A Mechanistic Analysis of Looped Reasoning Language Models

Hugh Blayney
University of Oxford
hugh@robots.ox.ac.uk

Álvaro Arroyo
University of Oxford
alvaro.arroyo@eng.ox.ac.uk

Johan Obando-Ceron
Mila – Quebec AI Institute
Université de Montréal

Pablo Samuel Castro
Mila – Quebec AI Institute
Université de Montréal

Aaron Courville
Mila – Quebec AI Institute
Université de Montréal

Michael Bronstein
University of Oxford
AITHYRA

Xiaowen Dong
University of Oxford

Abstract

Reasoning has become a central capability in large language models. Recent research has shown that reasoning performance can be improved by looping an LLM's layers in the latent dimension, resulting in *looped reasoning language models*. Despite promising results, few works have investigated how the internal dynamics and computation of looped models differ from that of standard feedforward models. In this paper, we conduct a mechanistic analysis of latent states and layer behavior in looped language models. We analyze cyclic recurrence and show that for many of the studied models each layer in the cycle converges to a distinct fixed point; consequently, the recurrent block follows a consistent cyclic trajectory in the latent space. We provide evidence that as these fixed points are reached, attention-head behavior stabilizes, leading to constant behavior – *stages of inference* – across recurrences. Empirically, we discover that recurrent blocks tend to organize their computation such that it closely mirrors that of feedforward models, repeating this behavior in depth with each iteration. We study how architectural choices influence the emergence and stability of these cyclic fixed points and stages of inference, providing practical guidance for looped language model design. Code implementation can be found here.

1 Introduction

The vast majority of current LLMs are based on the Transformer architecture (Vaswani et al., 2017), which comprises a sequence of blocks traversed in a feedforward manner to predict the next token. As the capability of these models increased, attention turned to eliciting *reasoning* capabilities in LLMs by increasing test-time computation, commonly through chain-of-thought (CoT) prompting (Wei et al., 2022) or reinforcement-learning-based fine-tuning, notably DeepSeek-R1 (Guo et al., 2025). More recently, research has explored building reason-

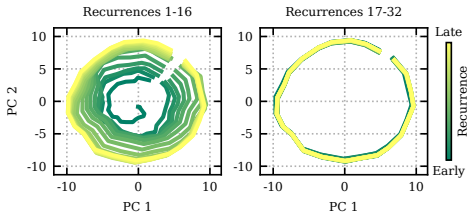


Figure 1: **Latent states after each block in a looped model frequently tend towards *separate* fixed points**: a consistent trajectory in latent space. Illustrative; from a randomly initialized model.

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

ing capabilities directly into the model architecture via *recurrent looping* (Wang et al., 2025; Jolicoeur-Martineau, 2025), where additional test-time compute is spent by taking more recurrent steps, echoing early designs in this direction (Graves, 2016). Despite growing empirical success, the mechanisms underlying these models remain poorly understood, as well as their benefits and limitations when compared to feedforward computation.

In this paper, we study the dynamics of looped LLMs and how this impacts their organization of computation in depth. To this end, we study the macroscopic behavior of attention patterns, and the functional utility of latent states throughout the model depth via their performance on downstream tasks: this draws on existing research into *stages of inference* (Lad et al., 2025; Queipo-de-Llano et al., 2026) and related work (Skean et al., 2025; Csordás et al., 2025; Ali et al., 2025). We show that looped LLMs often reach a fixed point with repeated looping, and we establish that stable recurrence necessitates the emergence of a *constant cyclic trajectory* in latent space and consistent, repeated computational cycles. Remarkably, we find that the macroscopic computational behavior of looped LLMs frequently mirrors that of comparable feedforward models, but repeating within each recurrent block. Finally, we conduct a controlled experiment whereby we isolate the *stability* property of a looped LLM and demonstrate that stable latent dynamics result in predictable test-time extrapolation and consistent computational patterns. These findings clarify both the relationship between looped and feedforward LLMs, and the role of stability in looped LLM computation, facilitating further development and the transfer of improvements between these fields.

Key insights:

Stability and Cyclic Trajectories: We establish that many looped models converge to *constant cyclic trajectories* in latent space. We show that this stability induces fixed, layer-specific attention patterns (Sec. 3) and identify the architectural drivers – specifically input injection and normalization – that facilitate this convergence (Sec. 3.2).

Recurrent Stages of Inference: We find that functional computational stages are repeated within each recurrence, and these are organised such that each loop mirrors the behavior of the entire depth of comparable feedforward models (Sec. 4). We provide preliminary evidence that stability of latent dynamics is a prerequisite for consistent extrapolation of these stages; unstable models fail to extrapolate to recurrences beyond those seen in training, while their functional stages diverge and drift across iterations (Sec. 5).

2 Preliminaries

2.1 Looped Transformers (LTs)

In this section we introduce LTs and define the notation that we will use throughout the paper. We represent the input sequence to our transformer of length T and dimension D as $\boldsymbol{X} \in \mathbb{R}^{T\times D}$. Following the notation of Yudin et al. (2026) we define $\texttt{Attn}$ as a map $f : \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}$:

$$\texttt{Attn}(\boldsymbol{X}) = \text{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q\boldsymbol{W}_K^\top\boldsymbol{X}^\top}{\sqrt{d}}\right)\boldsymbol{X}\boldsymbol{W}_V = \text{softmax}(A(\boldsymbol{X}))\boldsymbol{X}\boldsymbol{W}_V, \tag{1}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{D\times d}$ are projection matrices and A is defined for convenience. A transformer block typically comprises an attention mechanism and a position-wise MLP as follows:

$$\hat{\boldsymbol{X}} = n_2\left(\boldsymbol{X} + \texttt{Attn}(n_1(\boldsymbol{X}))\right), \qquad \boldsymbol{X}' = n_4\left(\hat{\boldsymbol{X}} + \text{MLP}(n_3(\hat{\boldsymbol{X}}))\right), \tag{2}$$

where n_1, n_2, n_3, n_4 are each optional norms, borrowing from the notation of Geiping et al. (2025). We denote the action of a Transformer block $\text{B} : \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}$ as $\boldsymbol{X}' = \text{B}(\boldsymbol{X})$, and refer to the intermediate hidden-state matrices $\boldsymbol{X}$ between blocks as the **residual stream**.

LTs are Transformers that utilize "recurrence in depth" – that is, they reapply layers to repeatedly act on the latent states. Recent research has identified that an effective method (Bae et al., 2025) to achieve this is via *cyclic recurrence*: a fixed sequence of layers is repeated in a "cyclic" pattern. We define a k-stacked cyclic block as $\text{S}_k(\boldsymbol{X}) = \text{B}_k(\text{B}_{k-1}(\dots\text{B}_1(\boldsymbol{X})\dots))$.

In the case of Geiping et al. (2025); McLeish et al. (2026) this stacked block may also take an additional input $\boldsymbol{Z} \in \mathbb{R}^{T\times D}$ which is typically initialized from a Normal distribution, as

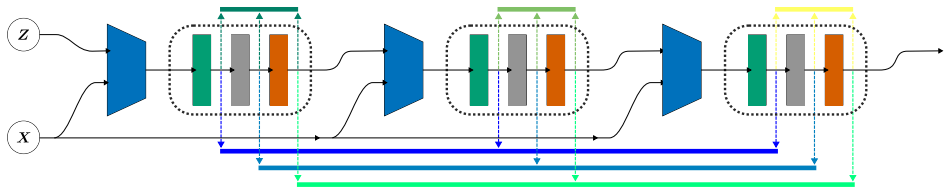


Figure 2: An illustrative LT architecture. Rectangles represent Transformer blocks; weights are shared between rectangles of the same color. Dashed regions denote stacks of Transformer blocks: the same stack is looped, resulting here in cyclic recurrence. The two inputs $\boldsymbol{Z}, \boldsymbol{X}$ denote the initial latent state and (optional) input injection respectively. For models with input injection (Huginn-0125, the Retrofitted series, Parcae), $\boldsymbol{X}$ is typically the output of the prelude layers and $\boldsymbol{Z}$ is the randomly initialised latent state. For Ouro, without input injection or prelude layers, $\boldsymbol{X} = \boldsymbol{0}$ and $\boldsymbol{Z}$ is the input to the Transformer (token embeddings). The **green-yellow** bars above the diagram indicate grouping metrics by recurrence and the **blue-green** bars indicate grouping by layer.

well as the original input to the recurrent section: this is known as *input injection* (Bai et al., 2019; Anil et al., 2022), and the two inputs are projected into common feature space $\mathbb{R}^D$ before the block is applied. In the case of input injection, a k-stacked block therefore becomes $\mathrm{S}_k(\boldsymbol{X}, \boldsymbol{Z}) = \mathrm{B}_k(\mathrm{B}_{k-1}(\ldots \mathrm{B}_1([\boldsymbol{X}, \boldsymbol{Z}]\boldsymbol{W}_I)\ldots))$, where $[\cdot,\cdot]$ denotes concatenation in the channel dimension and $\boldsymbol{W}_I \in \mathbb{R}^{2D\times D}$ is a learned projection matrix. This allows us to define a $(k \otimes l)$-Recurrent block as a k-stacked block repeated l times, $R_{l,k}(\boldsymbol{X}) = \mathrm{S}_k(\mathrm{S}_k(\ldots \mathrm{S}_k(\boldsymbol{X})\ldots))$, which with input-injection becomes $R_{l,k}(\boldsymbol{X}, \boldsymbol{Z}) = \mathrm{S}_k(\boldsymbol{X}, \mathrm{S}_k(\ldots \boldsymbol{X}, \mathrm{S}_k(\boldsymbol{X}, \boldsymbol{Z}))\ldots))$. An example $(3 \otimes 3)$ LT utilizing input injection is visualised in Fig. 2. An LT may include feedforward layers both before and after the Recurrent block: following the convention of Geiping et al. (2025), we refer to these as *prelude* and *coda* layers respectively.

We will often refer to *virtual layers*: each virtual layer is one execution of a transformer layer (attention and MLP) during a forward pass. Crucially, this is not the same as the number of unique layers: an LT with 2 layers repeated 8 times has 16 virtual layers. In presenting our results, we **group** metrics computed over these virtual layers in the following ways:

- **By recurrence**: Separate lines are visualized for each complete pass through the recurrent block. The x-axis is percentage-scaled to represent *relative* depth within that block (including prelude/coda), allowing us to overlay and compare successive passes. Shown with **green-yellow** colorbar, with later recurrences colored more yellow.
- **By unique layer**: Separate lines show evolution of each layer across recurrences. Shown with **blue-green** colorbar, with later layers colored more green.

These groupings are visualised above (grouping by recurrence) and below (grouping by layer) Fig. 2, where each color-coded line represents a plotted line with the same color.

2.2 Stages of Inference

Layers in feedforward Transformers exhibit distinct and consistent behaviors that change sharply in depth: Lad et al. (2025) originate the term "stages of inference" and demonstrate how different mechanisms emerge at different depths. Queipo-de-Llano et al. (2026) further develop this viewpoint, proposing three distinct stages:

Mixing: Transformer layers vary significantly in how they "mix" information across tokens (Queipo-de-Llano et al., 2026; Lad et al., 2025). This mixing often peaks in early layers, but architectures like Llama can also exhibit high mixing in middle layers (Barbero et al., 2025, Fig. 6). To quantify this, we define the *Mixing Score* – the average Shannon entropy of attention pattern rows (excluding the first), normalized by the maximum entropy permitted by the causal mask:

$$\text{Mixing Score} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{H(\boldsymbol{A}_{i,:})}{\log(i)} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{-\sum_{j=1}^{i}\alpha_{ij}\log\alpha_{ij}}{\log(i)}. \tag{3}$$

A score of 0 implies attention concentration on a single token e.g. identity, "off-by-one" or sink; 1 implies even distribution across all preceding tokens. In this way, Mixing Score acts as a unified heuristic for information dispersion, capturing the intuition of sink rate (Gu et al., 2025) and preceding attention weight (Lad et al., 2025) without specifying to a particular head taxonomy.

We highlight that this is a modified version of the Mixing Score introduced in Queipo-de-Llano et al. (2026): see App. G.2 for the rationale behind our modifications.

Embedding: Queipo-de-Llano et al. (2026) define this as the "compression" stage, where residual entropy is sharply reduced by massive activations. This compression correlates with a peak in downstream embedding task performance (Skean et al., 2025; Lad et al., 2025). While the looped models studied here do not exhibit consistent residual stream compression – likely due to residual stream norms suppressing the required massive activations, discussed in Sec. 4 and App. G.2.2 – we do observe a clear alignment in downstream embedding performance. We quantify this by training linear probes at every virtual layer to map hidden representations to multiple-choice answers.

Refinement: Later layers are primarily responsible for refining the next-token distribution (Queipo-de-Llano et al., 2026; Lad et al., 2025). We track this behavior using the logit lens (Nostalgebraist, 2020), projecting intermediate hidden states through the model's final normalization and unembedding layers to obtain vocabulary logits. By measuring the accuracy of these projections at every virtual layer, we observe how the looped block iteratively converges on the correct token distribution.

3 Latent Fixed Points in Looped LLMs

It is known that weight-tied Transformers can converge to a fixed point[1] $\boldsymbol{X}' = \mathrm{S}_k(\boldsymbol{X}')$ (Bai et al., 2019; Bansal et al., 2022; Anil et al., 2022). In this section, we explore for which looped LLMs this is true; we investigate the implications on layerwise dynamics; we prove that convergence results in consistent attention patterns; and we investigate the architectural drivers behind fixed-point behavior.

We motivate our work by noting that in cyclic architectures, this block-wise fixed point implies that every layer within the loop reaches its own fixed point:

Proposition 3.1 (Cyclic recurrent blocks reach cyclic fixed points)**.** *Let (l, k)-Recurrent block reach a fixed point $\boldsymbol{X}'$ such that $\mathrm{S}_k(\boldsymbol{X}') = \boldsymbol{X}'$. Then any cyclic permutation of blocks $1, \ldots, k$ will also have reached a fixed point.*

These fixed points are not necessarily the same: the recurrent block can trace a stable cyclic trajectory in latent space as illustrated in Fig. 1. In Sec. 3.1 we confirm that this *cyclic fixed point* behavior is observed frequently in practice.

As the residual stream stabilizes, the attention patterns must similarly stabilize. Assuming both this converged residual stream and bounded inputs (guaranteed by pre-norm in all studied models), tied weights ensure that attention remains consistent across recurrences as shown in Prop. 3.2.

Proposition 3.2 (Recurrent attention patterns change slowly under state convergence)**.** *Fix a layer ℓ in the recurrent block and consider its attention weight matrices to be tied across recurrences (so $\boldsymbol{W}_{Q,\ell}, \boldsymbol{W}_{K,\ell}$ are the same for all t). Let $\|\cdot\|$ denote a submultiplicative matrix norm that is invariant under transposition (e.g. Spectral or Frobenius norms). Assume the corresponding attention inputs are bounded under this norm as $\|\boldsymbol{X}_{\ell,t}\| \le B$ for all t. Define $\kappa_\ell = \|\boldsymbol{W}_{Q,\ell}\boldsymbol{W}_{K,\ell}^\top\|$. Then, writing $\mathcal{S}_\ell(\boldsymbol{X}) := \mathrm{softmax}(A_\ell(\boldsymbol{X}))$ with $A_\ell(\cdot)$ as defined above, for any $t \ge 1$,*

$$\left\|\mathcal{S}_\ell(\boldsymbol{X}_{\ell,t}) - \mathcal{S}_\ell(\boldsymbol{X}_{\ell,t-1})\right\| \;\le\; L_{\mathrm{sm}}\,\frac{2B\,\kappa_\ell}{\sqrt{d}}\,\left\|\boldsymbol{X}_{\ell,t} - \boldsymbol{X}_{\ell,t-1}\right\|,$$

where L_{sm} is a Lipschitz constant of the row-wise softmax with respect to the chosen norm.

3.1 Empirical Validation

Throughout the main body of this paper we investigate the same four looped LLMs: Ouro 1.4B (Zhu et al., 2025), Huginn-0125 (Geiping et al., 2025), retrofitted recurrent Llama (McLeish et al., 2026)

[1] As observed by Geiping et al. (2025), other limiting behaviors can occur. In App. D we show that 1) this is rare, the vast majority of tokens reach a fixed-point and 2) even with other limiting behavior, stages of inference remain constant.

and Parcae 770m (Prairie et al., 2026). These models span a range of design choices (Table 1), the architectural impact of which we isolate in Sec. 3.2. Unless stated otherwise, results are evaluated on 256 random examples from the GSM8k test set; in App. G.2.3 we verify that the conclusions hold on non-reasoning tasks.

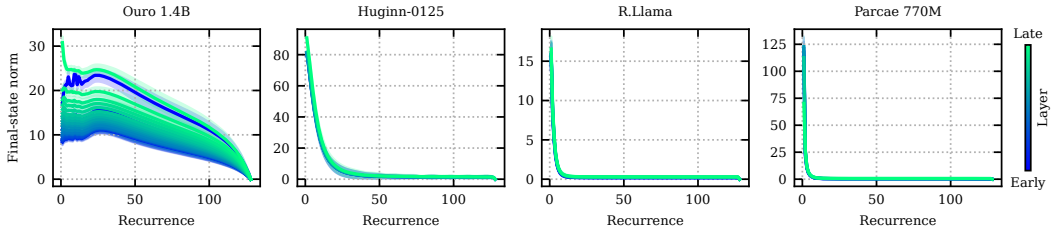


Figure 3: **Residual stream stability across recurrences.** Visualises the norm of the difference from each layer to an "approximate fixed point" (the 128th recurrence). While Huginn-0125, Retrofitted Llama and Parcae converge rapidly, Ouro continually drifts without reaching a fixed point.

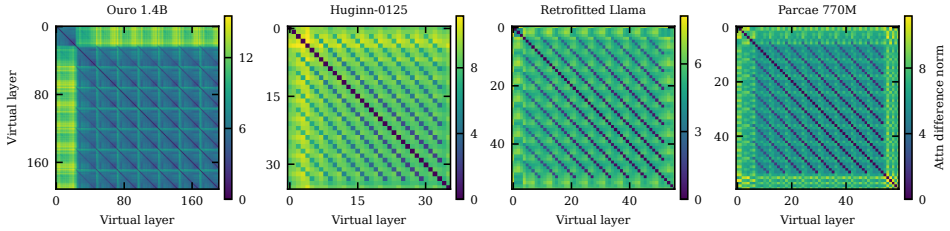


Figure 4: **Cyclic similarity in attention patterns.** Frobenius norms between attention patterns at all virtual layer pairs (8 loops), averaged across the batch and head dimensions. Strong diagonal patterns indicate that attention at any layer is similar across recurrences, confirming Prop. 3.2.

Fig. 3 plots for each layer l the distance from the residual stream at recurrence r to an approximate fixed point after 128 recurrences, $|\boldsymbol{h}_{l,r} - \boldsymbol{h}_{l,128}|_2$. Of the four models tested, only Ouro does not reach a fixed point – and the other models stabilize very quickly. This validates the premise of Prop. 3.2: as a fixed point is approached, the layerwise residual stream changes minimally between recurrences. As predicted by this result, heatmaps of the Frobenius norms between attention patterns (Fig. 4) over eight loops demonstrate high cyclic similarity; the distinct diagonal patterns indicate that a layer's attention pattern is most similar to its own pattern in other recurrent steps. Even though Ouro does not reach a fixed point, over the range visualised in Fig. 4 it still exhibits highly similar attention patterns: Sec. 5 demonstrates that this does not hold at later recurrences.

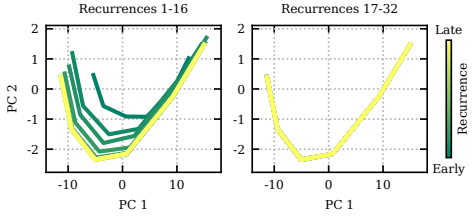


Figure 5: **Latent space (PCA) trajectory for Retrofitted Llama** hidden states at the final sequence position. Overlap in the second plot implies a stable cyclic fixed point.

For models that do reach a fixed point, the recurrent block effectively traces a consistent cycle in latent space: we illustrate this phenomenon via PCA for retrofitted Llama in Fig. 5.

3.2 Impact of Architecture Choices

To isolate stability properties arising from architectural choices from learned behaviors, we evaluate randomly initialized 12-layer models across the architectural variants found in Huginn-0125, Ouro, and retrofitted models, testing each with and without input injection. Fig. 6 plots mean results over five random seeds. We see that while input injection enables stable fixed-point behavior across most norm types, omitting it renders all configurations unstable. Note that the high cosine similarity of pre-norm without input injection is pathological: its residual stream norm grows unbounded, overwhelming the nonlinear block's contribution at each layer and creating the illusion of convergence in cosine

similarity space. Curiously, we discover that the Ouro norm structure renders even input injection unstable.

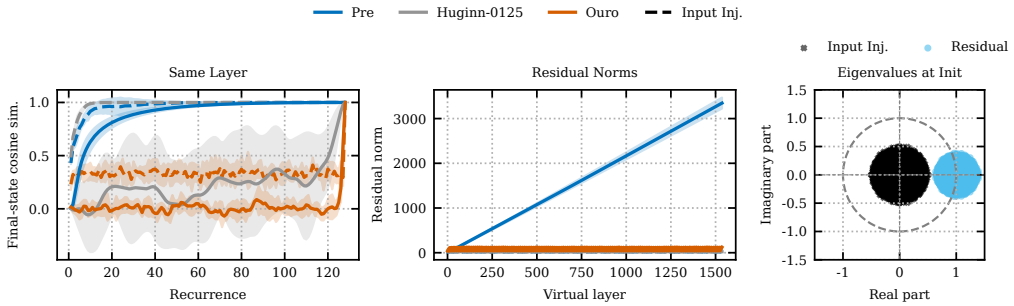


Figure 6: **Impact of Input Injection and Norm Structure on Stability. Left:** Cosine similarity between the first layer's residual stream and its approximate fixed point. **Center:** Residual stream norms. **Right:** Spectral distribution of the recurrent block Jacobian for pre-norm variants, showing how input injection pulls eigenvalues within the stable unit circle. Left and center plots averaged over 5 random seeds.

We can partially explain the stabilizing effect of input injection using the dynamical system framing of Prairie et al. (2026), wherein the recurrent update is written as:

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{e} + \mathcal{R}(\boldsymbol{X}_t, \boldsymbol{E}), \tag{4}$$

where $\mathcal{R}$ captures nonlinear components, $\boldsymbol{e}$ is the input embedding, and $\boldsymbol{x}$ is the hidden state. Linearizing the system (omitting $\mathcal{R}$) yields a necessary convergence condition: the spectral radius $\rho(\boldsymbol{A})$ must be strictly less than 1. Without input injection, $\boldsymbol{A} = \mathbb{I}$ and $\rho(\boldsymbol{A}) = 1$, leaving the system only marginally stable. Prop. 3.3 shows that the standard initialization of input injection layers encourages $\rho(\boldsymbol{A}) < 1$ in high dimensions.

Proposition 3.3. *Assume a looped model which permits the dynamical system formulation of Eq. (4) (e.g. it has no norm on the residual stream). Assume that input injection is via projection by matrix $\boldsymbol{W}_I \in \mathbb{R}^{D\times 2D}$ acting on the concatenated vector $[\boldsymbol{h}_t, \boldsymbol{e}] \in \mathbb{R}^{2D}$, and that $\boldsymbol{W}_I$ is initialized with elements drawn from $\mathcal{N}(0, \sigma_I^2)$. If $\sigma_I < 1/\sqrt{D}$ (as is the case for Xavier initialization (Glorot & Bengio, 2010)) then $\rho(\boldsymbol{A}) < 1$ with probability 1 as $D \to \infty$.*

Empirical measurements of stable *trained* models confirm that $\rho(\boldsymbol{A}) < 1$: Huginn-0125 (see caveat below), Retrofitted Llama, Retrofitted TinyLlama and Retrofitted OLMo-2 have spectral radius 0.24, 0.48, 0.45 and 0.52 respectively. This suggests that training does not necessarily push the spectral radius into the unstable regime. However, this stability is not guaranteed: we direct the reader to Prairie et al. (2026) for a discussion of how $\boldsymbol{A}$ can be constrained to encourage stability.

We highlight that this linear analysis only holds for the Parcae and retrofitted models: Ouro and Huginn-0125 both normalise the residual stream, and as such Eq. (4) does not hold.

4 Stages of Inference in Looped Models Mirror Feedforward Computation

As established in Sec. 3, looped models converge to a regime where attention patterns within individual layers stabilize across recurrences. This stability contrasts sharply with feedforward Transformers, where unconstrained weights allow for abrupt shifts in attention dynamics. Prior work links these transitions to layer behavior and specific stages of inference (Lad et al., 2025; Queipo-de-Llano et al., 2026); we investigate how cyclic weight-sharing reconfigures these stages within the looped architecture.

We first examine attention **mixing** behavior by plotting the Mixing Score across virtual depth (Fig. 7). Rather than distinct mixing behaviors distributed across the entire model depth, we find clear repeating patterns of behavior that repeat within every iteration of the recurrent block. The behavior of individual layers (visualized as solid lines) converges rapidly toward constant behavior, indicating that each unique layer specializes in a specific portion of this mixing cycle.

Fig. 8 demonstrates that this per-cycle organization often mirrors the stages of inference found in feedforward models (App. G.2.1 for correlations). We observe the following: (1) Ouro, despite being trained from scratch with recurrence, adopts a mixing profile that closely mirrors the Llama family. (2) Huginn-0125 does not exhibit similar mixing stages to any known Feedforward models. We attribute this to the repeated normalization of the residual stream at every layer, which suppresses the formation of massive activations that typically drive these stages (Queipo-de-Llano et al., 2026). We provide extended evidence for this hypothesis in App. G.2.2, showing that ablating the MLP responsible for the formation of massive activations results in stages of inference that deviate from those of the feedforward model. (3) Retrofitted models preserve the stages of their base architectures, with the prelude and coda handling the initial and final stages respectively, while the recurrent block repeatedly cycles the "middle" stages. (4) The Parcae models mirror the stages of inference of a "feedforward control" parameter-matched Parcae model trained with only one recurrence. These stages are different from prior analyses of older LLMs, which we attribute to architectural modifications of Nanochat, such as Value Embeddings. However, as with the Ouro and retrofitted results, Parcae models exhibit stages of inference that mirror those of a comparable Feedforward model.

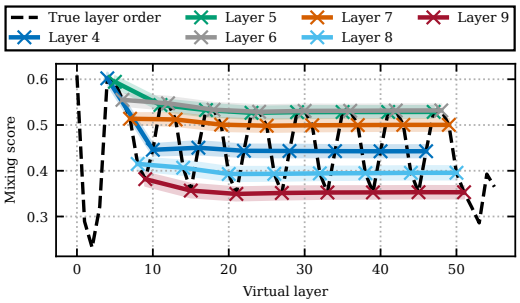


Figure 7: **Cyclic Mixing Stages in Retrofitted Llama.** Solid lines denote unique layers, dashed line the sequence of virtual layers across 8 recurrences. Individual layers converge to constant mixing scores, resulting in a stable, repeating cycle.

The fact that the Ouro and Parcae models independently learn stages of inference during pretraining that mirror comparable feedforward models suggests that this functional decomposition is a robust emergent property. We provide further evidence for this self-organization in App. G.4.

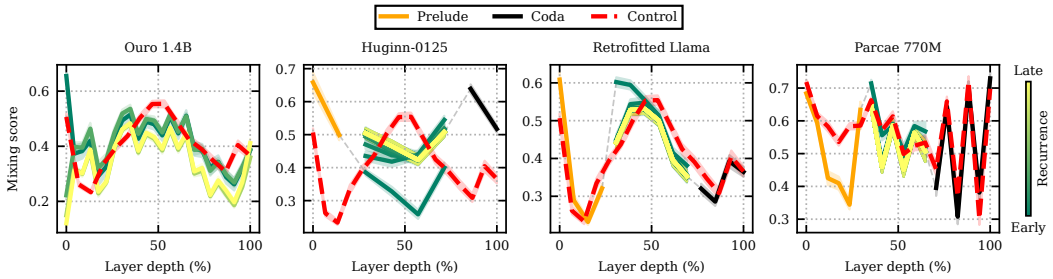


Figure 8: **Internal Mixing Stages per Recurrent Loop.** Ouro resembles Llama mixing stages of inference; the retrofitted models retain the stages of inference of their base models; Parcae resembles a feedforward control of the same architecture.

We characterize **embedding** and **refinement** stages by evaluating per-layer accuracy on downstream tasks: Figs. 9 and 10 respectively. These functional trajectories broadly align with feedforward observations: embedding performance peaks in middle layers, while next-token refinement accuracy increases toward the end of the block. In addition, two looped-specific behaviors emerge: **(1) Repeated "Detokenization":** In Ouro, which lacks a coda to delegate refinement to, early layers in later recurrences transform representations back *out* of the refined next-token space, evidenced by the U-shaped LogitLens accuracy seen within later loops. **(2) Iterative Conceptual Refinement:** In Huginn-0125, embedding performance increases dramatically for the first ~10 recurrences before plateauing. This demonstrates that the model uses looping to iteratively refine the latent representation in a way that is beneficial for downstream embedding tasks.

5 Stability to Unseen Numbers of Recurrences

Sec. 3 establishes that some LT architectures converge to a fixed point while others do not. However, when evaluating these LTs within the range of recurrences on which they were trained (Sec. 4),

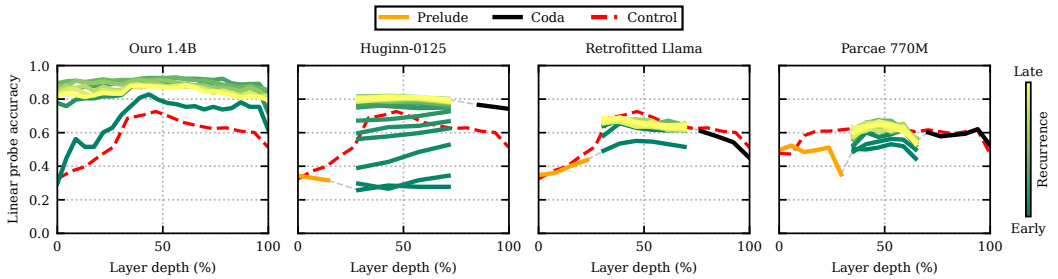


Figure 9: **Linear-probe multiple-choice accuracy on ARC-Easy across model depth.** Residual stream representations for every question–answer pair are mean-pooled over all non-padding tokens; linear probes at each virtual layer are trained on the ARC-Easy training split (underlying model remains frozen); accuracy is the fraction of held-out test questions answered correctly.

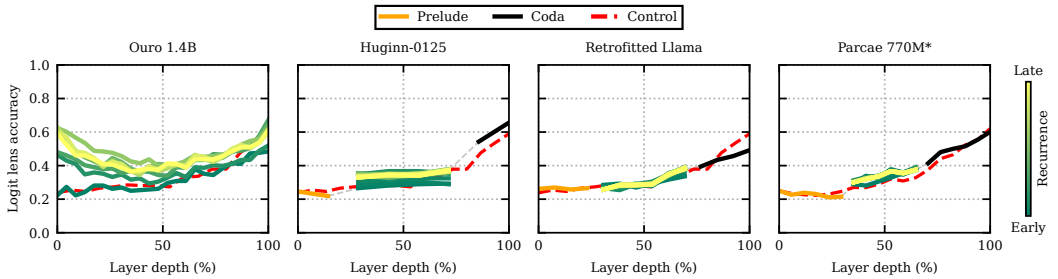


Figure 10: **Logit-lens multiple-choice accuracy on ARC-Easy across model depth.** Residual stream is decoded using the model's final layer normalisation and unembedding; candidate answers are appended to the question and scored by log-probability; accuracy computed as the fraction of held-out questions answered correctly. Parcae additionally applies pre-coda linear projection.

we see both types converge to feedforward stages of inference within each recurrent block. In this section, we demonstrate that stable and unstable LTs exhibit different behavior when *extrapolating* to unseen recurrent depths: models that do not reach a fixed point exhibit **unstable stages of inference**. Conversely, models with recurrent-block-wise stages of inference that also converge to a fixed point will keep enacting these stages of inference for arbitrary test time recurrences.

We first demonstrate this behavior by extending the recurrence depth of Retrofitted Llama and Ouro. For retrofitted Llama (representative of models using input injection), individual layers rapidly reach stable states, ensuring the mixing cycles remain constant to arbitrary depth. By contrast, individual layers in Ouro continually deviate, preventing the model from maintaining a consistent functional profile. The impact on block-wise stages is visualized in Fig. 11: at later out of distribution recurrences, Ouro's stages of inference drift significantly from their trained configuration.

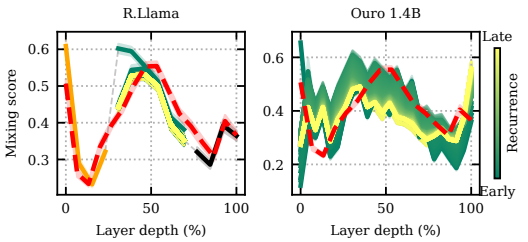
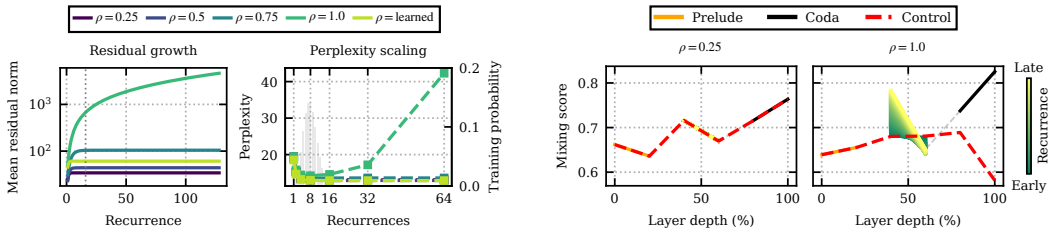


Figure 11: **Impact of fixed-point behavior on Stages of Inference:** stable models preserve their functional stages whereas unstable models deviate

This mechanistic stability appears to correlate with a model's ability to utilize additional test-time compute. Architectures that maintain stable stages of inference avoid the performance deterioration typically associated with "overthinking" in deep networks (Bansal et al., 2022). For instance, existing work shows that while extending recurrence beyond the training budget harms the performance of the unstable Ouro (Zhu et al., 2025, Tab. 10), the performance of the stable Huginn-0125 remains constant or improves in the same extrapolation regime (Geiping et al., 2025, Fig. 1). These existing results suggest that functional consistency – underpinned by fixed-point convergence – encourages reliable test-time scaling.

We further test this hypothesis by attempting to *isolate* the stability of a Looped LLM with minimal other changes to the training recipe. To this end we define a variant of the Parcae 140M (Prairie et al., 2026) model in which we replace the $\boldsymbol{A}$ matrix (the state-transition that carries the residual stream between recurrences, see Eq. (4)) with a single ρ scalar, allowing us direct (and trivial) control over the linearized spectral radius. We train these models with Poisson-sampled recurrent depth, mean 8 and capped at 16. As we sweep this ρ value from 0.25 up to 1.0, visualized in Fig. 12a, we see that a spectral radius of 1.0 results in an unstable system, with continued residual growth and perplexity that increases outside of the depth range seen at training. Fig. 12b links this failure of test-time depth extrapolation to a loss of stable stages of inference: the mixing stages of the $\rho = 0.25$ model remain stable, adhering closely to the stages seen at the largest training depth, while those of the $\rho = 1.0$ model drift significantly.



(a) Residual and performance scaling

(b) Stages of Inference stability

Figure 12: **Isolated impact of stability.** A controlled set of Parcae-style models where we directly control the spectral radius ρ of the residual state transition. Stable models exhibit stable residual norms and consistent perplexity in the extrapolation regime (Fig. 12a), and consistent stages of inference in the extrapolation regime (Fig. 12b, left). Unstable models exhibit the opposite. Models are trained with Poisson-sampled recurrences of mean 8 and capped at 16, visualized in grey in Fig. 12a, right. The controls in Fig. 12b are the mixing scores at the 16th recurrence: the largest seen at train time. Overlaid on this are the mixing stages for all recurrences *after* the 16th, up to 128 recurrences. "Learned" ρ follows the standard Parcae 140m model architecture with a per-channel diagonal $\boldsymbol{A}$ matrix.

6 Related Work

Looped and Recurrent Transformers. Weight-tied Transformers originated with Universal Transformers (Dehghani et al., 2019), with recent extensions focusing on sparse/conditional computation (Tan et al., 2023; Csordás et al., 2024) and reasoning-centric designs like HRM (Wang et al., 2025) and TRM (Jolicoeur-Martineau, 2025). Within language modeling, we highlight Huginn-0125 (Geiping et al., 2025), Ouro (Zhu et al., 2025), and Mixture-of-Recursions (Bae et al., 2025) as models that have been pretrained from random initialization, as well as recent work by McLeish et al. (2026); Koishekenov et al. (2025) that retrofit recurrence into pretrained LLMs.

In terms of mechanistic studies, we highlight Pappone et al. (2025) who analyze latent dynamics in recurrent Transformers. However, the setting of their analysis differs from ours: they analyze a model in which each recurrent block comprises either 1 or 2 layers, and the model comprises multiple *separate* recurrent blocks. We instead study a single looped block with deeper cycles (4+ layers) and analyze the *internals* of these cyclic blocks by examining the latent states of each separate layer. Other relevant work addresses looped model expressivity (Xu & Sato, 2025; Saunshi et al., 2025) and fixed-point stability (Bai et al., 2019; Anil et al., 2022; Ke et al., 2024; Yudin et al., 2026). The only work of which we are aware that analyses the internal states of the cyclic recurrent blocks is Lu et al. (2025), who demonstrate cyclic behavior in logit lens prediction throughout recurrent blocks, but do not analyse this "per-recurrence" or draw the comparison to feedforward models. We highlight also the work of Fumero et al. (2026), who examine the latent dynamics of autoencoders, with additional theoretical analysis around fixed point behavior.

Stages of Inference and Attention Dynamics. The idea that LLMs organize their feedforward computation into several distinct *stages of inference* was first proposed by Lad et al. (2025). Building on this, Queipo-de-Llano et al. (2026) explain the emergence of these stages through the behavior of

attention heads, driven by massive activations (Sun et al., 2024). We also highlight a complementary line of work that analyzes LLM learning dynamics via attention patterns through the lens of *mixing* (Barbero et al., 2024; Arroyo et al., 2026; Veličković et al., 2025; Barbero et al., 2025), motivated by information propagation challenges originally studied in Graph Neural Networks (GNNs) (Cai & Wang, 2020; Alon & Yahav, 2021; Arroyo et al., 2025; Hariri et al., 2025; Blayney et al., 2026).

Test-time Computation. Recurrence is a primary mechanism for scaling compute at test time, offering an alternative to early-exit (Schuster et al., 2022) or continuous-thought architectures (Darlow et al., 2025). Classic approaches to adaptive test-time compute include Adaptive Computation Time (Graves, 2016) and subsequent probabilistic halting frameworks such as PonderNet (Banino et al., 2021). Building on these foundations, recent work has begun to characterize when additional inference compute actually generalizes beyond training-time budgets (Schwarzschild et al., 2021), how to mitigate failures due to excessive computation ("overthinking") (Bansal et al., 2022), how to ensure stable dynamics with repeated iteration (Bear et al., 2024), and how LTs can better generalize to out of distribution tasks at test time (McLeish et al., 2024).

7 Discussion

Summary. This paper examines the limiting behavior of LTs, exploring when and why fixed points are reached. We demonstrate across a range of architectures that recurrent blocks tend to "mirror" the stages of a feedforward Transformer. We investigate the implications for these mixing stages when models converge to a stable fixed point, and when they do not.

Implications of Findings. The implications of our findings extend to theoretical understanding of Transformers and practical design of looped architectures. By decoupling functional depth from parameter count, looped models provide a control for investigating *why* inference stages emerge. While prior work suggested these stages serve to mitigate the cumulative "harms" of extreme depth (Barbero et al., 2025; Queipo-de-Llano et al., 2026), our results demonstrate that looped models can develop these same stages while improving performance through deeper recurrence. Our work shows that architectural insights from feedforward models may be transferable to the looped setting, such as stage-dependent attention sparsification or leaner, low-rank parameterization of middle-stage MLPs where representations are most compressed. We also provide evidence of the link between stability and consistent stages of inference, showing that without stability functional behavior can drift across recurrences. Finally, our analysis identifies specific architectural constraints: prelude and coda layers appear useful to delegate embedding and unembedding behavior, but normalization applied to the residual stream can be counterproductive, suppressing massive activations and mixing behavior.

Limitations and Future Work. We focus exclusively on cyclic recurrence as this appears to be the dominant approach in the literature. However, this means our analysis does not extend to sequential recurrence with multiple separate recurrent blocks; for analysis of this setting we refer the reader to Pappone et al. (2025). Several limitations remain open for future work: we have limited theoretical understanding of when and why LTs are stable; and it remains unclear whether this stable limiting behavior is desirable or restrictive for reasoning tasks.

Acknowledgments and Disclosure of Funding

HB acknowledges funding support from the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems No. EP/S024050/1. MB is partially supported by the EPSRC Turing AI World-Leading Research Fellowship No. EP/X040062/1 and EPSRC AI Hub No. EP/Y028872/1. AITHYRA is supported by the Austrian Academy of Sciences and the not-for-profit Boehringer Ingelheim Stiftung.

HB would like to thank Enrique Queipo-de-Llano for many highly useful discussions particularly around Mixing Score and ColSum Concentration, and Jonas Geiping for discussions around the existence of Orbit and Slider limiting behavior. The authors would like to thank all reviewers of this manuscript for their constructive comments.

References

Ali, R., Caso, F., Irwin, C., and Liò, P. Entropy-lens: Uncovering decision strategies in LLMs. *arXiv preprint arXiv:2502.16570*, 2025.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Alt, J., Erdős, L., and Krüger, T. Spectral radius of random matrices with independent entries. *Probability and Mathematical Physics*, 2(2):221–280, 2021.

Anil, C., Pokle, A., Liang, K., Treutlein, J., Wu, Y., Bai, S., Kolter, J. Z., and Grosse, R. B. Path independent equilibrium models can better exploit test-time computation. *Advances in Neural Information Processing Systems*, 35:7796–7809, 2022.

Arroyo, A., Gravina, A., Gutteridge, B., Barbero, F., Gallicchio, C., Dong, X., Bronstein, M. M., and Vandergheynst, P. On vanishing gradients, over-smoothing, and over-squashing in GNNs: Bridging recurrent and graph learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Arroyo, A., Barbero, F., Blayney, H., Bronstein, M. M., Dong, X., Lio, P., Pascanu, R., and Vandergheynst, P. A survey on over-smoothing and over-squashing: Unified propagation perspectives on graph neural networks and transformers. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Bae, S., Kim, Y., Bayat, R., Kim, S., Ha, J., Schuster, T., Fisch, A., Harutyunyan, H., Ji, Z., Courville, A., and Yun, S.-Y. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Bai, Z.-D. and Yin, Y.-Q. Limiting behavior of the norm of products of random matrices and two problems of Geman-Hwang. *Probability Theory and Related Fields*, 73:555–569, 1986.

Banino, A., Balaguer, J., and Blundell, C. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Bansal, A., Schwarzschild, A., Borgnia, E., Emam, Z., Huang, F., Goldblum, M., and Goldstein, T. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. *Advances in Neural Information Processing Systems*, 35:20232–20242, 2022.

Barbero, F., Banino, A., Kapturowski, S., Kumaran, D., Araújo, J. G. M., Vitvitskyi, A., Pascanu, R., and Veličković, P. Transformers need glasses! information over-squashing in language tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Barbero, F., Arroyo, A., Gu, X., Perivolaropoulos, C., Veličković, P., Pascanu, R., and Bronstein, M. M. Why do LLMs attend to the first token? In *Second Conference on Language Modeling*, 2025.

Bear, J., Prugel-Bennett, A., and Hare, J. Rethinking deep thinking: Stable learning of algorithms using Lipschitz constraints. *Advances in Neural Information Processing Systems*, 37:97027–97052, 2024.

Blayney, H., Arroyo, A., Dong, X., and Bronstein, M. M. gLSTM: Mitigating over-squashing by increasing storage capacity. In *The Fourteenth International Conference on Learning Representations*, 2026.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Csordás, R., Irie, K., Schmidhuber, J., Potts, C., and Manning, C. D. MoEUT: Mixture-of-experts universal transformers. *Advances in Neural Information Processing Systems*, 37:28589–28614, 2024.

Csordás, R., Manning, C. D., and Potts, C. Do language models use their depth efficiently? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Darlow, L. N., Regan, C., Risi, S., Seely, J., and Jones, L. Continuous thought machines. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. In *International Conference on Learning Representations*, 2019.

Fumero, M., Moschella, L., Rodolà, E., and Locatello, F. Navigating the latent space dynamics of neural models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Geiping, J., McLeish, S. M., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

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

Gu, X., Pang, T., Du, C., Liu, Q., Zhang, F., Du, C., Wang, Y., and Lin, M. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081): 633–638, 2025.

Gurnee, W., Horsley, T., Guo, Z. C., Kheirkhah, T. R., Sun, Q., Hathaway, W., Nanda, N., and Bertsimas, D. Universal neurons in GPT2 language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Hariri, A., Arroyo, A., Gravina, A., Eliasof, M., Schönlieb, C.-B., Bacciu, D., Dong, X., Azizzadenesheli, K., and Vandergheynst, P. Return of ChebNet: Understanding and improving an overlooked GNN on long range tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Karpathy, A. nanochat: The best ChatGPT that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

Ke, Y., Li, X., Liang, Y., Shi, Z., and Song, Z. Advancing the understanding of fixed point iterations in deep neural networks: A detailed analytical study. *arXiv preprint arXiv:2410.11279*, 2024.

Koishekenov, Y., Lipani, A., and Cancedda, N. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. *arXiv preprint arXiv:2510.07358*, 2025.

Lad, V., Lee, J. H., Gurnee, W., and Tegmark, M. Remarkable robustness of LLMs: Stages of inference? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Lu, W., Yang, Y., Lee, K., Li, Y., and Liu, E. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.

McLeish, S., Bansal, A., Stein, A., Jain, N., Kirchenbauer, J., Bartoldson, B., Kailkhura, B., Bhatele, A., Geiping, J., Schwarzschild, A., et al. Transformers can do arithmetic with the right embeddings. *Advances in Neural Information Processing Systems*, 37:108012–108041, 2024.

McLeish, S. M., Li, A., Kirchenbauer, J., Kalra, D. S., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Geiping, J., Goldstein, T., and Goldblum, M. Teaching pretrained language models to think deeper with retrofitted recurrence. In *Third Conference on Language Modeling*, 2026.

Nair, P. Softmax is $1/2$-Lipschitz: A tight bound across all ℓ_p norms. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Nostalgebraist. Interpreting GPT: The logit lens. *Blog Post*, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Pappone, F., Crisostomi, D., and Rodolà, E. Two-scale latent dynamics for recurrent-depth transformers. *arXiv preprint arXiv:2509.23314*, 2025.

Prairie, H., Novack, Z., Berg-Kirkpatrick, T., and Fu, D. Y. Parcae: Scaling laws for stable looped language models. In *Third Conference on Language Modeling*, 2026.

Queipo-de-Llano, E., Arroyo, A., Barbero, F., Dong, X., Bronstein, M. M., LeCun, Y., and Shwartz-Ziv, R. Attention sinks and compression valleys in LLMs are two sides of the same coin. In *The Fourteenth International Conference on Learning Representations*, 2026.

Sandoval-Segura, P., Wang, X., Panda, A., Goldblum, M., Basri, R., Goldstein, T., and Jacobs, D. Using attention sinks to identify and evaluate dormant heads in pretrained LLMs. *arXiv preprint arXiv:2504.03889*, 2025.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V., Tay, Y., and Metzler, D. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35: 17456–17472, 2022.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34:6695–6706, 2021.

Skean, O., Arefin, M. R., Zhao, D., Patel, N. N., Naghiyev, J., LeCun, Y., and Shwartz-Ziv, R. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025.

Sun, M., Chen, X., Kolter, J. Z., and Liu, Z. Massive activations in large language models. In *First Conference on Language Modeling*, 2024.

Tan, S., Shen, Y., Chen, Z., Courville, A., and Gan, C. Sparse universal transformer. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Perivolaropoulos, C., Barbero, F., and Pascanu, R. Softmax is not enough (for sharp size generalisation). In *Forty-second International Conference on Machine Learning*, 2025.

Wang, G., Li, J., Sun, Y., Chen, X., Liu, C., Wu, Y., Lu, M., Song, S., and Yadkori, Y. A. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

Xu, K. and Sato, I. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. In *Forty-second International Conference on Machine Learning*, 2025.

Yudin, N., Kudriashov, S., Gaponov, A., and Rakhuba, M. Pay attention to attention distribution: A new local Lipschitz bound for transformers. In *European Conference on Computer Vision*, pp. 241–258. Springer, 2026.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhu, R.-J., Wang, Z., Hua, K., Zhang, T., Li, Z., Que, H., Wei, B., Wen, Z., Yin, F., Xing, H., et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

Appendix Contents

A Potential Societal Impacts **16**

B Proofs of Propositions **16**

C Additional Experimental Details **17**

D Non-Fixed-Point Limiting Behavior **20**

D.1 How Frequent is Non-Fixed-Point Behavior? 20

D.2 Do Intermediate Layers Exhibit Non-Fixed-Point Behavior? 22

D.3 How Does Non-Fixed-Point Behavior Impact Stages of Inference? 24

E Additional Fixed Point Results **25**

E.1 Additional Architecture Results 26

F Additional Latent Trajectories **27**

G Additional Stages of Inference Results **29**

G.1 Prediction and Suppression Neuron Classification 29

G.2 Cyclic Attention Behavior 30

G.3 Cyclic Functional Behavior 37

G.4 Self-Organization Into Stages of Inference 38

H Code, Data and Model Licenses **43**

A Potential Societal Impacts

The potential for negative societal impacts of this work is minor: we seek to further understand existing looped language models, but no new models or techniques are introduced in this work.

However, we believe it is relevant to highlight that this research field of exploring recurrent depth for improved reasoning and test time compute has a clear negative outcome that must be avoided: by moving inference time compute away from the current standard of Chain of Thought into latent space we lose controllability and interpretability over the "thoughts" of these LLM systems. More research must be done on retaining this control before systems are deployed at scale.

B Proofs of Propositions

Proof of Prop. 3.1

Proof sketch. Note

$$\mathrm{S}_k(\boldsymbol{X}') = \mathrm{B}_k(\mathrm{B}_{k-1}(\ldots \mathrm{B}_1(\boldsymbol{X}')\ldots)) = \boldsymbol{X}'$$

Applying B_1 to both sides yields $\mathrm{B}_1(\mathrm{B}_k(\mathrm{B}_{k-1}(\ldots \mathrm{B}_1(\boldsymbol{X}')\ldots))) = \mathrm{B}_1(\boldsymbol{X}')$. Defining $\boldsymbol{Y}' = \mathrm{B}_1(\boldsymbol{X}')$, we obtain $\mathrm{B}_1(\mathrm{B}_k(\mathrm{B}_{k-1}(\ldots \mathrm{B}_2(\boldsymbol{Y}')\ldots))) = \boldsymbol{Y}'$. The general proof follows by induction, and extends trivially to input injection. □

Proof. Assume that (l, k)-Recurrent block S_k reaches a fixed point such that $S_k(\boldsymbol{X}') = \boldsymbol{X}'$. Note that

$$\mathrm{S}_k(\boldsymbol{X}') = \mathrm{B}_{k-1}(\mathrm{B}_{k-2}(\ldots \mathrm{B}_0(\boldsymbol{X}')\ldots)) = \boldsymbol{X}'$$

define the cyclic shift function $f_n(i) = (i + n) \mod k$. Now we aim to prove by induction

$$\forall n \in \mathbb{Z}_+ \cup \{0\} : \mathrm{B}_{f_n(k-1)}(\mathrm{B}_{f_n(k-2)}(\ldots \mathrm{B}_{f_n(0)}(\boldsymbol{Z}')\ldots)) = \boldsymbol{Z}'$$

for some $\boldsymbol{Z}'$. The base case $n = 0$ is trivial as $\forall i < k : f_0(i) = i$. Now assume for some $n = j$

$$\mathrm{B}_{f_j(k-1)}(\mathrm{B}_{f_j(k-2)}(\ldots \mathrm{B}_{f_j(0)}(\boldsymbol{Z}')\ldots)) = \boldsymbol{Z}' \tag{5}$$

Now, let $n = j + 1$. Note

$$f_{j+1}(i) = (i + j + 1) \mod k = f_j(i + 1)$$

Therefore

$$\mathrm{B}_{f_{j+1}(k-1)}(\mathrm{B}_{f_{j+1}(k-2)}(\ldots \mathrm{B}_{f_{j+1}(0)}(\boldsymbol{Y})\ldots)) = \mathrm{B}_{f_j(k)}(\mathrm{B}_{f_j(k-1)}(\ldots \mathrm{B}_{f_j(1)}(\boldsymbol{Y})\ldots)) \tag{6}$$

$$= \mathrm{B}_{f_j(0)}(\mathrm{B}_{f_j(k-1)}(\ldots \mathrm{B}_{f_j(1)}(\boldsymbol{Y})\ldots)) \tag{7}$$

Now take Eq. (5) and apply $\mathrm{B}_{f_j(0)}$ to both sides, defining a new fixed point $\boldsymbol{Z}'' = \mathrm{B}_{f_j(0)}(\boldsymbol{Z}')$:

$$\mathrm{B}_{f_j(0)}(\mathrm{B}_{f_j(k-1)}(\mathrm{B}_{f_j(k-2)}(\ldots \mathrm{B}_{f_j(0)}(\boldsymbol{Z}')\ldots))) = \mathrm{B}_{f_j(0)}(\boldsymbol{Z}') \tag{8}$$

$$\mathrm{B}_{f_j(0)}(\mathrm{B}_{f_j(k-1)}(\mathrm{B}_{f_j(k-2)}(\ldots \mathrm{B}_{f_j(1)}(\boldsymbol{Z}'')\ldots))) = \boldsymbol{Z}'' \tag{9}$$

Combining Eq. (9) and Eq. (7), we see that there exists a fixed point $\boldsymbol{Z}''$ such that

$$\mathrm{B}_{f_{j+1}(k-1)}(\mathrm{B}_{f_{j+1}(k-2)}(\ldots \mathrm{B}_{f_{j+1}(0)}(\boldsymbol{Z}'')\ldots)) = \boldsymbol{Z}''$$

Therefore completing the induction step and proving the proposition. □

Proof of Prop. 3.2

Proof. Define

$$\mathcal{S}_\ell(\boldsymbol{X}) := \mathrm{softmax}(A_\ell(\boldsymbol{X})) = \mathrm{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q\boldsymbol{W}_K^\top\boldsymbol{X}^\top}{\sqrt{d}}\right)$$

Let L_{sm} be the Lipschitz constant of the row-wise softmax such that

$$\left\|\mathcal{S}_\ell(\boldsymbol{X}_{\ell,t}) - \mathcal{S}_\ell(\boldsymbol{X}_{\ell,t-1})\right\| \le L_{\mathrm{sm}}\left\|A_\ell(\boldsymbol{X}_{\ell,t}) - A_\ell(\boldsymbol{X}_{\ell,t-1})\right\| \tag{10}$$

Recently Nair (2026) show that $L_{\mathrm{sm}} = 1/2$.

Define for convenience $\boldsymbol{M} = \boldsymbol{W}_Q \boldsymbol{W}_K^\top$ and $\Delta_{\ell,t} = \boldsymbol{X}_{\ell,t} - \boldsymbol{X}_{\ell,t-1}$. Then

$$\begin{aligned}
A_\ell(\boldsymbol{X}_{\ell,t}) - A_\ell(\boldsymbol{X}_{\ell,t-1}) &= \frac{\boldsymbol{X}_{\ell,t}\boldsymbol{W}_Q\boldsymbol{W}_K^\top\boldsymbol{X}_{\ell,t}^\top}{\sqrt{d}} - \frac{\boldsymbol{X}_{\ell,t-1}\boldsymbol{W}_Q\boldsymbol{W}_K^\top\boldsymbol{X}_{\ell,t-1}^\top}{\sqrt{d}} \\
&= \frac{1}{\sqrt{d}}\left(\left(\Delta_{\ell,t} + \boldsymbol{X}_{\ell,t-1}\right)\boldsymbol{M}\left(\Delta_{\ell,t} + \boldsymbol{X}_{\ell,t-1}\right)^\top - \boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\boldsymbol{X}_{\ell,t-1}^\top\right) \\
&= \frac{1}{\sqrt{d}}\Bigg(\Delta_{\ell,t}\boldsymbol{M}\Delta_{\ell,t}^\top + \Delta_{\ell,t}\boldsymbol{M}\boldsymbol{X}_{\ell,t-1}^\top + \boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\Delta_{\ell,t}^\top \\
&\quad + \cancel{\boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\boldsymbol{X}_{\ell,t-1}^\top} - \cancel{\boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\boldsymbol{X}_{\ell,t-1}^\top}\Bigg) \\
&= \frac{1}{\sqrt{d}}\left(\Delta_{\ell,t}\boldsymbol{M}\left(\Delta_{\ell,t} + \boldsymbol{X}_{\ell,t-1}\right)^\top + \boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\Delta_{\ell,t}^\top\right) \\
&= \frac{1}{\sqrt{d}}\left(\Delta_{\ell,t}\boldsymbol{M}\boldsymbol{X}_{\ell,t}^\top + \boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\Delta_{\ell,t}^\top\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\left\|A_\ell(\boldsymbol{X}_{\ell,t}) - A_\ell(\boldsymbol{X}_{\ell,t-1})\right\| &= \left\|\frac{1}{\sqrt{d}}\left(\Delta_{\ell,t}\boldsymbol{M}\boldsymbol{X}_{\ell,t}^\top + \boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\Delta_{\ell,t}^\top\right)\right\| \\
&\leq \frac{1}{\sqrt{d}}\left(\left\|\Delta_{\ell,t}\boldsymbol{M}\boldsymbol{X}_{\ell,t}^\top\right\| + \left\|\boldsymbol{X}_{\ell,t-1}\boldsymbol{M}\Delta_{\ell,t}^\top\right\|\right) \\
&\leq \frac{\|\boldsymbol{M}\|\left(\|\boldsymbol{X}_{\ell,t}\| + \|\boldsymbol{X}_{\ell,t-1}\|\right)}{\sqrt{d}}\|\Delta_{\ell,t}\|
\end{aligned}$$

Assuming $\|\boldsymbol{X}_{\ell,t}\| \leq B$ for all t, and defining $\kappa_\ell = \|\boldsymbol{W}_{Q,\ell}\boldsymbol{W}_{K,\ell}^\top\| = \|\boldsymbol{M}\|$, we see

$$\left\|A_\ell(\boldsymbol{X}_{\ell,t}) - A_\ell(\boldsymbol{X}_{\ell,t-1})\right\| \leq \frac{2B\kappa_\ell}{\sqrt{d}}\left\|\boldsymbol{X}_{\ell,t} - \boldsymbol{X}_{\ell,t-1}\right\| \tag{11}$$

Combining Eq. (10) and Eq. (11), we complete the proof:

$$\left\|\mathcal{S}_\ell(\boldsymbol{X}_{\ell,t}) - \mathcal{S}_\ell(\boldsymbol{X}_{\ell,t-1})\right\| \leq L_{\text{sm}}\frac{2B\kappa_\ell}{\sqrt{d}}\left\|\boldsymbol{X}_{\ell,t} - \boldsymbol{X}_{\ell,t-1}\right\| \tag{12}$$

□

Proof of Prop. 3.3

Proof. Note $\boldsymbol{A} = \boldsymbol{W}_I[:, :D]$ is a random matrix with elements drawn from $\mathcal{N}(0, \sigma_I^2)$. By the circular law (Bai & Yin, 1986; Alt et al., 2021) as $D \to \infty$, $\rho(\boldsymbol{A}) \leq \sigma_I\sqrt{D}$ with probability 1. Therefore, if $\sigma_I < 1/\sqrt{D}$ then $\rho(\boldsymbol{A}) < 1$. □

C Additional Experimental Details

Important architectural details for each of the models used in the paper are summarised in Table 1. All pretrained models are obtained from Huggingface, model references provided in Table 2. We use standard settings for the tokenizers of each model, and as such some models prepend a BOS token whereas others do not: we make this clear in the 'Prepends BOS' column of the same table.

Mixing Experiments Unless stated otherwise, all of our mixing experiments are averaged over the same subset of 256 random examples from the test split of the GSM8k (Cobbe et al., 2021) dataset. Each example is presented zero-shot, without few-shot demonstrations, system prompts or chat templates, as the question followed by its unmodified reference solution:

Table 1: LT architecture summary. Retrofitted Llama and OLMo use 6 layers in their recurrent block, TinyLlama uses 8. We combine and adapt the nomenclature of Geiping et al. (2025); Saunshi et al. (2025) and refer to an LT with p prelude layers, k recurrent layers and c coda layers with the tuple (p, k, c). Where input injection is used, we add an I subscript $(p, k, c)_I$, and when referring to a recurrent block looped a specific number of times l we write $(p, k \otimes l, c)$.

Model Name	Structure	Train Loops	Block Norm	Norm After Loop?
Ouro 1.4B (Zhu et al., 2025)	$(0, 24, 0)$	4	$\hat{\boldsymbol{X}} = \boldsymbol{X} + n\left(\texttt{Attn}(n(\boldsymbol{X}))\right)$ $\boldsymbol{X}' = \hat{\boldsymbol{X}} + n\left(\text{MLP}(n(\hat{\boldsymbol{X}}))\right)$	Yes
Huginn-0125 (Geiping et al., 2025)	$(2, 4, 2)_I$	32	$\hat{\boldsymbol{X}} = n\left(\boldsymbol{X} + \texttt{Attn}(n(\boldsymbol{X}))\right)$ $\boldsymbol{X}' = n\left(\hat{\boldsymbol{X}} + \text{MLP}(n(\hat{\boldsymbol{X}}))\right)$	Yes
Retrofitted (McLeish et al., 2026)	$(4, 6, 4)_I$ / $(4, 8, 4)_I$	32	$\hat{\boldsymbol{X}} = \boldsymbol{X} + \texttt{Attn}(n(\boldsymbol{X}))$ $\boldsymbol{X}' = \hat{\boldsymbol{X}} + \text{MLP}(n(\hat{\boldsymbol{X}}))$	No
Parcae (Prairie et al., 2026)	$(2, 2, 2)_I$ / $(4, 4, 4)_I$ / $(6, 6, 6)_I$	8	$\hat{\boldsymbol{X}} = \boldsymbol{X} + \texttt{Attn}(n(\boldsymbol{X}))$ $\boldsymbol{X}' = \hat{\boldsymbol{X}} + \text{MLP}(n(\hat{\boldsymbol{X}}))$	No

Table 2: Additional Huggingface details on pretrained Looped models used.

Model	Num Params	Huggingface ID	Base Model	Prepends BOS	Notes
Ouro 1.4B	1.4B	`ByteDance/Ouro-1.4B`	-	×	-
Ouro 2.6B	2.6B	`ByteDance/Ouro-2.6B`	`ByteDance/Ouro-1.4B`	×	"Upcycled" from Ouro 1.4B by repeating the same layers after the first training phase.
Huginn-0125	3.5B	`tomg-group-umd/huginn-0125`	-	✓	-
Retrofitted Llama	1B	`smcleish/Recurrent-Llama-3.2-train-recurrence-32`	`meta-llama/Llama-3.2-1B`	✓	Models trained with fewer recurrences exhibit similar mixing patterns.
Retrofitted OLMo-2	1B	`smcleish/Recurrent-OLMo-2-0425-train-recurrence-32`	`allenai/OLMo-2-0425-1B`	×	As above.
Retrofitted TinyLlama	0.8B	`smcleish/Recurrent-TinyLlama-3T-train-recurrence-32`	`TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T`	✓	As above.
Parcae	0.14/0.37/0.77B	`SandyResearch/parcae-(size)`	-	×	-

`Question: {question}\nAnswer:{solution}`

where `\n` is a newline. The solution keeps GSM8k's `<<...>>` calculator annotations and its final `#### {answer}` line.

The illustrative trajectory plot of Fig. 5 is instead produced with a *test sequence* that we obtain from Barbero et al. (2025): "Hello! I've been well. I hope that you're doing well." Additional trajectory plots using random samples from GSM8k can be found in App. F.

Additional results targeting non-reasoning behavior using the HellaSwag dataset (following an identical setup of running inference on the same 256 random examples from the test split) can be found in App. G.2.3.

Downstream (Embedding and Refinement) Experiments All of the "downstream" tasks presented in this paper are evaluated on multiple-choice tasks. In the main body of the paper, in Figs. 9 and 10 we present results on ARC-Easy and in App. G.3 we present results on additional datasets: ARC-Challenge, HellaSwag and WinoGrande. We note that for the ARC datasets we remove the few questions which do not have four answers: this removes fewer than 1% of questions in all of the splits used. We evaluate on the test set of the ARC datasets, and the validation sets for HellaSwag and WinoGrande.

Our experimental setting closely follows that of Queipo-de-Llano et al. (2026). For all multiple-choice benchmarks, we represent each answer option as an independent prompt: for ARC and HellaSwag, we concatenate the question context with the option text, while for WinoGrande each option is already a complete filled-in sentence. These option prompts are passed through the frozen model to extract hidden states at every virtual layer.

To measure *embedding* representational quality at each layer and recurrence (the **linear probe** experiments), per virtual layer we train a probe: an unscaled (non elementwise-affine, per PyTorch docs) LayerNorm followed by a Linear layer with bias. The output dimension of this linear layer is 1: it maps to a single scalar. The input to this probe is a mean pool taken over every token index in the sequence, thus aggregating the latent representations. This results in one scalar (logit) per answer: we train these logits with cross entropy loss against the one-hot encoded true multiple choice answer. We use Adam optimizer with learning rate 5×10^{-4}, and train on the entire train sets of the ARC datasets, and a random 5000 sample split of the HellaSwag and WinoGrande train sets. Accuracy is computed against the test/validation set via selecting the option with the highest probe score.

To measure the next token distribution *refinement* (the **logit lens** experiments) we project hidden states into vocabulary log-probabilities using the model's final layernorm and pre-trained unembedding weights. We score each answer option by summing the log-probabilities assigned to its constituent tokens, conditioned on the preceding context. LogitLens accuracy is defined as the fraction of examples where the highest-scoring option corresponds to the ground-truth answer. Parcae models have an additional learned linear projection between the core block and the coda layers: for all layers before the coda we compute the logit-lens by applying first this linear map before the final norm and unembedding. We denote this difference with an asterisk.

Isolating Stability Experiments For the experiments in Fig. 12 we train a variety of models based on the 140M Parcae (Prairie et al., 2026) configuration. Typically the Parcae models learn a diagonal $\boldsymbol{A}$ matrix such that the spectral radius of this matrix is constrained to < 1 – we instead replace this matrix with a single scalar ρ, thus directly (trivially) controlling the spectral radius. We train each of these models with a Poisson recurrence schedule with mean 8, and we cap the number of recurrences at 16, allowing us to control when the models are operating "out of distribution". We test $\rho = 0.25, 0.5, 0.75, 1.0$ and a learned variant, for which we retrain a model using the standard Parcae 140M architecture but with a maximum recurrence of 16.

Self-Organizing Nanochat Experiments Our small training runs in App. G.4 are performed by adapting a publicly available fork of Nanochat (Karpathy, 2025), `https://github.com/TrelisResearch/nanochat/tree/recursive`. For all experiments we use a model dimension (residual stream) of 512 (4 heads of dimension 128) and train for 3.7B tokens. As discussed in App. G.4, loss is the same as that of a regular feedforward model: cross entropy loss on the final output representation (as opposed to the summed loss of Zhu et al. (2025)). Each model is trained for a *constant* 4 recurrences (as opposed to the Poisson sampling of Geiping et al. (2025)). All models use pre-norm only: this

norm does not result in the stability issues reported by Geiping et al. (2025); Zhu et al. (2025), but this is likely because we are operating at a far smaller scale.

Compute Requirements All stages of inference experiments are easily reproducible and can run in less than 12 hours on an A10 GPU. Each of the Nanochat training runs took fewer than 12 hours on an H100 GPU.

D Non-Fixed-Point Limiting Behavior

D.1 How Frequent is Non-Fixed-Point Behavior?

In this section we investigate more closely the "orbits" and "sliders" initially observed by Geiping et al. (2025). These are important as they appear to represent stable limiting behavior that is **not a fixed point**.

We develop a heuristic algorithm to detect these behaviors, presented in Algorithm 1. We use the sequence of cosine similarities from the final recurrent layer (per token), where the output residual stream after each of 128 recurrences is compared to the final residual stream. This is visualized in the leftmost column of Fig. 13. We set threshold $\tau = 0.05$ and fixed-point fraction $\rho = 0.9$.

Using this algorithm, we classify the limiting behavior over *all* tokens in the GSM8k test set for the Huginn-0125 and Retrofitted Llama models. We discover that the *system prompt* used before presenting the GSM8k question has a large impact on the limiting behavior types and as such we test the following prompts across both models:

Long Persona This is the system prompt used by Geiping et al. (2025) to produce their Orbit plots:

> You are Huginn, an AI assistant who embodies careful thought and deliberation. Your responses demonstrate:
> Methodical reasoning, breaking complex problems into clear steps
> Mathematical and programming expertise grounded in fundamentals
> The ability to acknowledge uncertainty and correct course when needed
> Clear communication that illuminates rather than just informs
> When engaging with questions, you first seek to understand their deeper structure before answering. Like your namesake who flew the nine worlds seeking wisdom, you explore problems from multiple angles, helping users build genuine understanding rather than providing shallow answers. You express warmth and intellectual curiosity while maintaining professionalism. When faced with errors or confusion, you model honest reflection and careful correction. Your goal is not just to provide answers, but to help humans develop clearer, deeper thinking.

Long Persona (Padded) This is the same prompt as above but with all tokens replaced with the padding token: we include this to test whether the behavior arises purely due to the length of the input.

Short Math This is the system prompt used by Geiping et al. (2025) in their GSM8k benchmark evaluation:

> You are a helpful assistant that is capable of helping users with mathematical reasoning.

We present the percentage of GSM8k tokens that exhibit each classification of limiting behavior in Table 3 and the percentage of GSM8k examples that exhibit each classification of limiting behavior *across any of their tokens* in Table 4.

These results reveal that these non-fixed-point limiting behaviors appear to be extremely rare in practice: without a system prompt (the setting used throughout this paper) only approximately 0.02% of tokens exhibit non-fixed-point behavior. This percentage can be significantly increased with the longer system prompt, but these behaviors remain rare at 0.14%. Curiously, the "Persona" system

Table 3: Percentage of tokens exhibiting each behavior type, by model and system prompt.

Model	System Prompt	Non-Fixed-Point %	Orbit %	Slider %	Unknown %
Huginn	Long Persona	0.14	0.13	0.00	0.01
Huginn	Long Persona (Padded)	0.05	0.02	0.02	0.01
Huginn	No System Prompt	0.02	0.01	0.01	0.00
Huginn	Short Math	0.01	0.00	0.00	0.00
Retrofitted-Llama	Long Persona	0.00	0.00	0.00	0.00
Retrofitted-Llama	Long Persona (Padded)	0.00	0.00	0.00	0.00
Retrofitted-Llama	No System Prompt	0.00	0.00	0.00	0.00
Retrofitted-Llama	Short Math	0.00	0.00	0.00	0.00

prompt used by Geiping et al. (2025) seems to increase the occurrence of these orbits and sliders more than a comparable prompt of padding tokens, suggesting that the effect is not purely to do with sequence length.

We highlight that these exact results need to be treated with caution: the absolute values vary significantly depending on the algorithm hyperparameters chosen, and in the absence of a good external metric we selected these hyperparameters manually via visual inspection of the trajectories and cosine similarities to ensure the results agreed with intuition. However, across different hyperparameters the results consistently showed that non-fixed-point behavior is rare and that the long persona prompt results in a greater rate of non-fixed-point behavior. We leave in-depth classification and explanation of these phenomena to future work.

Algorithm 1 Per-token limiting behaviour classification

Require: similarity (or norm) series $\mathbf{s} \in \mathbb{R}^n$ for one token over the last n recurrent firings (final firing excluded); threshold τ; fixed-point fraction ρ
Ensure: Label $\ell \in \{\text{FIXEDPOINT}, \text{ORBIT}, \text{SLIDER}, \text{UNKNOWN}\}$

— *Detrend* —
Fit linear trend: $[\hat{a}, \hat{b}] \leftarrow \arg\min_{a,b} \sum_i (s_i - ai - b)^2$
$\tilde{\mathbf{s}} \leftarrow \mathbf{s} - (\hat{a}\,\mathbf{t} + \hat{b})$ ▷ $\mathbf{t} = [0, 1, \ldots, n-1]^\top$
— *Spectral amplitude* —
$\mathbf{w} \leftarrow \tilde{\mathbf{s}} \odot \text{Hann}(n)$ ▷ reduce spectral leakage
$\mathbf{M} \leftarrow |\,\text{RFFT}(\mathbf{w})|_{1:}$ ▷ discard DC bin
$k^* \leftarrow \arg\max_k M_k$
$A \leftarrow 4\,M_{k^*}/n$ ▷ Hann-corrected amplitude
— *Classify (first match wins)* —
$n_{\text{close}} \leftarrow \#\{i : s_i \geq 1 - \tau\}$ ▷ (or $s_i \leq \tau$ for norm)
if $n_{\text{close}} \geq \rho\, n$ **then**
  return FIXEDPOINT
end if
▷ peak-to-peak $= 2A \geq \tau$; at least 2 full cycles in window
if $A \geq \tau/2$ **and** $(k^*+1)/n \geq 2/n$ **then**
  return ORBIT(freq $= (k^*+1)/n$, amp $= A$)
end if
$g \leftarrow \hat{a}$ ▷ linear-fit slope; negate for norm series
▷ sim increases by $\geq \tau$ over the full window
if $g > \tau/n$ **then**
  return SLIDER(g)
end if
return UNKNOWN

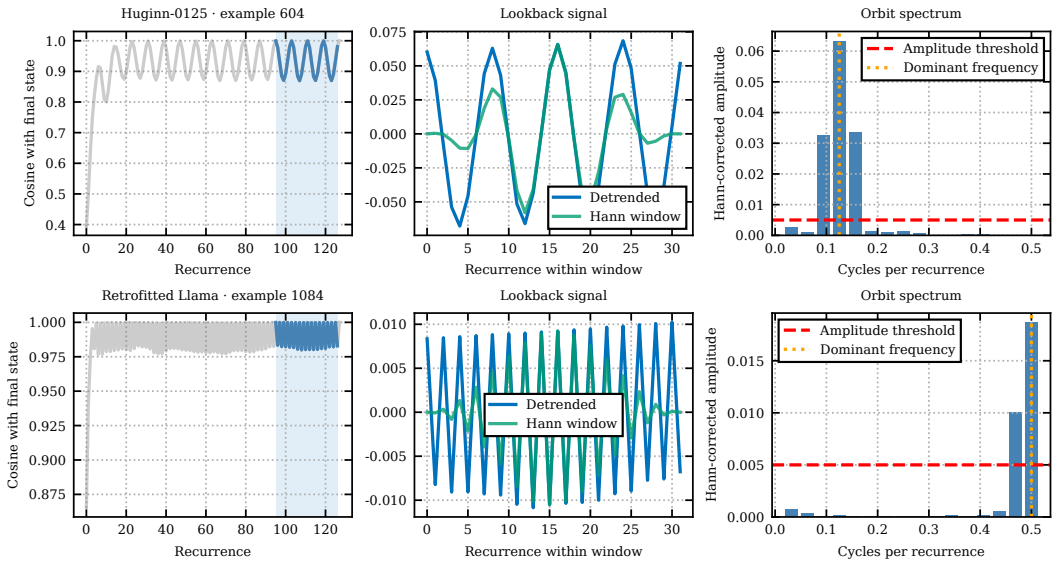


Figure 13: Visualizing the component parts of the Orbit detection algorithm of Algorithm 1. The input sequence (cosine similarities for the residual streams of a given token and layer and successive recursions, as compared to their final residual stream) is visualized in the leftmost column. The center column visualizes the effect of windowing and de-trending, and the rightmost column shows the FFT magnitudes. The top row visualizes the detected Orbit with the largest amplitude for the Huginn-0125 model (which occurs with the "Long Persona" prompt) and the bottom row visualizes the largest amplitude for the Retrofitted Llama model (which occurs with no system prompt). We note the appearance of complex, multi-frequency oscillation in this latter case.

Table 4: Percentage of examples exhibiting each behavior type at least once on any question token, by model and system prompt.

Model	System Prompt	Non-Fixed-Point %	Orbit %	Slider %	Unknown %
Huginn	Long Persona	2.81	2.50	0.15	1.06
Huginn	Long Persona (Padded)	0.83	0.23	0.61	0.15
Huginn	No System Prompt	0.76	0.53	0.15	0.08
Huginn	Short Math	0.45	0.38	0.08	0.00
Retrofitted-Llama	Long Persona	0.00	0.00	0.00	0.00
Retrofitted-Llama	Long Persona (Padded)	0.00	0.00	0.00	0.00
Retrofitted-Llama	No System Prompt	0.08	0.08	0.00	0.00
Retrofitted-Llama	Short Math	0.00	0.00	0.00	0.00

D.2 Do Intermediate Layers Exhibit Non-Fixed-Point Behavior?

Geiping et al. (2025) observe orbits and sliders in the latent states after each application of the entire recurrent block. Here we investigate what occurs in the latent states of the intermediate layers within the recurrent block.

We first extend Figure 16 of Geiping et al. (2025) (visualizing latent trajectories on a math prompt), visualizing also the trajectories of the intermediate layer residual streams: this can be found in Fig. 15. This demonstrates that in this particular case, where Orbits occur, they also occur in the intermediate layers. This implies – viewed in realized depth – latent trajectories exhibiting multi-scale cyclic behavior.

We investigate this across all GSM8k test examples in Fig. 14 by plotting the conditional probability of observing each behavior on a given token for any layer, given an observation of another behavior on that same token. We discover that orbits and sliders do not co-occur across looped layers, but that orbits and sliders both frequently co-occur with fixed point behavior. "Unknown" behavior often co-occurs with orbits, which we suggest may be due to mis-classification of the behavior algorithm.

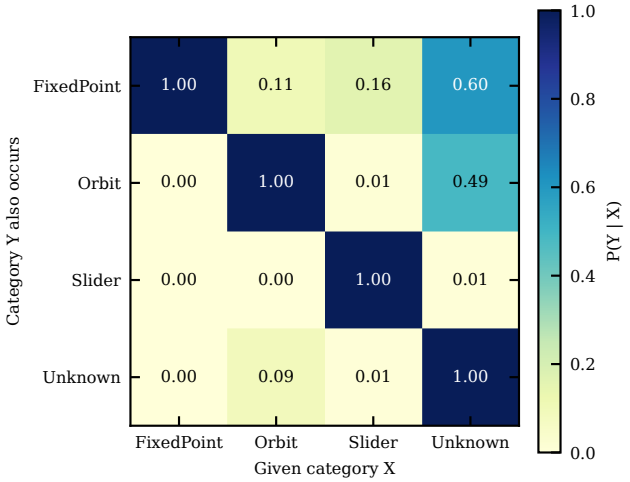


Figure 14: Conditional probabilities of co-occurrence for the different limiting behaviors.

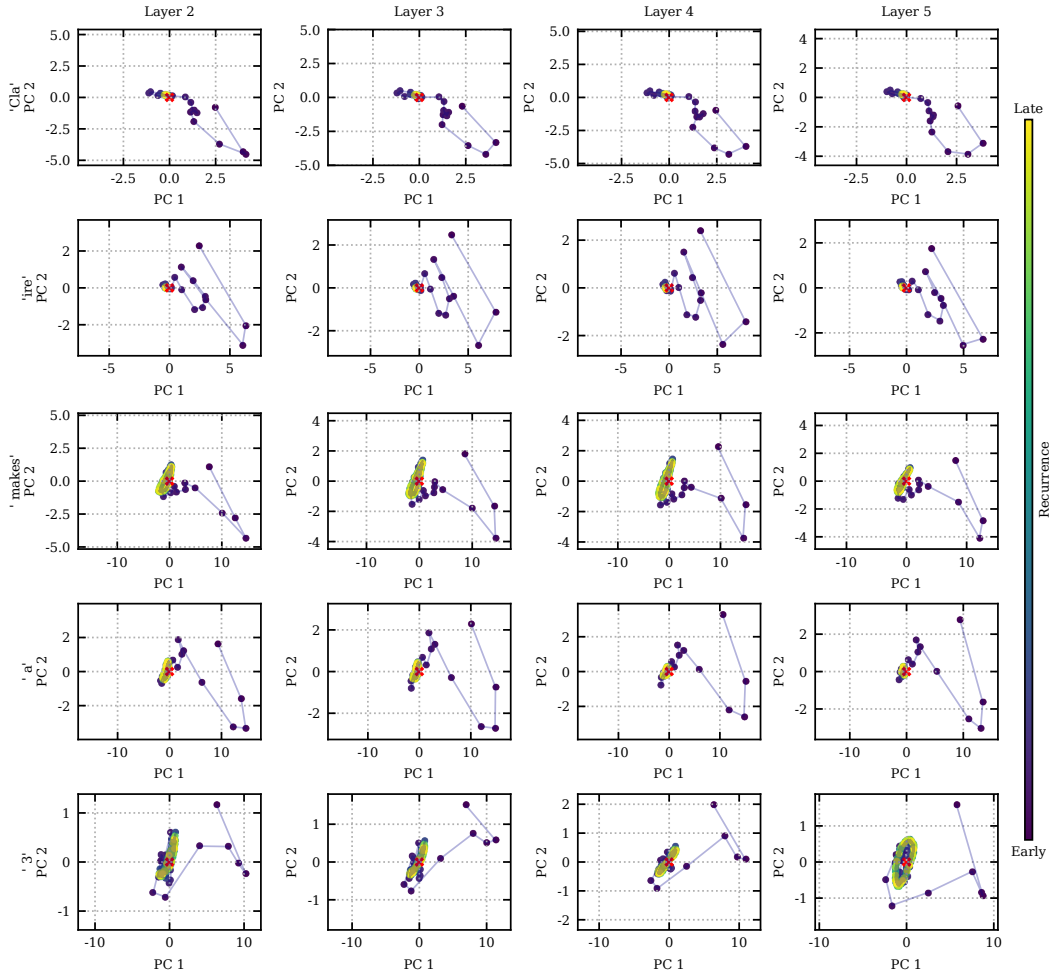


Figure 15: PCA trajectories in the intermediate layers of Huginn-0125: this reproduces the leftmost column of Fig. 16 in Geiping et al. (2025) (the first two principal components) and additionally plots the latent trajectories for the intermediate layers in the recurrent block.

D.3 How Does Non-Fixed-Point Behavior Impact Stages of Inference?

To complete the link between non-fixed-point behavior and our work we additionally investigate how this behavior impacts the observed stages of inference.

To attempt to isolate the "worst case" scenario for stages of inference stability, we plot stages of inference for the GSM8k test prompt that exhibits the greatest orbit amplitude. We first plot in realized depth the extended stages of inference metrics used throughout the paper, see Fig. 16. This demonstrates that sink rates show some variability with the orbiting behavior, but the other metrics remain broadly consistent. We plot also the same metrics in block depth (showing only the final 64 loops) in Fig. 17: this demonstrates clearly that the stages of inference remain very consistent despite the orbiting behavior.

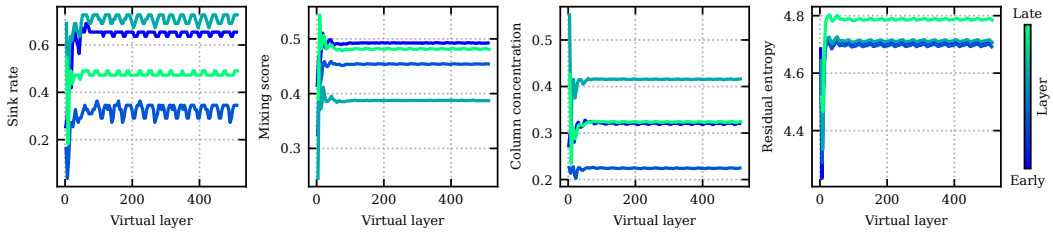


Figure 16: Stages of inference metrics (sink rate, mixing score and colsum concentration) for Huginn-0125 across 128 recurrences, for the GSM8k test prompt that exhibited the largest orbit amplitude. Visualized in realized depth.

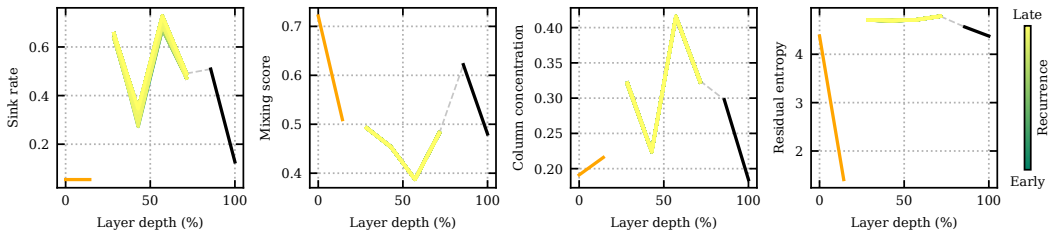


Figure 17: Stages of inference metrics (sink rate, mixing score and colsum concentration) for Huginn-0125 across 128 recurrences, for the GSM8k test prompt that exhibited the largest orbit amplitude: only the final 64 recurrences are visualized, to isolate the impact of the orbit. Visualized in percentage block depth.

We plot the same for the largest amplitude orbit on the Retrofitted Llama model in Fig. 18, demonstrating that the same behavior holds.

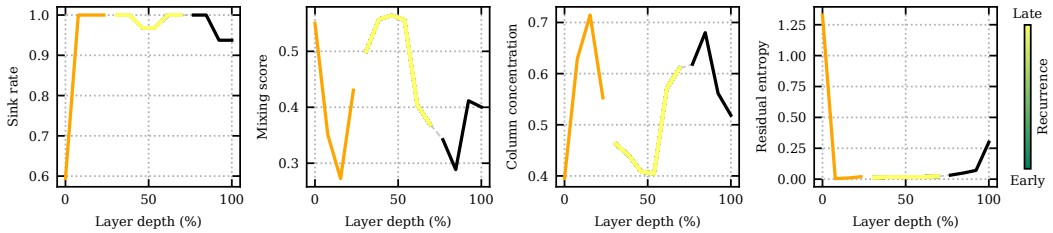


Figure 18: Stages of inference metrics (sink rate, mixing score and colsum concentration) for Retrofitted Llama across 128 recurrences, for the GSM8k test prompt that exhibited the largest orbit amplitude: only the final 64 recurrences are visualized, to isolate the impact of the orbit. Visualized in percentage block depth.

E Additional Fixed Point Results

In this appendix we include additional plots to validate and provide further context to our fixed point claims in Sec. 3.

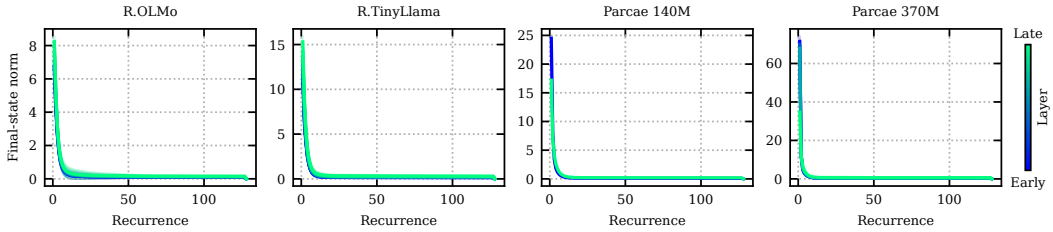


Figure 19: Complement to Fig. 3 with additional models, visualizing the norm of the difference from each layer to an "approximate fixed point" (the 128th recurrence).

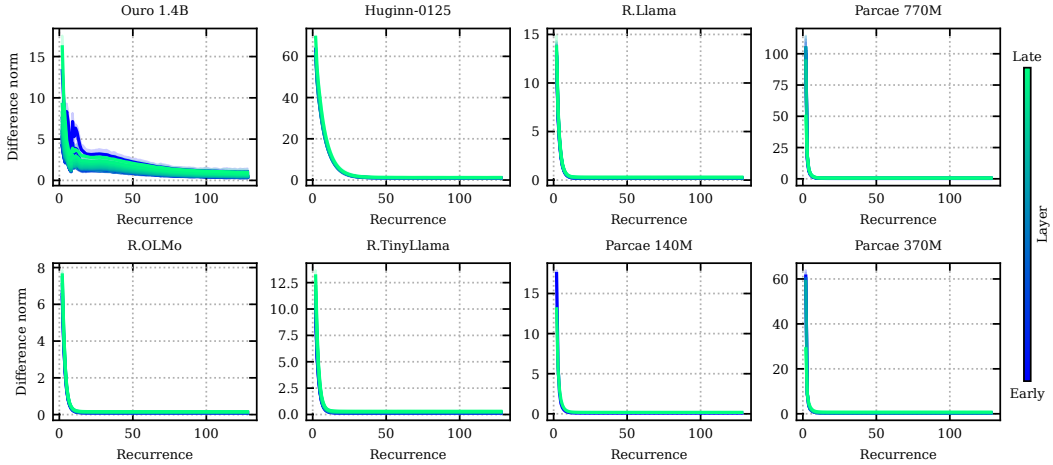


Figure 20: Direct validation of the premise of Prop. 3.2, visualizing the norm of the difference between successive residual streams for each layer l, $|\boldsymbol{h}_{l,r} - \boldsymbol{h}_{l,r-1}|_2$. This successive difference continues to decrease to a low level even for Ouro, which does not reach a fixed point.

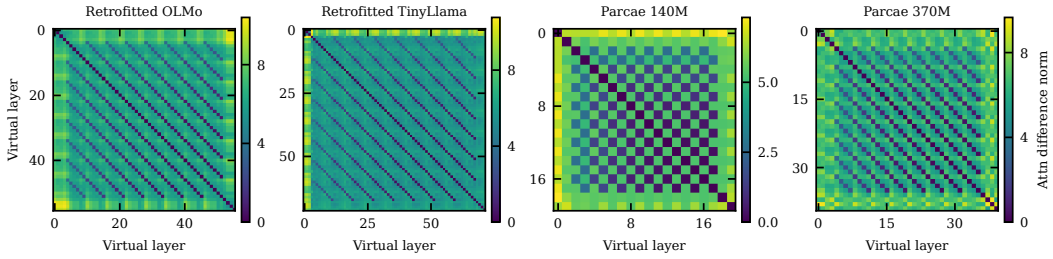


Figure 21: Complement to Fig. 4 with additional models, visualizing Frobenius norms between attention patterns at all virtual layer pairs (8 loops), averaged across the batch and head dimensions.

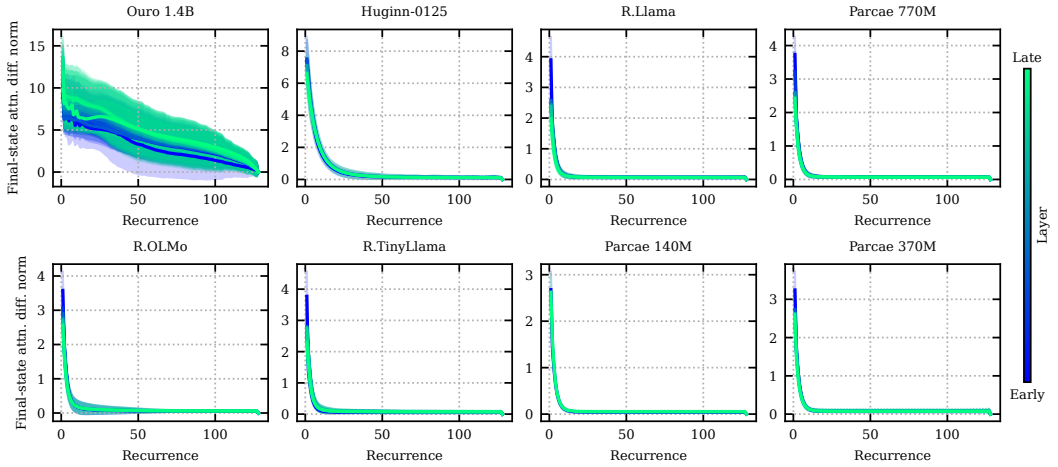


Figure 22: Attention pattern stability across recurrences. Visualises the Frobenius norm of the difference from the attention patterns at each layer to an "approximate fixed point" (the 128th recurrence): for each layer l, $|\boldsymbol{A}_{l,r} - \boldsymbol{A}_{l,128}|_F$, averaged over batch and head dimensions.

E.1 Additional Architecture Results

In Fig. 23 we present the norm of the residual difference, complementing the cosine similarity results of Fig. 6. In Fig. 24 we further check that our stability conclusions hold for varying layer counts: we find that broadly they do, with the partial exception that in the case of pre-norm with input injection, some seeds with 16 layers exhibit oscillation rather than a true fixed point.

We note that the continued residual growth of pre-norm models without input injection initially seems at odds with the pretrained convergence results of Fumero et al. (2026, Appendix B.4.1) who analyze the convergence of pretrained models, however their experiments include the final layernorm immediately pre-softmax in their recurrent block, explaining the bounded norm.

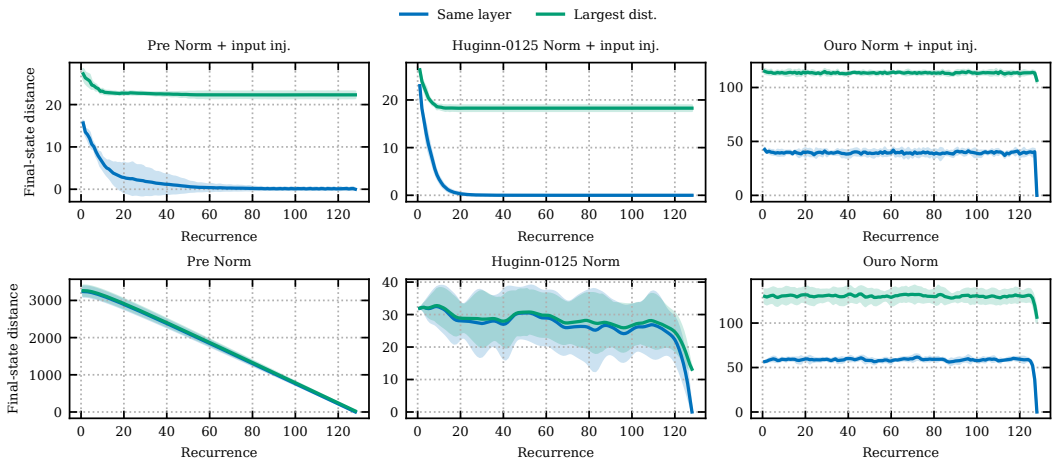


Figure 23: Complement to Fig. 6, showing instead the difference norm to fixed point per layer l (in blue) $|\boldsymbol{h}_{l,r} - \boldsymbol{h}_{l,128}|_2$ and also to the layer with the most distant fixed point $\tilde{l}$ (in green) $|\boldsymbol{h}_{l,r} - \boldsymbol{h}_{\tilde{l},128}|_2$, in order to potentially capture "degenerate fixed points" which would be constant across layers. As in the main body of the paper, averaged over 5 seeds.

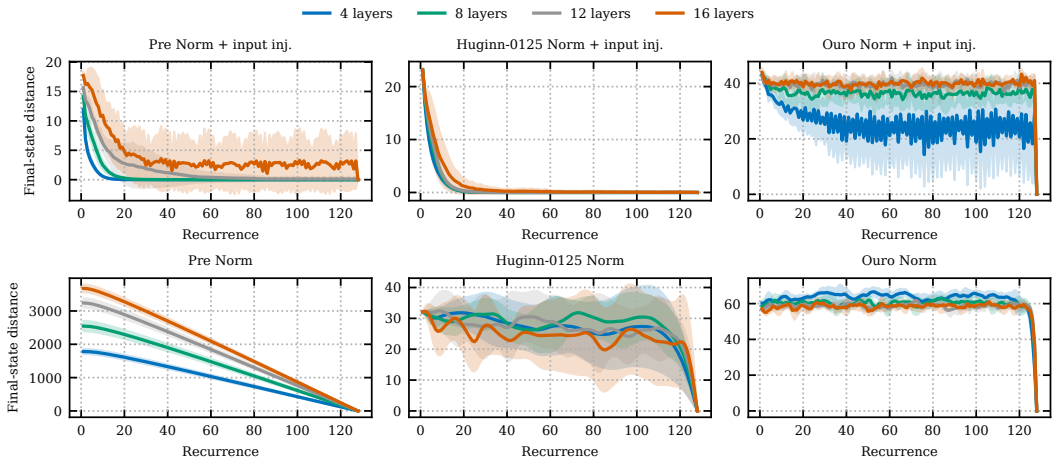


Figure 24: Complement to Fig. 6, showing the impact of different numbers of layers in the recurrent block. Broadly the same trends hold across the architectural choices tested, but for more layers some of the prenorm seeds begin to exhibit oscillation, explaining the larger deviations in the top left plot. As in the main body of the paper, averaged over 5 seeds.

F Additional Latent Trajectories

To provide additional intuition on the latent space trajectories followed by these models and support the qualitative conclusions drawn from Fig. 5, in this section we plot the latent trajectories at multiple token positions (beginning, ~ middle, end) and for multiple different prompts (randomly selected from the GSM8k test set) for the main models presented in the paper:

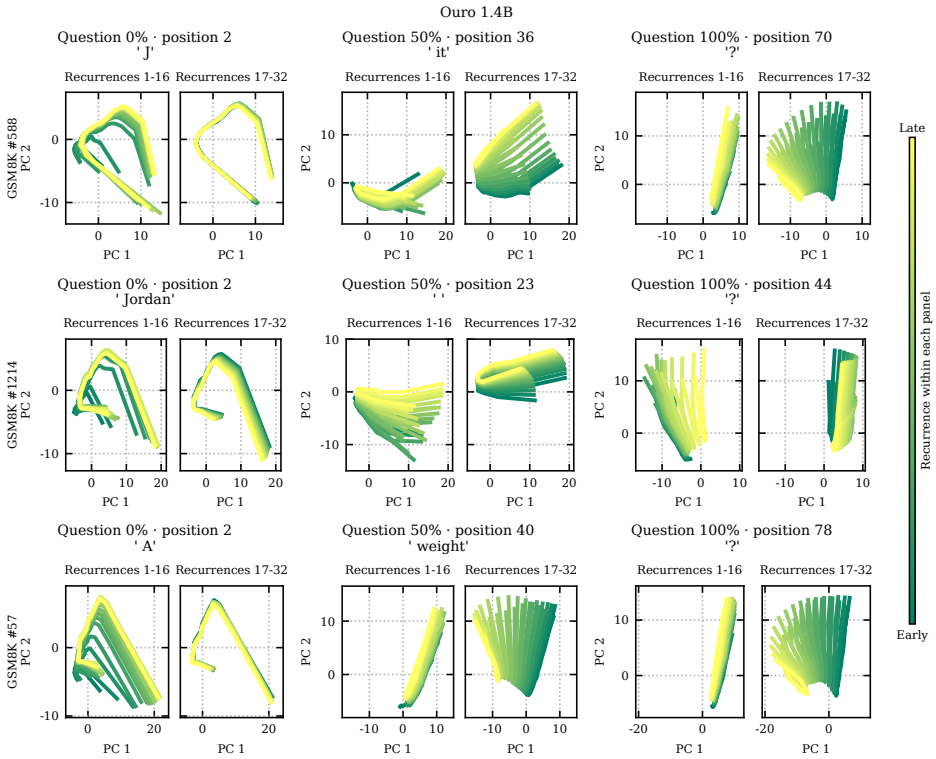


Figure 25: Example latent trajectories at different GSM8k prompts and different token positions for Ouro.

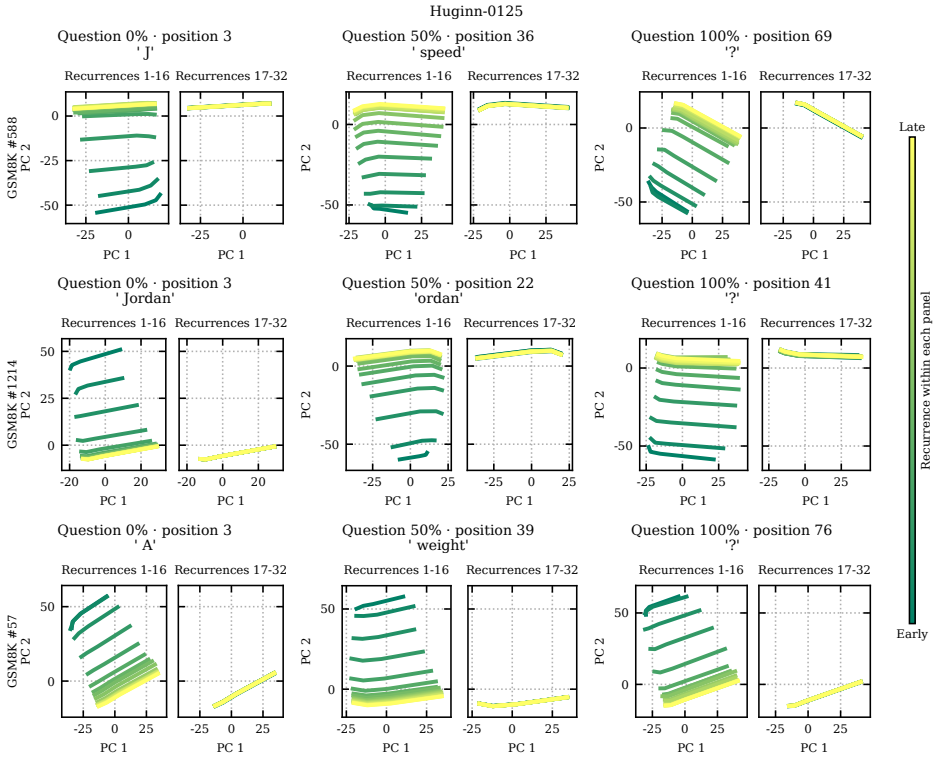


Figure 26: Example latent trajectories at different GSM8k prompts and different token positions for Huginn-0125.

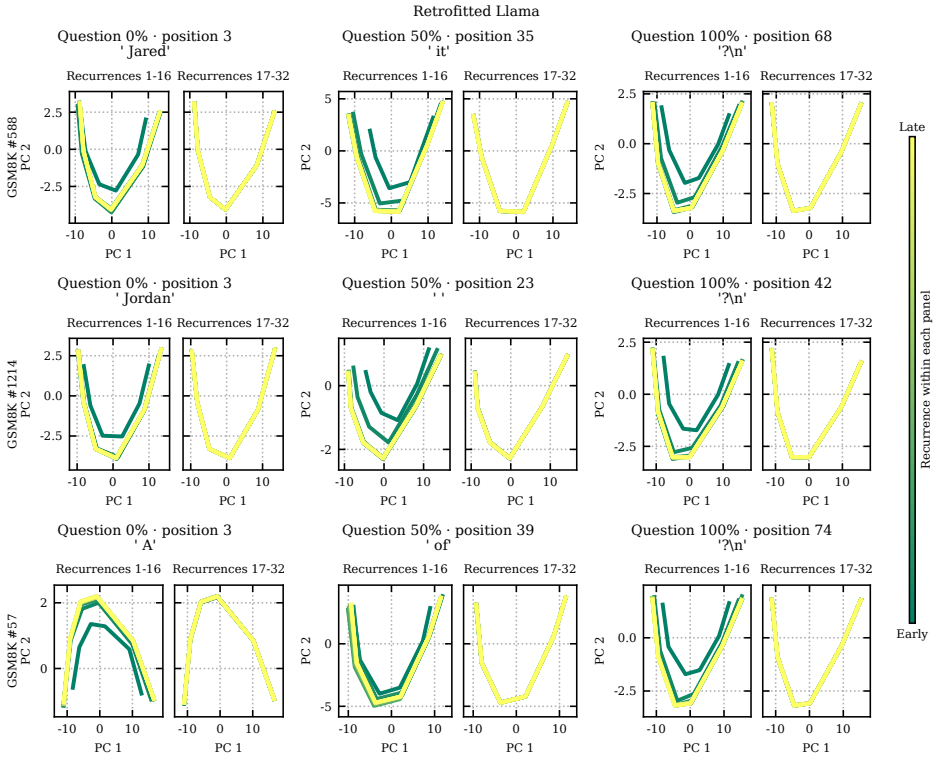


Figure 27: Example latent trajectories at different GSM8k prompts and different token positions for Retrofitted Llama.

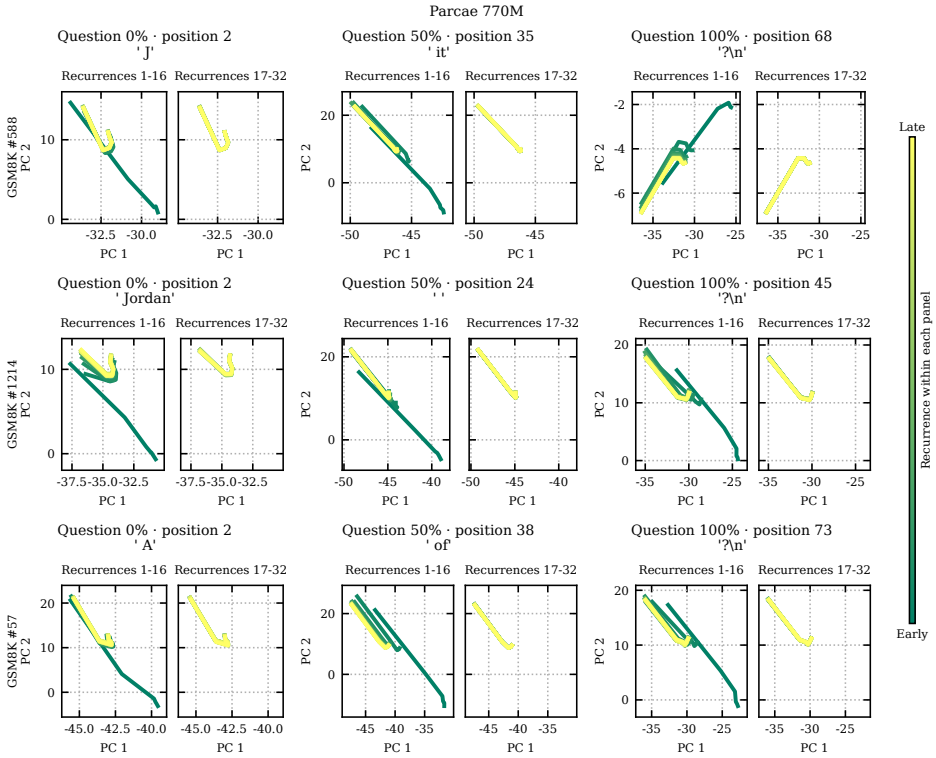


Figure 28: Example latent trajectories at different GSM8k prompts and different token positions for Parcae.

G Additional Stages of Inference Results

In this appendix we explore in more detail the mixing behavior presented in the main body via additional metrics, and considering a wider range of models.

G.1 Prediction and Suppression Neuron Classification

As our work is concerned largely with how the functionality of layers changes throughout depth as the residual stream is iteratively updated, our primary concern is with *input-dependent* measures of stages of inference: Mixing Score as presented in the main text of the paper is one such input-dependent metric, and the later sections of this appendix will introduce and present results for a wider range of such metrics.

However, to further link our work to that of Lad et al. (2025), we also present results for the fraction of *prediction and suppression* neurons (Gurnee et al., 2024) in successive layers of both feedforward and LTs. We highlight however that *these are unable to change with successive recurrences*, and are thus secondary to our focus. Fig. 29 presents results for a selection of feedforward models used throughout the paper, and Fig. 30 presents results for a selection of looped models used throughout the paper. Similar to our results elsewhere, we find that the looped model stages of inference in these metrics tend to mirror those of feedforward models. We highlight that the existence of the prediction and suppression neurons in the repeated block of Ouro in the leftmost figure of Fig. 30 supports the results presented in Fig. 10, wherein initial layers in later loops must "decode" back out of this token space. In other models, these neurons tend to be delegated to the coda, which runs only once.

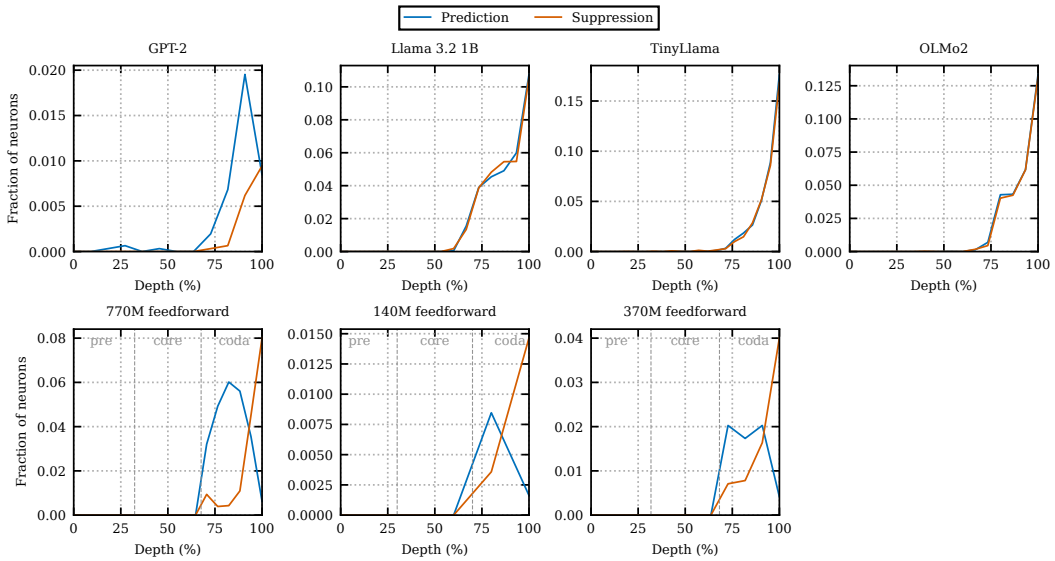


Figure 29: **Fraction of prediction and suppression neurons** in a selection of feedforward models used throughout the paper.

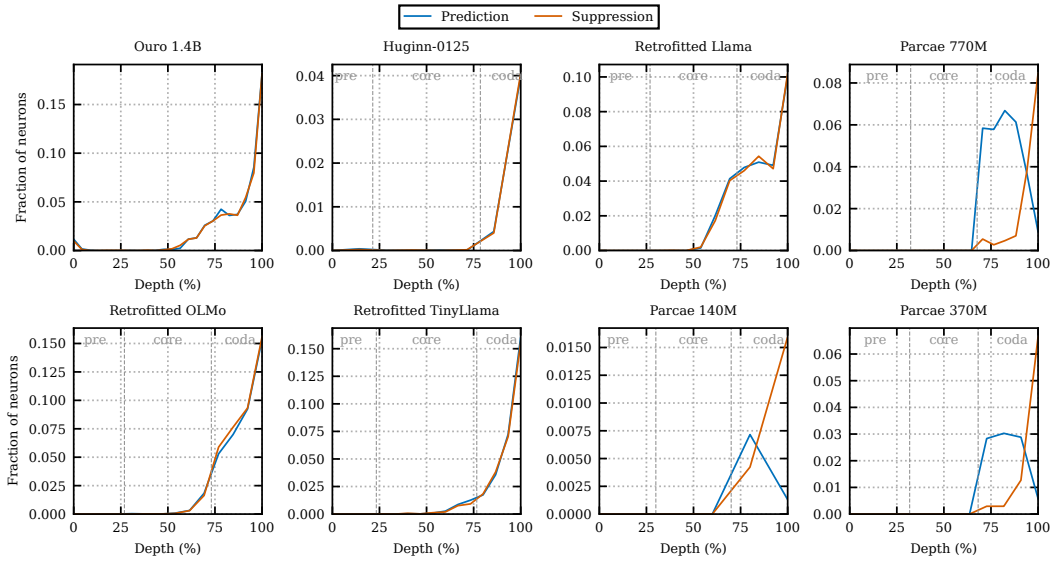


Figure 30: **Fraction of prediction and suppression neurons** in a selection of looped models used throughout the paper.

G.2 Cyclic Attention Behavior

One well-studied phenomenon by which Transformers drastically *reduce* the mixing in a given layer is that of the *attention sink* (Xiao et al., 2024; Barbero et al., 2025), whereby the layer focuses the majority of the attention "weight" onto the first token in the sequence; often this is the BOS (beginning of sequence) token, and thus completely uninformative. To measure this behavior, we adopt the **attention sink score** and **sink rate** of Gu et al. (2025): for token position k and sequence length T, the sink score at layer ℓ and head h is defined as:

$$\text{sink-score}_k^{(\ell,h)} = \frac{1}{T}\sum_{t=0}^{T-1} A_{tk}^{(\ell,h)}, \tag{13}$$

where $A_{tk}^{(\ell,h)}$ corresponds to the realized attention pattern of Eq. (1) at layer ℓ, head h. The sink rate is then defined as the fraction of heads for which the sink score lies above a certain threshold:

$$\text{sink-rate}_k^{(\ell)} = \frac{1}{H}\sum_{h=1}^{H}\mathbb{I}\Big(\text{sink-score}_k^{(\ell,h)} \geq \tau\Big), \tag{14}$$

where following related work we define a threshold of $\tau = 0.3$, and $\mathbb{I}$ denotes the indicator function.

We also adapt the *Mixing score* of Queipo-de-Llano et al. (2026). This was initially defined as follows: for sequence length T, layer ℓ and head h the Mixing Score is the average row entropy of the attention patterns:

$$\text{mixing-score}^{(\ell,h)} = \frac{1}{T}\sum_{i=1}^{T} H(A_{i,:}^{(\ell,h)}), \tag{15}$$

and there exists a trivial extension of this that normalizes by $\log T$ to achieve a value in $[0, 1]$. However, this has some unintuitive properties, as illustrated in the "Uniform Causal" example in Fig. 31: for a uniform ("maximally mixed") causal attention pattern, the Mixing Score still doesn't take a value of 1 for two reasons:

1. The Shannon entropy of row i is normalized by $\log T$ but due to the causal mask the maximum value it can take is $\log i$. Therefore, all rows other than the last are constrained to a normalized entropy below 1.
2. Due to the causal mask, the entropy of the first row will always be $H(A_{1,:}) = -1 \log 1 = 0$.

We address these two issues in our modified Mixing Score of Eq. (3) by omitting the first row and normalizing each row by $\log i$: this allows the score to truly take values in the entire range of $[0, 1]$, and means that uniform causal attention patterns have a Mixing Score of 1.

We also use the ColSum Concentration from Queipo-de-Llano et al. (2026). For this we first define the Column sum $c_j = \sum_i A_{ij}$ to capture how much attention mass is received by token j. We normalize this as $\hat{c}_j = c_j/T$ to obtain a probability distribution, noting that $\sum_{i,j} A_{ij} = T$ since A is row-stochastic. From this distribution, we define the **ColSum Concentration** via its normalized entropy as

$$C = 1 - H_{\text{col}} \in [0, 1] = 1 + \frac{1}{\log T}\sum_j \hat{c}_j \log \hat{c}_j. \tag{16}$$

Large values of C indicate a high *concentration* of attention mass: few columns receive most of the mass. We note therefore that this metric captures the well-studied *attention sink* (Xiao et al., 2024; Barbero et al., 2025) behavior, but generalizes to capture concentration over any token position. This property is useful for our investigation as not all models studied herein exhibit sinks on the first token; in particular OLMo-2 frequently concentrates attention mass on punctuation, echoing a result in Sandoval-Segura et al. (2025).

In the plots that follow we average sink rates, Mixing Scores and ColSum concentrations over all the heads in a layer, and over all input sequences.

We plot all of these metrics as they measure subtly different things. The sink rate measures the specific occurrence of the first-token attention sink phenomena but misses split sinks and other "sharp" attention patterns; the ColSum concentration addresses this and captures split sinks and sinks on other tokens but still misses other "sharp" attention patterns like identity and previous token heads; the Mixing Score loses fidelity on specific sinking behavior but captures a wide range of "sharp" vs "sparse" attention patterns. See Fig. 31 for a visualization of some of these differences.

We first supplement the mixing / attention pattern conclusions drawn from Fig. 8 with both additional models and additional attention metrics: Ouro (Fig. 32), Huginn-0125 (Fig. 33), Retrofitted Llama (Fig. 34), Retrofitted OLMo-2 (Fig. 35), Retrofitted TinyLlama (Fig. 36), Parcae 370M (Fig. 37), Parcae 770M (Fig. 38). These figures demonstrate that for **sink rates and mixing scores**, the cyclic behavior and lack of significant depth-wise changes hold both for the other investigated models, and the additional stages of inference metrics.

Following Skean et al. (2025); Queipo-de-Llano et al. (2026) we additionally measure the compression of the residual stream $\boldsymbol{X}$ via the matrix-based entropy $H(\boldsymbol{X})$. This is averaged over all input sequences.

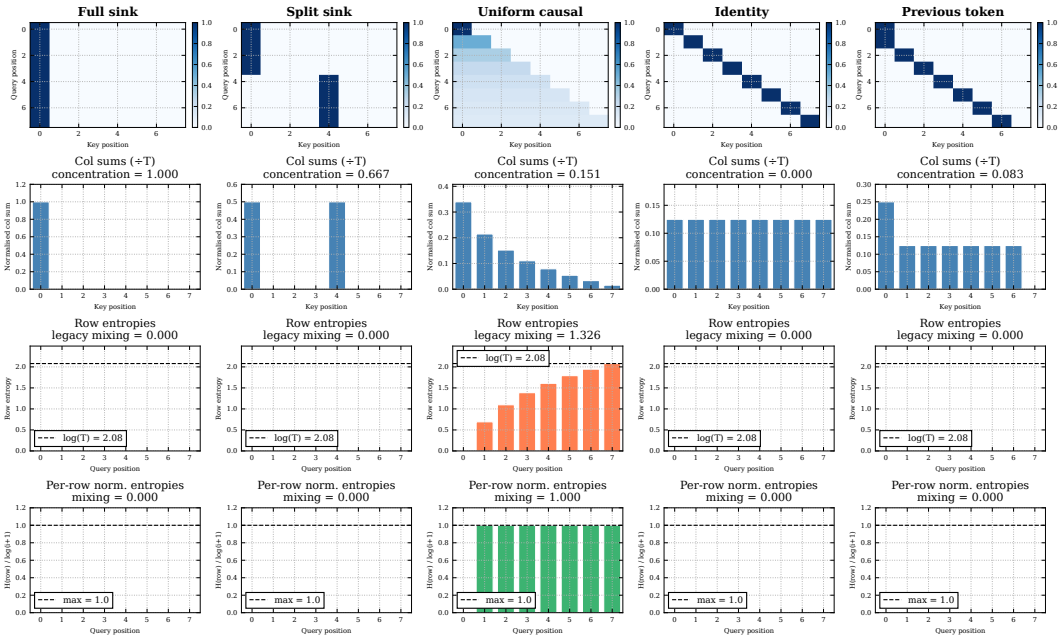


Figure 31: **Stages of Inference metrics on representative attention patterns:** ColSum Concentration, (legacy) Mixing Score (Queipo-de-Llano et al., 2026) and our adapted Mixing Score.

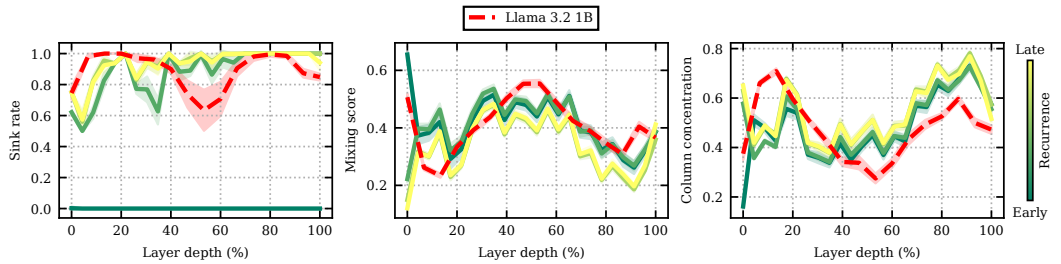


Figure 32: Complement to Fig. 8, visualising additional attention metrics for Ouro.

The results for residual entropy, visualized in Fig. 39, show almost the same story: Huginn-0125 as before shows no recognizable stages of inference, while all the retrofitted and Parcae models broadly modulate residual stream entropy throughout recurrent depth similarly to their feedforward counterparts. However, Ouro demonstrates different behavior: the residual entropies both change significantly with successive recurrences and do not closely follow the feedforward behavior. We believe that the divergence from feedforward behavior may derive from the norm structure of this model: the residual stream is normalised after each recurrent block, as visualised in Fig. 43, thus periodically shutting down massive activations. This is not the case for the retrofitted or Parcae series of models, which lack this norm and show much closer alignment with feedforward stages of inference.

We plot stages of inference for Ouro 2.6B in Fig. 41. This model is interesting due to the training regime followed by Zhu et al. (2025), which "upcycles" a 48 layer model from the 24 layer 1.4B parameter model. As a consequence, the first and second half of each recurrent block each independently align with the Llama feedforward stages of inference.

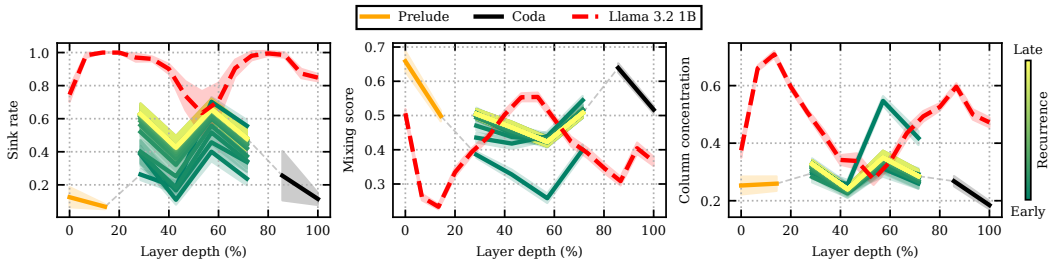


Figure 33: Complement to Fig. 8, visualising additional attention metrics for Huginn-0125.

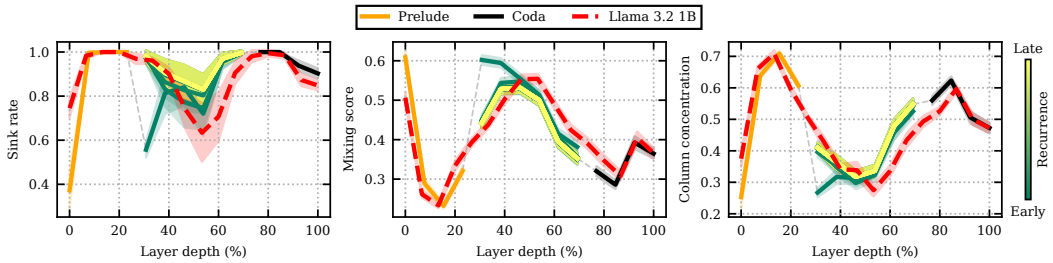


Figure 34: Complement to Fig. 8, visualising additional attention metrics for Retrofitted Llama.

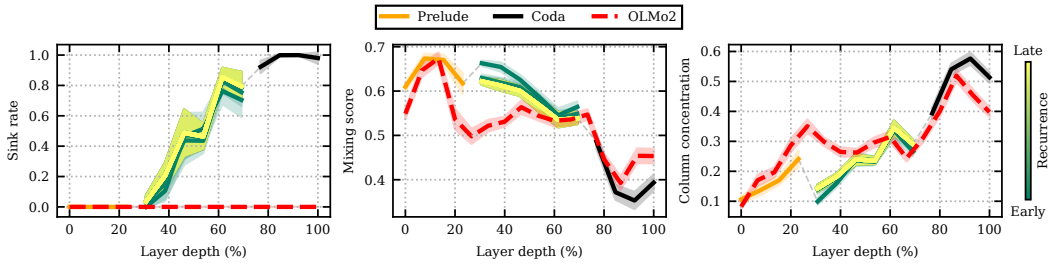


Figure 35: Complement to Fig. 8, visualising additional attention metrics for Retrofitted OLMo-2.

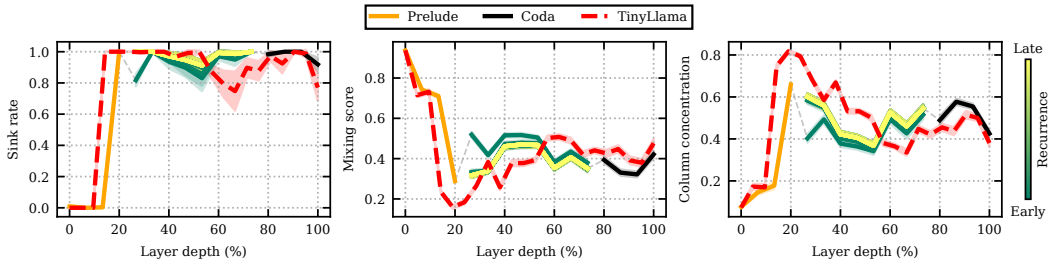


Figure 36: Complement to Fig. 8, visualising additional attention metrics for Retrofitted TinyLlama.

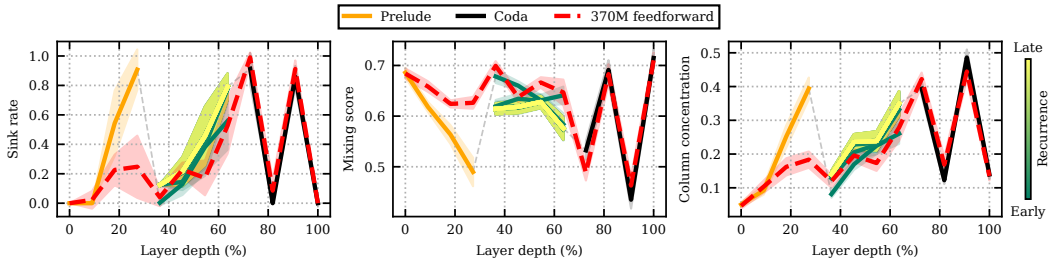


Figure 37: Complement to Fig. 8, visualising additional attention metrics for Parcae 370M.

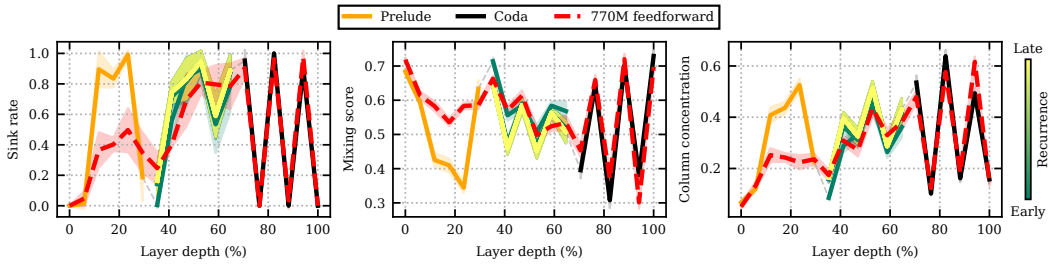


Figure 38: Complement to Fig. 8, visualising additional attention metrics for Parcae 770M.

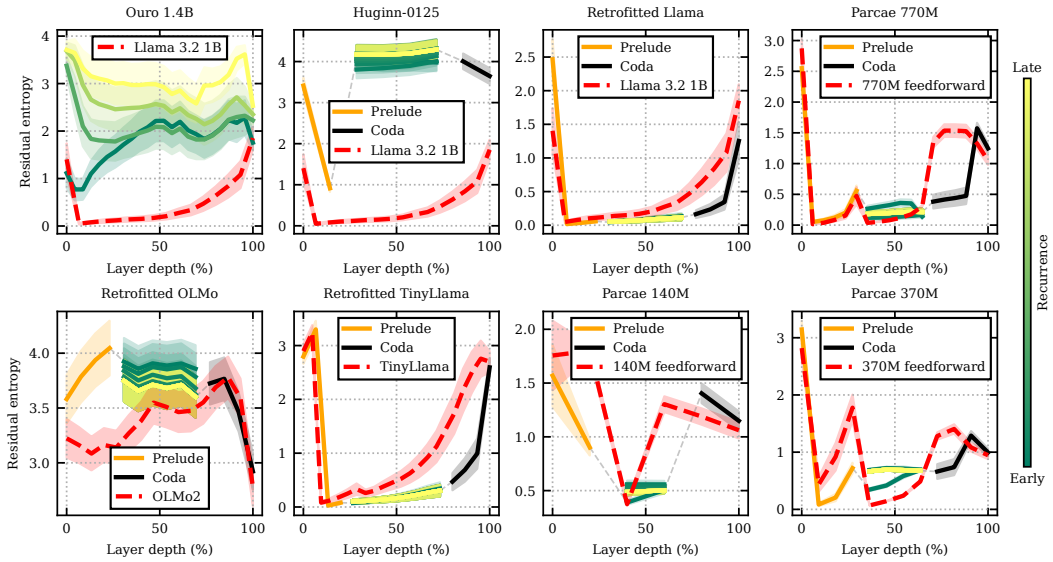


Figure 39: **Residual entropy with recurrent depth for all models explored in this paper**.

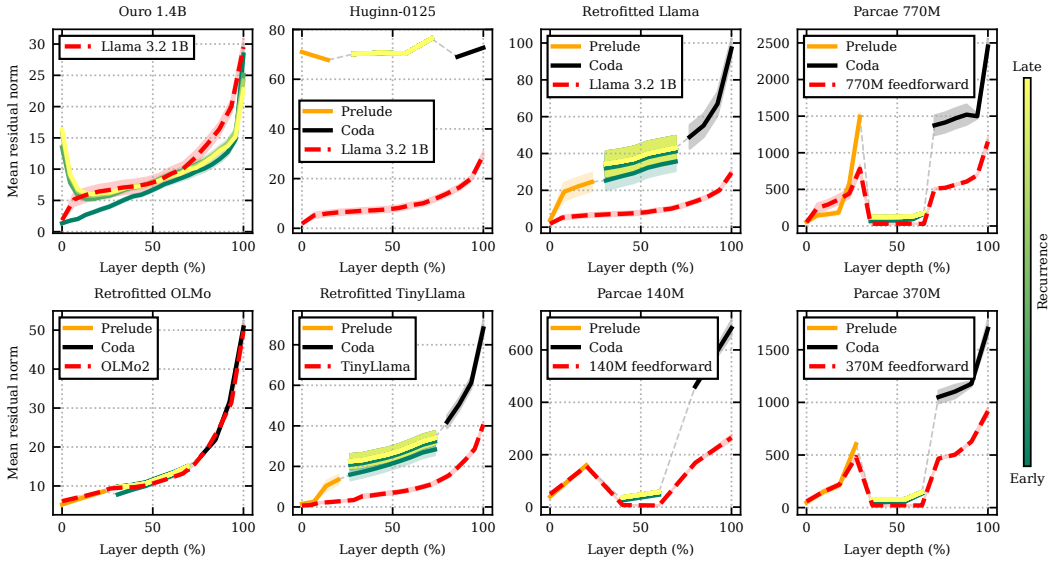


Figure 40: **Residual norm with recurrent depth for all models explored in this paper**. Mean taken across tokens and input sequences.

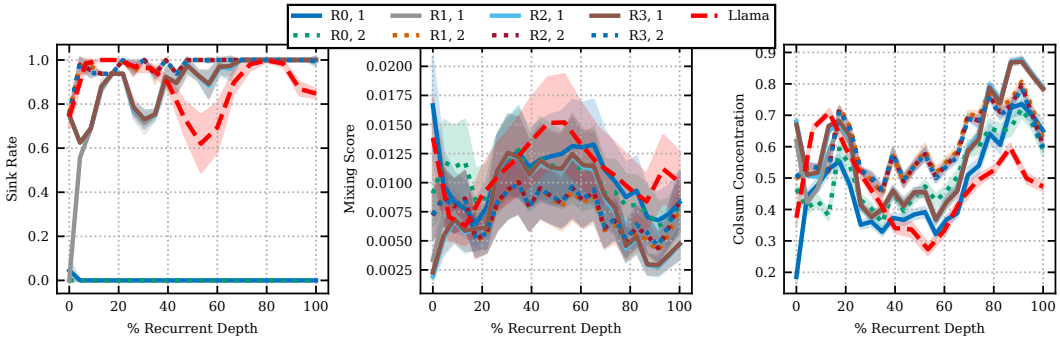


Figure 41: **Stages of inference for each recurrent loop in Ouro 2.6B.** For this model we separate out the first and second half of the recurrent block and overlay them, demonstrating that both halves have close alignment with the Llama feedforward stages of inference. We suggest that this likely arises due to the training regime of Zhu et al. (2025), which first trains a single 24 layer 1.4B parameter model, and then "upcycles" this into a 48 layer 2.6B parameter model by duplicating these layers, consequently duplicating the stages of inference as well.

G.2.1 Mixing Score Correlations

To quantify the similarity between the within-recurrence mixing-score profiles of each looped model and its matched feedforward control (Fig. 8), we correlate the two profiles across layer depth. For recurrence k, the looped profile consists of the prelude, the k-th application of the recurrent block, and the coda, in layer order. Ouro has no prelude or coda, so its profile is a single recurrence. Where the looped model and its control have different numbers of layers, we linearly interpolate the control's profile at the looped model's normalised layer depths. We report two summaries, shown in Fig. 42: the correlation between the example-averaged profiles (as visualised elsewhere in the paper), and the mean (and standard deviation) of correlations computed separately per example. The two agree closely. Retrofitted Llama ($r = 0.89$, averaged over recurrences) and Parcae 770M ($r = 0.81$) closely mirror their controls, and Ouro does so moderately ($r = 0.47$, $r = 0.64$ on the first recurrence). Huginn-0125, as expected, has no positive correlation, with $r = -0.24$. Spearman's ρ, comparing only the rank order of layers, gives the same qualitative picture.

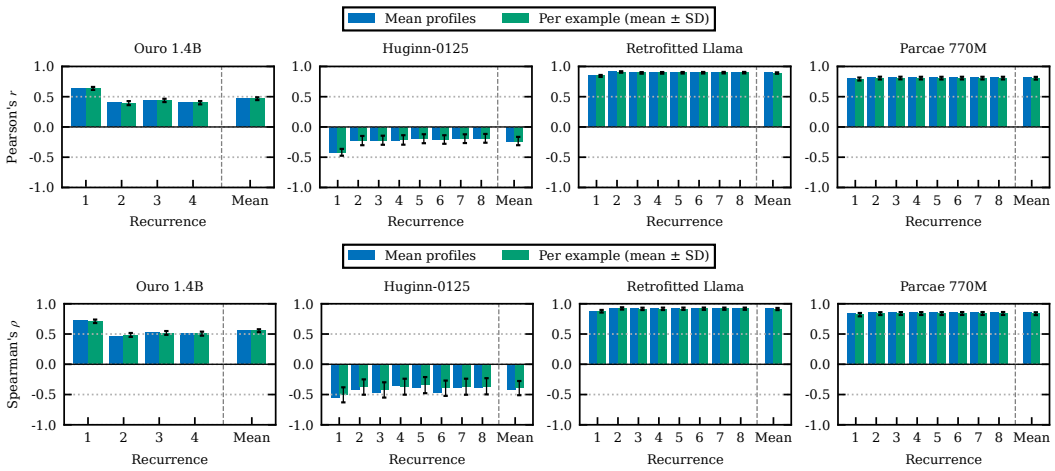


Figure 42: Correlation between the mixing-score profile of each recurrence and that of the matched feedforward control, measured by Pearson's r (top) and Spearman's ρ (bottom), on 256 GSM8K examples. Each looped profile comprises the prelude, one recurrence, and the coda. Controls are Llama 3.2 1B for Ouro, Huginn-0125 and Retrofitted Llama, and a size-matched one-recurrence Parcae model for Parcae 770M. Controls with a different layer count are interpolated at the looped model's layer depths. Blue bars correlate the example-averaged profiles. Green bars average per-example correlations, with whiskers showing one standard deviation across examples. The final bar in each panel is the mean over the recurrences shown. Ouro uses four recurrences. Other models show the first eight recurrences of a 32-recurrence rollout, whose coda follows the final recurrence.

G.2.2 Huginn-0125 Negative Mixing Stages Result

In Sec. 4 we suggested that the lack of stages of inference in Huginn-0125 is likely due to the normalization of the residual stream resulting in massive activations being unable to form: we demonstrate the impact of different norms by plotting residual stream magnitudes for a range of models in Fig. 43. This residual normalization means that Huginn-0125 is unable to develop the growth in residual stream magnitude that Queipo-de-Llano et al. (2026, Section 3.3) note *causes* compression behavior, leading to sink formation and mixing stages of inference.

We further support this hypothesis by ablating the massive activations from the Retrofitted Llama model (which *does* display stages of inference) via zeroing the output of the MLP in the second layer, which is responsible for its massive activations. In this setting, visualized in Fig. 44, we see that the model no longer exhibits stages of inference comparable to the feedforward model, suggesting that the presence of massive activations is required for stages of inference to emerge in looped models.

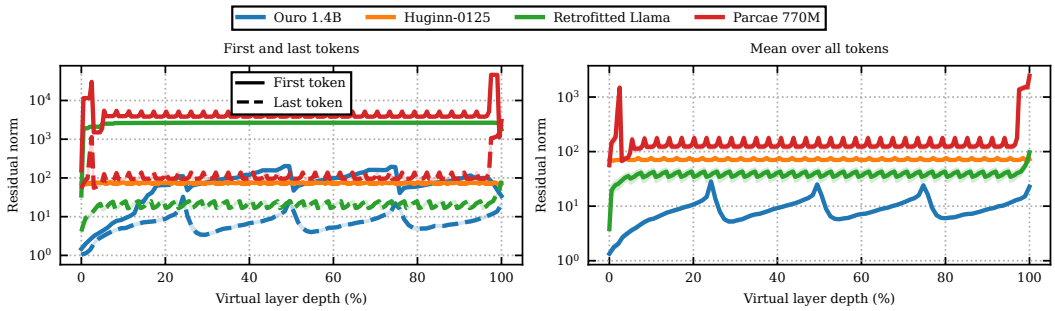


Figure 43: **Norms of the residual stream for a range of models,** demonstrating that Huginn-0125 is unable to develop the activation magnitude changes required for stages of inference due to its repeated normalization of the residual stream.

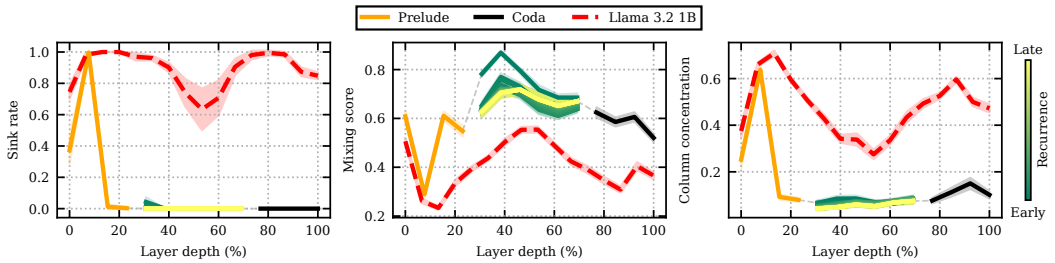


Figure 44: **Stages of inference for each recurrent loop in Retrofitted Llama for which the massive activations have been ablated.**

G.2.3 Generalization to Non-Reasoning Tasks

Throughout the main text of the paper, mixing experiments are conducted on the GSM8k dataset. In this appendix we verify that the attention pattern stages of inference we observe are not specific to this dataset, and also occur in a non-reasoning setting, for which we use the HellaSwag dataset (Zellers et al., 2019). We follow an identical experimental setup to the GSM8k experiments, running inference on 256 random examples from the test split.

To this end, we present in Fig. 45 the equivalent of Fig. 8. We see that the mixing stages of inference on this dataset closely follow the GSM8k dataset. In the interest of space we do not reproduce every other figure in this paper on Hellaswag, however these figures are easily reproduced using our codebase.

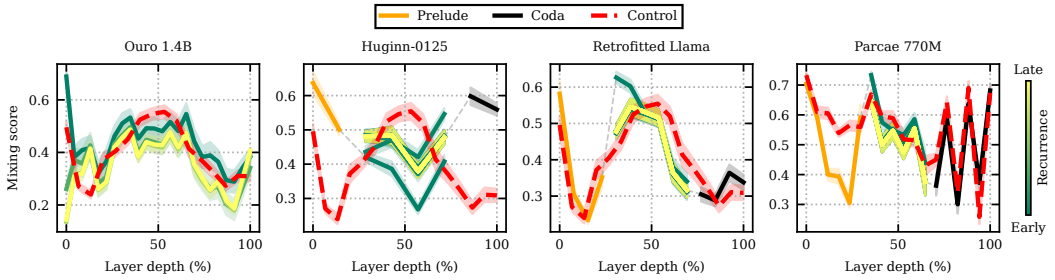


Figure 45: **Internal Mixing Stages per Recurrent Loop on the Hellaswag Dataset.**

G.3 Cyclic Functional Behavior

In this section we provide further evidence for the cyclic "functional" behavior of looped LLM layers, including performance on downstream tasks and metrics capturing output token prediction behavior.

We begin by examining the multiple choice "downstream" behavior. Feedforward comparisons are shown in Fig. 46.

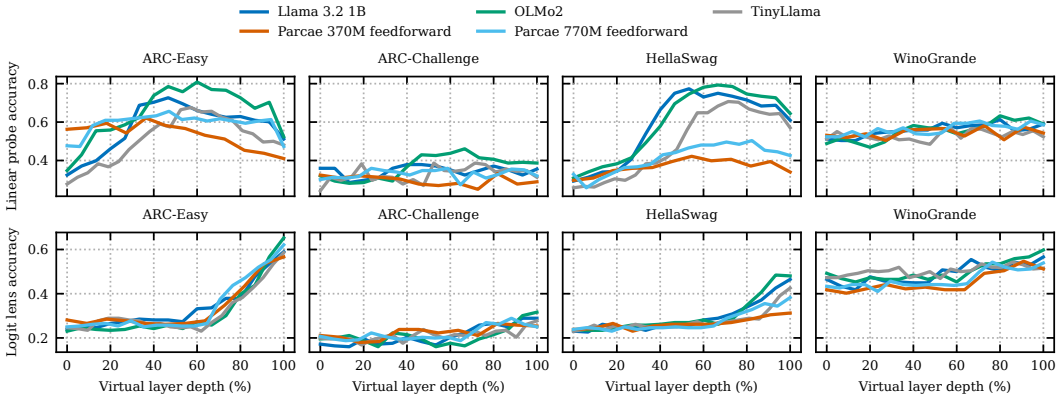


Figure 46: Linear probe and logitlens multiple choice performance per layer for a range of feedforward models.

We next present the linear probe and logitlens accuracy on the downstream multiple choice task for a wider range of datasets over all models in the main body of the paper: Ouro (Fig. 47), Huginn-0125 (Fig. 48), Retrofitted Llama (Fig. 49), Parcae 770M (Fig. 50).

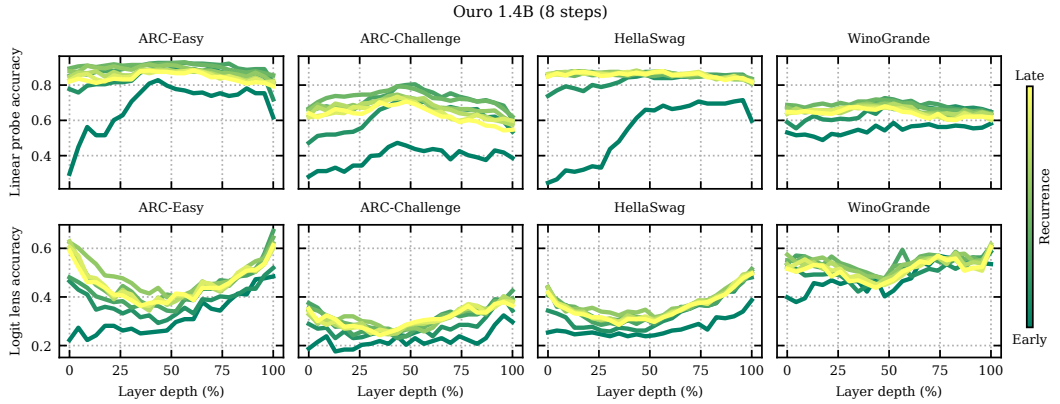


Figure 47: Linear probe and logitlens multiple choice performance per layer, per recurrence for Ouro

G.3.1 Do Looped LLMs Use Recurrences to Refine the Token Distribution?

We supplement this investigation into functional downstream behavior by plotting Entropy-Lens (Ali et al., 2025), investigating how these profiles compare to feedforward controls and whether looped LLMs use later recurrences to refine the output token distribution. We find in Fig. 51, as elsewhere, that the looped models mirror their feedforward counterparts with each recurrence. We do not, however, observe any meaningful reduction in entropy across successive recurrences.

G.4 Self-Organization Into Stages of Inference

One interesting question arising from our work is whether stages of inference emerge naturally or whether they are instead induced by specific aspects of the looped training procedure. The Parcae models provide strong evidence that they can emerge naturally, however these models are trained using a Poisson recurrence sampler and as such are trained "in part" as a feedforward model (although this makes up a minimal fraction of training examples).

To further isolate training method and provide additional evidence around the question of self-organisation, in this section we pretrain small-scale LTs from random initialization using a constant

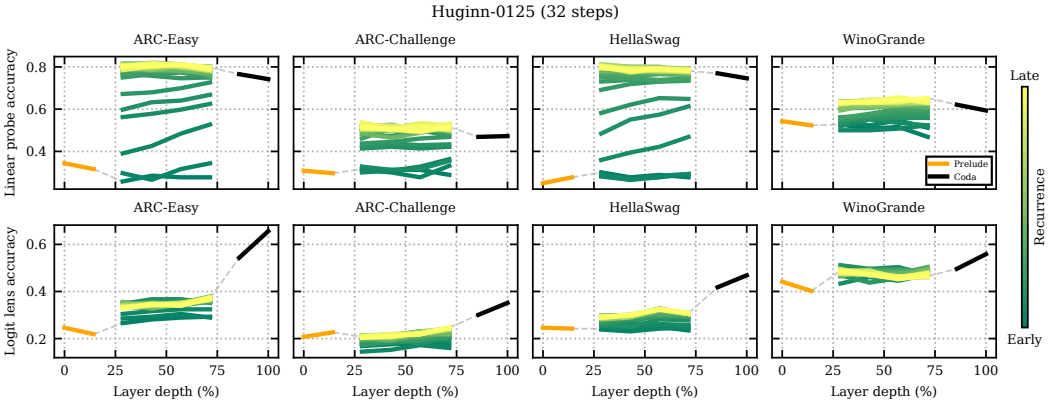


Figure 48: Linear probe and logitlens multiple choice performance per layer, per recurrence for Huginn-0125

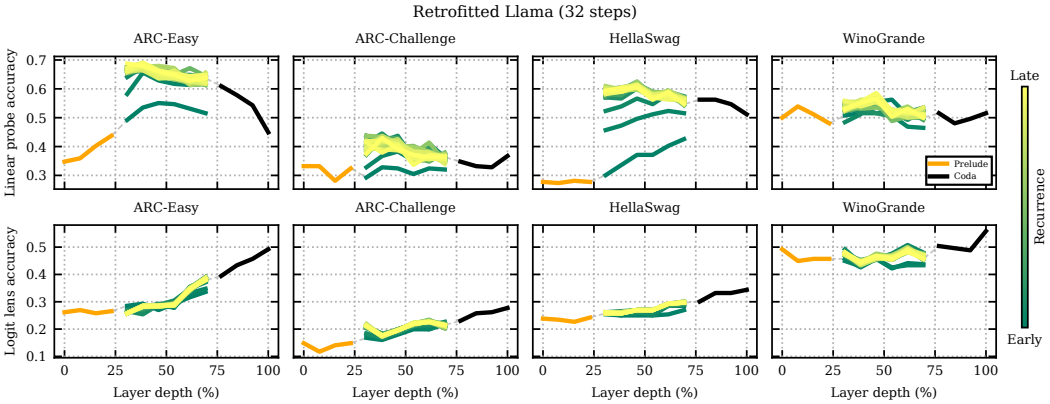


Figure 49: Linear probe and logitlens multiple choice performance per layer, per recurrence for Retrofitted Llama

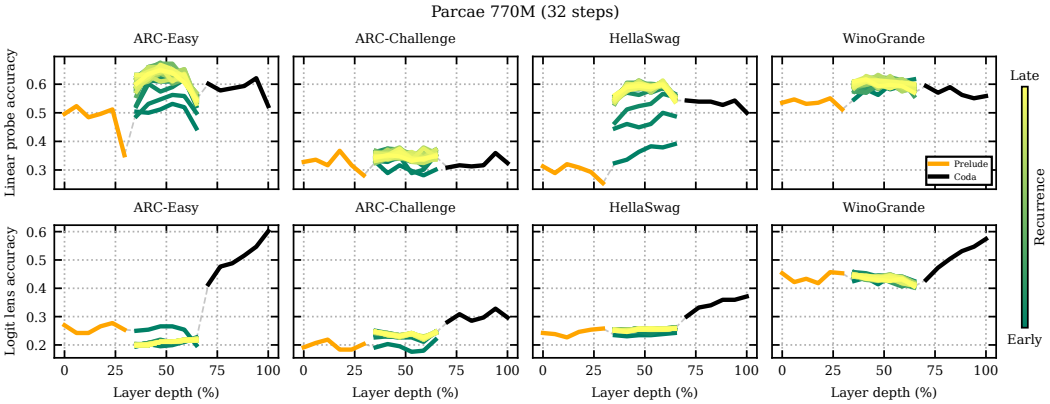


Figure 50: Linear probe and logitlens multiple choice performance per layer, per recurrence for Parcae 770M

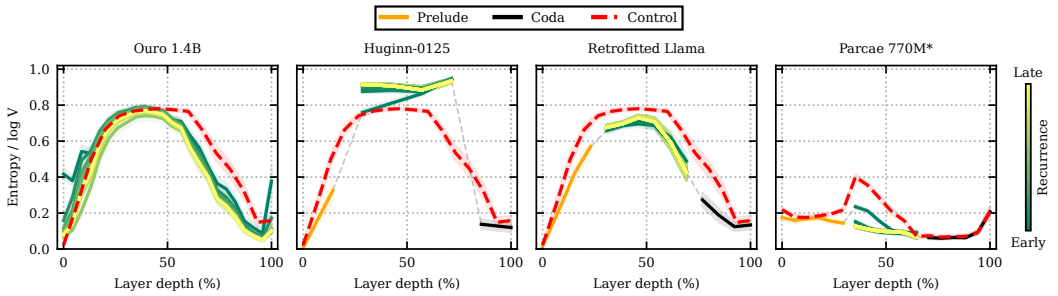


Figure 51: Entropy-lens plots for the main models investigated.

recurrence schedule ($T = 4$) and a single loss term applied only to the final hidden state. This setup removes any explicit pressure for intermediate loops to produce meaningful or "feedforward-like" representations. Our training protocol is adapted from Karpathy (2025); full experimental details are provided in App. C.

Results for configurations with 4, 8, and 12 recurrent layers are visualized in Fig. 52, alongside a depth-12 feedforward "control". Despite the absence of feedforward-biasing objectives, the models clearly *self-organize* into distinct mixing stages within each recurrent block. While these results should be viewed with caution due to their scale, they provide evidence that the functional decomposition of inference into mixing and refinement phases may not purely be a training artifact. Rather, it may be a computationally advantageous strategy that emerges even under the constraints of weight sharing.

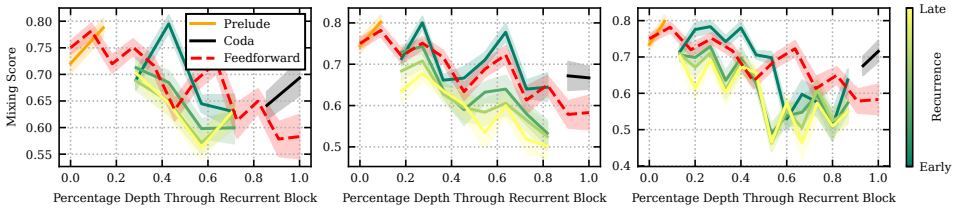


Figure 52: **Emergent Mixing Stages in LTs Trained from Scratch.** Mixing Scores for models trained with a constant $T = 4$ recurrence and loss applied only to the final state. Configurations follow $(2, L \otimes 4, 2)$ architecture for $L = 4, 8, 12$ respectively (left to right). Red lines indicate a 12-layer feedforward control. Even without feedforward-biasing loss terms, the models self-organize into repeating functional cycles that mirror feedforward stages.

The extended stages of inference metrics for the models visualized in Fig. 52 are presented in Figs. 53 to 55. Equivalent models with added input injection are visualized in Figs. 56 to 58, and equivalent models without prelude/coda layers in Figs. 59 to 61.

It appears in these small scale experiments that feedforward stages of inference are most closely replicated without input injection, and using prelude/coda layers, as in Figs. 53 to 55. The addition of input injection appears to mean that the *final* recurrence follows the feedforward stages of inference more closely, but to the detriment of earlier recurrences.

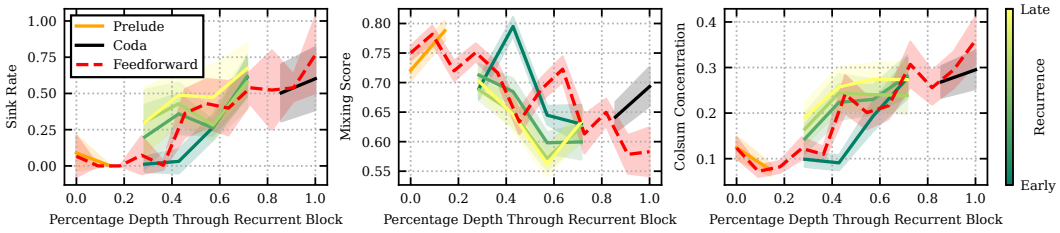


Figure 53: **Stages of inference metrics for a small-scale LT of configuration** $(2, 4 \otimes 4, 2)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

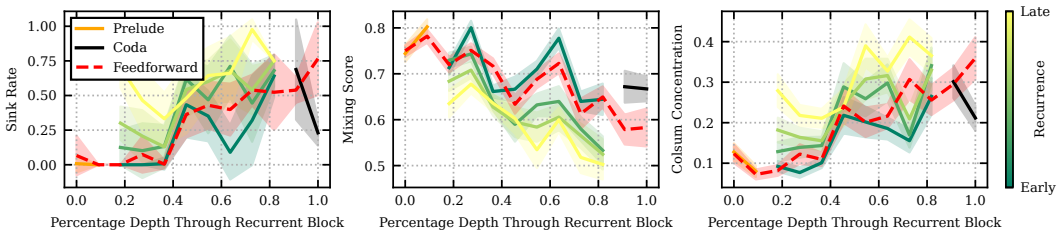


Figure 54: **Stages of inference metrics for a small-scale LT of configuration** $(2, 8 \otimes 4, 2)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

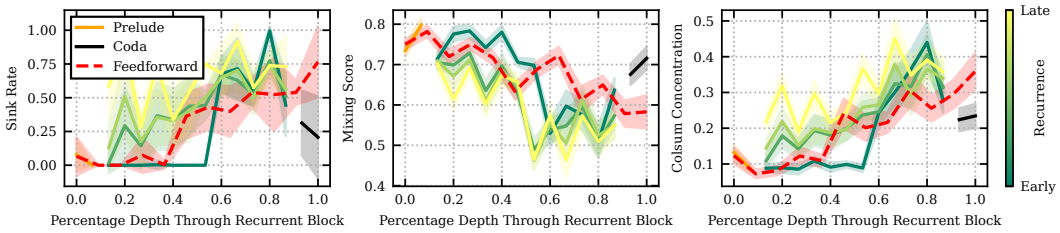


Figure 55: **Stages of inference metrics for a small-scale LT of configuration** $(2, 12 \otimes 4, 2)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

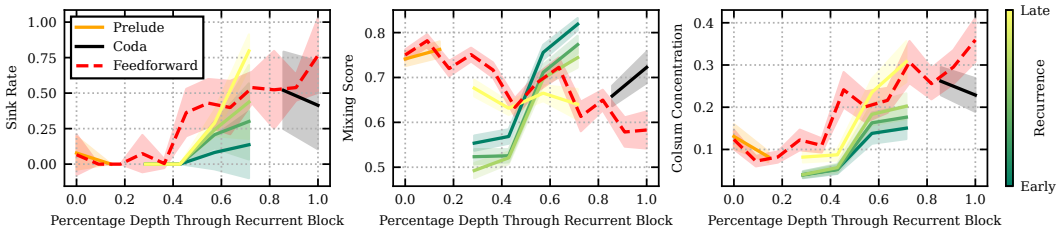


Figure 56: **Stages of inference metrics for a small-scale LT of configuration** $(2, 4 \otimes 4, 2)_I$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

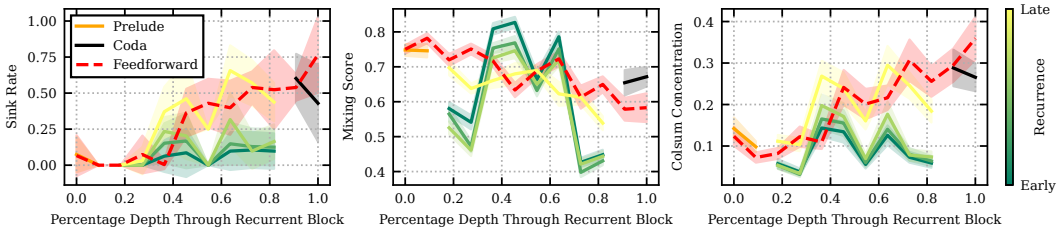


Figure 57: **Stages of inference metrics for a small-scale LT of configuration** $(2, 8 \otimes 4, 2)_I$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

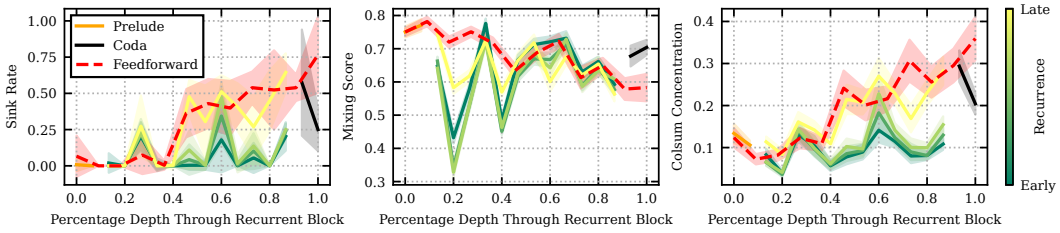


Figure 58: **Stages of inference metrics for a small-scale LT of configuration** $(2, 12 \otimes 4, 2)_I$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

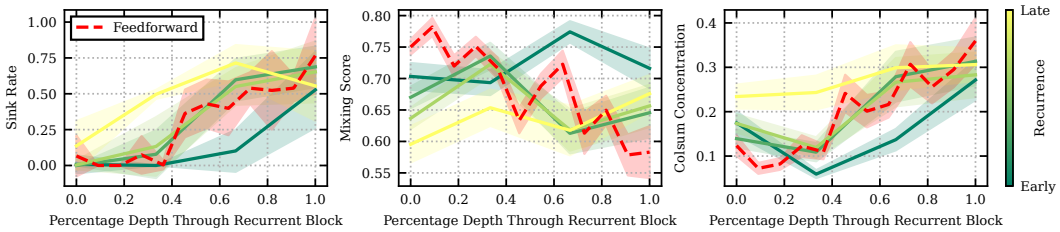


Figure 59: **Stages of inference metrics for a small-scale LT of configuration** $(0, 4 \otimes 4, 0)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

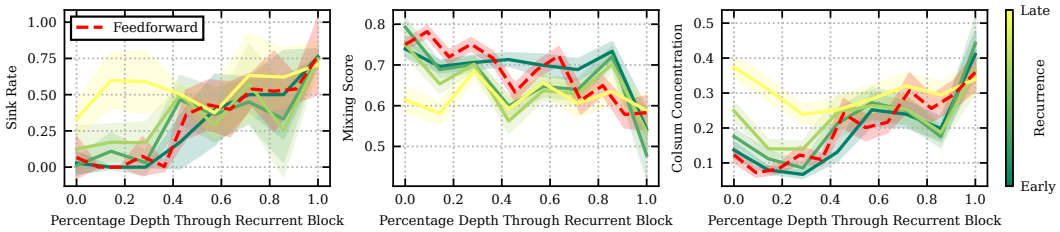


Figure 60: **Stages of inference metrics for a small-scale LT of configuration** $(0, 8 \otimes 4, 0)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

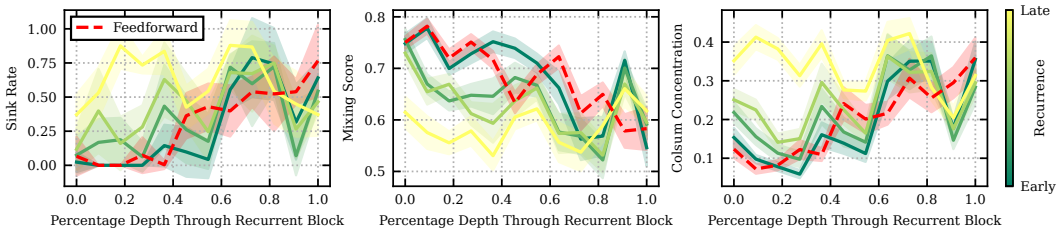


Figure 61: **Stages of inference metrics for a small-scale LT of configuration** $(0, 12 \otimes 4, 0)$, compared to a "control" feedforward Transformer with the same training configuration and depth 12.

H Code, Data and Model Licenses

Table 5: Existing assets used in this work, with their licenses.

Asset	Identifier / version	License
Code		
LM Evaluation Harness	`lm-eval` 0.4.13	MIT
TransformerLens	`transformer-lens` 3.9.0	MIT
nanochat	`karpathy/nanochat`	MIT
Parcae	`sandyresearch/parcae`	MIT
Recurrent pretraining (Huginn)	`seal-rg/recurrent-pretraining`	Apache 2.0
Hugging Face Transformers	`transformers` 5.17.0	Apache 2.0
Data		
GSM8K	`openai/gsm8k` (`main`)	MIT
HellaSwag	`Rowan/hellaswag`	MIT
ARC-Easy, ARC-Challenge	`allenai/ai2_arc`	CC BY-SA 4.0
WinoGrande	`allenai/winogrande` (`winogrande_xl`)	CC BY
FineWeb-Edu[a]	`SandyResearch/fineweb-edu-shuffled`	ODC-By 1.0
Models		
Huginn-0125	`tomg-group-umd/huginn-0125`	Apache 2.0
Ouro 1.4B	`ByteDance/Ouro-1.4B`	Apache 2.0
Ouro 2.6B	`ByteDance/Ouro-2.6B`	Apache 2.0
Retrofitted Llama[b]	`smcleish/Recurrent-Llama-3.2-train-recurrence-32`	Apache 2.0
Retrofitted OLMo	`smcleish/Recurrent-OLMo-2-0425-train-recurrence-32`	Apache 2.0
Retrofitted TinyLlama	`smcleish/Recurrent-TinyLlama-3T-train-recurrence-32`	Apache 2.0
Parcae 140M, 370M, 770M[c]	`SandyResearch/parcae-140m`, `-370m`, `-770m`	Not specified
Llama 3.2 1B	`meta-llama/Llama-3.2-1B`	Llama 3.2 Community License
OLMo 2 1B	`allenai/OLMo-2-0425-1B`	Apache 2.0
TinyLlama 1.1B	`TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T`	Apache 2.0
GPT-2 (124M)	`openai-community/gpt2`	MIT

[a]Pretraining data for the Parcae recurrence controls and stability-sweep models trained in this work; derived from `HuggingFaceFW/fineweb-edu` (ODC-By 1.0, also subject to the Common Crawl terms of use). [b]Fine-tuned from Llama 3.2 1B, so the Llama 3.2 Community License also applies. [c]The released weights and tokenizer state no license; the Parcae code is MIT.

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 attempt to accurately convey the paper's contributions and scope in the abstract and introduction. We try to make this brief and explicitly clear with a "key insights" box at the end of the introduction.

   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 highlight specific limitations throughout where appropriate (e.g. small scale of "self-organization" experiments), and discuss some high-level limitations in the conclusion.

   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: To the best of our knowledge, we articulate all relevant assumptions in our proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: This information is covered in some detail in the appendices.

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: Full code and documentation are provided. We have attempted to make reproduction extremely simple.

   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: The training side of this is only slightly applicable to our work via the Nanochat experiments: here we used the standard nanochat splits and training setup, as made clear in the paper. Our test data and splits are made clear in the appendices.

   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: We have included error bars in all appropriate plots.

   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: Compute requirements are relatively small: all of the stages of inference experiments can be run in less than half a day on an A10 GPU. Nanochat training runs take a bit longer: up to 12 hours for one run on an H100. We specify this in the experimental details section of the appendix.

   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 research complies with the code of ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

Justification: We believe the societal impacts of this particular work are negligible, but the field of recurrent depth for test time compute has clear potential negative impacts: we highlight these in the appendices.

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: We do not release any new models or data.

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: We make it clear what work we build upon, licenses are explicitly mentioned in App. H.

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: We do not introduce new assets in this paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We do not crowdsource or work 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: We have no study participants.

    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: We do not use LLMs in the ways described here. LLMs were used to improve clarity of writing, and for code assistance: as per the Main Track Handbook this does not necessitate declaration.

    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.