue λ_i of $\mathbf{J}$, the corresponding eigenvalue of the damped Jacobian is

$$\mu_i(\eta) \;=\; 1 - \eta + \eta\lambda_i \;=\; 1 + \eta(\lambda_i - 1).$$

Local asymptotic stability is implied by $|\mu_i(\eta)| < 1$ for all i. Writing $\lambda_i = a_i + \mathrm{i}\, b_i$ with $a_i = \Re(\lambda_i)$,

$$|\mu_i(\eta)|^2 \;=\; 1 + 2\eta(a_i - 1) + \eta^2 |\lambda_i - 1|^2.$$

Hence $|\mu_i(\eta)| < 1$ if and only if $\eta|\lambda_i - 1|^2 < 2(1 - a_i)$, i.e.,

$$0 \;<\; \eta \;<\; \frac{2(1 - \Re(\lambda_i))}{|\lambda_i - 1|^2}.$$

By assumption $\Re(\lambda_i) < 1$ for every i, so each upper bound is strictly positive. Setting

$$\eta_0 \;=\; \min\left\{ 1,\; \min_i \frac{2(1 - \Re(\lambda_i))}{|\lambda_i - 1|^2} \right\} \;>\; 0,$$

we obtain $|\mu_i(\eta)| < 1$ for every i and every $\eta \in (0, \eta_0)$. Therefore $\mathbf{z}^\star$ is locally asymptotically stable under the damped iteration $\mathbf{z}_{t+1} = g_{\eta,\theta}(\mathbf{z}_t;\mathbf{x})$, and the iterates converge to $\mathbf{z}^\star$ from any sufficiently close initialization. □

C.2. A toy failure mode for recurrent PostLN

Figure 5 gives a minimal version of the normalization tradeoff discussed in the main text. We take random $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{100\times 2}$, set $\mathbf{z}^0 = \mathbf{x}$, and tie the same rank-one map for $L = 20$ iterations,

$$W(\mathbf{w}) = \mathbf{w}\mathbf{1}^\top, \qquad \mathbf{w} = (w_1, w_2)^\top.$$

This choice allows 2d visualization while making the normalization operation non-trivial. Here N is row-wise ℓ_2 normalization,

$$\mathsf{N}(\mathbf{z})_{i,:} = \frac{\mathbf{z}_{i,:}}{\|\mathbf{z}_{i,:}\|_2 + \varepsilon}.$$

The post-norm recurrence is

$$\mathbf{z}_{\text{post}}^{\ell+1} = \mathsf{N}\big(\mathbf{z}_{\text{post}}^{\ell} + \mathbf{z}_{\text{post}}^{\ell} W(\mathbf{w})\big)\,,$$

while the scaled pre-norm recurrence is

$$\mathbf{z}_{\text{pre}}^{\ell+1} = (1-\beta)\mathbf{z}_{\text{pre}}^{\ell} + \beta\,\mathsf{N}\big(\mathbf{z}_{\text{pre}}^{\ell} W(\mathbf{w})\big)\,, \qquad \beta = \frac{1}{2}.$$

For both models we sweep a 200×200 grid over $[-5,5]^2$ and plot

$$\Delta\mathcal{L}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) - \min_{\mathbf{v}\in\mathcal{G}} \mathcal{L}(\mathbf{v}), \qquad \mathcal{L}(\mathbf{w}) = \frac{1}{nd}\left\|\mathbf{z}^{L}(\mathbf{w}) - \mathbf{y}\right\|_F^2.$$

The figure illustrates that **boundedness does not imply trainability**. Post-norm keeps the recurrent state bounded by construction: after every step, each row is projected back to the unit sphere. However, the same projection also removes radial information at every iteration. In this two-parameter slice, the resulting loss is organized into thin angular sectors with sharp ridges and narrow low-loss regions. Thus a random initialization of $\mathbf{w}$ is likely to start in a bounded but poorly conditioned part of the landscape, where the gradient does not point into a useful basin. This is the toy analogue of the optimization difficulty of recurrent post-norm blocks.

The right panel is not bare pre-norm; it is pre-norm with residual scaling. This matters because naive pre-norm removes the projection that controls the recurrent state and can lead to activation growth, as discussed above. With the scaled update, each row satisfies

$$\|\mathbf{z}_{\text{pre},i}^{\ell+1}\|_2 \le (1-\beta)\|\mathbf{z}_{\text{pre},i}^{\ell}\|_2 + \beta,$$

so the toy dynamics remain bounded while preserving a live residual stream. The broader low-loss region in Figure 5 is therefore consistent with the architecture we use in Theorem 1: pre-normalization improves signal propagation, while residual scaling replaces the boundedness mechanism that post-normalization provided.

D. Further Details About the Architecture

Optimization of fixed-point models. One advantage of contractive fixed-point models is that they can be trained using truncated back-propagation through time (BPTT) [13]. Specifically, following the implicit function theorem we can write the gradient w.r.t. the parameters of the model as [2]:

$$\frac{\partial \mathbf{z}^\star}{\partial \theta} \;=\; \left(\mathbf{I} - \frac{\partial f_\theta}{\partial \mathbf{z}}(\mathbf{z}^\star;\mathbf{x})\right)^{-1} \frac{\partial f_\theta}{\partial \theta}(\mathbf{z}^\star;\mathbf{x}). \tag{18}$$

As the contractiveness entails a contractive Jacobian, the Neumann series corresponding to the first term on the RHS of Equation (18) is converging, allowing us to estimate the gradient as:

$$\frac{d\mathbf{z}^\star}{d\theta} \;\approx\; \sum_{j=0}^{k-1} \mathbf{J}^j \,\frac{\partial f_\theta}{\partial \theta}(\mathbf{z}^\star;\mathbf{x}), \tag{19}$$

which in our setting translates into doing k backward passes after we reach the fixed-point. The following proposition shows that truncated BPTT becomes exponentially accurate as an estimation of the gradient of fixed-point models:

Let $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star;\mathbf{x})$ *and* $\mathbf{J} = \frac{\partial f_\theta}{\partial \mathbf{z}}(\mathbf{z}^\star;\mathbf{x}) \in \mathbb{R}^{D\times D}$. *If* $\mathbf{J}$ *is contractive in spectral norm,* $\|\mathbf{J}\|_2 = \sigma < 1$, *then for every* $k \ge 0$,

$$\left\| (\mathbf{I}-\mathbf{J})^{-1} - \sum_{j=0}^{k-1} \mathbf{J}^j \right\|_F \;\le\; \sqrt{D}\,\frac{\sigma^k}{1-\sigma}.$$

The proof is in Section C. An essentially equivalent result appears in the proof of Theorem 2 of [13].

Proposition 1 allows for a fixed memory footprint during training, essentially decoupling the number of loops from the memory complexity of the model. In the same spirit, HRM [32] and TRM [16] approximate the gradient with a small number of backward passes at the fixed-point, although TRM argues that the fixed-point condition is unnecessary in practice. However, we note that reaching fixed-points in this paper is not strictly done for the purpose of optimization, but rather as a halting mechanism to replace ACT. In the following, we provide a full description of the proposed method.

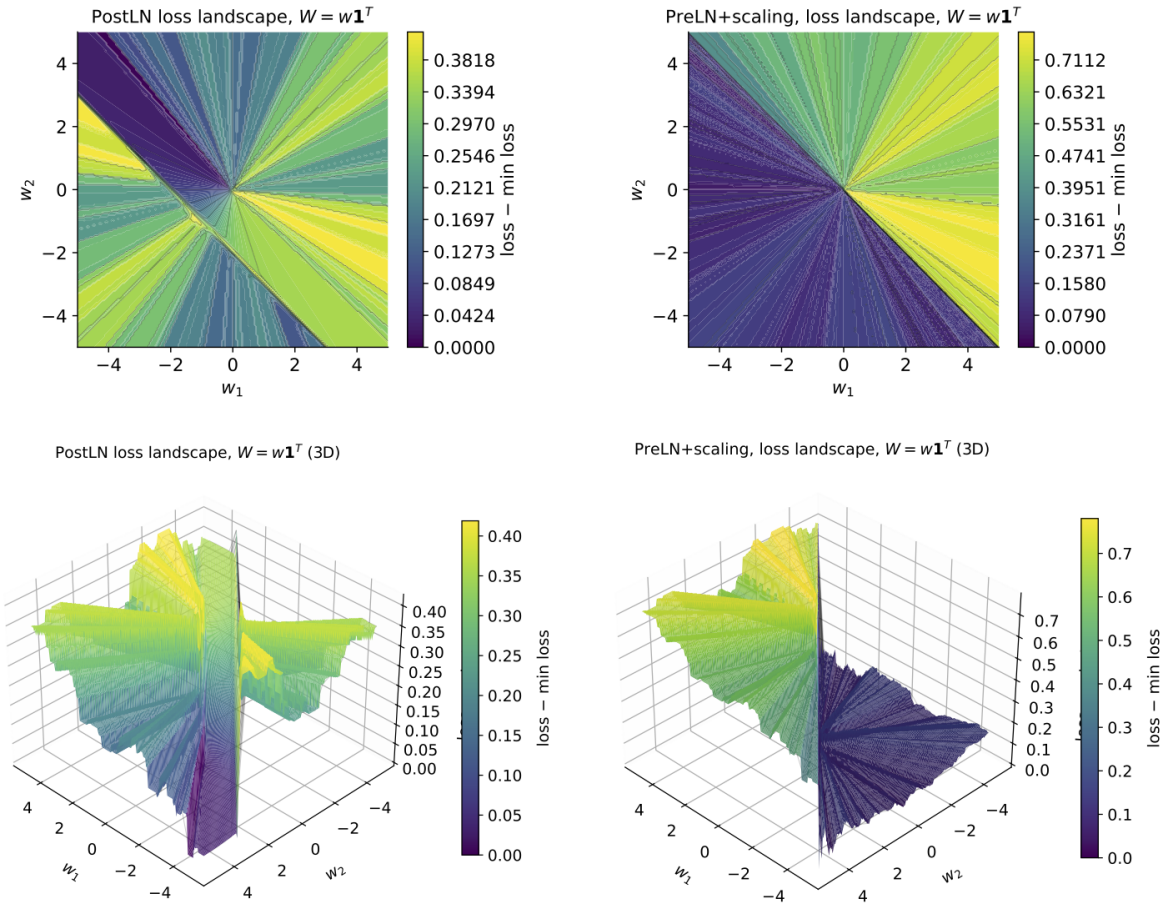


Figure 5. Landscape visualization for the setup proposed in Section C.2

Algorithm 1 Fixed-point optimiser FPOPT: one damped step with patience-based decay

Proposition 1 (Exponential decay of truncated-BPTT error)**. Require:** initial damping η_0; decay $\gamma \in (0, 1)$; patience P

Internal state: $\eta \leftarrow \eta_0,\ p \leftarrow P,\ r^\star \leftarrow \infty$

procedure STEP($\mathbf{z}$, $\tilde{\mathbf{z}}$)
  $r \leftarrow \|\mathbf{z} - \tilde{\mathbf{z}}\|_\infty / (\|\tilde{\mathbf{z}}\|_\infty + \epsilon)$ ▷ residual
  $\mathbf{z} \leftarrow \eta\, \tilde{\mathbf{z}} + (1 - \eta)\, \mathbf{z}$ ▷ damped update
  if $r < r^\star$ **then**
    $r^\star \leftarrow r,\ p \leftarrow P$ ▷ progress: reset
  else
    $p \leftarrow p - 1$
    if $p = 0$ **then**
      $\eta \leftarrow \gamma\, \eta,\ p \leftarrow P$ ▷ decay η
    end if
  end if
  return $\mathbf{z}$, r
end procedure

Algorithm 2 FPRM training loop with truncated BPTT and deep supervision

Require: Model f_θ; prediction head h_ϕ; fixed-point optimiser FPOPT; model optimiser MODELOPT; input $\mathbf{x}$; target $\mathbf{y}$; BPTT depth K; initial state $\mathbf{z}_0$

$\mathbf{z} \leftarrow \mathbf{z}_0$
while FPOPT.CONT() **do** ▷ outer loop
  for $k = 1, \ldots, K$ **do** ▷ BPTT window
    $\mathbf{y}_k \leftarrow f_\theta(\mathbf{z};\, \mathbf{x})$
    $\mathbf{z} \leftarrow$ FPOPT.STEP($\mathbf{z}$, $\mathbf{y}_k$)
  end for
  $\hat{\mathbf{y}} \leftarrow h_\phi(\mathbf{z})$ ▷ deep supervision
  $\mathcal{L} \leftarrow$ CROSSENTROPY($\hat{\mathbf{y}}$, $\mathbf{y}$)
  MODELOPT.BACKWARD($\mathcal{L}$)
  $\mathbf{z} \leftarrow \mathrm{detach}(\mathbf{z})$
end while

Fixed-point solver. Let $\mathbf{z}^{(k)} \in \mathbb{R}^{B \times T \times d}$ denote the iterate at step k of the solver (with B batch, T sequence, d hidden), and $\mathbf{z}^{(k+1)}$ the proposed update. Convergence is measured per sample $\mathbf{z}_b^{(k)}$ by the relative ℓ_∞ residual,

$$\mathbf{r}_b^{(k)} = \frac{\left|\left|\mathbf{z}_b^{(k+1)} - \mathbf{z}_b^{(k)}\right|\right|\infty}{\left|\left|\mathbf{z}_b^{(k+1)}\right|\right|\infty + \epsilon}, \mathbf{r}_b^{(k)} \in \mathbb{R}.$$

A sample is declared converged when $r_b^{(k)} < \tau$. In practice, we set τ to 0.1. During training, the solver terminates as soon as the fraction of converged samples falls below a tolerated quantile p,

$$\sum_{b=1}^{B} \mathbf{1}\left[r_b^{(k)} \geq \tau\right] < p \cdot B.$$

Two safeguards bound the loop: a hard cap $k \leq K_{\max}$ on iterations, and early termination if the adaptive step size collapses below a minimum. At evaluation we set $p = 0$, requiring every sample to satisfy $r_b^{(k)} < \tau$.

Deep supervision. We adopt a similar deep supervision mechanism to HRM [32] and TRM [16]. Let T_{sup} denote the deep supervision interval. Every T_{sup} iterations the intermediate activations of the model are decoded through the output head, the task loss is computed, and backpropagation is performed through the most recent step only. Then, the activations and the solver state are detached, implementing truncated backpropagation through time (TBPTT). After the outer solver converges,

Table 3. Effect of adding FPRM's architectural modifications to TRM: pre-norm, residual scaling (α_1, α_2), and a final post-activation output norm, individually and in combination. Values are the change in test sequence accuracy on Sudoku Extreme (%) relative to the TRM baseline of 74.32%.

Configuration	Δ Seq. Acc. (%)
Original TRM (post-norm, no scale)	—
+ output norm	-0.47
+ residual scaling (α_1 only)	-13.96
+ residual scaling (α_1, α_2)	-6.16
+ pre-norm & residual scaling (α_1, α_2)	$\mathbf{-54.55}$
+ pre-norm, residual scaling, final norm (full FPRM stack)	-9.45

a final iteration is performed with gradients enabled and its output is supervised as well. The number of backward passes per forward pass is therefore $\lceil K/T_{\text{sup}} \rceil + 1$, where K is the number of iterations actually taken by the solver on that batch. Setting $T_{\text{sup}} = 1$ supervises every intermediate-step, which is the typical configuration.

E. Additional Experiments

Introducing residual scaling to TRM. Since TRM could potentially benefit from FPRM's architectural changes, we add them to TRM individually and in combination, measuring test sequence accuracy on Sudoku Extreme. Results are shown in Table 3, reported as the change relative to the TRM baseline (74.32%). No configuration matches the baseline, and the effects are not additive. The optimal configuration for FPRM turned out to be the most detrimental for TRM.

F. Additional Experimental Details

Weight initialization. It seems like initializing the weights using a truncated normal distribution (LeCun initialization) is common practice in looped architectures. In our experiments, it accelerates the convergence but there's very little material difference in sequence accuracy after convergence.

Grokking. There is some evidence for grokking in looped architectures, but in maze we observe convergence on the train data. And training the models for a longer period (up to 7 days) did not yield better performance.

Hyperparameters, device specification and estimated GPU hours Based on our approximation, around 1k GPU-hours produced the numbers in the paper, out of 14k GPU-hours burned overall in the project. Next, we provide the values for some of the most important hyperparameters in the paper, per each model and dataset.

Table 4. Hyperparameters for Sudoku-Extreme experiments (Table 1 of the paper). Shared across all models: 1×A100-40GB, batch 768, 60 000 epochs, ADAM_ATAN2 optimizer, lr= 10^{-4} (constant after 2 000-step warm-up), weight-decay 1.0, EMA enabled, puzzle-embedding length 16.

	HRM	TRM	FPRM
Looping structure			
H-cycles	2	3	3
L-cycles	2	6	6
H-layers	4	0	0
L-layers	4	2	2
$n_{\text{back},L}$	–	$= L$-cycles	4
Halting			
mechanism	ACT	ACT	fixed-point
`halt_max_steps`	16	16	–
`max_iter`	–	–	1000
iter. distribution	–	–	exponential (s=16)
Block / signal-prop modifications			
norm type	post-norm	post-norm	pre-norm
norm placement	–	–	output
residual scaling	–	–	input-independent
α_1, α_2 init	–	–	0.5, 0.5
conv branch	–	none	conv2d (k=3)

Table 5. Hyperparameters for Maze-Hard experiments (Table 1 of the paper). HRM uses the published configuration; TRM and FPRM both train on `maze-30x30-hard-1k` with 4×A100-80GB, 60 000 epochs, lr= 10^{-4} constant, weight-decay 1.0, EMA enabled, puzzle-embedding length 16.

	HRM	TRM	FPRM
optimizer	ADAM_ATAN2	ADAM_ATAN2	ADAM_ATAN2
batch size	768	512	768
lr warm-up steps	2000	0	2000
Looping structure			
H-cycles	2	3	3
L-cycles	2	4	6
H-layers	4	0	0
L-layers	4	2	2
$n_{\text{back},L}$	–	$= L$-cycles	4
Halting			
mechanism	ACT	ACT	fixed-point
`halt_max_steps`	16	16	–
`max_iter`	–	–	1000
iter. distribution	–	–	exponential (s=16)
Block / signal-prop modifications			
norm type	post-norm	post-norm	pre-norm
norm placement	–	–	output
residual scaling	–	–	input-independent
α_1, α_2 init	–	–	0.75, 0.5
conv branch	–	none	conv2d (k=3)

Table 6. Hyperparameters for state-tracking experiments on A_5 (Figure 3 of the paper) and for the Universal Transformer signal-propagation analysis (Figure 2). All runs use 1×A100-80GB, global batch 1024, ADAM_ATAN2, lr-warm-up 0, weight-decay 10^{-2}, EMA disabled, no puzzle embedding (`puzzle_emb_len`=0). Trained at k_{train}=32, evaluated for $k \in [2, 128]$.

	TRM	TRM + ACT	FPRM	UT (Fig. 2)
epochs	50	50	20	30
learning rate	10^{-4}	10^{-4}	10^{-3}	10^{-4}
Looping structure				
H-cycles	2	2	3	3
L-cycles	4	4	6	6
L-layers	4	4	2	2
$n_{\text{back},L}$	$= L$-cycles	$= L$-cycles	4	4
Halting				
mechanism	fixed iters	ACT	fixed-point	fixed iters
`halt_max_steps`	16	16	–	–
`max_iter`	–	–	128	$= k_{\text{train}}$
iter. distribution	–	–	deterministic	deterministic
Block / signal-prop modifications				
norm type	post-norm	post-norm	pre-norm	sweep†
norm placement	–	–	none	none
residual scaling	–	–	input-indep.	sweep†
α_1, α_2 init	–	–	0.5, 0.5	sweep†
spec.-norm linear	–	–	yes	no
conv branch	none	none	conv1d (k=4)	none

† UT row sweeps the {post-norm, pre-norm, pre-norm + residual-scaling} variants from Figure 2a; the residual-scaling variant uses α_1=0.75, α_2=0.5 and $k_{\text{train}} \in \{8, 16, 32, 64\}$.