Stability and Generalization in Looped Transformers

Asher Labovich
Brown University, Liquid AI
asherlabovich@gmail.com

Abstract

Looped transformers promise test-time compute scaling by spending more iterations on harder problems, but it remains unclear which architectural choices let them extrapolate to harder problems at test time rather than memorize training-specific solutions. We introduce a fixed-point based framework for analyzing looped architectures along three axes of stability – local attractivity, input-dependence, and geometry – and use it to characterize when fixed-point iteration yields meaningful predictions. Theoretically, we prove that looped networks without recall have countable fixed points and cannot achieve strong input-dependence at any spectral regime, while recall with outer normalization reliably produces a regime in which fixed points are simultaneously locally-attractive, locally smooth in the input, and supported by stable backpropagation. Empirically, we train small looped transformers on chess, sudoku, and prefix-sums and find that downstream performance tracks the framework's predictions across tasks and architectural configurations. We additionally introduce internal recall, a novel recall placement variant allowing an identity residual path, and exploit its contrasting fixed-point behavior to test our framework's predictions in a controlled setting.

1 Introduction

Recent progress in reasoning with large language models has come largely from chain-of-thought (CoT) methods, in which models produce a hidden scratchpad of tokens before responding [Wei et al., 2023, Guo et al., 2025]. CoT has several structural limitations: it requires autoregressive decoding that scales poorly in latency and energy, it forces intermediate computation to reside in discrete tokens, and its reasoning depth is bounded by the token budget the model was trained on. Looped transformers offer an alternative path to test-time compute scaling by instead training a single weight-tied network whose iteration count can, in principle, be scaled with problem difficulty. Recent work has shown that such models can match or exceed much larger fixed-depth transformers on reasoning benchmarks [Geiping et al., 2025, Wang et al., 2025, Jolicoeur-Martineau, 2025], and that weight-tying itself induces useful inductive biases towards algorithmic reasoning [Saunshi et al., 2024, 2025]. Beyond matching fixed-depth performance with few parameters, looped transformers offer a capability unavailable to fixed-depth models: a model that has learned a stable algorithm may be able to extrapolate beyond its training iteration depth, solving problems harder than any it was trained on simply by iterating more.

Whether looped transformers actually achieve this extrapolation remains unclear. Empirical work has converged on two architectural ingredients – recall (conditioning each iterate on the original input) and outer normalization – as apparently necessary for stable looped computation [Bansal et al., 2022, Anil et al., 2022, Geiping et al., 2025]. However, neither is theoretically justified, and generalization results across tasks and scales remain inconsistent [Yang et al., 2024]. It is not obvious why recall should be *necessary* rather than merely helpful, why outer normalization (typically disliked in fixed-depth transformers due to gradient instability [Xiong et al., 2020]) becomes beneficial in looped settings, or how these ingredients interact to enable stable computation.

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

This paper[1] provides a unified account of why these choices matter, framed around a fixed-point analysis of the looped architecture. We argue that a looped model can be trusted to run for arbitrarily many iterations without degrading (avoiding the problem known as "overthinking") only if it has fixed points which are locally attractive, input-dependent, and geometrically robust in parameter space – three properties we call the "axes of stability". We show that each architectural choice – recall existence, recall placement, and outer normalization – affects a distinct subset of the axes in ways that directly predict downstream performance.

Our key theoretical result is that recall combined with outer normalization reliably yields a regime in which all three axes are simultaneously satisfied. We validate this framework empirically by training single-layer looped transformers on chess, sudoku, and prefix-sums across a grid of normalization and recall choices, and find that downstream performance – both on the training distribution and harder out of distribution (OOD) problems – tracks the framework's predictions across tasks and configurations. Over the course of this analysis, we also introduce *internal recall*, a novel placement variant whose narrow stability region without outer normalization provides direct empirical support for the geometry axis of our framework.

2 Related Work

In this section, we contextualize our work on looped transformers among the many different architecture, training, and benchmarking choices made in previous work.

Inductive Biases. Recent work finds that weight-tied architectures exhibit useful inductive biases, trading memorization capacity for reasoning ability [Saunshi et al., 2024, 2025]. In addition, Merrill and Sabharwal [2025] show that looped transformers with padding can efficiently solve parallelizable problems in a manner unavailable to fixed-depth transformers. However, these results consider models run for a fixed loop count, and none address whether additional test-time iterations can solve problems harder than those seen during training.

Extrapolation in Looped Models. Bansal et al. [2022] first study extrapolation in CNNs via looped iterations, finding that recall is empirically necessary to avoid overthinking, and introduce the progressive loss mechanism used in this work. Geiping et al. [2025] scale this to a large transformer, finding outer normalization to be additionally necessary for strong results, though performance plateaus beyond the number of loops seen in training. Yang et al. [2024] corroborate that stable, recall looped transformers can nonetheless fail to generalize to OOD problems. Across all three, recall is consistently identified as necessary but never theoretically justified, and generalization[2] results are mixed.

Other Related Work. Deep Equilibrium Models [Bai et al., 2019] reframe looped networks as fixed-point finders, directly motivating our stability analysis. Unlike DEQ, which treats convergence as a *design* objective enforced by the solver, we study fixed points as a property of the model itself, analyzing when and why it arises based on architectural choices. Adaptive computation methods [Graves, 2017, Dehghani et al., 2018, Banino et al., 2021] attempt to *reduce* loops for computational efficiency rather than increase them, which makes them complementary to our study. Wang et al. [2025] and Jolicoeur-Martineau [2025] propose training looped models with "deep supervision" and find they outperform compute-matched models on reasoning tasks. However, since they only evaluate on in-distribution tasks on training depth, it remains unclear to what extent these gains reflect the inductive biases of weight-tied architectures versus an ability to extrapolate with additional iterations.

3 Preliminaries

3.1 Definitions

This section provides a short description of looped architectures from a mathematical perspective, and is useful for the propositions proven in the upcoming sections. The framework and most of our theoretical results apply to smooth looped models generally. Transformer-specific structure enters

[1] Code available at `https://github.com/ashlab11/generalization`

[2] In the context of this paper we use "generalization" to refer specifically to the ability of looped models to extrapolate beyond the number of iterations seen during training to solve OOD problems.

in Section 5 through our analysis of normalization, and via the token-based transversality condition in Proposition 6.

Definition 3.1 (Looped Network)**.** Given parameters $\theta \in \mathbb{R}^p$, embedding dimension d, and sequence length L, a *looped network* is a discrete dynamical system on the state space $\mathbb{R}^{d\times L}$ defined by the recurrence

$$x_{t+1} = f_\theta\left(x_t,\, \{x_i\}_{i=0}^{t-1}\right) \qquad t = 1, \ldots, T-1$$

where f_θ is some network parameterized by θ and $x_t \in \mathbb{R}^{d\times L}$ is the hidden state at time t. We call T the *iteration depth*. The network is initialized with $x_1 = f_\theta(e, x_0)$ where e is a free parameter often chosen to be zero, x_0, or Gaussian noise. This formulation of looped networks is quite broad; the following definitions narrow this to the two cases studied in this paper.

Definition 3.2 (Autonomous Network)**.** A looped network is *autonomous* if f_θ depends only on the current state, i.e.,

$$x_{t+1} = f_\theta(x_t).$$

We choose the term "autonomous" to mimic autonomous systems in differential equations theory, which arise when the rate of change depends only on the current state.

Definition 3.3 (Recall Network)**.** A looped network is a *recall network* if f_θ depends on only the current state and the initial input, i.e.,

$$x_{t+1} = f_\theta(x_t,\, x_0). \qquad t = 1, \ldots, T-1$$

Recall models are initialized at x_1 equivalently to Definition 3.1.

A recall network can be thought of as "middle ground" between autonomous and the more general looped models, taking in a second parameter (unlike autonomous models) but limiting that parameter to only the initial state (unlike general looped models). Figure 1 compares autonomous and recall models visually.

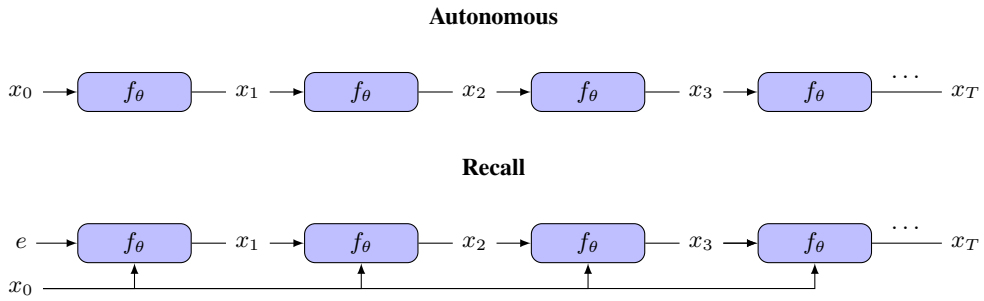


Figure 1: Comparison of autonomous and recall networks. In the former, the network only depends on the previous state x_{t-1}; in the latter, it also depends on the initial state x_0.

Definition 3.4 (Outer Normalization)**.** A looped network uses *outer normalization* if some normalizing function is applied to the output of each iteration:

$$x_{t+1} = \phi_\theta\left(f_\theta\left(x_t, \{x_i\}_{i=0}^{t-1}\right)\right)$$

where f_θ may itself contain arbitrary internal normalization.

Remark 1. Practitioners have considerable flexibility in their choice of ϕ; we consider RMSNorm and GRU normalization. When ϕ is RMSNorm or LayerNorm, this matches what the standard transformer literature calls *post-norm*, and we use that term in this setting. In fixed-depth transformers, pre-norm – applying normalization only inside f_θ – is more common due to gradient instability problems with post-norm [Xiong et al., 2020]; we show in Section 5.2 that recall architectures create a stability regime which uniquely benefits from outer normalization. We also note that post-norm constrains each token to a compact set, which we exploit in Section 4.

Smoothness Assumption. Separately from our definitions, we also note that throughout this paper, we assume all models are *smooth*, which can be guaranteed by using smooth activations only[3].

[3]Many modern transformers use GeLU or SwiGLU activations, both of which are smooth. In addition, RMSNorm/LayerNorm are both smooth as long as the denominator contains an ϵ term.

3.2 Axes of Stability

Throughout this paper, we discuss architectural choices that improve or harm the ability of looped transformers to learn algorithms rather than memorize solutions. To make this analysis precise, we first need a vocabulary for the distinct ways such a model can succeed or fail. We emphasize here that our analysis focuses on fixed points of the looped computation. In practice, we want looped models we can trust to run for any number of iterations without overthinking or degrading. Only fixed points offer this guarantee, so we take reaching a correct fixed point as our criterion for success.

For each task, we separate easy data (matching the training distribution) from hard data (out-of-distribution along the generalization axis of interest – longer sequences, harder instances). We identify three properties ("axes") of a looped model that jointly determine its success on easy and/or hard data: **local attractivity, input-dependence, and geometry**. We describe each in turn below.

Locally attractive. A looped model is locally attractive if repeated iteration near a fixed point converges to that fixed point rather than diverging or cycling. In practice, this means running the model for more iterations moves it toward a stable answer rather than away from one. Local attractivity is the most basic dynamical requirement for looped computation. Due to floating-point perturbations, function iteration will rarely if ever reach a fixed point if the fixed point is not locally attractive.

Input-dependence. It is trivial to build a model with perfect attraction by ensuring a contraction map and relying on the Banach fixed point theorem. Such a map is, of course, useless as an algorithm – its output is the same regardless of input. A useful looped model must therefore have fixed points that vary meaningfully with the input. Architectures differ in *how* they achieve this, and as we will see, the mechanism matters as much as whether input-dependence exists at all.

Geometry. Even among models that have parameter regimes of fixed-point attraction and input-dependence, the *shape* of that regime matters. Stability regions that are highly direction-dependent – narrow slivers rather than round, broad neighborhoods – force stable configurations into a narrow band of the parameter space, making training outcomes fragile and hyperparameter-sensitive. We refer to this directional bias as anisotropy, and show in Section 5.1 and Appendix B.2 that internal and external recall without outer normalization differ sharply along this axis.

Having defined the axes of stability, we now examine how specific architectural choices – recall existence, recall placement, and outer normalization – succeed or fail along each axis, and its effect on downstream performance.

4 Why Autonomous Networks Fail

Prior work finds that autonomous looped networks often achieve low accuracy and degrade under repeated iteration [Bansal et al., 2022], with networks lacking input injection producing "essentially random or unpredictable" outputs [Yang et al., 2024]. In this section, we show this failure is *structural*, not merely an artifact of poor optimization or architecture choice: under mild assumptions, autonomous models exhibit extremely weak input-dependence across all spectral regimes.

Assumption 1. Consider function $g(x_t, \theta) : \mathbb{R}^{d\times L} \times \mathbb{R}^p \to \mathbb{R}^{d\times L} := f_\theta(x_t) - x_t$ chosen so that $g(x_t, \theta) = 0 \Leftrightarrow x_t$ is a fixed point of f_θ. We assume that at all points where $g(x_t, \theta) = 0$, $\begin{bmatrix} \frac{\partial g(x_t,\theta)}{\partial x_t} & \frac{\partial g(x_t,\theta)}{\partial \theta} \end{bmatrix}$ spans $\mathbb{R}^{d\times L}$ (i.e. g is transversal to $\{0\}$[4]).

This assumption is quite mild in practice, failing only when *all* $p+dL$ variables conspire to lose rank at a fixed point, which is an extreme coincidence. Examples of the weakness of this assumption can be found in Appendix A.1. Despite its weakness, this assumption helps us prove powerful results for autonomous networks.

Proposition 1 (Restrictions on the Jacobian of f_θ at fixed points)**.** *For almost all parameter vectors $\theta \in \mathbb{R}^p$, $J_{f_\theta}(x^*) = \frac{\partial f_\theta(x^*)}{\partial x^*}$ contains no eigenvalues equal to 1 for any fixed points x^*.*

Proof. Since g is smooth and transversal to $\{0\}$ (Assumption 1), we can apply the transversality theorem ([Guillemin and Pollack, 1974, p. 68]) to get that for almost all θ, g_θ is transversal to 0. By

[4]Transversality is defined as follows: a smooth function $f : N \to M$ is transverse to a manifold T if at all points in $f^{-1}(T)$, the Jacobian of f combined with the tangent space of T generates the tangent space of M at the image point.

definition of a transversal, we have that J_{g_θ} together with $T_0\{0\} = \{0\}$ span $\mathbb{R}^{d\times L}$ and thus J_{g_θ} is non-singular at $g_\theta^{-1}(0)$. However,

$$J_{g_\theta} = \frac{\partial g_\theta(x)}{\partial x} = \frac{\partial f_\theta(x) - x}{\partial x} = J_{f_\theta} - I \tag{1}$$

(1) implies that $J_{f_\theta} = J_{g_\theta} + I$. Since J_{g_θ} is non-singular, J_{f_θ} has no eigenvalues of 1 at $g_\theta^{-1}(0)$ (the fixed points of f_θ). □

With this, we can characterize the fixed-point structure of autonomous networks.

Proposition 2 (Fixed points are a dimension-zero manifold)**.** *Let $f_\theta : \mathbb{R}^{d\times L} \to \mathbb{R}^{d\times L}$ be an autonomous looped network. Let $S = \{x^* : f_\theta(x^*) = x^*\}$ be the set of fixed points of f_θ. Then, for almost all parameterizations θ, S is a dimension-zero manifold.*

Proof. From Proposition 1, we know that for almost all θ, $J_{g_\theta} = J_{f_\theta(x)-x}$ is non-singular at points x^* where $g_\theta(x^*) = 0$. Thus, 0 is a regular value and thus $g_\theta^{-1}(0)$ is a manifold of dimension $dL - dL = 0$ [Milnor, 1965, p. 11]. Since $g_\theta^{-1}(0)$ represents the set of all fixed points of f_θ the fixed points are a manifold of dimension 0 (countable). □

Proposition 2 bounds the number of fixed points an autonomous network can have: for almost all parameters, the set is countable, and finite when x is confined to a compact set (as under post-norm; see Remark 1). This limits the input-dependence of an autonomous model to simple basin selection, where the network simply chooses *which* of its predetermined isolated fixed points to output given an input. Recall networks, which condition each step on x_0, are not subject to this constraint. We note that this argument extends naturally to *joint fixed points* — points where the output of a final readout head w_θ is unchanged by one application of f_θ even if the hidden state is not – with an analogous transversality assumption; see Appendix A.2 for details.

The previous proposition argues that autonomous networks have severely limited input-dependence in theory. The following proposition shows that even within this limited regime, the gradient dynamics make basin selection hard to learn.

Proposition 3 (Input-Gradient Vanishing of Autonomous Looped Networks)**.** *Let f_θ be an autonomous looped network. Suppose x_t converges to x^* with $\rho(J_{f_\theta}) < 1$. Then $\frac{\mathrm{d}x_T}{\mathrm{d}x_0}$ converges to 0 exponentially as $T \to \infty$.*

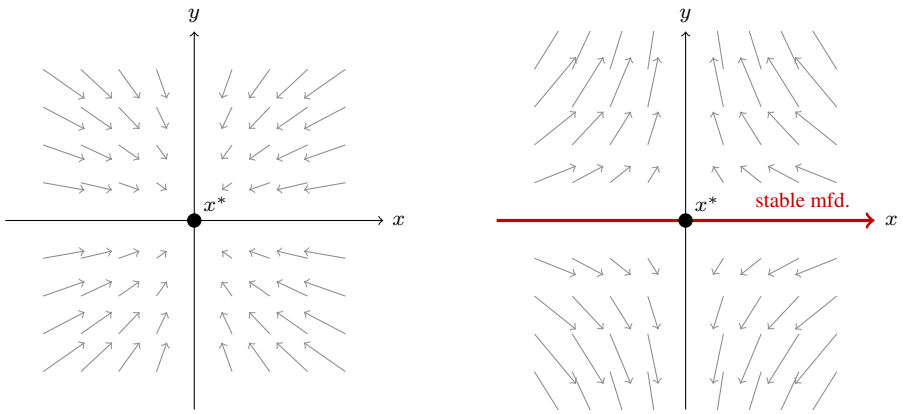
Proof. See Appendix A.3 for details. □

This representation provides a stronger understanding of the dynamics of the gradient at a fixed point of f_θ. In Appendix A.4, we show that the behavior of $\frac{\partial x_T}{\partial x_0}$ depends strongly on the value of $\rho(J_{f_\theta}(x^*))$, the spectral radius of the Jacobian of f_θ at the fixed point. In particular, we find three regimes, none of which provides useful training signal. If $\rho(J) < 1$, x^* is locally attractive; thus Proposition 3 holds, and the input-gradient vanishes exponentially with T. If $\rho(J) > 1$, the fixed point is not locally attractive and thus not practically reached with fixed-point iteration in the first place. Finally, if $\rho(J) = 1$, fixed-point iteration faces very slow convergence with possible exploding parameter gradients. Figure 2 illustrates the contrast between the cases where $\rho(J) < 1$ versus $\rho(J) > 1$: when all eigenvalues have modulus less than 1, every trajectory converges to x^*; when any eigenvalue exceeds 1 in magnitude, there exists a direction which is locally repelling, and trajectories almost always escape the neighborhood.

Together with Proposition 2, this paints a difficult picture for autonomous networks: their expressible outputs are countable, and reaching those fixed points trades off against the quality of the training signal. This helps explain the empirical observations of Bansal et al. [2022], where autonomous networks fail to reach fixed points and substantially degrade under repeated iterations.

By defining the "best" models as those that achieve top hard accuracy *at 500 iterations*, we naturally choose those that overthink the least, avoiding the failure mode noted in Bansal et al. [2022][5]. But the bottleneck remains: avoiding overt degradation under extra iterations is easier than having

[5]Many of our runs *did* see this overthinking, but some norm/lr combinations did not.



(a) $|\lambda_x|, |\lambda_y| < 1$ (b) $|\lambda_x| < 1 < |\lambda_y|$

Figure 2: Phase portraits for the map $(x, y) \mapsto (\lambda_x x,\ \lambda_y y)$, corresponding to a 2×2 matrix with eigenvectors along the coordinate axes. **(a)** Both eigenvalues satisfy $|\lambda| < 1$: every trajectory converges to x^*. This is *input-gradient vanishing.* **(b)** One eigenvalue exceeds 1 in magnitude: only the stable manifold (the x-axis, shown in red) converges to x^*; *all other trajectories diverge*.

locally attractive, input-dependent fixed points. Consistent with this, our autonomous models (Appendix B.6, "none" rows) achieve non-trivial val accuracy in some configurations, but still exhibit substantial val-to-hard degradation and never approach the hard accuracy of recall variants on the same task.

Notably, prefix-sums is the only problem on which autonomous networks achieve 0% accuracy across all configurations. It is also the only problem whose input dimensionality changes between training and testing: the train distribution is 32-bit sequences and the test distribution is 512-bit. Since the set of fixed points depends on the state dimension, basin-selection learned at training dimension largely cannot transfer to test dimension. Sudoku and chess preserve input dimensionality across train and test, so basin-selection can at least partially transfer – consistent with the non-trivial val and hard accuracies we observe there.

Recall networks aim to fix this by providing additional paths from $x_T \to x_0$ by adding it as an input at every step. However, simply introducing recall is not sufficient for fixed-point stability – the *placement* of recall, as well as the presence of outer normalization, critically determines where on the axes of stability a given recall architecture lies. We analyze this in the following section.

5 Stability and Generalization of Recall Networks

Although the formal definition of recall is simple – merely conditioning each iterate on the original input – practitioners have considerable flexibility in exactly how this conditioning is applied. This flexibility is compounded in models with multiple sublayers, where a nontrivial additional question arises: *where* in the network is the initial state injected? Below, we define two architecturally distinct recall variants, one standard in the literature and one novel, whose fixed-point structures differ in ways that directly determine stability and downstream performance. Our formal definitions assume a **two-sublayer** model for generality (e.g. Attention and MLP); Figure 3 gives a visual explanation of both variants under such a model. All functions depend on θ (omitted for brevity), and ϕ_i acts as an outer normalization function that may be present or absent depending on architecture choice.

Definition 5.1 (External Recall)**.** An *external* recall network is one such that for $t > 1$,

$$\begin{aligned} x_{t+1} &= \phi_2(z_t + h_2(z_t)) \\ z_t &= \phi_1(g(x_t, x_0) + h_1(g(x_t, x_0))) \end{aligned}$$

External recall is the most common architectural choice in the literature, as in Bansal et al. [2022], Geiping et al. [2025], Wang et al. [2025], Jolicoeur-Martineau [2025]. Common choices for g are

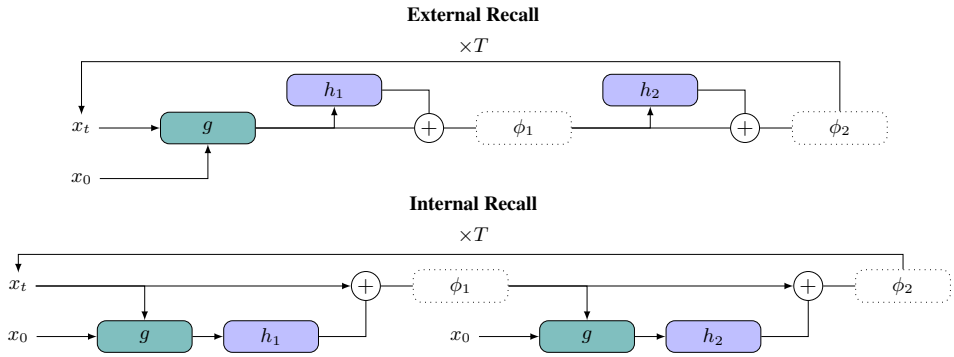


Figure 3: Comparison of two-layer external and internal recall architectures. In external recall, the recalled representation replaces the residual stream; in internal recall, the current state remains on the residual path and recall only affects the update.

pointwise addition [Wang et al., 2025, Jolicoeur-Martineau, 2025] or a linear combination with trainable matrices W_1, W_2 [Bansal et al., 2022, Geiping et al., 2025]. We note that recall only enters in the *first* sublayer, with all later sublayers acting as pseudo-autonomous layers.

Definition 5.2 (Internal Recall)**.** An *internal* recall network is one such that for $t > 1$,

$$x_{t+1} = \phi_2(z_t + h_2(g(z_t, x_0)))$$
$$z_t = \phi_1(x_t + h_1(g(x_t, x_0)))$$

In internal recall, the recall function g never enters into the residual directly, only ever affecting the update to the residual. For a two-sublayer model, internal and external recall use the same parameter count and nearly identical compute (internal applies g once more per iteration, but g is generally inexpensive). This allows for controlled comparisons between the two architectures in the remainder of Section 5.

5.1 Recall Without Outer Normalization

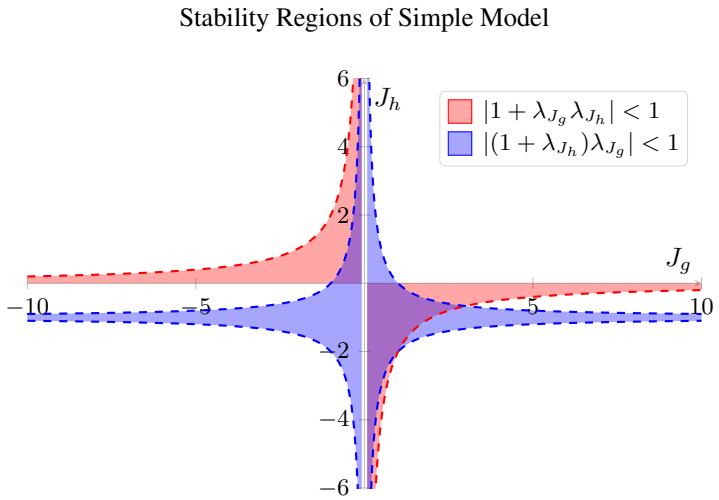


Figure 4: Stability regions for a simplistic (single-layer, equivalent eigenvectors) model of external (blue) and internal (red) recall models. In this figure, we shade the regions for which the two eigenvalues produce a stable fixed point.

In Proposition 4, we discuss the stability of these models in the scenario where outer normalization is not applied; i.e. $\phi_i = I$.

Proposition 4 (Local attractivity conditions for a recall model without outer normalization)**.** *Let x^* be a fixed point of a recall model for a fixed input x_0.*
External Recall: *Define*

$$M_{ext} = \left(I + \frac{\mathrm{d}h_2(z^*)}{\mathrm{d}z^*}\right)\left(I + \frac{\mathrm{d}h_1(g(x^*, x_0))}{\mathrm{d}g(x^*, x_0)}\right)\frac{\partial g(x^*, x_0)}{\partial x^*}$$

Then, $\rho(M_{ext}) < 1$ is a sufficient condition for x^* to be a locally attracting fixed point, while $\rho(M_{ext}) \leq 1$ is a necessary one.
Internal Recall: *Define*

$$M_{int} = \left(I + \frac{\mathrm{d}h_2(g(z^*, x_0))}{\mathrm{d}g(z^*, x_0)}\frac{\partial g(z^*, x_0)}{\partial z^*}\right)\left(I + \frac{\mathrm{d}h_1(g(x^*, x_0))}{\mathrm{d}g(x^*, x_0)}\frac{\partial g(x^*, x_0)}{\partial x^*}\right)$$

Then, $\rho(M_{int}) < 1$ is a sufficient condition for x^* to be a locally attracting fixed point, while $\rho(M_{int}) \leq 1$ is a necessary one.

Proof. See Appendix A.5. □

We note that while recall models do indeed add an additional path from $x_t \to x_0$, improving input-dependence, without outer normalization they are harder to stabilize than it may first appear. In particular, the *geometry* of the locally-attractive, input-dependent fixed-point regime changes drastically depending on whether the chosen architecture uses internal or external recall. Figure 4 visualizes this for a simplified single-layer model in which the eigenvectors of $J_h = \frac{\mathrm{d}h(g(x^*,x_0))}{\mathrm{d}g(x^*,x_0)}$ and $J_g = \frac{\partial g(x^*,x_0)}{\partial x^*}$ align, so that the stability condition reduces to a simple multiplication of scalar values.

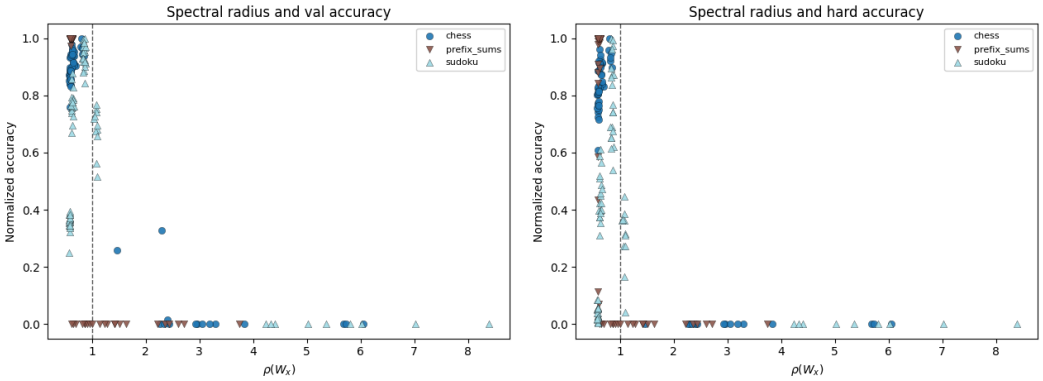


Figure 5: External unnormalized recall validation and hard (normalized by task) accuracy mapped against $\rho(\frac{\partial g(x^*,x_0)}{\partial x^*})$. External recall achieves substantially worse performance on all tasks when $\rho(\frac{\partial g(x^*,x_0)}{\partial x^*}) > 1$, as suggested by Proposition 4.

In the case where $\frac{\partial g(x^*,x_0)}{\partial x^*}$ is strongly contractive, $M_{\textbf{ext}}$ allows substantial flexibility on the other terms while retaining stability. The same is not true for internal recall, whose identity shifts require a careful balancing of $\frac{\partial h_2(g(z^*,x_0))}{\partial g(z^*,x_0)}$ and $\frac{\partial h_1(g(x^*,x_0))}{\partial g(x^*,x_0)}$ to retain stability even under a contractive recall Jacobian. Appendix B.2 conducts a simplified projection-based experiment on the single-layer eigenvector-matched model and finds that internal recall's stable region is substantially more anisotropic than external's. Working directly with models trained on sudoku data, we find in Appendix B.3 that internal models with pre-norm never converge to a fixed point, whereas external models can. Together, these theoretical and empirical results suggest that while internal recall may provide expressivity benefits by allowing each layer's attention and MLP to receive different linear combinations of the input and current state, it may also face unfavorable optimization geometry even in the contractive-Jacobian regime.

Appendix B.4 shows that lower learning rates help keep $\rho(\frac{\partial g(x^*,x_0)}{\partial x^*})$ small throughout training, placing both architectures in this contractive-recall regime. Consistent with the previously-mentioned asymmetry, external recall beats internal recall on all low/medium LR comparisons on hard data, and in 10 out of 12 on validation data (each by at least 2 SEs); the remaining two are statistical ties.

We note that unnormalized internal recall is the *only* configuration studied here with an identity residual path. Its poor performance suggests that identity residuals, while beneficial in fixed-depth transformers, can be **actively harmful for recall architectures**. This result is not surprising upon further reflection: identity residuals are beneficial in fixed-depth transformers because they help the gradient propagate through many different layers back to the original input. However, looped models with recall use the same layer at each step and have numerous gradient paths to the input via the recall connection, reducing reliance on an uninterrupted identity path. Meanwhile, as shown above, the identity path with recall can complicate local attraction, making outer normalization beneficial for reaching a fixed point.

By contrast, when $\frac{\partial g(x^*,x_0)}{\partial x^*}$ is *expansive*, the balance becomes more mixed. Figure 4 shows how, for J_g large, external recall must actively *shift* the eigenvalues of J_h towards -1, whereas internal recall must merely keep its eigenvalues small and negative. The empirical results here are substantially more task-dependent than the contractive recall scenario; external fails to find any fixed points whatsoever for chess with pre-norm (seeing 0% accuracy), but is able to find more usable fixed points than internal recall in sudoku with peri-norm. Overall, as $\rho(W_x)$ grows –which Appendix B.4 shows is driven by learning rate – both architectures move into the expansive-Jacobian regime predicted by Proposition 4 to be unstable, and both degrade accordingly. In several configurations, this expansiveness eliminates usable fixed points entirely.

Taken together, while recall *improves* input-dependence and thus generalization capacity – the best recall model beats the best autonomous model across all norm and task combinations on hard data – it is not *sufficient* to ensure a favorable geometry, especially in higher-lr regimes where $\rho(W_x) > 1$[6]. In the next section, we consider outer normalization, which broadens the geometrically-stable regime and, in doing so, enables stronger generalization behavior for both internal and external recall models.

5.2 Recall with Outer Normalization

Section 5.1 showed that without outer normalization, the stable regime is narrow for both recall architectures and especially restrictive for internal recall. Outer normalization broadens this regime: since the outer normalization Jacobians appear as factors throughout the fixed-point Jacobian (see Appendix A.7 for the explicit form), a contractive ϕ can shrink the spectral radius and keep the model within this stable regime – a mechanism that unnormalized models lack a direct lever for.

In addition, we also require the fixed point to depend non-trivially on x_0 – that is, not merely as basin selection among a discrete set (a key problem of autonomous networks as proven in Proposition 2) but with a well-defined, nonzero local sensitivity $\frac{\partial x^*}{\partial x_0}$. The following proposition characterizes the benefits of this locally-attractive regime for recall models, regardless of the use of outer normalization. We then examine empirically whether outer normalization helps models enter this regime.

Proposition 5. *Suppose we have a generic recall model $x_{t+1} = f(x_t, x_0)$ with fixed starting iterate e. Say that x_T converges to x^*, a fixed point of the model, with $\rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right) < 1$. Then, the input-gradient of the iterates converges to the input-gradient of the limiting fixed-point. Formally:*

$$\lim_{T\to\infty} \frac{\mathrm{d}x_T}{\mathrm{d}x_0} = \frac{\mathrm{d}}{\mathrm{d}x_0}\left(\lim_{T\to\infty} x_T\right) = \frac{\mathrm{d}x^*}{\mathrm{d}x_0} = \left(I - \frac{\partial f(x^*, x_0)}{\partial x^*}\right)^{-1} \frac{\partial f(x^*, x_0)}{\partial x_0}$$

Proof. This proof consists of two steps. First, we show that $\frac{\partial x_*}{\partial x_0} = \left(I - \frac{\partial f(x^*,x_0)}{\partial x^*}\right)^{-1} \frac{\partial f(x^*,x_0)}{\partial x_0}$ by showing invertibility of $I - \frac{\partial f(x^*,x_0)}{\partial x^*}$ and applying the implicit function theorem. Then, we show

[6]While we only discuss large $\rho(W_x)$ in the context of high learning rates, it can also occur under poor initializations of the W_x matrix.

$\lim_{T\to\infty} \frac{\mathrm{d}x_T}{\mathrm{d}x_0} = (I - \frac{\partial f(x^*,x_0)}{\partial x^*})^{-1} \frac{\partial f(x^*,x_0)}{\partial x_0}$ via a norm-bounding argument that shows the limit of the unrolled left hand side converges to the right side.

See Appendix A.6 for proof details, and Appendix A.7 for explicit Jacobian forms for external and internal models with outer normalization. □

Together, Propositions 4 and 5 identify a regime that does not exist for autonomous networks. When $\rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right) < 1$, the limit of gradients $\frac{\mathrm{d}x_T}{\mathrm{d}x_0}$ converges to the gradient at the fixed point, a finite value which need not be zero. Thus, fixed points are locally attractive and *vary smoothly* with the input x_0, unlike autonomous models which merely choose which of a predetermined set of fixed points to output. In addition, a further consequence of this regime is that the fixed point is *locally independent* of e, the initial recall iterate (Definition 3.3), as proven in Appendix A.8. This initialization-independence is both intuitively helpful (e is problem-independent, so it should not affect the final result) and empirically beneficial: Anil et al. [2022] find that path-independence improves reasoning performance on a wide range of problems.

In Appendix B.3, we find that outer normalization does indeed help achieve this regime. In our tests with external and internal models on sudoku problems, all converged hidden states had full-loop Jacobian spectral radii below 1. In addition, removing the outer-normalization Jacobian factors at the same states substantially increased the spectral radius, justifying our argument that outer normalization provides benefits by shrinking the full-loop spectral radius rather than simply ensuring bounded hidden states. These qualitative results also held for two-block models, providing evidence that outer normalization provides a similar role in more complicated architectures. In Table 7, we find that these benefits indeed transfer to real-world tasks, with the best normalized looped model outperforming the best unnormalized looped model by 13 and 4.5 points on sudoku and chess, respectively[7].

We note that although normalized models generally outperform un-normalized ones, the exact best architectural choices – post vs gru norm, internal vs external recall – differ across tasks, since other factors can dominate once the axes are satisfied (e.g. exponential token clustering in post-norm [Karagodin et al., 2025, Geiping et al., 2025], or representation capacity differences between internal and external recall). Thus, the axes should be treated as *necessary conditions rather than sufficient*: models without them degrade on validation and/or hard data, but models with them do not always succeed across tasks. We leave possible sufficient conditions for future work.

6 Conclusion

This paper introduced a framework for understanding when looped transformers learn generalizable, scalable algorithms rather than memorize training-depth solutions, organized around three axes of stability: local attraction, input-dependence, and geometry. We analyzed three architectural choices – the existence of recall, the placement of recall, and the presence of outer normalization – and found that each affects a distinct subset of the axes in ways that predict downstream performance.

Autonomous networks exhibit extremely weak input-dependence (Propositions 1 - 3); recall without outer normalization possesses input-dependent fixed points but suffers from a narrow, anisotropic stability geometry (Figure 4, Proposition 4); outer normalization broadens the stable regime and improves stability across all three axes. Our empirical results across chess, sudoku, and prefix-sums track these predictions. Internal recall, a novel placement variant, performs poorly with an identity residual but is competitive with external recall once outer normalization is added – though the relative ranking across tasks remains unresolved, and whether these findings extend to larger models is an object for future work. Our stability conditions are also local; while they provide for attraction near a fixed point, they do not characterize its basin size, so they do not ensure that x_0 reaches it. While we primarily apply these axes to describe and justify existing architectural choices, they also suggest a strategy for developing future architectures designed for local attraction, input-dependence, and favorable geometry from the start.

[7] Both unnormalized and normalized models saturate prefix sums, though autonomous models never reach non-zero hard accuracy.

Acknowledgments and Disclosure of Funding

We thank our advisors at Brown University: Peyam Tabrizian (Applied Math Department), and Andras Zsom (Department of Data Science).

This work was generously supported via a grant from the Brown University Library. Additional compute for the rebuttal and camera-ready phases was provided by Liquid AI.

References

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter, and Roger Baker Grosse. Path independent equilibrium models can better exploit test-time computation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=kgT6D7Z4Xv9.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models, 2019. URL https://arxiv.org/abs/1909.01377.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder, 2021. URL https://arxiv.org/abs/2107.05407.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end Algorithm Synthesis with Recurrent Networks: Logical Extrapolation Without Overthinking. *Advances in Neural Information Processing Systems*, 35, February 2022. ISSN 9781713871088. URL https://arxiv.org/pdf/2202.05826.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. *7th International Conference on Learning Representations, ICLR 2019*, July 2018. URL https://arxiv.org/pdf/1807.03819.

Tom Dillon. Tdoku: A fast sudoku solver and generator, 2025. URL https://t-dillon.github.io/tdoku.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. February 2025. URL http://arxiv.org/abs/2502.05171.

Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

V. Guillemin and A. Pollack. *Differential Topology*. AMS Chelsea Publishing. AMS Chelsea Pub., 1974. ISBN 978-1-4704-8112-4. URL https://bookstore.ams.org/chel-370-h.

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

Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL https://arxiv.org/abs/2010.04245.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

Alexia Jolicoeur-Martineau. Less is More: Recursive Reasoning with Tiny Networks, October 2025. URL http://arxiv.org/abs/2510.04871. arXiv:2510.04871.

Nikita Karagodin, Shu Ge, Yury Polyanskiy, and Philippe Rigollet. Normalization in attention dynamics, 2025. URL https://arxiv.org/abs/2510.22026.

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, 2025. URL https://arxiv.org/abs/2502.02732.

Lichess. Lichess. URL https://lichess.org.

limefax. Rope-nd, 2024. URL https://github.com/limefax/rope-nd/tree/main.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

William Merrill and Ashish Sabharwal. Exact expressive power of transformers with padding, 2025. URL https://arxiv.org/abs/2505.18948.

John Milnor. *Topology from the Differentiable Viewpoint*. University Press of Virginia, 1965.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. On the Inductive Bias of Stacking Towards Improving Reasoning. September 2024. URL http://arxiv.org/abs/2409.19044.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with Latent Thoughts: On the Power of Looped Transformers. February 2025. URL http://arxiv.org/abs/2502.17416.

Michael Shub. *Global stability of dynamical systems*. Springer, New York, NY, December 2010.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical Reasoning Model, August 2025. URL http://arxiv.org/abs/2506.21734. arXiv:2506.21734.

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

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

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, 2020. URL https://arxiv.org/abs/2002.04745.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, 2024. URL https://arxiv.org/abs/2311.12424.

A Proofs

Notation. Unless stated otherwise, throughout this appendix we treat the parameters θ fixed as a constant of the problem. We write $\frac{\mathrm{d}}{\mathrm{d}x_0}$ for total derivatives that consider all dependencies of x_0 through the unrolled recurrence. For derivatives of single-argument functions, we similarly use total derivatives: for example, $\frac{\mathrm{d}h(g(x_t,x_0))}{\mathrm{d}g(x_t,x_0)}$ denotes the derivative of h with respect to its argument. We reserve ∂ for multi-argument functions: for example, $\frac{\partial g(x_t,x_0)}{\partial x_0}$ denotes the partial derivative of g with respect to the second slot, holding x_t constant.

This distinction matters because several proofs below involve chain rule computations where the same variable appears as both a direct argument of a function and an indirect dependency through earlier states. We specify notation here to provide an unambiguous meaning throughout algebra-heavy proofs.

A.1 Weakness of Transversality Assumption

As a concrete illustration, consider a linear network $f : \mathbb{R}^d \to \mathbb{R}^d$ defined as $f(x) = Ax + b$. Then $\frac{\partial f}{\partial b} = I$, which is full-rank regardless of A or x, so transversality applies trivially. Below, we prove that a similar argument on A extends this result to length-L inputs whenever the tokens are linearly independent and $L \leq d$.

Proposition 6. *Consider a network $f : \mathbb{R}^{d\times L} \times \mathbb{R}^{d\times d} \times \mathbb{R}^d \to \mathbb{R}^{d\times L}$, linear in its first argument, defined as $f(X, A, b) = AX + b\mathbf{1}^T$. Assume that $L \leq d$ and that the columns of X are linearly independent (rank L). Then, $\frac{\partial f(X,A,b)}{\partial A}$, a $dL \times d^2$ matrix, is full-rank.*

Proof. Fix X and b. We note that $f(X, A + T, b) = AX + TX + b\mathbf{1}^T$. Thus,

$$\lim_{T\to 0} \frac{f(X, A+T, b) - f(X, A, b) - TX}{\|T\|} =$$
$$\lim_{T\to 0} \frac{AX + TX + b\mathbf{1}^T - AX - b\mathbf{1}^T - TX}{\|T\|} =$$
$$0$$

And thus $Df(T) = TX$ is the partial derivative of f with respect to A (though, it does not *depend* on A). Rather than directly analyze the $dL \times d^2$ matrix, we instead show this map $Df : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times L}$ is surjective. Consider arbitrary $Y \in \mathbb{R}^{d\times L}$. Since X has full column rank, X^TX is invertible. Then, let $T = Y(X^TX)^{-1}X^T \in \mathbb{R}^{d\times d}$ so that $Df(T) = Y(X^TX)^{-1}X^TX = Y$. Since Y was arbitrary, we have that the partial derivative is surjective, and thus $\frac{\partial f(X,A,b)}{\partial A}$ is full-rank regardless of the starting matrix A. Thus, the transversality assumption holds at fixed points X with linearly independent columns and $L \leq d$. □

We note two facts about this proof. First, it does not apply to our experiments on prefix-sums, given that they use 512-bit sequences with a hidden dimension of just 256. As the proof is merely a sufficient condition, not a necessary one – the combined state-parameter Jacobian could be full-rank, even if $\frac{\partial f(X,A,b)}{\partial A}$ alone is not – this proof simply gives no information in that case. Second, this example does not establish transversality for transformer architectures: while greater model flexibility and parameter count may help in ensuring the required independent directions, this proof remains a heuristic rather than a guarantee for nonlinear models.

A.2 Joint Fixed Points in Autonomous Networks

Proposition 2 proves that for almost all parameterizations θ, the set of fixed points of f_θ is a dimension-zero manifold (finite if x_t is compact). However, many looped models include a final "head" layer after concluding the looped layers, which may map multiple distinct values to the same value. For example, applying a normalization after the looped layers maps all λx_t to the same point. Thus, it is possible for there to exist points x_t wherein $f_\theta(x_t) \neq x_t$, but $w_\theta(f_\theta(x_t)) = w_\theta(x_t)$ for some head $w_\theta : \mathbb{R}^{d\times L} \to \mathbb{R}^{d\times L}$. We call these points "joint fixed points". As before we define

$h_\theta = w_\theta(f_\theta(x_t)) - w_\theta(x_t)$ such that $h_\theta(x_t) = 0 \leftrightarrow x_t$ is a joint fixed point. If we augment Assumption 1 so that h is transversal to 0 instead, we get that $h_\theta^{-1}(0)$ (the set of all joint fixed points) is a dimension-zero set (finite if x_t is bounded). This proves that, while head layers may be useful in some formulations of looped models, for autonomous models, they do not change the dimension of the set of fixed points prior to applying the loss.

A.3 Gradient Instability for Autonomous Looped Networks

Proof. Write $\rho^* = \rho(J_{f_\theta}(x^*))$. Fix $\alpha \in (\rho^*, 1)$, and let $\epsilon = \frac{\alpha - \rho^*}{2}$. By Horn and Johnson [1985] Lemma 5.6.10, there exists a submultiplicative matrix norm, written $\|.\|_*$, such that $\|J_{f_\theta}(x^*)\|_* \leq \frac{\alpha+\rho^*}{2} = \rho^* + \epsilon$. Assuming smoothness of f, $x \to J_{f_\theta}(x)$ is continuous. Since all norms are equivalent in finite-dimensional vector spaces, there exists a δ such that for $\|x - x^*\| < \delta$, $\|J_{f_\theta}(x) - J_{f_\theta}(x^*)\|_* < \epsilon$. Since $x_t \to x^*$, we also know that $\exists N \; s.t. \; \forall t > N, \|x_t - x^*\| < \delta$. Combined, we have

$$\exists N \; s.t. \; \forall t > N, \|J_{f_\theta}(x_t) - J_{f_\theta}(x^*)\|_* < \epsilon$$

Using the triangle inequality we get

$$\begin{aligned} \|J_{f_\theta}(x_t)\|_* \leq \\ \|J_{f_\theta}(x_t) - J_{f_\theta}(x^*)\|_* + \|J_{f_\theta}(x^*)\|_* < \\ \epsilon + \rho^* = \\ \alpha \end{aligned}$$

We can now plug this into the gradient as T grows. Since the network is autonomous, each iterate satisfies $x_{t+1} = f_\theta(x_t)$. By the chain rule,

$$\begin{aligned} \frac{\mathrm{d}x_T}{\mathrm{d}x_0} = \\ \prod_{t=0}^{T-1} J_{f_\theta}(x_t) = \\ \prod_{t=N+1}^{T-1} J_{f_\theta}(x_t) \prod_{t=0}^{N} J_{f_\theta}(x_t) < \\ \alpha^{T-N-2} \prod_{t=0}^{N} J_{f_\theta}(x_t) \end{aligned}$$

Since $\alpha < 1$, as $T \to \infty$, this vanishes exponentially with T. □

A.4 Stability Cases for Autonomous Looped Networks

Proposition 3 finds that the behavior of $\frac{\mathrm{d}x_T}{\mathrm{d}x_0}$ depends strongly on the spectral radius of $J_{f_\theta} = \frac{\partial f_\theta(x^*)}{\partial x^*}$. Here, we split the spectral radius into three cases and analyze the regime of each.

1. **Case 1: $\boldsymbol{\rho(J_{f_\theta}(x^*)) < 1}$.** Gelfand's formula implies that $\left\|J_{f_\theta}(x^*)^{T-K}\right\|$ decays exponentially with T. In particular, $\forall \alpha \in (\rho(J_{f_\theta}(x^*)), 1), \exists C_\alpha \in \mathbb{R}^+$ with $\left\|J_{f_\theta}(x^*)^{T-K}\right\| \leq C_\alpha \alpha^{T-K}$. Thus, we have that

$$\left\|\frac{\mathrm{d}x_T}{\mathrm{d}x_0}\right\| \leq \|J_{f_\theta}(x^*)\|^{T-K} \left\|A_{(x_0,\theta)}\right\| \leq C_{\alpha,x_0,\theta}\alpha^{T-K}$$

   which decays exponentially with T. This is *input-independence*: as $T \to \infty$, $\left\|\frac{\mathrm{d}x_T}{\mathrm{d}x_0}\right\| \to 0$ exponentially, meaning x_T becomes asymptotically insensitive to the initial state x_0. Since x_0 is the only information provided to the model f_θ, this means the model essentially "forgets" the information it was given.

2. **Case 2: $\rho(J_{f_\theta}(x^*)) > 1$**. This implies that there is at least one eigenvalue with modulus greater than 1. In this scenario, the center-stable manifold theorem ([Shub, 2010, p.65]) states the local center-stable manifold around x^* has dimension $< dL$, and is thus measure zero. Thus among points in a sufficiently small neighborhood of x^*, almost every point eventually leaves the neighborhood, with at least one direction repelling away from x^*. Figure 2 provides two example systems, one with all $|\lambda| < 1$ and one with an eigenvalue greater than 1; in the latter case, any starting point except the $y = 0$ line is repelled away from the fixed point. Unlike the $\rho(J_{f_\theta}) < 1$ scenario, wherein fixed points are easily reached but fail to depend on the input, fixed points in this case are practically never reached at all.

3. **Case 3: $\rho(J_{f_\theta}(x^*)) = 1$**. This case provides very little useful information, given that x^* is a non-hyperbolic fixed point, of which much less can be derived.

   We conduct two short analyses on this regime. First, we consider what occurs as the Jacobian approaches 1 from a locally-attractive regime; i.e. as $\rho(J_{f_\theta}(x^*)) \to 1^-$. Under this scenario, we use the fact that $\rho(J_{f_\theta}(x^*)) < 1$ along with equation (1) from Proposition 1 to note that $\frac{\partial g(x^*,\theta)}{\partial x^*}$ is nonsingular for $g(x^*, \theta) = f_\theta(x^*) - x^*$. Thus the implicit function theorem applies and we can write x^* as a smooth function of θ nearby the zero point.

   Note that our gradient notation shifts at this point: here we treat θ as the key variable and let $\frac{\mathrm{d}g(x^*(\theta),\theta)}{\mathrm{d}\theta}$ represent the total derivative with respect to θ. We can then write that the total gradient $\frac{\mathrm{d}g(x^*(\theta),\theta)}{\mathrm{d}\theta}$ = 0 at the zero point (due to the existence of a neighborhood with $g(x^*(\theta), \theta) = 0$), so we use the chain rule to get

$$\frac{\partial g(x^*,\theta)}{\partial x^*}\frac{\mathrm{d}x^*(\theta)}{\mathrm{d}\theta} + \frac{\partial g(x^*,\theta)}{\partial \theta} = 0 \to$$
$$\frac{\mathrm{d}x^*(\theta)}{\mathrm{d}\theta} = -\left(\frac{\partial g(x^*,\theta)}{\partial x^*}\right)^{-1}\frac{\partial g(x^*,\theta)}{\partial \theta} \to$$
$$\frac{\mathrm{d}x^*(\theta)}{\mathrm{d}\theta} = -\left(\frac{\partial f_\theta(x^*)}{\partial x^*} - I\right)^{-1}\frac{\partial f_\theta(x^*)}{\partial \theta}$$

   Thus, if the largest eigenvalue of the Jacobian is near-1 (which occurs when the sublayer is near-zero), the smallest eigenvalue of $\frac{\partial f_\theta(x^*)}{\partial x^*} - I \to 0$ and thus $\frac{\mathrm{d}x^*(\theta)}{\mathrm{d}\theta}$ explodes in magnitude (as long as it is not zeroed by the right-multiplication of $\frac{\partial f_\theta(x^*)}{\partial \theta}$). Therefore the *parameter* gradient becomes arbitrarily unstable in this incentivized regime.

   We simultaneously consider the *speed* of convergence in the $\rho(J_{f_\theta}(x^*)) = 1$ regime. Shub [2010] shows that in such a regime, fixed-point convergence occurs no faster than *sub-exponentially*, and thus may require far more compute than the exponential convergence of fixed points with $\rho(J_{f_\theta}(x^*)) < 1$. Thus, in this regime, the model has a greater probability to fail to reach a fixed point during training in the first place, making generalization more difficult.

A.5 Stability Regimes of Recall Models without Outer Normalization

Proof. For external recall, we have that

$$\frac{\mathrm{d}x_{t+1}}{\mathrm{d}x_t} =$$
$$\frac{\mathrm{d}x_{t+1}}{\mathrm{d}z_t}\frac{\mathrm{d}z_t}{\mathrm{d}x_t} =$$
$$\left(I + \frac{\mathrm{d}h_2(z_t)}{\mathrm{d}z_t}\right)\left(\frac{\partial g(x_t,x_0)}{\partial x_t} + \frac{\mathrm{d}h_1(g(x_t,x_0))}{\mathrm{d}g(x_t,x_0)}\frac{\partial g(x_t,x_0)}{\partial x_t}\right) =$$
$$\left(I + \frac{\mathrm{d}h_2(z_t)}{\mathrm{d}z_t}\right)\left(I + \frac{\mathrm{d}h_1(g(x_t,x_0))}{\mathrm{d}g(x_t,x_0)}\right)\frac{\partial g(x_t,x_0)}{\partial x_t}$$

Once a fixed-point is reached, this becomes

$$M_{\text{ext}} = \left(I + \frac{\mathrm{d}h_2(z^*)}{\mathrm{d}z^*}\right)\left(I + \frac{\mathrm{d}h_1(g(x^*,x_0))}{\mathrm{d}g(x^*,x_0)}\right)\frac{\partial g(x^*,x_0)}{\partial x^*}$$

As explained in Section 4 and Appendix A.4, a fixed point can only be practically reachable (given floating-point perturbations) if it is locally attractive, which requires $\rho\left(\frac{\partial f_\theta(x^*,x_0)}{\partial x^*}\right) = \rho(M_{\text{ext}}) \leq 1$.

For internal recall, we have that

$$\frac{\mathrm{d}x_{t+1}}{\mathrm{d}x_t} = \frac{\mathrm{d}x_{t+1}}{\mathrm{d}z_t}\frac{\mathrm{d}z_t}{\mathrm{d}x_t} = \left(I + \frac{\mathrm{d}h_2(g(z_t,x_0))}{\mathrm{d}g(z_t,x_0)}\frac{\partial g(z_t,x_0)}{\partial z_t}\right)\left(I + \frac{\mathrm{d}h_1(g(x_t,x_0))}{\mathrm{d}g(x_t,x_0)}\frac{\partial g(x_t,x_0)}{\partial x_t}\right)$$

Once a fixed point is reached, this becomes

$$M_{\text{int}} = \left(I + \frac{\mathrm{d}h_2(g(z^*,x_0))}{\mathrm{d}g(z^*,x_0)}\frac{\partial g(z^*,x_0)}{\partial z^*}\right)\left(I + \frac{\mathrm{d}h_1(g(x^*,x_0))}{\mathrm{d}g(x^*,x_0)}\frac{\partial g(x^*,x_0)}{\partial x^*}\right)$$

The equivalent argument as in external recall requires $\rho(M_{\text{int}}) \leq 1$ for internal recall models to be locally stable. □

A.6 Stability of Recall Models with Outer Normalization

We prove this in two steps.

Step 1: $\frac{\mathrm{d}x^*}{\mathrm{d}x_0} = \left(I - \frac{\partial f(x^*,x_0)}{\partial x^*}\right)^{-1}\frac{\partial f(x^*,x_0)}{\partial x_0}$

To begin with this proof, we must first show that the left term in this equation is well-defined and differentiable in the first place. We define the function $F(x_t, x_0) = f(x_t, x_0) - x_t$ so that fixed points of f are zeros of F. At a fixed point x^*, we have (by assumption) that $\rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right) < 1$. Thus, 1 is not an eigenvalue of $\frac{\partial f(x^*,x_0)}{\partial x^*}$. In that case, 0 is not an eigenvalue of

$$\frac{\partial F(x^*,x_0)}{\partial x^*} = \frac{\partial f(x^*,x_0)}{\partial x^*} - I$$

and thus $J_F = \frac{\partial F(x^*,x_0)}{\partial x^*}$ is invertible, so the implicit function theorem applies. Thus, there exists a differentiable function $x^*(x_0)$ in a neighborhood of x_0 such that $F(x^*(x_0), x_0) = 0$ for all points on that neighborhood. Further, we have that

$$\frac{\mathrm{d}x^*}{\mathrm{d}x_0} = -\left(\frac{\partial F(x^*(x_0),x_0)}{\partial x^*}\right)^{-1}\frac{\partial F(x^*(x_0),x_0)}{\partial x_0}$$

and

$$\frac{\mathrm{d}x^*}{\mathrm{d}x_0} = \left(I - \frac{\partial f(x^*(x_0),x_0)}{\partial x^*}\right)^{-1}\frac{\partial f(x^*(x_0),x_0)}{\partial x_0}$$

Step 2: $\lim_{T\to\infty}\frac{\mathrm{d}x_T}{\mathrm{d}x_0} = \left(I - \frac{\partial f(x^*(x_0),x_0)}{\partial x^*}\right)^{-1}\frac{\partial f(x^*(x_0),x_0)}{\partial x_0}$

From our assumption, we have that $\rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right) < 1$. While useful for the prior step, here we instead need to bound the *norm* of the Jacobian. Horn and Johnson [1985] Lemma 5.6.10 tells us that for any $\epsilon > 0$, matrix A, there exists a norm $*$ such that $\|A\|_* \leq \rho(A) + \epsilon$. First defining $\rho = \rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right)$ for notational simplicity, we choose $\epsilon = \frac{1-\rho}{4}$, so that $\left\|\frac{\partial f(x^*,x_0)}{\partial x^*}\right\|_* < \frac{1+3\rho}{4}$.

We continue with two stated facts: first, that $x_T \to x^*$, and second, that $f(x_t, x_0)$ is smooth in both of its arguments. Thus, $\frac{\partial f(x_t,x_0)}{\partial x_t}$ is continuous in x_t, and since norms are a continuous function of a matrix, $\left\|\frac{\partial f(x_t,x_0)}{\partial x_t}\right\|_*$ is also continuous in x_t. Applying this to the fixed point x^*, we have that

for any $\epsilon > 0, \exists \delta > 0 \ s.t. \ \|x' - x^*\| < \delta \implies \left| \left\| \frac{\partial f(x', x_0)}{\partial x'} \right\|_* - \left\| \frac{\partial f(x^*, x_0)}{\partial x^*} \right\|_* \right| < \epsilon$. We choose $\epsilon = \frac{1-\rho}{4}$ so that $\left\| \frac{\partial f(x', x_0)}{\partial x'} \right\|_* < \frac{1+3\rho}{4} + \frac{1-\rho}{4} = \frac{1+\rho}{2}$.

Similarly, since x_T converges to x^*, we know that for $\epsilon > 0, \exists N s.t. \forall n > N, \|x_n - x^*\| < \epsilon$. We choose ϵ as δ found earlier, and get that $\exists N \ s.t. \ \forall n > N, \left\| \frac{\partial f(x_n, x_0)}{\partial x_n} \right\|_* < \frac{1+\rho}{2}$. We can follow the same argument for $\left\| \frac{\partial f(x_n, x_0)}{\partial x_0} - \frac{\partial f(x^*, x_0)}{\partial x_0} \right\|_* < \epsilon$, and choose N large enough so that for small, prechosen α, we have all three conditions fulfilled:

1. $\left\| \frac{\partial f(x_n, x_0)}{\partial x_n} \right\|_* < \frac{1+\rho}{2}$
2. $\left\| \frac{\partial f(x_n, x_0)}{\partial x_0} - \frac{\partial f(x^*, x_0)}{\partial x_0} \right\|_* < \alpha$
3. $\left\| \frac{\partial f(x_n, x_0)}{\partial x_n} - \frac{\partial f(x^*, x_0)}{\partial x^*} \right\|_* < \alpha$

With this in mind, we consider the recurrence $\frac{\mathrm{d}x_t}{\mathrm{d}x_0} = \frac{\partial x_t}{\partial x_{t-1}} \frac{\mathrm{d}x_{t-1}}{\mathrm{d}x_0} + \frac{\partial x_t}{\partial x_0}$, with x_t both a direct function of x_0 (recall) and an indirect one through x_{t-1}. Unrolling this to zero, we get:

$$\frac{\mathrm{d}x_t}{\mathrm{d}x_0} = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \frac{\partial x_j}{\partial x_{j-1}} \right) \frac{\partial x_i}{\partial x_0}$$

Since we care about the result as $t \to \infty$, we consider what occurs as $t > N$. We split up the sum into before and after N:

$$\sum_{i=1}^{T} \left(\prod_{j=i+1}^{T} \frac{\partial x_j}{\partial x_{j-1}} \right) \frac{\partial x_i}{\partial x_0} =$$

$$\sum_{i=1}^{N} \left(\prod_{j=i}^{N} \frac{\partial x_j}{\partial x_{j-1}} \right) \left(\prod_{j=N+1}^{T} \frac{\partial x_j}{\partial x_{j-1}} \right) \frac{\partial x_i}{\partial x_0} + \sum_{i=N+1}^{T} \left(\prod_{j=i+1}^{T} \frac{\partial x_j}{\partial x_{j-1}} \right) \frac{\partial x_i}{\partial x_0}$$

And since $\left\| \frac{\partial x_i}{\partial x_{i-1}} \right\|_* = \left\| \frac{\partial f(x_i, x_0)}{\partial x_i} \right\|_* < \frac{1+\rho}{2}$ as long as $i > N$, the first term goes to 0 as $T \to \infty$. We are left with only the final term.

We now wish to prove that

$$\lim_{T \to \infty} \sum_{i=N+1}^{T} \left(\prod_{j=i+1}^{T} \frac{\partial x_j}{\partial x_{j-1}} \right) \frac{\partial x_i}{\partial x_0} = \left(I - \frac{\partial f(x^*, x_0)}{\partial x^*} \right)^{-1} \frac{\partial f(x^*, x_0)}{\partial x_0}$$

We proceed by first naming matrices for ease of reading. $A_t = \frac{\partial x_t}{\partial x_{t-1}}, B_t = \frac{\partial x_t}{\partial x_0}$, and $A_* = \frac{\partial f(x^*, x_0)}{\partial x^*}$ with $B_* = \frac{\partial f(x^*, x_0)}{\partial x_0}$. We now reindex the sum backwards $i \to T - i$ and substitute these names to get:

$$S_T = \sum_{i=0}^{T-N-1} \left(\prod_{j=T-i+1}^{T} A_j \right) B_{T-i}$$

We compare this sum with the sum of the fixed-point matrices, namely

$$S_T^* = \sum_{i=0}^{T-N-1} A_*^i B_*$$

Note that as $T \to \infty$, as long as $\|A_*\| < 1$ for some norm (which we previously proved for norm $*$), S_T^* converges to $\left(I - \frac{\partial f(x^*, x_0)}{\partial x^*} \right)^{-1} \frac{\partial f(x^*, x_0)}{\partial x_0}$.

We wish to prove that $\lim_{T\to\infty} \|S_T^* - S_T\|_* = 0$; i.e. the Jacobians are the same in the limit. By the triangle inequality, we have that

$$\|S_T^* - S_T\|_* \leq \sum_{i=0}^{T-N-1} \left\| \left(\prod_{j=T-i+1}^{T} A_j \right) B_{T-i} - A_*^i B_* \right\|_*$$

We can add and subtract $A_*^i B_{T-i}$ and simplify to get:

$$\begin{aligned}
&\sum_{i=0}^{T-N-1} \left\| \left(\prod_{j=T-i+1}^{T} A_j \right) B_{T-i} - A_*^i B_* \right\|_* = \\
&\sum_{i=0}^{T-N-1} \left\| \left(\prod_{j=T-i+1}^{T} A_j \right) B_{T-i} - A_*^i B_{T-i} + A_*^i B_{T-i} - A_*^i B_* \right\|_* = \\
&\sum_{i=0}^{T-N-1} \left\| \left(\left(\prod_{j=T-i+1}^{T} A_j \right) - A_*^i \right) B_{T-i} + A_*^i (B_{T-i} - B_*) \right\|_* \leq \\
&\sum_{i=0}^{T-N-1} \left(\left\| \left(\left(\prod_{j=T-i+1}^{T} A_j \right) - A_*^i \right) \right\|_* \|B_{T-i}\|_* + \|A_*\|_*^i \|B_{T-i} - B_*\|_* \right)
\end{aligned}$$

We now bound each of the individual norms. First, note that i goes from 0 to $T - N - 1$, so $T - i > N$ across the entire sum. Thus, $\|B_{T-i} - B_*\|_* < \alpha$ as described earlier. Also, we have that $\|A_*\|_* < \frac{1+\rho}{2} = \frac{1+\rho}{2}$ (renaming), so that $\|A_*\|_*^i < (\frac{1+\rho}{2})^i$. By the triangle inequality, we have that

$$\|B_{T-i}\|_* = \|B_{T-i} - B_* + B_*\|_* \leq \|B_{T-i} - B_*\|_* + \|B_*\|_* < \alpha + \|B_*\|_*$$

For simplicity, we notate $M = \alpha + \|B_*\|_*$, a constant regardless of T or i. Thus, the only term left to bound is the very first term. To do so, we prove a short lemma by induction:

Lemma 1. *For matrices $(A_n)_{n=1}^K, B$ that can be multiplied,*

$$\prod_{i=1}^{K} A_i - B^K = \sum_{i=1}^{K} \left(\prod_{j=1}^{i-1} A_j \right) (A_i - B) B^{K-i}$$

Proof. We prove by induction.

With K = 1, we have $A_1 - B = I(A_1 - B)B^0 = A_1 - B$, since the product term is the identity when the bottom is less than the top.

Now we suppose this is true for K. Then, for K + 1, we have

$$\begin{aligned}
\prod_{i=1}^{K+1} A_i - B^{K+1} = \\
\left(\prod_{i=1}^{K} A_i\right) A_{K+1} - B^K B = \\
\left(\prod_{i=1}^{K} A_i\right) A_{K+1} - \left(\prod_{i=1}^{K} A_i\right) B + \left(\prod_{i=1}^{K} A_i\right) B - B^K B = \\
\left(\prod_{i=1}^{K} A_i\right) (A_{K+1} - B) + \left(\prod_{i=1}^{K} A_i - B^K\right) B = \\
\left(\prod_{i=1}^{K} A_i\right) (A_{K+1} - B) + \sum_{i=1}^{K} \left(\prod_{j=1}^{i-1} A_j\right) (A_i - B) B^{K-i} B = \\
\left(\prod_{i=1}^{K} A_i\right) (A_{K+1} - B) + \sum_{i=1}^{K} \left(\prod_{j=1}^{i-1} A_j\right) (A_i - B) B^{(K+1)-i} = \\
\sum_{i=1}^{K+1} \left(\prod_{j=1}^{i-1} A_j\right) (A_i - B) B^{(K+1)-i}
\end{aligned}$$

□

We then apply Lemma 1 to the first term and get that

$$\begin{aligned}
\left\| \left(\left(\prod_{j=T-i+1}^{T} A_j \right) - A_*^i \right) \right\|_* = \\
\left\| \left(\left(\prod_{j=1}^{i} A_{j+T-i} \right) - A_*^i \right) \right\|_* = \\
\left\| \sum_{j=1}^{i} \left(\prod_{k=1}^{j-1} A_{k+T-i} \right) (A_{j+T-i} - A_*) A_*^{i-j} \right\|_* \leq \\
\sum_{j=1}^{i} \left\| \prod_{k=1}^{j-1} A_{k+T-i} (A_{j+T-i} - A_*) A_*^{i-j} \right\|_* \leq \\
\sum_{j=1}^{i} \prod_{k=1}^{j-1} \|A_{k+T-i}\|_* \|(A_{j+T-i} - A_*)\|_* \|A_*\|_*^{i-j}
\end{aligned}$$

Each factor in the first term has index greater than N, so we can bound $\|A_{k+T-i}\|_* < \frac{1+\rho}{2}$. The latter term is similarly bound: $\|A_*\|_*^{i-j} < (\frac{1+3\rho}{4})^{i-j} < (\frac{1+\rho}{2})^{i-j}$. The middle term is bound by α,

as it also has index greater than N. Thus the entire Jacobian is bounded by

$$\sum_{j=1}^{i}\left(\prod_{k=1}^{j-1}\frac{1+\rho}{2}\right)\alpha\left(\frac{1+\rho}{2}\right)^{i-j}=$$
$$\sum_{j=1}^{i}\left(\frac{1+\rho}{2}\right)^{j-1}\alpha\left(\frac{1+\rho}{2}\right)^{i-j}=$$
$$\sum_{j=1}^{i}\left(\frac{1+\rho}{2}\right)^{i-1}\alpha\leq$$
$$i\left(\frac{1+\rho}{2}\right)^{i}\alpha$$

Going back to the original series, we have that

$$\sum_{i=0}^{T-N-1}\left(\left\|\left(\left(\prod_{j=T-i+1}^{T}A_j\right)-A_*^i\right)\right\|_*\|B_{T-i}\|_*+\|A_*\|_*^i\|B_{T-i}-B_*\|_*\right)\leq$$
$$\sum_{i=0}^{T-N-1}\left(\left(i\left(\frac{1+\rho}{2}\right)^{i-1}\alpha M\right)+\alpha\left(\frac{1+\rho}{2}\right)^{i}\right)=$$
$$\alpha\sum_{i=0}^{T-N-1}\left(\left(i\left(\frac{1+\rho}{2}\right)^{i-1}M\right)+\left(\frac{1+\rho}{2}\right)^{i}\right)$$

Both terms converge to a positive number, since the right term is a geometric series with $|r|<1$ and the first is the derivative of a geometric series with converged sum equal to $\frac{1}{(1-r)^2}$ for $|r|<1$. Thus, the sum is bounded by a constant, and the series is bounded by $C\alpha$.

We then have by the triangle inequality that

$$\lim_{T\to\infty}\left\|S_T-(I-A_*)^{-1}B_*\right\|_*\leq\lim_{T\to\infty}\|S_T-S_T^*\|_*+\lim_{T\to\infty}\left\|S_T^*-(I-A_*)^{-1}B_*\right\|_*$$

The first term can be made arbitrarily small by setting $\alpha\to 0$ (and thus choosing large T and N), and the latter term is equal to 0. Thus,

$$\lim_{T\to\infty}\frac{\mathrm{d}x_T}{\mathrm{d}x_0}=\left(I-\frac{\partial f(x^*,x_0)}{\partial x^*}\right)^{-1}\frac{\partial f(x^*,x_0)}{\partial x_0}$$

completing the proof.

A.7 Explicit Jacobian Forms for External/Internal Recall

Here, we provide explicit forms of the Jacobians $\frac{\partial x_t}{\partial x_{t-1}}=\frac{\partial f(x_{t-1},x_0)}{\partial x_{t-1}}$ and $\frac{\partial x_t}{\partial x_0}=\frac{\partial f(x_{t-1},x_0)}{\partial x_0}$ for both internal and external recall. For notational simplicity, we let $\tilde{z}_t,\tilde{x}_t$ represent the result prior to applying ϕ_1,ϕ_2 respectively; i.e. $z_t=\phi_1(\tilde{z}_t)$ and $x_t=\phi_2(\tilde{x}_t)$. Note below the presence of the outer normalization Jacobians in the explicit forms, which can shrink the spectral radius if they are small.

For ease of memory, we redefine external and internal recall below:

External:

$$\begin{aligned}x_{t+1}&=\phi_2(z_t+h_2(z_t))\\z_t&=\phi_1(g(x_t,x_0)+h_1(g(x_t,x_0)))\end{aligned}$$

Internal:

$$\begin{aligned}x_{t+1}&=\phi_2(z_t+h_2(g(z_t,x_0)))\\z_t&=\phi_1(x_t+h_1(g(x_t,x_0)))\end{aligned}$$

External Recall:

$$\frac{\mathrm{d}x_t}{\mathrm{d}x_{t-1}} = \frac{\mathrm{d}x_t}{\mathrm{d}\tilde{x}_t}\frac{\mathrm{d}\tilde{x}_t}{\mathrm{d}z_{t-1}}\frac{\mathrm{d}z_{t-1}}{\mathrm{d}\tilde{z}_{t-1}}\frac{\partial\tilde{z}_{t-1}}{\partial x_{t-1}} = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(I + \frac{\mathrm{d}h_2(z_{t-1})}{\mathrm{d}z_{t-1}}\right)\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\left(\frac{\partial g(x_{t-1},x_0)}{\partial x_{t-1}} + \frac{\mathrm{d}h_1(g(x_{t-1},x_0))}{\mathrm{d}g(x_{t-1},x_0)}\frac{\partial g(x_{t-1},x_0)}{\partial x_{t-1}}\right) = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(I + \frac{\mathrm{d}h_2(z_{t-1})}{\mathrm{d}z_{t-1}}\right)\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\left(I + \frac{\mathrm{d}h_1(g(x_{t-1},x_0))}{\mathrm{d}g(x_{t-1},x_0)}\right)\frac{\partial g(x_{t-1},x_0)}{\partial x_{t-1}}$$

Internal Recall:

$$\frac{\mathrm{d}x_t}{\mathrm{d}x_{t-1}} = \frac{\mathrm{d}x_t}{\mathrm{d}\tilde{x}_t}\frac{\partial\tilde{x}_t}{\partial z_{t-1}}\frac{\mathrm{d}z_{t-1}}{\mathrm{d}\tilde{z}_{t-1}}\frac{\partial\tilde{z}_{t-1}}{\partial x_{t-1}} = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(I + \frac{\mathrm{d}h_2(g(z_{t-1},x_0))}{\mathrm{d}g(z_{t-1},x_0)}\frac{\partial g(z_{t-1},x_0)}{\partial z_{t-1}}\right)\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\cdot \left(I + \frac{\mathrm{d}h_1(g(x_{t-1},x_0))}{\mathrm{d}g(x_{t-1},x_0)}\frac{\partial g(x_{t-1},x_0)}{\partial x_{t-1}}\right)$$

A.7.1 Jacobian of $\frac{\partial x_t}{\partial x_0}$

External Recall:

$$\frac{\partial x_t}{\partial x_0} = \frac{\mathrm{d}x_t}{\mathrm{d}\tilde{x}_t}\frac{\mathrm{d}\tilde{x}_t}{\mathrm{d}z_{t-1}}\frac{\mathrm{d}z_{t-1}}{\mathrm{d}\tilde{z}_{t-1}}\frac{\partial\tilde{z}_{t-1}}{\partial x_0} = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(I + \frac{\mathrm{d}h_2(z_{t-1})}{\mathrm{d}z_{t-1}}\right)\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\left(\frac{\partial g(x_{t-1},x_0)}{\partial x_0} + \frac{\mathrm{d}h_1(g(x_{t-1},x_0)}{\mathrm{d}g(x_{t-1},x_0)}\frac{\partial g(x_{t-1},x_0)}{\partial x_0}\right) = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(I + \frac{\mathrm{d}h_2(z_{t-1})}{\mathrm{d}z_{t-1}}\right)\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\left(I + \frac{\mathrm{d}h_1(g(x_{t-1},x_0)}{\mathrm{d}g(x_{t-1},x_0)}\right)\frac{\partial g(x_{t-1},x_0)}{\partial x_0}$$

Internal Recall:

$$\frac{\partial x_t}{\partial x_0} = \frac{\mathrm{d}x_t}{\mathrm{d}\tilde{x}_t}\left(\frac{\partial\tilde{x}_t}{\partial x_0} + \frac{\mathrm{d}\tilde{x}_t}{\mathrm{d}z_{t-1}}\frac{\mathrm{d}z_{t-1}}{\mathrm{d}\tilde{z}_{t-1}}\frac{\partial\tilde{z}_{t-1}}{\partial x_0}\right) = \frac{\mathrm{d}\phi_2(\tilde{x}_t)}{\mathrm{d}\tilde{x}_t}\left(\frac{\mathrm{d}h_2(g(z_{t-1},x_0))}{\mathrm{d}g(z_{t-1},x_0)}\frac{\partial g(z_{t-1},x_0)}{\partial x_0} + \left(I + \frac{\mathrm{d}h_2(g(z_{t-1},x_0))}{\mathrm{d}g(z_{t-1},x_0)}\frac{\partial g(z_{t-1},x_0)}{\partial z_{t-1}}\right)\cdot\frac{\mathrm{d}\phi_1(\tilde{z}_{t-1})}{\mathrm{d}\tilde{z}_{t-1}}\frac{\mathrm{d}h_1(g(x_{t-1},x_0))}{\mathrm{d}g(x_{t-1},x_0)}\frac{\partial g(x_{t-1},x_0)}{\partial x_0}\right)$$

A.8 Local-Independence of e in Stable Regimes

Proof. We wish to prove that in the stable regime found in Section 5.2, the resulting fixed point is locally independent of e, the initial input to recall. Importantly, in both external and internal recall, this initial iterate *does not affect* x_0, only later x_t. Thus, we have that $\frac{\mathrm{d}x_t}{\mathrm{d}e} = \frac{\mathrm{d}x_t}{\mathrm{d}x_{t-1}}\frac{\mathrm{d}x_{t-1}}{\mathrm{d}e}$. We unroll this back and see that

$$\frac{\mathrm{d}x_T}{\mathrm{d}e} = \left(\prod_{i=2}^{T}\frac{\mathrm{d}x_t}{\mathrm{d}x_{t-1}}\right)\frac{\partial x_1}{\partial e}$$

As proven in Appendix A.6, we know that when $\rho\left(\frac{\partial f(x^*,x_0)}{\partial x^*}\right) < 1$ (i.e. when in the stable regime) the term $\prod_{i=2}^{T} \frac{\mathrm{d}x_t}{\mathrm{d}x_{t-1}}$ converges to 0 as $T \to \infty$. Thus the fixed point becomes *locally-independent* of the initial iterate e while remaining *dependent* on x_0 as proven in Appendix A.6. □

B Empirical Analyses

B.1 Experimental Setup

B.1.1 Data

We train on three datasets – prefix-sums, sudoku, and chess – each designed to test stability and generalization along a different axis.

Prefix-sums: We obtained prefix-sum data from Bansal et al. [2022]. Each question inputs an N-bit binary sequence, and outputs a same-length sequence with each bit K the sum of all preceding input bits, mod 2. This task is also known as cumulative parity. We train and validate the model on 32-bit data (8000 and 2000 points, respectively) and test on 512-bit data (10000 points). We note that this is the *only* problem whose difficulty is directly determined by its *length*, rather than some alternative internal mechanism.

Sudoku: We obtained sudoku data from Sapient Intelligence's sudoku-extreme dataset ([Wang et al., 2025]) that defines difficulty as the number of backtracks needed on the Tdoku solver ([Dillon, 2025]) to solve the puzzle. To separate "easy" from "hard" examples, we calculated the 80th percentile of backtracks needed throughout the dataset (38 backtracks). We then trained on 40000 sudoku puzzles with fewer than the 80th percentile of backtracks, validated on 10000 of the same difficulty, and tested on 10000 puzzles with more than the 80th percentile of backtracks. Sudoku uses 10 input channels and 10 output channels (one each from 0-9).

Chess: We train our models on data from the Lichess ([Lichess]) puzzles database, which provides FEN data for given chess board states as well as the corresponding best next move. We train our model on puzzles with ELO ratings less than 1600, and validate on the same; we test the model on harder puzzles with ratings between 1600-2000. Lichess puzzles remain the same size across training and testing (8×8) and have 12 input channels (6 pieces, 2 players) and 2 output channels (1s on the from and to spots on the board). We normalize orientation so the model always sees "white to move". We use 40000 training examples and 10000 of both validation and testing examples.

Table 1: Dataset summary. For prefix-sums, train and test use different sequence lengths (32-bit and 512-bit, respectively), as length directly determines difficulty. Note that prefix-sums and chess both use two-logit classification.

Dataset	Dimensions	In Channels	Out Channels	Train	Test
Prefix-Sums	$32 \to 512$ bits	1	2	8,000	10,000
Sudoku	9×9	10	10	40,000	10,000
Chess	8×8	12	2	40,000	10,000

B.1.2 Training Details

Across all datasets, we use a single-layer transformer with a hidden dimension of 256. The attention module uses 8 heads (head dimension 32), QK-normalization [Henry et al., 2020] for training stability, and multi-dimensional RoPE [limefax, 2024] to provide consistent positional structure across both 1D (prefix-sums) and 2D (sudoku, chess) inputs. For sudoku and chess, we use full bidirectional attention. For prefix-sums, RoPE causes length generalization failures, so we instead use a sliding attention window of width 5. The MLP uses a hidden dimension of 1024 with GELU activations [Hendrycks and Gimpel, 2023]. We apply RMSNorm at all normalization sites. We sweep over three RMSNorm-based normalization configurations: pre-norm (pre), post-norm (post), peri-norm (peri) [Kim et al., 2025], in addition to GRU outer normalization (gru). For models with GRU outer normalization, we replace the standard residual connection $x_{t+1} = x_t + h_\theta(x_t)$ with

$x_{t+1} = \text{GRU}(x_t, h_\theta(x_t))$, where the recurrent state is x_t and the update is $h_\theta(x_t)$. For recall models, we use linear recall, i.e., $g(x_t, x_0) = W_x x_t + W_0 x_0$. Looped models with GRU had 1.39M parameters, while models without it had 0.98M. Fixed (non-looped) models used 15 layers and had 11.9M parameters without GRU and 17.8M parameters with it. Recall accounted for 0.13M parameters in models that used it.

We note that while recall does add parameters, its benefits are not primarily attributable to parameter count. If they were, fixed-depth models – which have an order of magnitude more parameters – would dominate across tasks, yet they are outperformed by recall variants on sudoku and prefix-sums, and only match them on chess (the least naturally iterative of the three). Bansal et al. [2022] make a similar observation, noting that the gains from recall far exceed what additional depth or width provides in non-recall models.

We train with AdamW [Loshchilov and Hutter, 2019] (weight decay 0.01) for 100 epochs with a batch size of 500. We apply an exponential warmup over the first 10 epochs, reaching 63% of the target learning rate by epoch 9, followed by a constant learning rate through epoch 60, after which we apply a $10\times$ cooldown. Gradients are clipped at norm 1.

Within each batch, we apply the progressive loss scheme of Bansal et al. [2022]. We sample $N \sim \mathcal{U}[0, T-1]$ and $K \sim \mathcal{U}[1, T-N]$, where $T = 30$ is the maximum number of iterations. The model runs N iterations without gradients followed by K iterations with gradients; loss is computed and backpropagated only over the final K iterations. This is intended to discourage the model from learning iteration-specific shortcuts. At test-time, we unroll the model to 500 loops to ensure it has enough time to solve harder problems.

Experiments were conducted in PyTorch [Paszke et al., 2019], and analyses used NumPy [Harris et al., 2020] and Pandas [Wes McKinney, 2010].

Table 2: Hyperparameter summary for all experiments.

Hyperparameter	**Value**
Hidden dimension	256
Attention heads	8 (head dim 32)
MLP hidden dimension	1024
Activation	GELU
Normalization	RMSNorm
Optimizer	AdamW
Weight decay	0.01
Batch size	500
Epochs	100
LR warmup	Epochs 0–9 (exponential)
LR cooldown	$10\times$ reduction at epoch 60
Gradient clip norm	1
Max iterations (T)	30

B.1.3 Compute Usage

Our final experiments used 120 NVIDIA B200 GPU-hours and 35 MI325X GPU-hours. Additional experiments not included approximately doubled used compute. All models were trained on a single GPU. We trained all models with BF16 casting and sped up training via PyTorch's compile feature.

B.2 Anisotropy Analysis

Figure 4 in Section 5.1 showcases the stability regions for a simple one-layer model with overlapping eigenvectors. To understand how these regions affect the resulting chosen Jacobian eigenvalues, we conduct a simple projection-based experiment.

In particular, we begin by sampling points from $\mathcal{N}(0, \sigma^2 I_{2\times 2})$ and then project them onto the internal and external stability regions. For each projected point, we calculate two metrics to analyze anisotropy:

Table 3: Projected-point anisotropy; $n = 10000$ draws per σ. Parens after means: SE ($\hat{\sigma}/\sqrt{n}$)

	log-range				balance			
	mean		median		mean		median	
σ	int	ext	int	ext	int	ext	int	ext
0.5	9.3359 (0.0831)	1.1524 (0.0102)	4.6835	0.8946	0.2250 (0.0030)	0.4417 (0.0028)	0.0092	0.4088
1	9.6840 (0.0865)	1.2031 (0.0098)	4.6835	1.0320	0.2225 (0.0030)	0.4144 (0.0027)	0.0092	0.3563
2	10.1580 (0.0888)	1.4412 (0.0093)	4.9472	1.4045	0.1803 (0.0026)	0.3315 (0.0025)	0.0071	0.2455
4	11.1402 (0.0884)	1.9594 (0.0106)	18.1294	1.7632	0.0943 (0.0017)	0.2197 (0.0020)	0.0000	0.1715

- Log-range: $\left|\log \frac{|x|+\epsilon}{|y|+\epsilon}\right|$, where more isotropic points have values closer to 0.
- Balance: $\frac{\min(|x|,|y|)+\epsilon}{\max(|x|,|y|)+\epsilon}$, where more isotropic points have values closer to 1.

Table 3 showcases the result of this simulation. Across all choices of σ, internal recall is *substantially* more anisotropic, having a median log-range at least three times as large as external recall and a median balance at least 4 times as small.

We note that this experiment is purely a toy model designed for intuition, and degrades with misaligned eigenvectors since the spectral radius depends on a product of non-commuting matrices whose eigenvalues do not give the complete information. However, the result transfers empirically to real-world models: in B.3, we find that internal models with pre-norm never converge to a fixed point while external pre-norm can – suggesting that indeed, the difference in Jacobian geometry at the fixed point for internal and external models do have a substantive effect on stability.

B.3 Jacobian spectral shrinkage analysis

Outer normalization could stabilize iteration and improve convergence in two ways: directly, by shrinking the Jacobian at the fixed point, or indirectly, by pushing tokens towards a shared direction (the token clustering known to arise under post-norm [Karagodin et al., 2025]) and collapsing the state. Here, we directly separate the two of these by conducting experiments analyzing both the spectral radii at convergence and the extent to which tokens are clustered at convergence.

In particular, we conduct power iteration on the full-loop Jacobian after convergence on Sudoku runs, to compare the spectral radius for models with and without outer normalization. We conduct these analyses on both single-block and double-block models, to confirm that all findings transfer beyond just one FFN-Attention combination. We find multiple important results.

First, all converged models have a full-loop spectral radius less than 1, while many have a spectral radius of W_x greater than one. This showcases the importance of considering the full-loop Jacobian rather than just the individual recall matrix. Second, removing the post-norm from converged models drastically increases the full-loop Jacobian spectral radius, indicating that post-norm does indeed act as a lever for controlling the spectral radius. At the same time, converged states are *less* clustered than at initialization, with cosine similarity dropping by more than half for all one-block converged sudoku runs, with similar results for both normalized and unnormalized models. Two-block models show the same qualitative pattern, but in three of four post-norm configurations their fixed points are noticeably more clustered than two-block pre-norm ones (0.29–0.36, vs 0.20-0.21). These results suggest that while post-norm token clustering is not the mechanism behind its stability at this scale, post-norm at scale may indeed induce token clustering in looped models as in Geiping et al. [2025].

B.4 Effect of learning rates on $\rho\left(\frac{\partial g(x_t,x_0)}{\partial x_t}\right)$

In Section 5.1, we discuss the effects of learning rate on the spectral radius of $\frac{\partial g(x^*,x_0)}{\partial x^*}$. Figure 6 makes this explicit: across all problems and recall choices (averaged over pre/peri norm), higher LR corresponds to larger spectral radius for $\frac{\partial g(x^*,x_0)}{\partial x^*}$. We use $g(x_t, x_0) = W_x x_t + W_0 x_0$, so $\frac{\partial g(x^*,x_0)}{\partial x^*} = W_x$.

Table 4: Full-loop Jacobian spectral radii for Sudoku models with one or two blocks, evaluated on five validation inputs per configuration. Full-loop spectral radii are averaged over inputs for which model iteration converged. The "no ϕ" column tests the full-loop spectral radius with the post-norm removed, and is thus only valid on post-norm models.

Norm	Recall	LR	Blocks	$\rho(W_x)$	$\rho(M_{\text{full}})$	$\rho(M_{\text{full}}, \text{no } \phi)$	Conv.	Hard acc.
post	external	1e-3	1	1.61	0.96	5.59	4/5	16.2
post	external	1e-3	2	0.86	0.93	18.7	3/5	13.2
post	external	3e-3	1	3.90	0.96	39.1	4/5	2.4
post	external	3e-3	2	1.54	0.91	1,248	3/5	2.6
post	internal	1e-3	1	0.78	0.89	3.96	2/5	37.1
post	internal	1e-3	2	0.75	0.89	17.4	2/5	33.8
post	internal	3e-3	1	0.87	0.89	28.3	3/5	9.3
post	internal	3e-3	2	0.97	0.95	2,130	3/5	6.4
pre	external	1e-3	1	0.84	0.96	n/a	5/5	21.0
pre	external	1e-3	2	0.65	0.96	n/a	3/5	17.2
pre	external	3e-3	1	1.06	0.97	n/a	4/5	0.0
pre	external	3e-3	2	0.63	0.93	n/a	4/5	0.0
pre	internal	1e-3	1	1.13	—	n/a	0/5	1.6
pre	internal	1e-3	2	1.10	—	n/a	0/5	5.5
pre	internal	3e-3	1	2.29	—	n/a	0/5	0.3
pre	internal	3e-3	2	2.27	—	n/a	0/5	0.2

Table 5: Token cosine similarity at initialization and at converged fixed points for one and two block Sudoku models, averaged over converged inputs (of five) per configuration. Only configurations with converged inputs are shown.

Norm	Recall	LR	Blocks	Initial	Fixed point
post	external	1e-3	1	0.58	0.23
post	external	1e-3	2	0.61	0.29
post	external	3e-3	1	0.54	0.24
post	external	3e-3	2	0.50	0.30
post	internal	1e-3	1	0.56	0.23
post	internal	1e-3	2	0.58	0.36
post	internal	3e-3	1	0.55	0.17
post	internal	3e-3	2	0.53	0.12
pre	external	1e-3	1	0.60	0.18
pre	external	1e-3	2	0.63	0.20
pre	external	3e-3	1	0.61	0.23
pre	external	3e-3	2	0.66	0.21

B.5 Effect of $\rho\left(\frac{\partial g(x_t, x_0)}{\partial x_t}\right)$ on External Recall Performance

In section 5.1, we discuss how, without outer normalization, external recall can have difficulty creating locally-attractive fixed points when $\frac{\partial g(x^*, x_0)}{\partial x^*}$ becomes large, as the other eigenvalues have to compensate by shifting towards a complex unit ball centered around -1. Here, we provide empirical evidence for this fact: Figure 5 compares final (normalized by task) accuracy for external recall, unnormalized models across tasks as a function of $\rho\left(\frac{\partial g(x^*, x_0)}{\partial x^*}\right) = \rho\left(\frac{\partial g(x_t, x_0)}{\partial x_t}\right) = \rho(W_x)$ for linear recall mechanism $g(x_t, x_0) = W_x x_t + W_0 x_0$. Indeed, we see that external recall performs substantially worse on both val and hard data as the spectral radius of W_x grows beyond 1, with no run with $\rho(W_x) > 1.1$ achieving non-zero accuracy on hard data.

B.6 Seed-Averaged Results for All Models

In Tables 6 and 7, we provide the validation and hard accuracy for all (problem, norm, lr, recall) tuples, averaged over 10 seeds. We also provide standard deviations (SD) over all seeds for uncer-

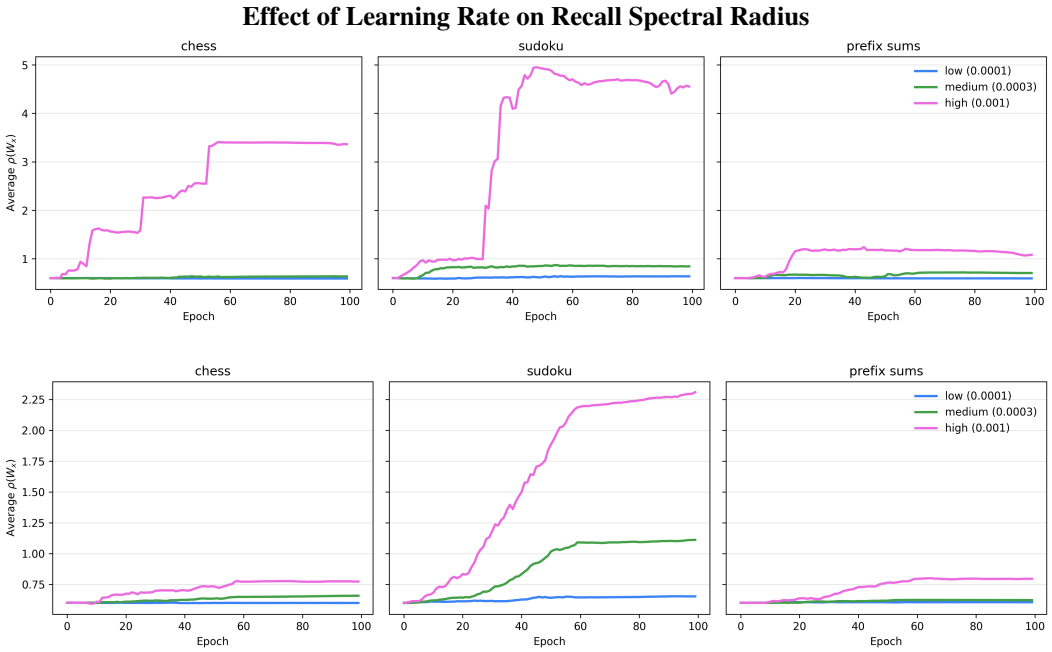


Figure 6: Effect of LR on non-outer-normalized internal and external recall Jacobian spectral radius. Top is external recall, bottom is internal recall. Chess and prefix sums use $(0.0001, 0.0003, 0.001)$; sudoku uses $(0.0003, 0.001, 0.003)$

tainty analysis. We bold the best result in each group (i.e. validation sudoku, or hard chess) and underline the second best.

Each value is "total" accuracy; i.e. not bitwise accuracy, but rather the percent of inputs whose answer is completely correct. Sudoku consistently required higher learning rates to reach equivalent accuracy; as such, we report higher LR here and report the full LR sweep in Appendix B.7.

We note that for fixed-length models on prefix-sums, using sliding-window attention does not provide a large enough receptive field over 15 layers, ensuring 0% accuracy. For fair comparisons, we remove this model from our analysis on hard data, as using a full-length model would unfairly correlate failures of positional embedding mechanisms with that of generalization ability.

B.7 Sudoku Results

For transparency, we report our results on sudoku across all four learning rates in Table 8. We found throughout our analysis that sudoku consistently needed larger learning rates regardless of model architecture.

Table 6: Accuracy on validation (easy distribution, but not trained on) data. SD is given as ±; bolding is best, underlined is second best.

Norm	Recall	Chess			Sudoku			Prefix_sums		
		1×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-4}	1×10^{-3}	3×10^{-3}	1×10^{-4}	3×10^{-4}	1×10^{-3}
gru	internal	57.1±6.1	71.8±1.7	67.4±6.3	28.9±10.3	35.2±9.0	25.3±5.1	**100.0±0.0**	84.9±32.3	86.3±30.5
	external	70.3±0.9	75.0±1.1	**78.3±1.0**	35.0±1.2	40.1±2.3	0.0±0.0	**100.0±0.0**	88.2±31.0	29.1±46.8
	none	37.3±14.2	43.9±5.2	21.1±15.0	6.7±14.0	4.9±8.8	19.3±8.1	95.0±7.8	55.5±48.0	8.2±17.1
	fixed	72.4±0.6	76.2±0.5	77.9±0.6	2.8±0.1	3.0±0.1	0.8±1.3	2.8±5.6	0.1±0.3	0.0±0.0
peri	internal	40.9±3.5	55.1±1.7	58.7±2.4	29.9±2.1	42.6±3.6	26.0±7.7	99.3±0.8	98.9±1.7	76.3±31.2
	external	66.1±2.5	72.7±1.5	45.7±39.4	45.8±2.2	51.7±2.0	37.6±13.4	**100.0±0.0**	90.0±31.6	10.0±31.6
	none	28.5±3.6	27.3±4.7	16.2±5.3	25.1±3.5	28.0±3.8	2.4±7.2	34.7±28.2	0.0±0.0	0.0±0.0
	fixed	71.4±0.7	74.3±0.8	76.7±0.6	2.9±0.1	3.1±0.1	3.2±0.2	26.5±26.2	46.1±25.5	5.0±9.0
post	internal	28.3±6.2	70.6±1.4	68.6±9.0	40.5±1.7	**57.8±1.5**	35.5±13.1	99.9±0.3	99.0±1.3	19.2±39.4
	external	62.1±1.8	72.5±1.6	73.3±3.5	32.0±1.6	47.3±2.2	30.0±4.3	99.7±0.3	99.6±0.8	29.1±46.8
	none	33.4±13.2	21.6±6.2	0.3±0.9	37.5±4.1	2.7±7.7	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
	fixed	70.4±0.4	75.0±0.7	63.8±33.6	2.7±0.0	3.3±0.1	0.0±0.0	23.6±10.5	15.3±17.1	0.0±0.0
pre	internal	38.7±3.3	47.7±7.4	45.5±3.6	11.2±7.3	27.1±4.9	20.8±3.4	98.7±1.1	96.8±7.8	79.8±33.2
	external	69.6±0.9	67.4±23.7	0.0±0.0	43.9±1.7	52.2±2.3	6.4±13.6	**100.0±0.0**	79.7±42.0	0.0±0.0
	none	31.3±4.7	27.0±5.6	14.2±7.4	5.4±5.7	14.2±11.0	13.8±7.1	80.4±32.1	9.3±27.4	0.0±0.0
	fixed	71.2±0.5	75.1±0.6	78.0±0.5	2.8±0.0	3.1±0.1	3.9±0.9	42.1±15.4	60.3±24.9	15.3±17.4

Table 7: Accuracy on hard data. SD is given as ±; bolding is best, underlined is second best.

Norm	Recall	Chess			Sudoku			Prefix_sums		
		1×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-4}	1×10^{-3}	3×10^{-3}	1×10^{-4}	3×10^{-4}	1×10^{-3}
gru	internal	26.7±3.4	36.2±1.3	32.7±5.2	3.5±3.3	6.3±4.0	0.8±0.7	92.9±15.5	78.5±41.4	72.1±31.5
	external	34.9±0.8	38.4±0.9	**41.3±1.2**	5.2±1.0	8.5±1.8	0.0±0.0	**99.9±0.2**	84.4±31.3	28.9±46.6
	none	14.5±5.8	17.2±2.7	7.9±5.7	1.2±2.6	0.4±0.9	0.4±0.7	0.0±0.0	0.0±0.0	0.0±0.0
	fixed	35.5±0.7	39.3±0.6	40.5±0.8	0.0±0.0	0.0±0.0	0.0±0.0	—	—	—
peri	internal	15.8±2.0	24.0±1.1	25.6±2.0	3.4±1.3	14.4±4.3	2.2±2.0	0.0±0.0	0.0±0.0	0.1±0.1
	external	31.1±2.1	36.8±1.5	23.8±20.5	12.8±2.2	20.3±3.6	8.2±3.2	81.5±30.1	90.0±31.6	10.0±31.6
	none	10.4±1.7	10.2±1.9	5.8±1.7	2.8±1.2	3.1±2.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
	fixed	34.6±0.5	37.1±0.7	39.0±0.6	0.0±0.0	0.0±0.0	0.0±0.0	—	—	—
post	internal	13.1±2.7	35.5±1.5	33.1±7.0	9.1±1.5	**34.7±2.8**	7.4±4.0	73.7±37.7	98.8±1.5	14.4±32.0
	external	28.0±1.5	36.3±1.4	37.0±3.1	3.4±0.9	16.8±3.9	2.6±1.4	80.1±33.3	99.5±0.9	19.5±40.3
	none	12.7±4.8	8.5±2.9	0.1±0.4	8.5±1.6	0.1±0.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
	fixed	34.0±0.6	37.5±0.6	33.7±17.8	0.0±0.0	0.0±0.0	0.0±0.0	—	—	—
pre	internal	14.9±1.6	19.6±3.7	18.1±1.9	0.0±0.0	2.1±1.7	0.1±0.1	0.0±0.0	0.0±0.0	0.1±0.3
	external	34.4±0.5	34.7±12.2	0.0±0.0	10.9±1.7	21.3±4.0	0.6±1.4	95.6±13.0	69.7±46.6	0.0±0.0
	none	11.6±2.0	10.1±2.0	5.0±2.0	0.0±0.0	0.5±1.5	0.1±0.2	0.0±0.0	0.0±0.0	0.0±0.0
	fixed	34.8±0.4	37.8±0.6	40.5±0.4	0.0±0.0	0.0±0.0	0.2±0.2	—	—	—

Table 8: Validation and hard accuracy for sudoku across all four learning rates. SD is given as ±; bolding is best, underlined is second best.

Norm	Recall	Validation Accuracy				Hard Accuracy			
		1×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-3}	1×10^{-4}	3×10^{-4}	1×10^{-3}	3×10^{-3}
gru	internal	25.5±1.0	28.9±10.3	35.2±9.0	25.3±5.1	0.3±0.2	3.5±3.3	6.3±4.0	0.8±0.7
	external	24.6±1.2	35.0±1.2	40.1±2.3	0.0±0.0	1.0±0.8	5.2±1.0	8.5±1.8	0.0±0.0
	none	8.8±5.4	6.7±14.0	4.9±8.8	19.3±8.1	0.0±0.0	1.2±2.6	0.4±0.9	0.4±0.7
	fixed	2.5±0.1	2.8±0.1	3.0±0.1	0.8±1.3	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
peri	internal	2.2±0.9	29.9±2.1	42.6±3.6	26.0±7.7	0.0±0.0	3.4±1.3	14.4±4.3	2.2±2.0
	external	25.0±0.9	45.8±2.2	51.7±2.0	37.6±13.4	1.0±0.6	12.8±2.2	20.3±3.6	8.2±3.2
	none	0.7±0.8	25.1±3.5	28.0±3.8	2.4±7.2	0.0±0.0	2.8±1.2	3.1±2.0	0.0±0.0
	fixed	2.5±0.1	2.9±0.1	3.1±0.1	3.2±0.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
post	internal	14.1±7.9	40.5±1.7	**57.8±1.5**	35.5±13.1	1.1±0.8	9.1±1.5	**34.7±2.8**	7.4±4.0
	external	15.3±3.7	32.0±1.6	47.3±2.2	30.0±4.3	0.7±0.5	3.4±0.9	16.8±3.9	2.6±1.4
	none	9.0±6.9	37.5±4.1	2.7±7.7	0.0±0.0	0.5±0.6	8.5±1.6	0.1±0.2	0.0±0.0
	fixed	2.1±0.1	2.7±0.0	3.3±0.1	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
pre	internal	1.7±0.5	11.2±7.3	27.1±4.9	20.8±3.4	0.0±0.0	0.0±0.0	2.1±1.7	0.1±0.1
	external	24.7±1.2	43.9±1.7	<u>52.2±2.3</u>	6.4±13.6	1.0±0.7	10.9±1.7	<u>21.3±4.0</u>	0.6±1.4
	none	0.4±0.4	5.4±5.7	14.2±11.0	13.8±7.1	0.0±0.0	0.0±0.0	0.5±1.5	0.1±0.2
	fixed	2.4±0.1	2.8±0.0	3.1±0.1	3.9±0.9	0.0±0.0	0.0±0.0	0.0±0.0	0.2±0.2

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We explicitly define our theoretical and empirical findings and justify them throughout the paper. We take care to explain but not over-exaggerate our findings.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We note limitations of our paper in Section 5.2 and the conclusion. In particular, we note that our theoretical results provide necessary but not sufficient conditions, that we test our results on only small models, and that we do not discuss basin size.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provide assumptions necessary for our proofs, and cite relevant theorems from the literature. Appendix A contains all formal proofs of theorems mentioned in the paper.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: In addition to providing a repository to reproduce our results, we describe all architectural decisions and hyperparameters necessary to exactly reproduce our experiments.

Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: In our supplementary materials, we provide an anonymized repository with details on reproducibility. We also provide a link to our repository on the second page.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We describe all hyperparameters and necessary components to reproduce our results. We also provide a github repository with exact methods for reproduction.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All accuracy results for the main sweep in Appendix B.6 are reported as mean $\pm$ standard deviation over 10 random seeds. Comparisons between configurations in the main text are judged by whether the difference in means exceeds two standard errors (calculated as the standard deviation divided by $\sqrt{10}$). Our anisotropy analysis (Appendix B.2) also reports uncertainty over its sampled matrices, and the full-loop Jacobian and cosine similarity experiments (Appendix B.3) are averaged over five validation inputs.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section B.1.3 describes the necessary compute to run the final experiments, as well as additional compute beyond those experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our work uses only publicly licensed datasets, no personal or sensitive data, no human subjects. All licenses and attributions are respected.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: We analyze the stability of looped transformers, and thus all potential societal impacts relate to the larger field of transformer models and not to our specific paper.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: This paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All third-party assets are open-source and credited. We release our code under an MIT license, compatible with previously-mentioned licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

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

13. **New assets**

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

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not include crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not include crowdsourcing or research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were used only for writing and organizing purposes.

    Guidelines:

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