Fixed-Point Reasoners: Stable and Adaptive Deep Looped Transformers

Sajad Movahedi [*1] **Vera Milovanovic** [*1 2] **Shlomo Libo-Feigin** [*1 2] **Alexander Theus** [*1 2]
Thomas Hoffmann [2] **Valentina Boeva** [†2] **Konstantin Rusch** [†1] **Antonio Orvieto** [†1]

Abstract

Looped architectures provide an inductive bias toward learning step-by-step procedures for tasks that require compositional reasoning. The depth of effective layers reached by looping determines the quality of the solution these models find. Similar to deep architectures, looped architectures are prone to a signal propagation problem induced by depth as the halting decision is postponed. In this paper, we address the signal propagation issue by using pre-norm layers and residual scaling. Building on these architectural modifications, we propose **FPRM**: a Transformer-based **F**ixed-**P**oint **R**easoning **M**odel 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, state-tracking and ARC-AGI. The implementation can be found here.

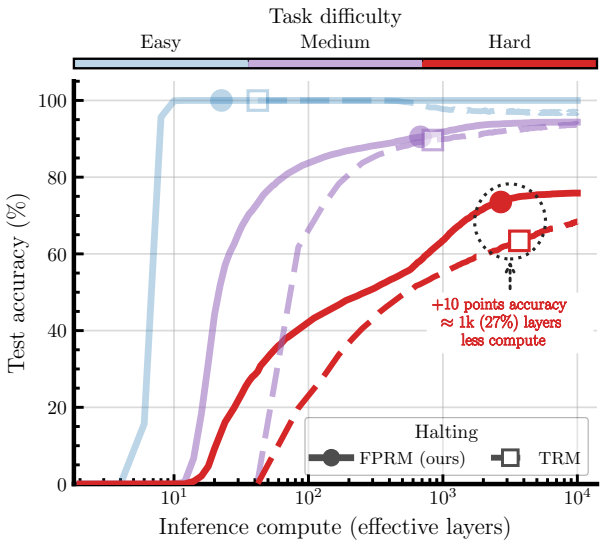


Figure 1. **Signal propagation and adaptivity, FPRM vs. TRM:** Sudoku-Extreme performance as a function of compute across difficulty. Despite being non-hierarchical, FPRM scales better, while correctly detecting the accuracy plateaus by using fixed-points for halting.

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 [48, 56]. However, doing so requires two ingredients. **(1) flexibility:** the possibility of spending a variable amount of compute on the problem. Once the model is flexible, the next step is **(2) adaptivity:** a way to scale the compute spent on the problem; i.e., when to halt the computation.

The standard way to achieve both is through a Chain-of-Thought (CoT) mechanism [62]. With 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 and hand-crafted reasoning traces [31]. This makes the method complex and undermines the desirable property of end-to-end training.

An alternative approach to end-to-end training for reasoning models is emerging in the form of looped architectures [14, 7, 54, 32]. Whereas in CoT, the compute scales along the sequence dimension, in looped architectures it scales along the depth dimension (i):

$$\mathbf{z}_{i+1} = f_\theta(\mathbf{z}_i; \mathbf{x}), \tag{1}$$

where $f_\theta(\mathbf{z}; \mathbf{x})$ is a neural network, $\mathbf{x}$ is the input, and $\mathbf{z}_i$ is the ith latent representation. Thus, compute can be increased at test-time by looping in depth, naturally introducing flexibility from the architecture. Looped models have been shown to have an inductive bias towards learning algorithms [64, 20] and have achieved remarkable success on reasoning benchmarks. For example, Hierarchical Reasoning Model

[*]Equal contribution [†]Equal advising. [1]ELLIS Institute Tübingen, Max Planck Institute for Intelligent Systems, Tübingen AI Center [2]ETH Zurich, Zurich, Switzerland. Correspondence to: Sajad Movahedi <sajad.movahedi@tue.ellis.eu>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

(HRM) [61] and Tiny Reasoning Model (TRM) [37] incorporate a hierarchical structure into the looping process, proving effective in solving puzzle tasks such as sudoku, maze, and ARC-AGI [13].

The halting decision (i.e., deciding when to stop iterating), however, is no longer trivial in looped models. Most approaches either fix or randomly sample the number of loops [36, 65, 25, 54, 11, 7], which eliminates adaptivity, or use external modules trained to make halting decisions [61, 37, 14]. The latter is associated with separate Adaptive Computation Time (ACT) networks [30], which introduce optimization challenges, as they require a continuous relaxation of a discrete objective. As a result, ACT can fail to provide adaptivity, which we demonstrate in the case of HRM and TRM (Figures 4, 5). We mitigate this issue by introducing a different halting mechanism: let the model loop until its hidden state converges to a fixed-point, and use the convergence itself as the halting signal. Unlike ACT, the fixed-point halting mechanism requires no external module and lets the model spend as much compute as a given input demands.

A further challenge arises as the number of loops in the model increases. This is desirable for harder tasks, but unrolling the recursions yields a deep effective layer. As a result, looped architectures suffer from the same **signal propagation** issue as deep non-looped networks. To this end, we adapt architectural techniques originally developed to optimize traditional transformers at deep effective layers, carefully modifying them for the Looped Transformer setting. Our key intuition is that *Looped Transformers are, in part, very deep transformers*.

Viewed this way, one design choice is surprising: looped models commonly use post-norm [14, 25, 61, 37]. Deep non-looped architectures, by contrast, prefer pre-norm, since post-norm causes unstable training [63, 47]. Yet in looped models, post-norm serves a specific purpose. It keeps the magnitude of activations bounded as the model iterates, preventing the hidden states from diverging [43] (see Figure 2). This raises 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. In this paper, we use residual scaling parameters to give an affirmative answer to this question. In Figure 1, we observe that fixed-point halting, together with better signal propagation, allows FPRM, a **non**-hierarchical reasoning model, to outperform TRM at a lower cost on the Sudoku-Extreme benchmark. We provide a description of how the figure was made in Section E.

We summarize our contributions as follows:

1. **Successfully training a looped pre-norm Transformer.** We modify a Transformer layer to be trainable over deeper effective layers by switching post-norm to pre-norm and adding residual scaling parameters.
2. **Reaching a fixed-point as a halting mechanism.** To enable stable training, we propose a theoretically motivated modification specific to fixed-point models to limit the oscillation around the fixed-point.
3. **Proposing the framework FPRM: Fixed-Point Reasoning Model.** We show that it outperforms the baselines on Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and state-tracking benchmarks A_5 and S_5. Notably, we achieve our results without the hierarchical structure of HRM and TRM. To the best of our knowledge, FPRM is the first Transformer-based reasoning model that exhibits adaptivity of compute to task difficulty (Figures 1,4).

We provide a thorough discussion of related work in Section A.

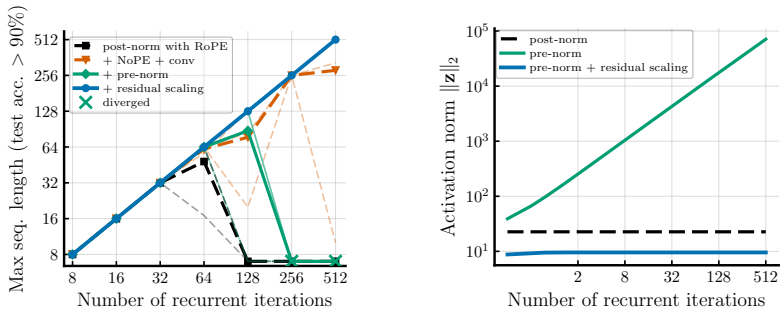
2. Method

Looped architectures rely on bounded activations for stability: as the model loops, an unbounded layer can cause the hidden states to grow without limit (Figure 2). Current methods mostly employ post-norm to satisfy the boundedness condition [37, 61, 25]. However, in fixed-depth models post-norm introduces a signal propagation issue, often associated with unstable training and restricted effective layer, as reported by [15, 47, 59]. In Section C, we provide evidence for trainability issues in a toy looped model with post-norm, despite stability induced by bounded activations.

Consequently, in this section we propose to **(a) switch to pre-norm** to recover trainability at higher depth, while preserving boundedness by **(b) scaling the residual stream and the sub-layer outputs** (the attention and feed-forward maps). Together, these modifications yield a layer that is both stable under looping and trainable at large depth. Additionally, we introduce **(c) a fixed-point halting mechanism** that decides the effective layer adaptively per input. Interestingly, we find that with these changes we can **(d) remove the hierarchy** common in recent Looped Transformers [61, 37], resulting in a much simpler model. We provide an overview of our proposed model in Figure 3.

2.1. Improving signal propagation with pre-norm

We start by introducing pre-norm and post-norm in a Transformer layer. A Transformer layer consists of two sub-layers (with $f^{\ell}_{\theta^{\ell}}(.)$ denoting the $\ell^{th} \in \{1, \ldots, 2L\}$ sub-layer): multi-head attention and a feed-forward network. We denote the Looped Transformer model as defined in Equation (1), consisting of multiple Transformer layers, by $f_{\theta}(.)$. In the Transformer layer, the two sub-layers are interleaved with



(a) Trainability vs. context length

(b) Activation norm at init.

Figure 2. **The blessing and the curse of depth in Looped Transformers.** A deep effective layer can unlock expressivity, but also creates a stability challenge: pre-norm models without residual scaling can diverge in activation norm, while post-norm models may struggle to utilize the signal.

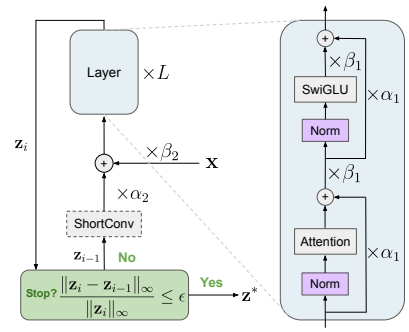


Figure 3. **FPRM architecture.** Our fixed-point Looped Transformer uses pre-norm and residual scaling for improved signal propagation.

layer normalization (Norm). Two canonical placements of the normalization define two variants of the layer. The original *post-norm* formulation ($\mathrm{Norm}_{\mathrm{post}}$) [60] applies normalization after the residual addition with the residual stream ($\mathbf{z}^{\ell-1}$), while the *pre-norm* variant ($\mathrm{Norm}_{\mathrm{pre}}$) [63] applies normalization to the input of each sub-layer without modifying the residual stream:

$$\mathbf{z}^{\ell} = \mathrm{Norm}_{\mathrm{post}}\left(\mathbf{z}^{\ell-1} + f^{\ell}_{\theta^{\ell}}\left(\mathrm{Norm}_{\mathrm{pre}}\left(\mathbf{z}^{\ell-1}\right)\right)\right), \quad \ell = 1, \ldots, 2L,$$

where L is the number of layers (Transformer blocks). In fixed-depth models, both normalization placements have been linked to training issues at large depth: post-norm bounds activation magnitudes but induces a signal propagation problem [41, 47], while pre-norm improves signal propagation but causes exponential growth in residual magnitude [41, 63]. Therefore, while using pre-norm in a deep neural network is desirable from the signal propagation standpoint, it can introduce instability due to unbounded activations.

Motivated by these observations, we investigate both normalization placements in Looped Transformers [54, 14]. In Figure 2a, we test the positive correlation between the effective layer and the expressivity (maximum sequence length with $>90\%$ test accuracy) of a Looped Transformer for the state-tracking task A_5 [45]. We observe that increasing the effective layer of a Looped Transformer with post-norm does not translate into improved expressivity, while a pre-norm variant diverges at larger depth. This divergence can be attributed to the exponential growth of the activations, apparent in Figure 2b. Therefore, to use pre-norm and improve signal propagation in deep looped models, we must first stabilize it, which we do in the following.

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 deeper effective layer, causing trainability issues observable in Figure 2a. Consequently, we propose to restore boundedness by introducing scaling parameters applied at two different scales: one over each sub-layer of the network, and one across the iterations.

Layer-wise residual scaling. Within a single application of $f_\theta(\mathbf{z}; \mathbf{x})$, the residual stream and sub-layer output $f^{\ell}_{\theta^{\ell}}(\mathbf{z}^{\ell-1})$ are weighted by tied scalars (α_1, β_1) shared across all L layers:

$$\mathbf{z}^{\ell} = \alpha_1\, \mathbf{z}^{\ell-1} + \beta_1\, f^{\ell}_{\theta^{\ell}}\left(\mathrm{Norm}_{\mathrm{pre}}\left(\mathbf{z}^{\ell-1}\right)\right), \quad \ell = 1, \ldots, 2L. \tag{2}$$

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 [4]:

$$\mathbf{z}^{0}_{i+1} = \alpha_2\, \mathbf{z}^{2L}_{i} + \beta_2\, \mathbf{x}. \tag{3}$$

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 [49]. 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* $\|f^{\ell}_{\theta^{\ell}}(\mathbf{u})\| \leq c_f$ *for all* ℓ *and input* $\mathbf{u}$*. Let* $0 \leq \alpha_1, \alpha_2 < 1$*, and set*

$$\beta_2 = 1 - \alpha_2 \alpha_1^{2L}, \qquad \beta_1 = \frac{\beta_2 (1 - \alpha_1)}{1 - \alpha_1^{2L}}.$$

Then the fixed-point iterates $\{\mathbf{z}_i^0\}_{i \geq 0}$ *from Equation (3) are bounded, and if* $\mathbf{z}_i^0 \to \mathbf{z}_\infty^0$*, then*

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

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, as shown by [40]. However, boundedness still does not guarantee that the looping to converge to a fixed-point, which we propose to utilize for adaptivity. In the following, we show that there exists a choice of α_2 that satisfies this requirement. Consequently, as empirically shown by [7], the model may become locally contractive during training.

Theorem 2 (Small α_2 implies convergence)**.** *Let* λ_f *be the Lipschitz constant of the* L*-layer model defined by Equation (2), i.e. the map* $\mathbf{z}^0 \mapsto \mathbf{z}^{2L}$*. Then the looped step* $f_\theta(\,\cdot\,;\mathbf{x})$ *of Equations 2–3 is Lipschitz in its first argument with constant* $\alpha_2 \lambda_f$*. In particular, if* $\alpha_2 \lambda_f < 1$*, then* $f_\theta(\,\cdot\,;\mathbf{x})$ *is a contraction and the iteration* $\mathbf{z}_{i+1} = f_\theta(\mathbf{z}_i;\mathbf{x})$ *converges to a unique fixed-point* $\mathbf{z}^\star = f_\theta(\mathbf{z}^*;\mathbf{x})$ *at a linear rate:*

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

The proof is in Section B.2.

While a contractive map is needed for convergence, an excessively contractive one severely limits expressivity [4, 2]. 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 2). In Figure 13, we observe that after training, the distribution of α values widens, with the median very close to the initial point. Intriguingly, these observations are also in line with the common solutions to signal propagation and rank-collapse problems in deep neural networks [47, 59], suggesting a connection between rank-collapse and the signal propagation issue 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 establish 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 of Theorem 2 is not itself guaranteed. 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 characterization of the iteration's behavior. 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 neighborhood 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 B.3.

Theorem 3 shows that a suitable damping factor η eliminates the oscillations while preserving the fixed-points of $f_\theta(\mathbf{z};\mathbf{x})$. We measure convergence to a fixed-point at iteration i as

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

which serves as the halting signal: the iteration stops once r_i falls below a tolerance τ. To choose η adaptively at inference time, we use a patience mechanism that decreases η whenever this residual stops improving. We track the smallest residual observed so far, $r^\star = \min_{j \leq i} r_j$. A geometric decay $\eta \leftarrow \gamma\,\eta$ with $\gamma \in (0,1)$ is applied to the step-size after P consecutive iterations with no improvement in the residuals. The full procedure is given in Algorithm 2, which is based on the implementation provided by [46].

2.4. 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) [27]. Let $\mathbf{J} = \frac{\partial f_\theta}{\partial \mathbf{z}}(\mathbf{z}^\star; \mathbf{x})$ and $\mathbf{P} = \frac{\partial f_\theta}{\partial \theta}(\mathbf{z}^\star; \mathbf{x})$ denote the Jacobians of f_θ at the fixed-point with respect to the state and the parameters, respectively. Following the implicit function theorem we can write the gradient w.r.t. the parameters of the model as [4]:

$$\frac{d\mathbf{z}^\star}{d\theta} \;=\; (\mathbf{I} - \mathbf{J})^{-1}\,\mathbf{P}. \tag{4}$$

The Neumann series $(\mathbf{I} - \mathbf{J})^{-1} = \sum_{j \geq 0} \mathbf{J}^j$ converges, assuming $f_\theta(\mathbf{z}; \mathbf{x})$ is contractive. Therefore, truncating the series at depth k yields the estimate:

$$\frac{d\mathbf{z}^\star}{d\theta} \;\approx\; \textstyle\sum_{j=0}^{k-1} \mathbf{J}^j\,\mathbf{P}, \tag{5}$$

which is closely related to Jacobian-free backpropagation, where the full implicit linear solve is replaced by cheaper approximate gradients [22]. The truncation depth k trades off computation against accuracy. The following proposition bounds the resulting error under contractivity.

Proposition 1 (Exponential decay of truncated-BPTT error)**.** *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 $k \geq 0$,*

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

The proof is in Section B.4. An essentially equivalent result appears in the proof of Theorem 2 of [27].

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 [61] and TRM [37] 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.

2.5. The fixed-point reasoning model

So far, we introduced modifications to improve signal propagation in Looped Transformers (Section 2.1) without sacrificing stability (Section 2.2). These modifications yield fixed-point iterations that can be made non-oscillatory (Section 2.3) and trainable through truncated-BPTT (Section 2.4), making adaptivity through fixed-points reliable. We assemble these components into FPRM, summarized in Algorithm 1 and illustrated in Figure 3. The result is a Looped Transformer that iterates until its hidden state converges, spending compute proportional to each input's difficulty—the adaptivity our halting mechanism was designed to provide. Since we observed no improvements in our experiments by using the hierarchical structure of [61], we opt for a classic looped architecture [14] instead. An overview of the framework is available in Algorithm 1. In Section D we provide further details about FPRM.

Algorithm 1 FPRM training loop with truncated BPTT and deep supervision

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

```
z ← z_0
while FPOPT.CONT() do                          ▷ outer loop
    for k = 1, ..., K do                       ▷ BPTT window
        z̃_k ← f_θ(z; x)
        z ← FPOPT.STEP(z, z̃_k)
    end for
    ŷ ← h_φ(z)                                 ▷ deep supervision
    L ← CROSSENTROPY(ŷ, y)
    MODELOPT.BACKWARD(L)
    z ← detach(z)
end while
```

During training, the model performs looping in windows of k iterations, with k being a hyperparameter, which determines the truncated-BPTT value. During inference, we set $k = 1$. At each forward pass, the fixed-point optimizer introduced in Algorithm 2 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 [61, 37]. To truncate the computation graph between deep supervision steps during training, we detach the state $\mathbf{z}$ from the graph and stop when the optimizer detects fixed-points. This happens when the residual falls below the tolerance, or the step-size becomes too small.

3. Experiments

In this section, we provide an experimental analysis of FPRM against other looped reasoning models. Our implementation builds on the public TRM codebase [37], where we also adopt the training procedure involving deep supervision. We provide the additional experimental details in Section I. We first summarize the benchmarks, then evaluate puzzle performance, adaptive compute, and depth-induced signal propagation.

3.1. Datasets

We evaluate on four families of reasoning benchmarks. Sudoku-Extreme and Maze-Hard test symbolic search on hard 9×9 Sudoku puzzles and 30×30 shortest-path mazes from [61]. ARC-1 and ARC-2 [13] test few-example grid-

rule induction, reported with exact pass@2 accuracy. Finally, the A_5 and S_5 state-tracking tasks [45] test out-of-distribution length generalization for permutation composition: models train up to length $k = 32$ and are evaluated up to $k = 128$. Additional dataset details are provided in **??**.

3.2. Puzzle tasks

We first evaluate FPRM on puzzle problems, namely the Sudoku-Extreme, Maze-Hard [61], ARC-1, and ARC-2 [13] benchmarks. These benchmarks were designed to test whether latent recurrent reasoning models can solve symbolic search problems from limited supervision. In the following, we discuss the experimental results. For more information about the baselines, we refer the reader to Section A.

Table 1 demonstrates the effectiveness of FPRM as the best performing model with 7M parameters on Sudoku-Extreme, Maze-Hard, and ARC-1, with performance on par with TRM on ARC-2. On Sudoku-Extreme, FPRM improves upon even larger models. It is worth noting that these results are achieved without the breadth-search fixed-point method proposed by [35], which is orthogonal to FPRM's modifications. On Maze-Hard, FPRM underperforms compared to the larger Attractor model with ˜$\times 4$ more parameters. However, we note that we were not able to reproduce the results reported in the paper [21].

On ARC-2, FPRM performs similarly to the publicly available TRM checkpoints, but underperforms compared to the TRM results reported in [37] and the URM baseline, which has roughly twice as many parameters. However, considering the recent trend, we emphasize that the ARC benchmarks appear to be much more sensitive to parameter count compared to the other puzzle tasks considered in this section [55, 34]. Therefore, we note that the comparison may not be fair in this case.

Given that FPRM performs on par with or better than the hierarchical baselines that use post-norm on the tasks from Table 1, we hypothesize that the hierarchy might be alleviating signal propagation issues in these models. However, if this hypothesis holds, hierarchy's benefit appears limited: in Figure 6, both models improve as effective layer grows, but the performance gap widens in favor of FPRM, and TRM's performance stays below FPRM at matched effective layers.

Moreover, in Table 3, we attempt to investigate the effect of introducing our architectural modifications to the Transformer layers used in TRM on its performance on Sudoku-Extreme. We find the proposed modifications to have a detrimental impact on the performance of the models, which we attribute to, among other things, hyperparameter optimization and a careful redesigning of the looping.

3.3. Adaptivity

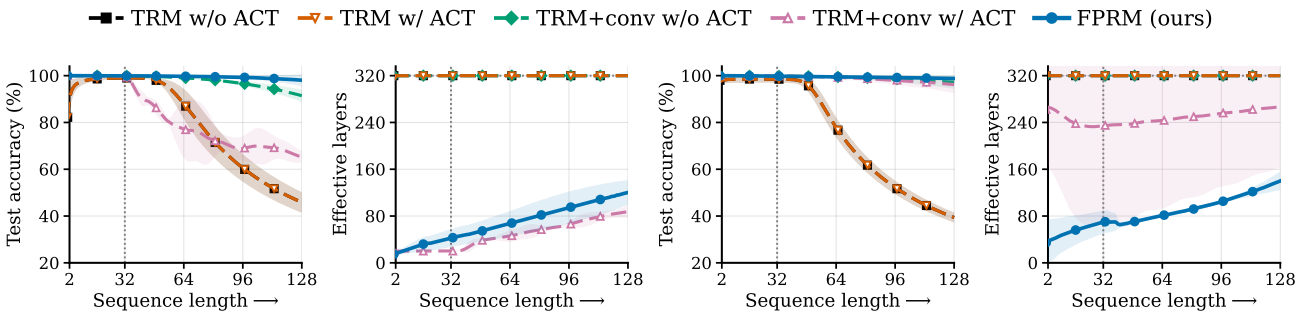
In this section, we demonstrate the adaptivity of FPRM, and compare it to TRM. We use three benchmarks: Sudoku-Extreme from Section 3.2 and state-tracking benchmarks A_5 and S_5 introduced in [45]. Each benchmark provides an intuitive measure of difficulty. For Sudoku-Extreme, this is the number of empty cells, an established proxy for difficulty [51]. For state-tracking, it is the sequence length.

State tracking. As shown in Figure 4, plain TRM fails to extrapolate: at length 128, TRM and TRM with ACT enabled both obtain $45.8\% \pm 3.9\%$ on A_5 and $39.4\% \pm 1.9\%$ on S_5. Adding a causal 1D convolution layer substantially improves length generalization, reaching $91.4\% \pm 2.3\%$ on A_5 and $97.2\% \pm 2.5\%$ on S_5. This modification is not part of the original TRM, but matches the task structure: group composition is a local left-to-right scan, $s_i = s_{i-1} \cdot g_i$, and causal convolution provides a shared translation-equivariant primitive for this operation.

However, 1D convolution does not make ACT reliable. On A_5, TRM+conv+ACT scales compute with sequence length but drops to $65.3\% \pm 2.2\%$, well below TRM+conv without ACT. On S_5, it reaches $96.2\% \pm 3.2\%$, but uses substantially more compute than FPRM; moreover, only a few seeds learn to adapt, while the others exhaust the full compute budget. In contrast, FPRM scales compute smoothly with sequence length while maintaining high accuracy, reaching $98.1\% \pm 2.2\%$ on A_5 and $98.8\% \pm 0.9\%$ on S_5 at length 128.

Sudoku-Extreme. We compare the performance and efficiency of the halting mechanisms in FPRM and TRM on Sudoku puzzles of varying difficulty, measured by the number of empty cells (Figure 5). The sample counts per difficulty level are shown in Figure 9. We observe that halting mechanisms in both TRM and FPRM adapt the number of loops, i.e., inference compute, to the sample difficulty. In contrast, the default behavior of TRM during inference (deactivated ACT) does not adapt the compute to the task difficulty. Even when we enable ACT with TRM, FPRM proves to be more efficient and accurate.

Compute–accuracy trade-off in fixed-point halting. The previous experiments show that FPRM adapts its compute to task difficulty. We now examine how this adaptation can be controlled. In Figure 12, we demonstrate the effect of the step-size decay rate (γ) and maximum patience (P) from Algorithm 2 on the halting time and performance at test-time. Max. inference-time budget was set to be the same (70k effective layers) for all experiments. We observe that larger decay rates improve the performance. Moreover, we observe that maximum patience has minimal impact on the performance, with its impact completely disappearing



(a) Accuracy, A_5 *(b)* Adaptivity, A_5 *(c)* Accuracy, S_5 *(d)* Adaptivity, S_5

Figure 4. **Length generalization and adaptive compute as a function of sequence length.** Shaded bands show 95% confidence intervals over seeds. The vertical dotted line marks the training length 32. The matched compute budget is 320 effective layers.

Table 1. Test accuracy on Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2. For each task, the **best overall** result is bold-face, and the best result with 7M parameters is underlined. The results denoted by † are reproduced using public checkpoints. The ARC results for URM, denoted by ‡, are only reported for Pass@1.

Model	# params.	Single Loop (No Hier.)	Sudoku-Ext. Pass@1	Maze-Hard Pass@1	ARC-1 Pass@2	ARC-2 Pass@2
Our reproduction attempt						
Attractor Model	27M	✗	71.4%	–	–	–
TRM	7M	✗	72.6%	79.0%	40.0%†	6.2%†
Reported						
HRM	27M	✗	55.0%	74.5%	40.3%	5.0%
Attractor Model	27M	✗	91.4%	**93.1%**	–	–
URM	14M	✗	77.6%	–	**≥53.8%**‡	**≥16.0%**‡
TRM	7M	✗	74.7%	85.3%	44.6%	7.8%
EqR	7M	✗	93.0%	–	–	–
Attractor Model	7M	✗	54.3%	46.7%	–	–
FPRM	7M	✓	**94.2%**	87.0%	47.5%	6.2%

for larger decay rates. On the other hand, the increased performance comes at the cost of reaching much deeper effective layers before halting. The largest decay rate that achieves the best performance also exhausts the max. inference budget.

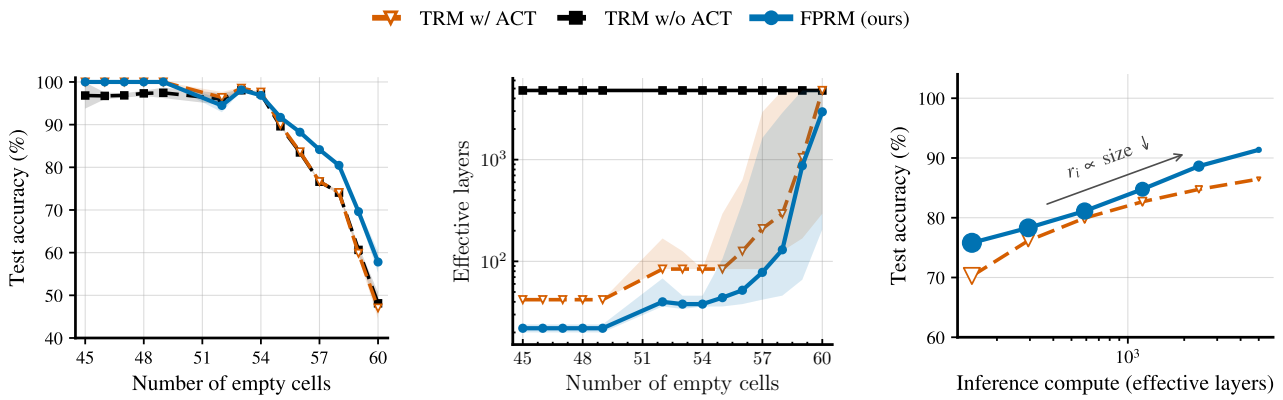
These trends follow from how γ and P control the step size η from Algorithm 2. A decay rate closer to 1 reduces η only slightly once the residual stops improving. The state therefore keeps evolving, halting is deferred, and the model reaches deeper effective layers. The accuracy gain follows directly from these additional iterations. Patience P sets only when a decay occurs, not its magnitude, so its effect is minor; as γ approaches 1 each decay becomes negligible and the two patience curves coincide. The decay rate thus controls the compute-accuracy trade-off directly: a larger γ spends more compute and yields higher accuracy, while P offers only secondary control that vanishes as compute saturates.

3.4. Depth-induced signal propagation issues

Here, we aim to investigate the role of pre-norm with residual scaling in mitigating the depth-induced signal propagation issues in looped models. For a comprehensive investigation, we approach the problem from three angles: **(1)** *trainability*, where we show that pre-norm with residual scaling keeps activations bounded and enables stable training at large depth (Section 3.4.1); **(2)** *depth utilization*, where we show that boundedness alone is not sufficient, and that pre-norm with residual scaling additionally improves depth utilization in FPRM [59] (Section 3.4.2); and **(3)** *residual scaling*, where we analyze the training dynamics of the scaling parameters (Section H.3).

3.4.1. Boundedness of activation norms and trainability

As noted in Section 2.1, the normalization scheme governs a trade-off between activation stability and signal propagation,



(a) Accuracy vs. difficulty

(b) Inference compute vs. difficulty

Figure 5. **FPRM achieves (a) better accuracy, while (b) adapting more efficiently to the task difficulty.** Difficulty is measured by the number of empty cells in the Sudoku grid. The max. compute budget is matched across models (4788 effective layers). From (b): effective layers are reported as medians with 25th–75th percentiles bands. The default behavior of TRM is without ACT at inference time (in black), which exhausts the max. budget for all sample difficulties.

Figure 6. **Test-time scaling lowers residuals while improving accuracy.** Both models run until the matched compute budget is exhausted. FPRM achieves higher accuracy across the entire range. Marker size denotes residue.

which sharpens with depth. Post-norm keeps activations bounded but suffers from signal propagation issues. Pre-norm enables better signal propagation but lets activations grow exponentially. To isolate this trade-off, we adopt the Looped Transformer framework [54, 14, 38] with a fixed number of effective layers. This controls for dynamic halting, which can reduce the effective layers, as introduced in Section 2.

In Figure 2b we show the norm of the final activations of the residual branch of Looped Transformer at initialization. From the figure, we conclude that the magnitude of the activations grows exponentially in the case of pre-norm Looped Transformer. In contrast, when using post-norm or pre-norm with residual scaling, the architecture benefits from bounded activations. In Figure 2a we give evidence that boundedness is a prerequisite for reaching deeper effective layers. Pre-norm architecture with its unbounded activations diverges and can't reach deep effective layers, while bounded architectures ensure trainability. Therefore, a Looped Transformer with pre-norm and without residual scaling cannot achieve deep effective layers, highlighting one aspect of the importance of residual scaling in pre-norm.

3.4.2. Pre-norm with residual scaling enables higher depth utilization

An important measure for evaluating signal propagation issues in deep neural networks is depth utilization [59]. Depth utilization measures whether all layers contribute meaningfully to the model. It is commonly probed by removing layers from a trained model and measuring the effect on performance at test-time. In models with signal propagation problems, removing deeper layers (closer to the output) usually does not impact the performance significantly, highlighting that these models fail to utilize the compute due to signal propagation issues. However, in a looped model there is no fixed stack of layers to remove, since the same block is applied to the previous latent representations. Consequently, we cannot perform the same experiment here. We therefore probe the same property from the opposite direction: instead of *removing computation* and measuring degradation, we *add computation* and measure improvement. We do so in two complementary regimes. In the first, using the state-tracking task, we ask whether the model can be trained to use the depth that a harder task requires. In the second, using the Sudoku-Extreme task, we ask whether a trained model can convert depth beyond its training regime into further gains at test time.

State-tracking. In Section 3.4.1, we discussed the necessity of using normalization layers that ensure the boundedness of the activations, which is required for the stable training. However, bounded activations are not a sufficient condition to ensure effective utilization of large depth, once it is reached. We demonstrate this in Figure 2a using a state-tracking task, where increasing difficulty, corresponding to longer sequences, requires training at greater depths [46]. We train a Looped Transformer model with its number of loops (effective layer) tied to the train sequence length. We choose the sequence length from the set $\{8, 16, 32, 64, 128, 256, 512\}$. We plot the maximum sequence length solved with $> 90\%$ accuracy against effective

layer. Because we also at test-time match effective layer to the training sequence length, a model with no signal-propagation bottleneck should solve exactly the length its depth permits, tracing the identity line $y = x$ (Figure 3 in [46]). We observe that this behavior is only present in the model equipped with pre-norm and residual scaling. We interpret this observation as strong evidence for improved trainability in Looped Transformers with pre-norm and residual scaling.

Sudoku-Extreme. Compared to the previous experiment on the state-tracking task, here we run each model far beyond its trained depth, trying to detect the point where more compute no longer translates into improvements at test-time. We expect the performance of a model with fewer signal propagation issues to saturate later and with more effective layers, indicating that the model is capable of reaching deeper effective layer. On the other hand, the performance of a model bottlenecked by signal propagation problems is expected to saturate early, indicating that it cannot convert the extra compute into better predictions. For this experiment, we focus on two variants of the FPRM model, one with pre-norm and residual scaling, and the other with post-norm, both trained on the Sudoku-Extreme task.

In Figure 11a, we demonstrate that scaling test-time compute improves performance for both types of normalization. Furthermore, considering Figure 11b, we also observe that the majority of the improvement comes before at least half of the samples halt. However, there are clear differences between the effective layer of the two normalization methods in Figure 11a, as the pre-norm model's performance saturates at almost twice as much compute, indicating improved signal propagation through depth.

The advantage of pre-norm with residual scaling also extends to the cross-model comparison in Figure 6, between FPRM (pre-norm with residual scaling) and TRM (post-norm). In this inference-time scaling experiment, all samples are run to a varied maximum looping compute budget. For TRM, compute is scaled through deep supervision steps, which we find optimal relative to scaling L- and H-steps (see Section G). FPRM outperforms TRM across a range of effective-layer depths reached, with the gap widening at higher compute budgets, consistent with FPRM making better use of its depth.

4. Discussion

Adaptivity and scaling. FPRM adapts its inference compute to problem difficulty on both state tracking and Sudoku-Extreme (Figures 4, 5). This is notable for state tracking, where Transformers lack the built-in sequential bias of state-space models or RNNs [46, 45]. The results also suggest a simple compute–accuracy control knob: because halting is governed by the fixed-point optimizer rather than a learned ACT module, the decay rate γ and patience P select different points on the test-time compute Pareto frontier.

Hierarchy and signal propagation. FPRM matches or improves on hierarchical baselines without using hierarchical looping. Since TRM and HRM are built from post-norm Transformer blocks, while Section 3.4 shows that post-norm looped models underutilize large effective depth, one possible explanation is that hierarchy partly alleviates signal-propagation difficulties. A theoretical account of hierarchy through this optimization lens remains an interesting direction for future work.

Limitations. Following prior work on end-to-end latent reasoning [38, 20, 61, 37, 17, 18], we evaluate only on algorithmic tasks. It remains open whether the compositional behavior seen here transfers to natural language and other domains. We also instantiate FPRM with Transformers, although the fixed-point framework could in principle be combined with CNNs, MLPs, or state-space models.

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 larger effective layer provided by looping. These modifications allow FPRM to outperform hierarchical baselines of similar size, such as HRM and TRM, on common symbolic reasoning benchmarks. We show that on state tracking and Sudoku-Extreme, FPRM is able to adapt its compute to the difficulty of the 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.

Acknowledgments and Disclosure of Funding

We thank Felix Sarnthein, Albert Catalan-Tatjer, Jonas Geiping, Philipp Nazari, Carl Richardson, and Nouha Dziri for the helpful discussions and comments. Alexander Theus, Vera Milovanović, and Shlomo Libo Feigin are supported by the Max Planck ETH Center for Learning Systems. Vera Milovanović and Antonio Orvieto are supported by the AI2050 program at Schmidt Sciences. Antonio Orvieto, T. Konstantin Rusch, and Sajad Movahedi acknowledge the financial support of the Hector Foundation.

References

[1] Donald G. M. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, 1965.

[2] 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 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.

[3] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron C. Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *CoRR*, abs/2507.10524, 2025.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 688–699, 2019.

[5] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing equilibrium models by jacobian regularization. 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 554–565. PMLR, 2021.

[6] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *CoRR*, abs/2107.05407, 2021.

[7] 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.

[8] David Belanger and Andrew McCallum. Structured prediction energy networks. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, JMLR Workshop and Conference Proceedings, pages 983–992. JMLR.org, 2016.

[9] David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 429–439. PMLR, 2017.

[10] Hugh Blayney, Alvaro Arroyo, Johan S. Obando-Ceron, Pablo Samuel Castro, Aaron C. Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *CoRR*, abs/2604.11791, 2026.

[11] Aleksandar Botev, Soham De, Samuel L. Smith, Anushan Fernando, George-Cristian Muraru, Ruba Haroun, Leonard Berrada, Razvan Pascanu, Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Sertan Girgin, Olivier Bachem, Alek Andreev, Kathleen Kenealy, Thomas Mesnard, Cassidy Hardin, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Armand Joulin, Noah Fiedel, Evan Senter, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, David Budden, Arnaud Doucet, Sharad Vikram, Adam Paszke, Trevor Gale, Sebastian Borgeaud, Charlie Chen, Andy Brock, Antonia Paterson, Jenny Brennan, Meg Risdal, Raj Gundluru, Nesh Devanathan, Paul Mooney, Nilay Chauhan, Phil Culliton, Luiz Gustavo Martins, Elisa Bandy, David Huntsperger, Glenn Cameron, Arthur Zucker, Tris Warkentin, Ludovic Peran, Minh Giang, Zoubin Ghahramani, Clément Farabet, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, Yee Whye Teh, and Nando de Frietas. Recurrentgemma: Moving past transformers for efficient open language models. *CoRR*, abs/2404.07839, 2024.

[12] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, 1965.

[13] François Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. ARC prize 2024: Technical report, 2024.

[14] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning*

Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[15] 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.

[16] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023.

[17] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, pages 5570–5582. PMLR, 2022.

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

[19] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019.

[20] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, volume 2025, pages 14502–14520, 2025.

[21] Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning, 2026.

[22] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley J. Osher, and Wotao Yin. JFB: jacobian-free backpropagation for implicit networks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6648–6656. AAAI Press, 2022.

[23] Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model. *CoRR*, abs/2512.14693, 2025.

[24] Renee Ge, Qianli Liao, and Tomaso A. Poggio. Hierarchical reasoning models: Perspectives and misconceptions. *CoRR*, abs/2510.00355, 2025.

[25] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38:41340–41391, 2026.

[26] Zhengyang Geng and J. Zico Kolter. Torchdeq: A library for deep equilibrium models, 2023.

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

[28] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 11398–11442. PMLR, 2023.

[29] Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-based transformers are scalable learners and thinkers. *CoRR*, abs/2507.02092, 2025.

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

[31] Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nat.*, 645(8081):633–638, 2025.

[32] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024.

[33] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*

2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[34] Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. ARC is a vision problem! *CoRR*, abs/2511.14761, 2025.

[35] Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026.

[36] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *CoRR*, abs/2602.11451, 2026.

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

[38] Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[39] 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.

[40] 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.

[41] Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

[42] Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *CoRR*, abs/2604.07822, 2026.

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

[44] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[45] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 35492–35506. PMLR / OpenReview.net, 2024.

[46] Sajad Movahedi, Felix Sarnthein, Nicola Muca Cirone, and Antonio Orvieto. Fixed-point rnns: From diagonal to dense in a few iterations, 2025.

[47] 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.

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

[49] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Çaglar Gülçehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 26670–26698. PMLR, 2023.

[50] Vardhan Palod, Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Performative thinking? the brittle correlation between cot length and problem complexity. *CoRR*, abs/2509.07339, 2025.

[51] Marcelo O. R. Prates and Luís C. Lamb. Problem solving at the edge of chaos: Entropy, puzzles and the sudoku freezing transition. In Lefteri H. Tsoukalas, Éric Grégoire, and Miltiadis Alamaniotis, editors, *IEEE 30th International Conference on Tools with Artificial*

Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece, pages 686–693. IEEE, 2018.

[52] Zirui Ren and Ziming Liu. Are your reasoning models reasoning or guessing? A mechanistic analysis of hierarchical reasoning models, 2026.

[53] Tim Salimans and Jonathan Ho. Should ebms model the energy or the score?, 2021. Energy-Based Models Workshop, ICLR 2021.

[54] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[55] Wen-Jie Shu, Xuerui Qiu, Rui-Jie Zhu, Harold Haodong Chen, Yexin Liu, and Harry Yang. Loopvit: Scaling visual ARC with looped transformers. *CoRR*, abs/2602.02156, 2026.

[56] 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.

[57] Shixiang Song, He Li, Zitong Wang, Boyi Zeng, Feichen Song, Yixuan Wang, Zhiqin John Xu, Ziwei He, and Zhouhan Lin. Adaponderlm: Gated pondering language models with token-wise adaptive depth. *CoRR*, abs/2603.01914, 2026.

[58] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

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

Algorithm 2 Fixed-point optimizer FPOPT: one damped step with patience-based decay

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 $p \leq 0$ **and** $r > \tau$ **then**
      $\eta \leftarrow \gamma\, \eta,\ p \leftarrow P$ ▷ decay η
    end if
  end if
  return $\mathbf{z}$, r
end procedure

A. Background and Related Works

Looped models. Looped architectures address the flexibility requirement of reasoning models by decoupling the number of effective layers 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 [38] and Universal Transformers [14]; more recent work shows that Looped Transformers can improve length generalization, learn algorithmic structure, and approximate much deeper untied models on reasoning tasks [64, 20, 54, 39, 25, 28, 42]. Recent recursive reasoning models such as HRM [61] and TRM [37] instantiate this principle in compact architectures. Using small networks, 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 and a slow-looping component. URM [23] follows-up on these works and shows that by using short-conv, performance can be further improved. However, as we describe in this paper, previous works leave two central design axes open to improvement: First, *how should a looped model decide how many iterations to run on each input at test-time?* FPRM addresses this through fixed-point halting. Second, *how to make the model utilize its large effective layer?* FPRM does so by switching post-norm to pre-norm and adding residual scaling. Importantly, FPRM achieves better performance than TRM and HRM without using an hierarchical structure for the loops.

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 [14, 6]. More recent works extend this idea by picking an adaptive depth for each token [3, 57]. Such mechanisms can, in principle, allocate more computation to harder instances, but they all add a separate learned decision process on top of the recurrent computation itself, and this process is hard to optimize because the halting decision is discrete rather than continuous. Moreover, analyses of HRM find that ACT does not always scale the inference compute with the actual difficulty of the input [52, 24], which can more generally also be the case with CoT [50] — the limitation we also observe and address by instead halting when the iteration reaches a fixed-point.

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 perspective is also supported by recent mechanistic analyses of looped language models, which observe that recurrent trajectories often converge to fixed-points, suggesting that weight-tied looped models can naturally implement an implicit fixed-point computation [10]. This view is most extensively developed in Deep Equilibrium Models [4], which replace a finite stack of layers by the equilibrium point of a

weight-tied transformation. However, DEQ models usually find the fixed-point via a quasi-Newton approach such Broyden's method [12, 4] or Anderson acceleration [1, 26] rather than fixed-point iterations, which makes them different from looped models. Moreover, DEQ models are difficult to optimize [5, 2, 37] , which we address by our proposed architectural changes. Concurrent to our work, Attractor Models [21] frame the iterations of TRM as a root-finding problem similar to DEQ, which they solve with Anderson acceleration. Additionally, they propose to optionally include a larger separate network to "guess" the initial latent. In another concurrent work is Equilibrium Reasoners (EqR) [35], which also adopt a fixed-point perspective and show that it is possible to scale the compute not only depth-wise, but also by making multiple initial guesses during training and inference time, favoring attractors with a wide basin. Our contributions are largely orthogonal: we target the signal-propagation issues that limit trainable depth, replacing post-norm with pre-norm and residual scaling, and damping the iteration to suppress oscillation around the fixed point. Moreover, in contrast to these works we do not use the hierarchical looping of TRM.

Score-based methods. Diffusion Models [33, 58] and Energy-Based Models (EBMs) [44] provide another way of iterative computation. An EBM learns a scalar energy function, the core idea being that the energy represents the negative log of the unnormalized density, and solutions correspond to low energy states. EBMs generate a prediction either by descending the energy landscape [8, 9] or by sampling from it with MCMC methods [19]. In comparison, diffusion models learn the score of the underlying distribution, which can be viewed as the gradient of a time-dependent energy function, thus the two are closely connected [53, 16]. While diffusion models typically integrate an SDE over a fixed time interval and in this sense are not adaptive, most relevant to our work are energy-based reasoning models [17, 18, 29], where the halting criterion is convergence to a local minimum of the energy landscape. Compared to these methods, we model the generative process via fixed-point iterations of a learned operator, as opposed to descending through a learned energy function. Learning the operator directly enables the theoretical analysis of our architectural modifications in Section 2.

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 [15, 47, 59]. 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 [25, 61, 37]. Indeed, in our own experiments a pre-norm looped model without further modification diverges in activation norm as the iteration count grows, and fails to train at the large loop counts where this growth is most severe (Figure 2). 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 simplify our notation by omitting the pre-norm layer in Equation (2). We start by unrolling the computation across the L layers within a single fixed-point iteration. By recursively applying the layer update, we obtain

$$
\begin{aligned}
\mathbf{z}_i^{2L} &= \alpha_1 \cdot \mathbf{z}_i^{2L-1} + \beta_1 \cdot f_{\theta^{2L}}^{2L}\left(\mathbf{z}_i^{2L-1}\right) \\
&= \alpha_1^2 \cdot \mathbf{z}_i^{2L-2} + \alpha_1\beta_1 \cdot f_{\theta^{2L-1}}^{2L-1}\left(\mathbf{z}_i^{2L-2}\right) + \beta_1 \cdot f_{\theta^{2L}}^{2L}\left(\mathbf{z}_i^{2L-1}\right) \\
&= \alpha_1^3 \cdot \mathbf{z}_i^{2L-3} + \alpha_1^2\beta_1 \cdot f_{\theta^{2L-2}}^{2L-2}\left(\mathbf{z}_i^{2L-3}\right) + \alpha_1\beta_1 \cdot f_{\theta^{2L-1}}^{2L-1}\left(\mathbf{z}_i^{2L-2}\right) + \beta_1 \cdot f_{\theta^{2L}}^{2L}\left(\mathbf{z}_i^{2L-1}\right) \\
&\;\;\vdots \\
&= \alpha_1^{2L} \cdot \mathbf{z}_i^{0} + \beta_1 \cdot \left(\sum_{j=0}^{2L-1} \alpha_1^{j} \cdot f_{\theta^{2L-j}}^{2L-j}\left(\mathbf{z}_i^{2L-j-1}\right)\right).
\end{aligned}
$$

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

$$\begin{aligned}\mathbf{z}_{i+1}^0 &= \alpha_2 \cdot \mathbf{z}_i^{2L} + \beta_2 \cdot \mathbf{x} \\ &= \alpha_2 \cdot \left(\alpha_1^{2L} \cdot \mathbf{z}_i^0 + \beta_1 \cdot \sum_{j=0}^{2L-1} \alpha_1^j \cdot f_{\theta^{2L-j}}^{2L-j}\left(\mathbf{z}_i^{2L-j-1}\right) \right) + \beta_2 \cdot \mathbf{x} \\ &= \alpha_2 \alpha_1^{2L} \cdot \mathbf{z}_i^0 + \alpha_2 \beta_1 \cdot \left(\sum_{j=0}^{2L-1} \alpha_1^j \cdot f_{\theta^{2L-j}}^{2L-j}\left(\mathbf{z}_i^{2L-j-1}\right) \right) + \beta_2 \cdot \mathbf{x}.\end{aligned}$$

For compactness, define $\rho = \alpha_2 \alpha_1^{2L}$ and $\mathbf{s}_i = \sum_{j=0}^{2L-1} \alpha_1^j \cdot f_{\theta^{2L-j}}^{2L-j}\left(\mathbf{z}_i^{2L-j-1}\right)$. 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

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

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

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

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

$$\limsup_{i \to \infty} \left\|\mathbf{z}_i^0\right\| \leq \frac{\beta_2}{1-\rho} \|\mathbf{x}\| + \frac{\alpha_2 \beta_1}{1-\rho} \left(\frac{1-\alpha_1^{2L}}{1-\alpha_1} \right) c_f.$$

Substituting back $\rho = \alpha_2 \alpha_1^{2L}$, we obtain

$$\limsup_{i \to \infty} \left\|\mathbf{z}_i^0\right\| \leq \frac{\beta_2}{1-\alpha_2 \alpha_1^{2L}} \|\mathbf{x}\| + \frac{\alpha_2 \beta_1 \left(1-\alpha_1^{2L}\right)}{\left(1-\alpha_2 \alpha_1^{2L}\right)\left(1-\alpha_1\right)} c_f.$$

We now set $\beta_2 = 1 - \alpha_2 \alpha_1^{2L}$. This makes the coefficient of $\|\mathbf{x}\|$ equal to 1. Furthermore, setting $\beta_1 = \frac{\beta_2 (1-\alpha_1)}{\left(1-\alpha_1^{2L}\right)}$ makes the coefficient of c_f equal to α_2. Therefore,

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

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

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

This completes the proof. □

B.2. Contractive mapping

Proof. We first show that $f_\theta(.;\mathbf{x})$ is a contraction with respect to $\mathbf{z}$. For any $\mathbf{z}, \mathbf{z}'$, using the Lipschitzness of the L-layer model, we have

$$\begin{aligned}\|f_\theta(\mathbf{z};\mathbf{x}) - f_\theta(\mathbf{z}';\mathbf{x})\| &\leq \lambda_f \|(\alpha_2 \cdot \mathbf{z} + \beta_2 \cdot \mathbf{x}) - (\alpha_2 \cdot \mathbf{z}' + \beta_2 \cdot \mathbf{x})\| \\ &= \alpha_2 \lambda_f \|\mathbf{z} - \mathbf{z}'\|.\end{aligned}$$

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

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

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

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

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

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 $\mathbf{z}_1 = f_\theta(\mathbf{z}_0; \mathbf{x})$, we obtain

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

This completes the proof. □

B.3. Mitigating oscillation through damping

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. □

B.4. Error of truncated-BPTT

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

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

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. A Toy Failure Mode for Recurrent Post-norm

In this section, we provide a toy example to showcase a failure mode of post-norm in a small setting. We take random $\mathbf{x},\mathbf{y} \in \mathbb{R}^{n\times d}$ to be the input-output pair of sequences, with sequence length $n = 100$ and hidden-size $d = 2$, and set $\mathbf{z}_0^0 = \mathbf{x}$. Let $f_\theta(\mathbf{z};\mathbf{x})$ be a neural network with a single sub-layer $f_{\theta^1}^1(\mathbf{z})$ defined as:

$$f_{\theta^1}^1(\mathbf{z}) = \mathbf{z}W(\mathbf{w}),$$

where we define the rank-one map:

$$W(\mathbf{w}) = \mathbf{w}\mathbf{1}^\top \in \mathbb{R}^{2\times 2}, \qquad \mathbf{w} = (w_1, w_2)^\top,$$

with $\theta^1 = \mathbf{w}$.

The **post-norm recurrence** is defined as

$$\mathbf{z}_{i+1}^0 = \mathrm{Norm}_{\mathrm{post}}\left(\mathbf{z}_i^0 + f_{\theta^1}\left(\mathbf{z}_i^0\right)\right),$$

while the **pre-norm recurrence** is defined as

$$\mathbf{z}_{i+1}^0 = (1-\beta)\cdot\mathbf{z}_i^0 + \beta\cdot f_{\theta^1}\left(\mathrm{Norm}_{\mathrm{post}}\left(\mathbf{z}_i^0\right)\right), \qquad \beta = \frac{1}{2}.$$

Figure 7 gives a minimal version of the normalization trade-off discussed in the main text based on this toy model. For both models we sweep a 200×200 grid over $\mathbf{w} \in \mathcal{G} = [-5,5]^2$ and plot

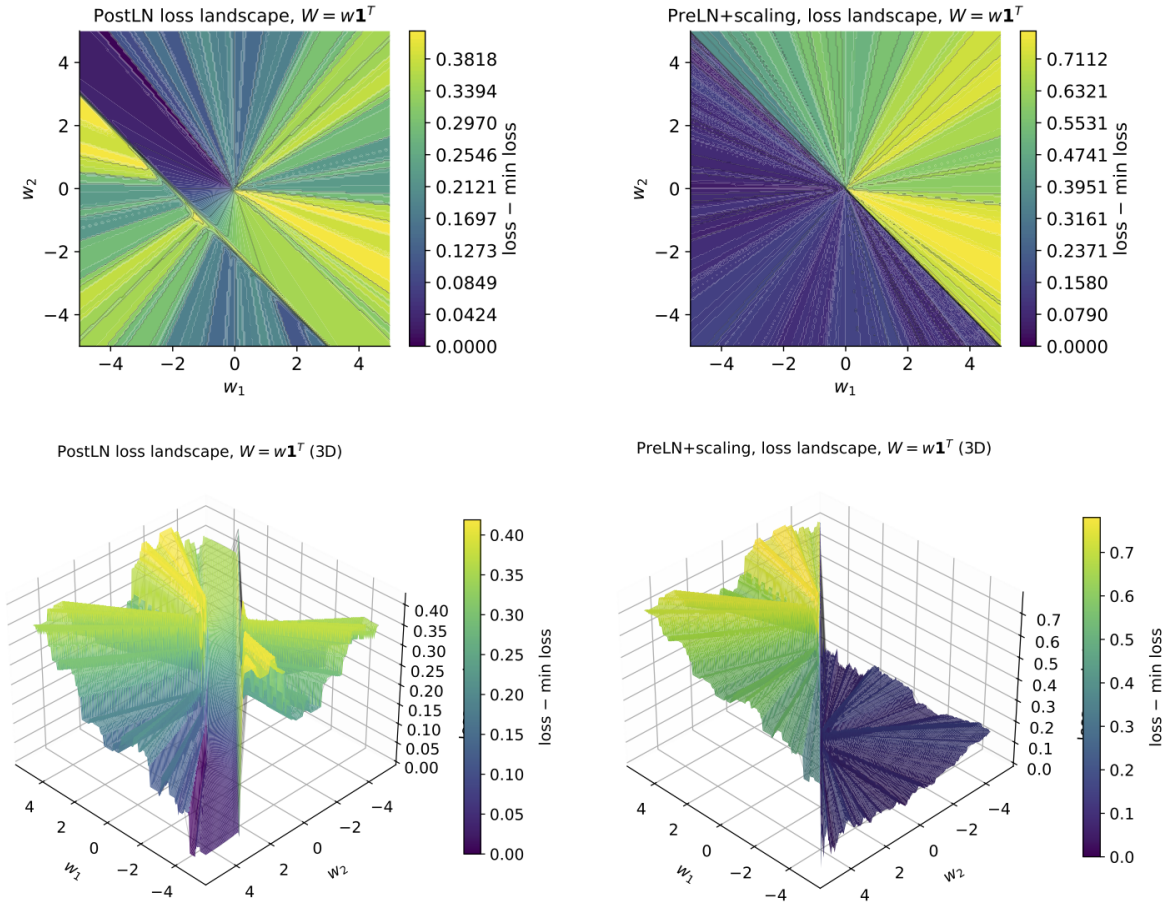


Figure 7. Landscape visualization for the setup proposed in Section C.

$$\Delta\mathcal{L}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) - \min_{\mathbf{v}\in\mathcal{G}} \mathcal{L}(\mathbf{v}),$$

where we define the loss as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{nd} \left\| \mathbf{z}_{20}^0(\mathbf{w}) - \mathbf{y} \right\|_F^2,$$

i.e., at the 20th effective layer.

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 onto 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 layers.

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 (Figure 2b). With the scaled update, each row satisfies

$$\|\mathbf{z}_{i+1}^0\|_2 \leq (1-\beta)\|\mathbf{z}_i^0\|_2 + \beta.$$

So the toy dynamics remain bounded while preserving a live residual stream. The broader low-loss region in Figure 7 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

Fixed-point solver. Let $\mathbf{z}_i \in \mathbb{R}^{B\times T\times d}$ denote the ith latent representation (with B denoting the batch index, T the sequence index, and d the hidden size), and $\mathbf{z}_{i+1}$ the next latent representation. We index the bth batch dimension as $\mathbf{z}_i[b]$. Convergence is measured per sample $\mathbf{z}_i[b]$ by the relative L_∞ norm of the residual,

$$\mathbf{r}_i[b] = \frac{\|\mathbf{z}_{i+1}[b] - \mathbf{z}_i[b]\|_\infty}{\|\mathbf{z}_{i+1}[b]\|_\infty + \epsilon} \in \mathbb{R}.$$

A sample is declared converged when $r_i[b] < \tau$. In practice, we set τ to 0.1. But we observe that for a reasonably small choice of τ, the model is not sensitive to this value. Two safeguards bound the loop: **(1)** a hard cap on the number of iterations, and **(2)** early termination if the adaptive step size collapses below a minimum.

Deep supervision. We adopt a similar deep supervision mechanism as HRM [61] and TRM [37]. Let T_{sup} denote the deep supervision interval. After every T_{sup} iterations, the intermediate activations of the model are decoded through the output head, the loss is computed, and truncated-BPTT is performed through the k latest iterations. Then, the computation graph is detached from the previous step. For each sequence, this process is continued until the fixed-point of the input is reached. The number of backward passes per forward pass is therefore $\lceil k/T_{\text{sup}} \rceil$. In our model, we set $T_{\text{sup}} = k$, while in TRM and HRM, T_{sup} is usually set to a larger number. However, we observe that in practice, FPRM performs a smaller number of forward and backward passes during training compared to TRM, lowering the training cost.

Depth-wise convolutions. Depth-wise convolutions have proven effective in improving the performance of looped models, at a small time and parameter complexity [55, 23]. Therefore, in FPRM we apply depth-wise convolutions on the latent representations at the beginning of each loop, which we find to be most effective. An overview of FPRM is available in Figure 3. We consider both 1D and 2D convolutions, and we find the 2D variant to be more effective at 2-dimensional tasks such as Sudoku and ARC, while the 1D variant is essential in state-tracking. However, as observed in Table 3, depth-wise convolutions seem to have a detrimental impact on the performance of TRM.

E. Description of Figure 1

In this figure, we categorize the puzzles into three groups: easy, medium, and hard. The grouping is based on difficulty, which is measured by the number of empty cells in the puzzle. The sample sizes for each difficulty level are balanced and set at around 1000 samples. For FPRM, we mark the halting decision for the entire group based on the residual of the group: if the mean residual is smaller than a pre-determined threshold (set to 0.1), then the model makes the halting decision. For TRM, the halting decision is marked when the ACT module signals halting for more than half of the samples. For the sake of exposition, we exclude the hardest puzzles from the groups, since they fail to halt at the current max. set budget of 10000 effective layers.

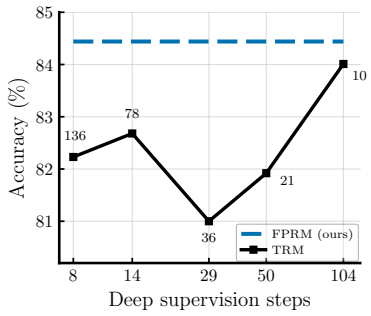


Figure 8. **The optimal way to spend the fixed looping compute is to maximize deep supervision steps.** The numbers next to markers are inner recurrence depths per each deep supervision step. The total depth of effective layers is approximately the same across all configurations of TRM and FPRM on the Sudoku-Extreme task.

F. Fixed-point Residuals and Halting

We provide the test accuracy and the fixed-point residuals achieved by FPRM as a function of effective layer in Figure 10. The residuals for more difficult problems decay at a much slower rate, indicating that they demand more compute. Furthermore, accuracy stops improving at roughly the same effective layer where the residual plateaus, supporting the use of fixed-points as a halting criterion.

G. How to effectively spend loops in TRM?

The default proposed TRM uses 16 deep-supervision steps (outer loops) and variable L- and H-steps (inner loops). The L-steps outnumber the H-steps, typically by about $2\times$. However, other configurations for the number of loops spent for deep supervision vs. inner loops are possible. We test the performance of other configurations with experiments shown in Figure 8. We fix the L-to-H ratio at 2 and vary deep-supervision steps (segments) against per-segment recurrence depth

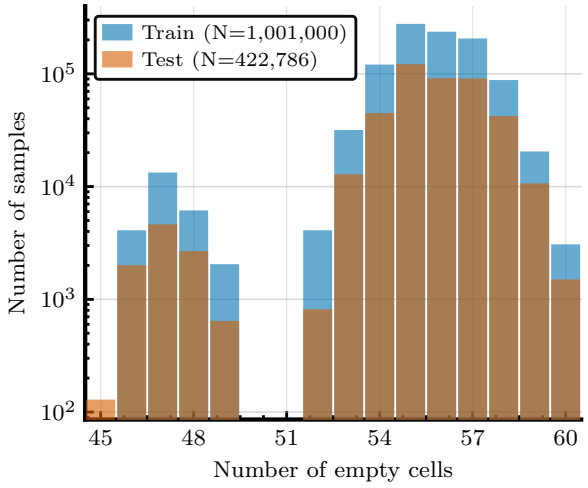


Figure 9. **Sudoku-Extreme dataset is imbalanced.** The number of samples per difficulty level (number of empty cells).

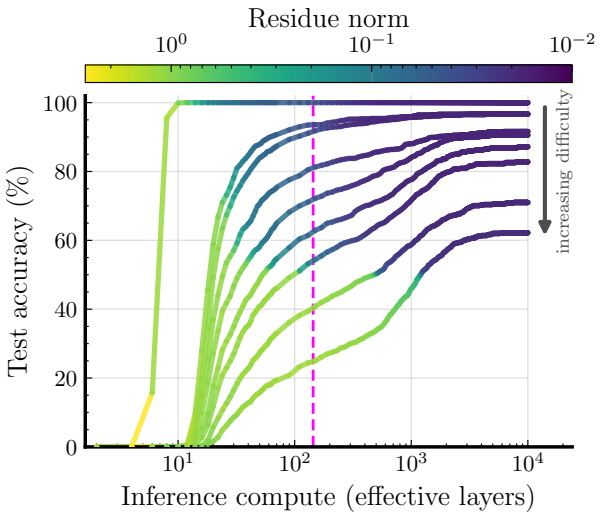


Figure 10. **FPRM allocates more compute to harder problems.** Harder inputs need more iterations before halting and peak beyond the training compute limit (dashed line); color shows residual norm.

(inner loops, shown with numbers next to the black markers in Figure 8). We measure test accuracy as a function of the number of deep-supervision steps, with fixed inference budget at approximately 1040 steps. This budget also matches the max. number of effective layers reached by the baseline FPRM on this task. This isolates how a fixed inference budget is best allocated: toward more outer refinement steps or deeper inner recurrence. Figure 8 shows that the budget is best spent on outer, deep-supervision steps. This matches the finding that, in TRM's post-norm Transformer, the gains from added effective layer depth get smaller compared to FPRM (Figure 6). Fewer effective layers per segment is therefore the better strategy at a fixed compute budget. We adopt it for all experiments where we scale TRM compute (effective layers).

H. Additional Empirical Analyses

H.1. Loop utilization

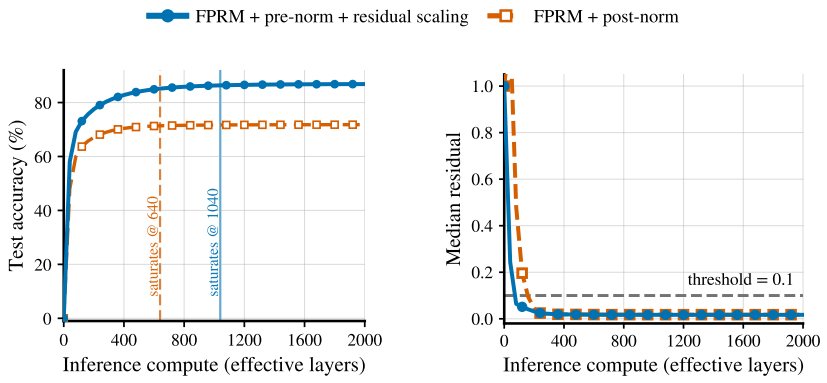
To probe whether improved signal propagation allows FPRM to make better use of additional recurrent depth, we run the model beyond its trained depth and measure where performance saturates. Figure 11 shows that the pre-norm model with residual scaling continues improving until a larger effective depth than the post-norm variant, while both exhibit comparable fixed-point residues. This suggests that the pre-norm variant does not merely converge more slowly, but converts additional iterations into more useful updates.

H.2. Fixed-point optimizer hyperparameters

We additionally study how the fixed-point optimizer controls the compute–accuracy trade-off at inference time. Figure 12 shows test accuracy and effective depth as a function of the decay rate γ for two values of the maximum patience P. Larger decay rates improve accuracy but require more effective layers before halting, whereas the choice of patience has a comparatively small effect.

H.3. The training dynamics of residual scales

While our original motivation for residual scaling was to prevent unbounded activations in pre-norm, we note a parallel relationship between our solution and common solutions to signal propagation problems [59, 47]. Specifically, it has been known that scaling down the output of the sub-layers when introducing them to the residual stream is beneficial to increasing effective layer, which is equivalent to increasing α_1 in Equation (2). Moreover, in Theorem 2, we show that the looping will become more stable for smaller α_2 in Equation (3). In order to investigate the impact of these two parameters, we perform a coarse-grained ablation on the initial value of α_1, α_2 on the Sudoku-Extreme task.



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

Figure 11. **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 residuals. This indicates similar latent-space convergence, with more meaningful updates in the pre-norm variant, resulting in improved performance.

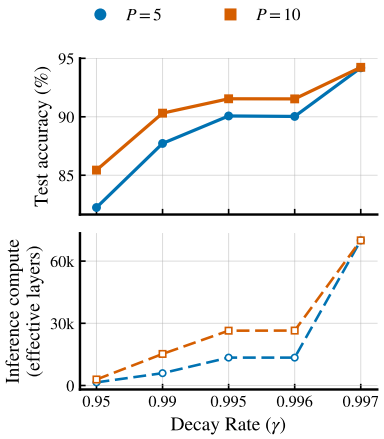


Figure 12. **Decay rate and patience.** Test accuracy and effective layer of FPRM with fixed-point halting as a function of decay rate γ, for maximum-patience $P \in \{5, 10\}$.

Table 2 demonstrates that the best initialization places α_1 at a high value and α_2 at a low value. Interestingly, the α_1 preference matches the common solutions to signal propagation problems: keeping the residual stream dominant. Moreover, the ablation also highlights the importance of having a more contractive mapping at initialization, as our convergence analysis (Theorem 2) requires a sufficiently small α_2 for the loop to reach a fixed-point. On the other hand, comparing the row with the smallest α_2 choice ($\alpha_2 = 0.25$) with the column with the largest α_1 ($\alpha_1 = 0.75$), we observe that increasing α_1 has a slightly higher positive impact on the accuracy than a decreasing α_2. We hypothesize that this is because it is easier for the model to recover from a bad choice of α_2 than α_1, as the gradients for α_1 come from two different sources (the MHA and MLP sub-layers), and thus can be noisier.

Table 2. 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	83.44	78.10	83.24
0.50	84.49	89.05	86.29
0.75	**94.23**	91.41	85.70

In Figure 13, we provide the distribution statistics of the residual scales after training. Interestingly, we observe that while the median of the α_1, α_2 values over channels does not deviate significantly from the initial point, the spread of the distributions widens significantly. In the case of α_1, the widening happens at a much larger scale, covering both very small and very large values. On the other hand, the α_2 distribution remains more concentrated, with the majority of the values actually becoming smaller than the initial point. This can be interpreted as the model learning to become more contractive during training, which is in line with the observations in [7]. Furthermore, this observation also supports our hypothesis that it might be easier for the model to learn the optimal α_2 values than the α_1 values.

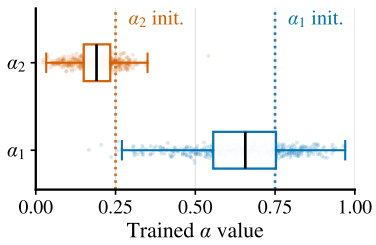


Figure 13. The distribution of the residual scales in FPRM after training on the Sudoku-Extreme dataset.

I. Additional Experimental Details

Weight initialization. It seems that 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 is very little material difference in sequence accuracy after convergence.

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

Table 3. Effect of adding FPRM's architectural modifications to TRM: pre-norm and residual scaling (α_2 only, or both α_1 and α_2), individually and in combination, evaluated with and without the conv2d layer in the TRM core. Each column reports the change in test sequence accuracy on Sudoku-Extreme (%) relative to its own post-norm, no-scale baseline measured in this sweep, with the absolute accuracy shown alongside.

	w/ conv		w/o conv	
Configuration	Δ (%)	Acc. (%)	Δ (%)	Acc. (%)
Original TRM (post-norm, no residual scaling)	—	63.98	—	72.60
+ residual scaling (α_2 only)	-6.71	57.27	-4.44	68.16
+ residual scaling (α_1, α_2)	-4.47	59.51	-12.24	60.36
− post-norm + pre-norm + residual scaling (α_2 only)	-49.00	14.98	-58.87	13.73
− post-norm + pre-norm + residual scaling (α_1, α_2)	-24.52	39.46	$\mathbf{-52.83}$	19.77

Hyperparameters, device specification 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, constant LR after 2 000-step warm-up, EMA enabled (rate 0.999), puzzle-embedding length 16. All models are trained with `AdamW` [**?**].

	TRM	FPRM
Looping structure		
H-cycles	3	–
L-cycles	6	–
H-layers	0	0
L-layers	2	2
n_{back}	$= L$-cycles $+ 1$	6
Halting		
mechanism	ACT	fixed-point
`halt_max_steps`	16	–
`max_iter` (train)	–	12
`max_iter` (eval)	–	35 000
stepsize-decay / patience (eval)	–	0.997 / 10
`fp_thresh`	–	0.1
Block / signal-prop modifications		
norm type	post-norm	pre-norm
residual scaling	✗	✓
α_1, α_2 init	–	0.75, 0.25
conv branch	–	2D-conv (3×3 kernel)
Optimizer		
learning rate	10^{-4}	10^{-3}
weight decay	1.0	10^{-3}
puzzle-emb LR	10^{-4}	10^{-3}
puzzle-emb WD	1.0	10^{-3}

Table 5. Hyperparameters for Maze-Hard experiments (Table 1 of the paper). Shared: trained on `maze-30x30-hard-1k` *without augmentation*, 4×A100-80GB, constant LR after a 2 000-step warm-up, EMA enabled (rate 0.999), puzzle-embedding length 16. FPRM trains for 60 000 epochs (TRM 50 000). FPRM is trained using `Adam-Atan2` [**?**]; TRM is trained using `AdamW`.

	TRM	FPRM
Looping structure		
H-cycles	3	–
L-cycles	4	–
H-layers	0	0
L-layers	2	2
n_{back}	$= L$-cycles $+ 1$	6
Halting		
mechanism	ACT	fixed-point
`halt_max_steps`	16	–
`max_iter` (train)	–	24
`max_iter` (eval)	–	35 000
stepsize-decay / patience (eval)	–	0.996 / 10
`fp_thresh`	–	0.1
Block / signal-prop modifications		
norm type	post-norm	pre-norm
residual scaling	✗	✓
α_1, α_2 init	–	0.75, 0.25
conv branch	–	1D-conv (1×4 kernel)
Optimizer		
learning rate	10^{-4}	10^{-4}
weight decay	1.0	1.0
puzzle-emb LR	10^{-4}	10^{-2}
puzzle-emb WD	1.0	1.0

Table 6. Hyperparameters for state-tracking experiments on A_5 and S_5 (Figure 4 of the paper) and for the Looped Transformer signal-propagation analysis (Figure 2). Shared: 1×A100-80GB, global batch 1024, `Adam-Atan2`, no LR warm-up, EMA disabled, no puzzle embedding (`puzzle_emb_len`=0). Trained at k_{train}=32, evaluated for $k \in [2, 128]$. TRM and FPRM train for 50 epochs; the Looped Transformer analysis (Fig. 2) for 30.

	TRM	FPRM	Looped Transformer (Fig. 2)
Looping structure			
H-cycles	2	–	–
L-cycles	4	–	–
H-layers	0	0	0
L-layers	4	2	2
n_{back}	$= L$-cycles $+ 1$	4	4
Halting			
mechanism	fixed iters or ACT (inference)	fixed-point	fixed iters
`halt_max_steps`	16	–	–
`max_iter`	–	128	$= k_{\text{train}}$
iter. distribution	–	deterministic	deterministic
Block / signal-prop modifications			
norm type	post-norm	pre-norm	sweep†
norm placement	–	none	none
residual scaling	✗	✓	sweep†
α_1, α_2 init	–	0.5, 0.5	sweep†
conv branch	–	1D-conv (1×4 kernel)	–
Optimizer			
learning rate	10^{-4}	10^{-3}	10^{-4}
weight decay	10^{-2}	10^{-2}	10^{-2}

† The Looped Transformer 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 $k_{\text{train}} \in \{8, 16, 32, 64\}$.

Table 7. Hyperparameters for the FPRM ARC-AGI experiments. The two runs share an identical configuration and differ only in the training corpus (ARC1-Concept vs. ARC2-Concept, both with 1000 augmentations per sample). Shared across both: 4×A100-80GB, batch 768, 100 000 epochs, constant LR with no warm-up, EMA enabled (rate 0.999), puzzle-embedding length 16, hidden size 512, 8 heads, MLP expansion 4, RoPE position encodings. Both models are trained with `Adam-Atan2` (β_1=0.9, β_2=0.95).

	FPRM
Looping structure	
H-cycles	–
L-cycles	–
H-layers	0
L-layers	2
n_{back}	6
Halting	
mechanism	fixed-point
`halt_max_steps`	–
`max_iter` (train)	8
`max_iter` (eval)	1000
stepsize / decay / patience (eval)	1.0 / 0.9 / 5
`fp_thresh`	0.1
Block / signal-prop modifications	
norm type	pre-norm
residual scaling	✓
α_1, α_2 init	0.75, 0.25
conv branch	1D-conv (kernel 4)
Optimizer	
learning rate	10^{-3}
weight decay	10^{-2}
puzzle-emb LR	10^{-2}
puzzle-emb WD	1.0