OVERCOMING THE KV CATCH: STATE-SPACE ANALYSIS OF TRANSFORMER GENERATION DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer-based large language models (LLMs) are feed-forward networks, yet they are deployed autoregressively, giving rise to rich dynamical phenomena such as repetition loops. Analyzing these phenomena with dynamical systems tools, such as fixed points and their stability, faces a basic obstacle: due to their growing memory, these systems lack a *fixed-size state*. Here, we take a first step towards state-space analysis of growing-memory transformers. We show that an attention head without positional embeddings is exactly described by a non-linear self-interacting Markov chain with a fixed-size state, and use its timescale separation to derive a self-consistency condition for fixed points. Building on this condition, we develop an optimization method that finds both stable and unstable fixed points. Applied to position-ablated heads of a pretrained 1-layer transformer, it reveals stable repetition attractors, asymptotically unstable yet persistent semantic clusters, and saddle states between neighboring attractors. We then introduce a multi-timescale mean-field approximation for general attention layers, which closely matches the full model, and use it to find an unstable repetition whose predicted escape dynamics we confirm with real-model rollouts. Our work opens the door to applying dynamical systems tools to understand LLM generation.

1 INTRODUCTION

Transformer-based LLMs are the most widely deployed artificial neural network today (Vaswani et al., 2017). However, despite their usefulness and rapid adoption in many domains, our understanding of their internal mechanisms remains lacking. The field of mechanistic interpretability has evolved rapidly over the last few years in an attempt to address this issue (Rai et al., 2025), finding different kinds of interpretable mechanisms inside pre-trained and deployed transformers (Meng et al., 2022; Elhage et al., 2021; Zou et al., 2025).

A relatively underexplored area of this problem is related to the dynamic nature of language generation. Despite the feedforward architecture of transformer models (figure 1a), language models are deployed in a "*closed-loop*" setting, where the model's outputs are fed back *autoregressively* as inputs to generate new sequences of text (figure 1b). This autoregressive process naturally gives rise to time-evolving dynamics that may (and indeed do) exhibit a range of dynamical behaviors – from stability and divergence (Nasr et al., 2025; Arora et al., 2023), to repetitive outputs (Holtzman et al., 2020). Understanding these dynamics is a core challenge that is crucial both for *interpreting* and *controlling* these models – both critical goals of the AI safety community.

Consequences of dynamics are classically treated in the fields of dynamical systems and control. The *state* of a system and its time evolution laws are defined, and various methods characterize the short- and long-term behavior of the system when starting from various states (Strogatz, 2018). Recently, these tools were applied in the context of mechanistic interpretability of artificial neural networks for reverse-engineering high-dimensional RNNs, where it exposed an interpretable low-dimensional structure in networks trained on cognitive tasks (Sussillo & Barak, 2013; Vyas et al., 2020).

Motivated by these examples, we want to analyze transformer language models in the framework of dynamical systems. However, transformers have a memory that grows with the sequence length, thereby precluding a straightforward definition of a state for the system. To overcome this, we introduce a definition of a constant-size state for 1-layer transformers. We show it is exact in the case

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

of position-ablated models, and can also approximate general attention layers. Using this definition, we develop an algorithm to find both stable and unstable fixed points of the autoregressive process.

Our **contributions** can be summarized as follows: (i) We **derive an exact fixed-size state** for a position-ablated semantic attention head and use its slow–fast structure to obtain a self-consistency condition for fixed points. (ii) We **develop an optimization method** that finds both stable and unstable fixed points, including states inaccessible through autoregressive generation. (iii) We **characterize fixed points** of attention heads, identifying stable repetition attractors, asymptotically unstable but persistent semantic clusters, and saddle-mixture states separating neighboring attractors. (iv) We **introduce the Multi-Timescale Mean-Field (MTMF) approximation** for general one-layer attention and validate its predicted unstable dynamics using real-model rollouts.

2 Related Work

Transformers as Dynamical Systems. Zekri et al. (2025) define LLMs with a finite context window as a Markov Chain using a sliding window assumption, and relate repetitions to its stationary distribution. Closest to our work is Ildiz et al. (2024), who map an attention head without positional embeddings to a Context-Conditioned Markov-Chain (CCMC), and find repetitions in its generation dynamics. However, they make a number of further simplifying assumptions in their model (namely, identity Value matrices and linearly independent token embeddings) that fundamentally restrict the dynamics of their model, in order for it to be analytically tractable. We make no further assumptions to model real pretrained heads.

Reverse-Engineering RNNs. Our work is inspired by uses of state-space analysis to reverse-engineer interpretable mechanisms in trained RNN networks (Vyas et al., 2020; Barak, 2017; Mante et al., 2013). This included locating fixed points (Sussillo & Barak, 2013; Katz & Reggia, 2017) and separatrices (Dabholkar & Barak, 2025). Trajectories themselves can be described as graphs (Brennan et al., 2023; Turner et al., 2021). In all cases, the resulting dynamics can be related to cognitive components of the relevant tasks.

Dynamical Phenomena in LLMs. Different kinds of dynamical phenomena are documented in LLMs. The most famous of these is collapse into degenerate text repetitions. Some studies of repetition loops are mostly empirical and offer ad-hoc mitigation techniques (Holtzman et al., 2020; Xu et al., 2022). Some mechanistic works found different explanations for repetitions (Hiraoka & Inui, 2025; Yao et al., 2025). Another phenomenon is divergence. Nasr et al. (2025) use model divergence for a training-data extraction attack. Arora et al. (2023) analyze accumulating errors origination from closed-loop generation exposure bias.

3 Results

3.1 The Challenge of LLM Dynamics – "The KV Catch"

During autoregressive generation, a transformer receives at position i an input token t_i, and outputs the probability distribution p_i (see appendix B for notations). Because of the causal masking applied in self-attention heads, all tokens before i are encapsulated in the *KV Cache* (figure 1a). If we denote by KV_i all of the KV vectors in the model up to position i, we can write the action of the model in the following way:

$$KV_i = f(KV_{i-1}, t_i), \qquad p_i = g(KV_i, t_i). \tag{1}$$

It is tempting to declare KV_i as the state of the system. But this fails because of a *dimensionality-mismatch* between the state at different steps (figure 1b):

$$KV_i \in \mathbb{R}^{i \times D} \implies \dim(KV_i) \neq \dim(KV_j) \quad \forall j \neq i, \tag{2}$$

where D represents the full concatenated dimension of all the KV vectors in the model for a single position. This mismatch prohibits the standard definition of dynamical objects such as fixed points, since there is no straightforward way of equating or comparing KV_i between different positions. Consider a model that collapsed into a repetitive loop: *"We are going to the game to the game to the game…"*. While the repetitions here are obvious in an intuitive sense, note that in a strict sense, the model never repeats itself: "We are going to the game" is a *different sequence* from "We are going to

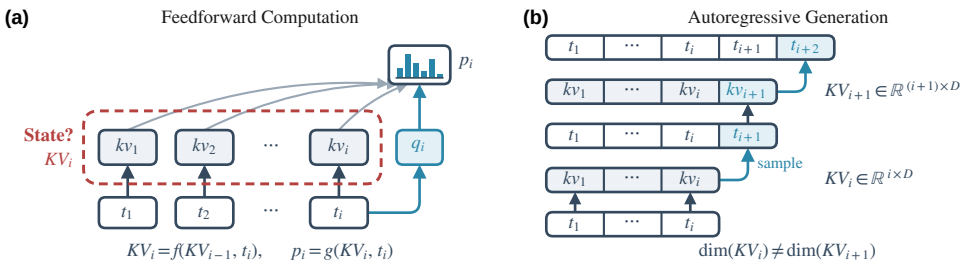

Figure 1: **The KV catch. (a)** A transformer is a feedforward structure, with the effect of past inputs captured by the key-value (KV) cache. **(b)** Autoregressive generation creates a growing KV cache whose dimension depends on sequence position, precluding a fixed-size state-space description.

the game to the game". Thus the input to the model is never the same sequence as a previous step in the generation. A few possible workarounds come to mind, such as restricting to a sliding context window, using simplified architectures or learning a latent dynamical space. While possible, these options either change the analyzed dynamics, or add further complications (see appendix K for a detailed discussion), and we instead follow the direction laid out by Ildiz et al. (2024).

3.2 State Representation of a Semantic Head

We first present our results and method for the simplest case of a transformer language model: a 1-layer, attention-only model with a single, purely "semantic" (i.e. with no positional embeddings) attention head. Although simplified, this derivation captures the core insights of our framework and methodology. In section 3.6 we explain how to extend them to the case of a 1-layer model with a general multi-head attention layer (see appendix L for discussion of multilayer models).

We begin by relating self-attention to a Markov Model, similar to Ildiz et al. (2024). A crucial difference is that we make no assumptions on the query, key and value matrices (W_Q, W_K, W_V) of the attention head, or the embedding matrices of the model. Since the inputs to the first layer are raw uncontextualized tokens, a *semantic* head has no way to distinguish between tokens in different positions, as $q_t^i = q_t^j \equiv q_t$ for all i, j (and similarly for k_t, v_t), and thus the attention score of each token will depend only on the token identity. The same holds for the value of tokens. The output of the head will then only depend on the identity and the counts of the tokens in the context window.

Formally, the current token t_i generates the query vector q_{t_i}. For any other token $t \in V$, its unnormalized attention score will be $s_{t_i,t} \triangleq q_{t_i}^\top k_t / \sqrt{d_{\text{head}}}$. The normalized attention weight $\alpha_{t_i,t}$ it receives is the output of the causally-masked softmax on these scores, and the final output o_i is the weighted sum:

$$\alpha_{t_i,t} = \frac{\exp(s_{t_i,t})}{\sum_{j\le i}\exp(s_{t_i,t_j})}, \qquad o_i = \sum_{j\le i} \alpha_{t_i,t_j}\, v_{t_j} \in \mathbb{R}^{d_{\text{head}}}. \tag{3}$$

Using our assumptions, we replace both position-wise sums to token-wise sums over all tokens in the vocabulary. Denoting the counts (the number of times token t appears in the context up to position i) by $N_i(t)$, we get:

$$o_i = \sum_{t\in V} N_i(t)\,\alpha_{t_i,t}\, v_t = \sum_{t\in V} \frac{N_i(t)\exp(s_{t_i,t})}{\sum_{t'\in V} N_i(t')\exp(s_{t_i,t'})}\, v_t = \sum_{t\in V} \frac{\pi_i(t)\exp(s_{t_i,t})}{\sum_{t'\in V} \pi_i(t')\exp(s_{t_i,t'})}\, v_t \tag{4}$$

where we defined $\pi_i(t) \triangleq N_i(t)/i$ as the proportion of token t in the context up to position i. Importantly, we note that the output turns out to be a function of only two variables (figure 2a): The *current token* $t_i \in V$, and the *current unigram distribution* in the context $\pi_i \in \mathbb{R}^{|V|}$. The final output of the model will also be a function of these two variables:

$$p_i(\pi_i, t_i) = \text{softmax}\big(W_U\big(e(t_i) + W_O\, o_i(\pi_i, t_i)\big)\big) \triangleq \mathcal{F}(\pi_i, t_i), \tag{5}$$

where $e(t) \in \mathbb{R}^{d_{\text{model}}}$ is the semantic embedding of token t, $W_O \in \mathbb{R}^{d_{\text{model}}\times d_{\text{head}}}$ is the output projection of the attention head, $W_U \in \mathbb{R}^{|V|\times d_{\text{model}}}$ is the unembedding matrix, and $\mathcal{F}(\pi_i, t_i)$ denotes the full

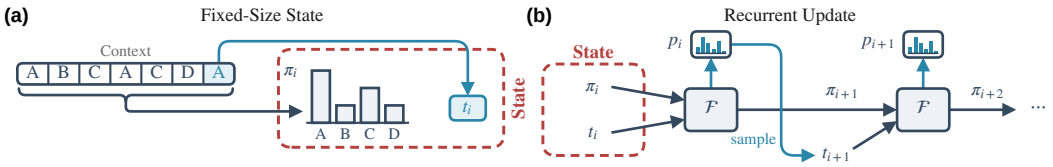

Figure 2: **Fixed-size state of a semantic head.** **(a)** The context is summarized by its unigram distribution π_i together with the current token t_i. **(b)** The recurrent update: $\mathcal{F}$ maps (π_i, t_i) to the next-token distribution p_i and the next state π_{i+1}; t_{i+1} is sampled from p_i and fed back.

forward pass of the model. Crucially, these are *fixed-size variables* – their dimensions are constant and are independent of the position i. We can write the state-space update equations for these variables (figure 2b):

$$t_{i+1} \sim p_i(\pi_i, t_i), \qquad \pi_{i+1} = \frac{i}{i+1}\pi_i + \frac{1}{i+1}t_{i+1} = \pi_i + \frac{1}{i+1}\left(t_{i+1} - \pi_i\right). \tag{6}$$

The next token t_{i+1} is simply sampled from the output distribution of the previous step. The context distribution is rescaled and updated according to the newly sampled token. We can interpret these equations in two perspectives – one emphasizing the current token, the other the probability distribution.

3.2.1 Self-Interacting Markov Chain Interpretation

We can interpret these equations as a *non-linear self-interacting Markov Chain* (SI-MC): Given a fixed unigram context distribution π, we denote the induced Markov Chain transition matrix as $P_\pi \in \mathbb{R}^{|V|\times|V|}$. The t-th row of P_π is given by the forward pass $\mathcal{F}(\pi, t)$ (equation 5). Importantly, since softmax returns strictly positive probabilities, P_π contains only non-zero entries and is thus ergodic and aperiodic, and has a unique stationary distribution which we denote as $\bar{\pi}(\pi)$. Unlike a standard Markov Chain, during autoregressive generation sampled tokens are fed back to update π_{i+1}, which updates the transition matrix for the next step $P_{\pi_{i+1}}$, leading to a non-linear self interaction.

3.2.2 Slow-Fast System Interpretation

A second interpretation can be achieved by invoking the *timescale separation* property of these equations. Timescale separation is widely applied in dynamical systems analysis to simplify the analysis of complex multi-scale systems (Strogatz, 2018), by assuming a fast variable converges rapidly to a state which depends on the value of a slow variable. From the decaying step size $1/(i+1)$ we can see that in the long context limit $i \to \infty$, the unigram distribution is updated at an increasingly slower rate (since it is an infinite memory accumulator). We can thus recognize π as our slow variable, and in the limit treat it as *frozen* compared to the fast token-sampling process. The tokens are then sampled according to the frozen Markov transition matrix P_π. For any finite mixing time chain, the tokens will eventually be sampled according to the stationary distribution $\bar{\pi}(\pi)$, which is fed back to the slow context distribution. Using this insight, we can define a reparameterized time $\tau = \ln i$ and obtain an averaged Ordinary Differential Equation (ODE) for the distribution (see appendix C.2 for the full derivation):

$$\frac{d\pi}{d\tau} = \bar{\pi}(\pi) - \pi. \tag{7}$$

3.3 Fixed Points and the Self-Consistency Condition

Nonlinear dynamical systems, such as our transformer, are not trivial to analyze. A common approach is to focus on special regions in state space that are easier to understand. The most natural candidates are fixed points of the dynamics. Using the slow-fast formalization we can readily see that a fixed point of the system demands $d\pi/d\tau = 0$, allowing us to define:

Self-Consistency Condition. A context distribution π^* is a fixed point of the slow process if and only if it equals the stationary distribution it induces:

$$\pi^* = \bar{\pi}(\pi^*) = \pi^* P_{\pi^*}. \tag{8}$$

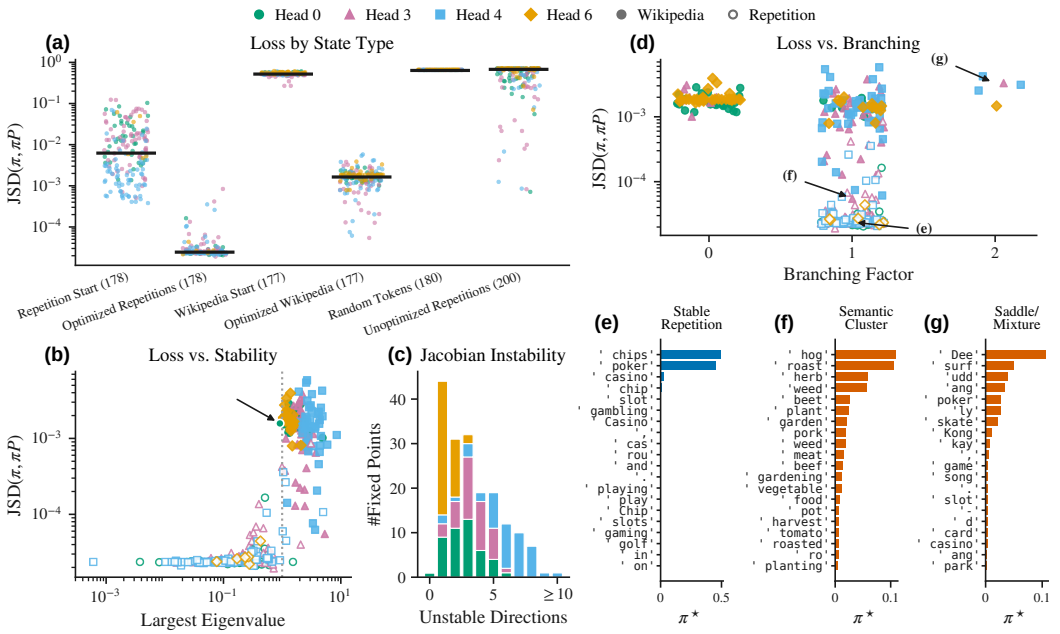

Figure 3: **Fixed points of semantic attention heads.** **(a)** Loss $\mathrm{JSD}(\boldsymbol{\pi}, \boldsymbol{\pi P_\pi})$ of optimized states and their starting points, compared with random-token contexts and unoptimized single-token and two-token repetitions; dots are states coloured by head, black bars are medians. Optimization lowers the loss by about two orders of magnitude for both repetitions and Wikipedia contexts. **(b)** Loss against the largest real part of the Jacobian eigenvalues; the dotted line marks the stability boundary at 1. 171 of 178 optimized repetitions are stable, against 1 of 177 Wikipedia fixed points (arrow). **(c)** Number of unstable directions of the Wikipedia fixed points, stacked by head (top 10 eigenvalues computed). **(d)** Loss against the measured branching factor - the number of consistent clusters reached by 20 rollouts after 2000 generated tokens (horizontal jitter added). Arrows mark the examples in (e–g). **(e–g)** Top-20 tokens of three fixed points of head 3: a stable repetition, a semantic cluster, and a saddle mixture (expanded in Fig. 4).

Note that this is a non-linear eigenvalue problem, since P_π depends nonlinearly on π.

3.4 Fixed-Point Finding Algorithm

In this section we present a simplified version of our fixed point finding algorithm. Outline and full implementation details are given in appendix D, and an extended version for a general multi-head attention layer is described in appendix J.

A crucial observation in designing the algorithm is that fixed points can be both *stable* and *unstable*. Simply running the dynamics can reach the stable points, but not the unstable ones. Similar methods in computational neuroscience (Sussillo & Barak, 2013) minimized the *speed* of the system: $L(x) = \frac{1}{2}\|\dot{x}\|^2$. The speed vanishes at all fixed points regardless of their stability, and thus applying gradient descent to it can lead to both kinds. We combine this approach with our analysis of semantic heads:

Key Insight 1. We can leverage the self-consistency condition (equation 8) to construct a loss function that vanishes if and only if π is a fixed point:

$$\mathcal{L}(\pi) = d(\pi, \pi P_\pi), \tag{9}$$

where $d(\cdot, \cdot)$ is some distance measure between probability distributions. We use the Jensen–Shannon Divergence (JSD) in all our experiments, as it is symmetric and bounded. The distance $d(\pi, \pi P_\pi)$ acts as our proxy for "speed".

Key Insight 2. Leveraging equations 4 and 5, we directly compute the induced transition matrix P_π as a differentiable function of π. This is achieved by simulating the model's forward pass using the

density vector π directly in the attention mechanism rather than an explicit sequence of tokens. While the full transition matrix contains $|V|^2$ entries, which could be prohibitive for modern vocabulary sizes, evaluating the loss only requires the vector-matrix product πP_π. Consequently, we never materialize P_π in memory, accumulating the product chunk-by-chunk instead.

Finally, to account for the fact that our optimization parameter is a probability distribution, we use a logit-space parameterization and apply Natural Gradient Descent (NGD) scaling to our gradients (details in appendix D).

3.5 Experiments - Fixed Points of a Semantic Head Model

We test our algorithm on a single semantic head. The hypothesis is that the algorithm will locate points with low loss, and some of them will be stable (and reachable by autoregressive generation), whereas others will not be. We use an open-source pretrained model, namely the 1-layer attention-only model from the TransformerLens (Nanda & Bloom, 2022) library (`attn-only-1l`), trained on web text and code to replicate the original 1-layer model of Elhage et al. (2021). This model has 8 attention heads and uses absolute positional embeddings with a context window of 1024 positions. Its vocabulary size is $|V| = 48{,}262$. In this section, we test our algorithm in the exact case – a single pure semantic head. We thus ablate the positional embeddings of the model and run only a single active attention head in the model by zero-ablating the rest of the heads. We perform separate experiments on heads 0, 3, 4 and 6 (head selection is explained in section 3.6).

We use two sets of initial states for our optimizations: putative stable points and generic states. The first is a set of 180 repetition loops (from 7 to 67 per head, depending on how many distinct loops each head produces), found by generating long sequences from the model until it converged to a repetition loop. The second is the result of generating short sequences (mean 231 tokens) with the semantic model, starting from 45 Wikipedia texts (Wikimedia Foundation, 2023), for each of the four heads (total 180). Figure 3a shows the loss for these initial conditions. As expected, the Wikipedia texts are far from being fixed points, and short generation sequences do not lead the dynamics to a fixed state either. In contrast, the repetition loops have a very low loss, suggesting they are indeed fixed points. For both sets we estimate the unigram distribution in the text as the initial condition for the optimizer. We omit optimizations that did not converge below a loss of 1×10^{-2}, and one repetition that had a tokenization error (5 of 360 runs). Full experiment details are in appendix E.

Figure 3a shows that the optimizer indeed reaches areas with low loss. For both sets, the median loss drops by two to three orders of magnitude. Note that random token combinations have losses similar to the Wikipedia starting points. To verify that the optimized states are approximately fixed on the timescale of a typical generation, we estimate their drift heuristically. For a state with loss ϵ, the induced distribution is $\pi P_\pi \approx \pi(1+\delta)$ with a small relative deviation δ, and since the JSD is quadratic in small deviations, $\epsilon \sim \delta^2$. By equation 7, π changes by $\pi\delta$ per unit of $\tau = \ln i$, so over n tokens generated from position i it moves by $\pi\delta \ln(1+n/i)$, i.e. by a JSD of order $\epsilon \ln^2(1+n/i)$. For $\epsilon \approx 3 \times 10^{-3}$, $i \in [200, 800]$ and a full context window ($n = 1024$), $\ln(1+n/i) \lesssim 2$, so the state moves by $\sim 10^{-2}$ nats, far below the ~ 0.5 nats separating neighbouring attractors (figure 4b). We therefore treat these points as fixed points of the model and characterize their behavior.

ODE Stability Analysis Returning to the two interpretations mentioned above, we utilize the deterministic ODE of equation 7, which is valid (and exact for semantic heads) in the long context limit. We determine the stability of points in this regime by estimating the spectrum of the flow's Jacobian at the fixed point (see appendix F for details). Figure 3b shows a distinct separation: almost all repetition loops are stable (171 of 178), while all but one of the points reached from the Wikipedia texts are unstable (the exception is marked by an arrow and explained below). Observing the number of unstable directions (figure 3c), we find that all but two points have fewer than 10 unstable directions, meaning the remaining $\approx 48{,}000$ directions are stable, suggesting a very structured, low-dimensional flow, despite the large state space.

An interesting question that arises from this is - are in fact *all* repetitions stable fixed points? We test this with a control group of unoptimized repetitions (figure 3a–b) and find the answer is no. We see that arbitrary repetitions are not necessarily stable, and in fact might not even be fixed points.

Finite-Time Stability. A real model generates stochastically in finite time, not infinite time. To further characterize the behavior in this regime, we run an empirical experiment using Monte Carlo

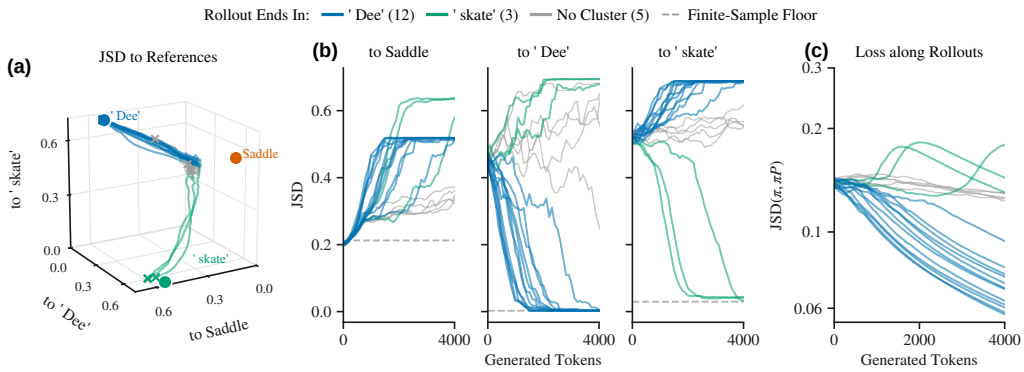

Figure 4: **Saddle point of a semantic head model.** 20 rollouts of head 3, each generating 4000 tokens after a 1500-token context sampled from the fixed point π^*, coloured by the attractor they end in. **(a)** Trajectories in the space of JSD distances to the saddle and to the two attractors (cluster centroids); crosses mark the final states. **(b)** The same distances against generated tokens. States in (a–b) are estimated from the last 750 tokens; dashed lines are the finite-sample floor of each reference distribution.**(c)** Loss $\mathrm{JSD}(\pi, \pi P_\pi)$ along the rollouts, with the state estimated from all tokens so far.

rollouts[1]. We sample 20 rollouts of 2000 new tokens initialized from each fixed point, and cluster the final states they reached (experiment details in appendix G). Figure 3d presents the fixed point's loss vs. its measured Branching Factor (BF, the number of distinct coherent clusters reached). As expected, the stable repetitions all have $\mathrm{BF} = 1$. Interestingly, we find that although unstable in the infinite time regime, many of the Wikipedia optimized states also have $\mathrm{BF} = 1$, meaning they are maintained in practice over a horizon of 2000 tokens ($2\times$ the context window of the model). Among them we find examples of "semantic clusters" – states with a wide unigram distribution over semantically related tokens (see figure 3f for an example) that is stochastic but approximately self-sustaining. Among states with $\mathrm{BF} > 1$, we find examples of saddle or "mixture" states – states with a distribution over unigrams from two or more semantic topics, which can collapse to either one of a few neighboring attractors. States with $\mathrm{BF} = 0$ are diffuse points whose rollouts do not form any cluster of at least three rollouts, typically because the finite-sample noise of their states exceeds the clustering threshold (appendix G). See figure 3g and figure 4 for a detailed example.

Overall, we are able to find both stable and unstable fixed points. We show that stable fixed points can be both repetitions and wider semantic concepts. On the other hand, not all repetitions are fixed points. We also find saddle points, which are borders between semantic concepts and can lead the system to converge to one or the other. Finally, we also observe one instance of what might be a *ghost attractor* (Sussillo & Barak, 2013; Koch et al., 2024), marked by an arrow in figure 3b. This point is linearly stable, but its loss does not vanish, so it is not an exact fixed point. It pulls the dynamics towards it for a prolonged time, and then releases them along its slowest mode towards a nearly deterministic next stage (appendix F).

3.6 Beyond Semantic Heads

Do these insights and methods apply to a general attention layer? In this section we show that by using several approximations we are able to find stable and unstable fixed points in this case as well.

We briefly describe how to extend our framework to a general attention layer, with extended details provided in appendix H. The semantic model of section 3.2 is exact only when positional embeddings are ablated. What happens when we look at the full `attn-only-1l` model, with all 8 heads active and positional embeddings? To measure how each head uses position, we profile the positional factor of its attention logits, $E_h(d)$, defined as the weight head h assigns to a key at a relative distance d, averaged over real query tokens (appendix H.2). We find that the heads split into two groups

[1]Note that since we are looking at the semantic model, we are not limited to the learned absolute positional embeddings of the model and can generate longer rollouts (appendix B).

(figure 5a). Four *fast* heads $(1, 2, 5, 7)$ decay within a few tokens, while four *slowly decaying* heads $(0, 3, 4, 6)$ keep an approximately exponential tail over tens to hundreds of tokens. These slowly decaying heads are the closest to the semantic limit, which is why we studied them in section 3.5.

We describe both types with a single approximation of each head's softmax, split into three blocks. The *local* block covers the last K tokens, i.e., the current K-gram $\mathcal{K}_i = (t_{i-K+1}, \ldots, t_i)$, and is computed exactly, with its real positional embeddings. The ***BOS*** block is the constant attention sink at position 0, which is also computed exactly, as it is a constant of the real model. The *far-field* block covers all remaining tokens and is the only approximated term. Here we fit each head's positional tail with an exponential decay, $E_h(d) \approx E_h(K)\,\gamma_h^{d-K}$ for $d \geq K$, which collapses the far field into a token-wise sum, as in equation 4:

$$\sum_{d \geq K} \alpha^h_{i,i-d}\, v^h_{t_{i-d}} \approx \frac{E_h(K)\, Z_h}{\mathcal{Z}^h_i} \sum_{t \in V} \pi^{(h)}_i(t)\, e^{s^h_{t_i,t}}\, v^h_t, \qquad \pi^{(h)}_{i+1} = \gamma_h\, \pi^{(h)}_i + (1-\gamma_h)\, t_{i+1-K}. \quad (10)$$

Here, $\pi^{(h)}_i$ is the discounted unigram distribution of tokens outside the local window, γ_h determines its effective timescale $L_h = (1-\gamma_h)^{-1}$, and $\mathcal{Z}^h_i$ is the attention normalizer. The exponential kernel gives the far field a fixed-size Markovian state. For fast heads its contribution is negligible, reducing the head approximately to a K-gram chain, whereas slowly decaying heads retain a discounted context distribution. Combining all heads yields the *Multi-Timescale Mean-Field* (MTMF) model,

$$p_i = \mathcal{F}\Big(\mathcal{K}_i, \pi^{(1)}_i, \ldots, \pi^{(H)}_i\Big),$$

a fast K-gram chain coupled to H context distributions with different timescales (App. H).

To test how well MTMF approximates the real model, we take a random position $i \in [700, 1000)$ in each of 1000 Wikipedia and 1000 C4 (Raffel et al., 2020) texts and compare the output next-token distribution of the real model with that of MTMF (details in appendix I). Figure 5b shows that the approximation with $K = 10$ has a low error (mean JSD 0.012, against 0.51 for a context-unigram baseline), that all components of the approximation are essential, and the compute-precision trade-off of increasing K.

Fixed Points of the MTMF Model. Because every $\pi^{(h)}$ is a discounted average of the same token stream, at a fixed point all of them coincide with the unigram marginal of the stationary K-gram distribution ν^*, whatever their timescales. The self-consistency condition therefore still holds, now as $\nu^* = \nu^* P_{\pi^*}$ with $\pi^* = \mathrm{marg}(\nu^*)$, and we can minimize the same JSD loss over a K-gram measure. This optimization is harder, because the state space grows as $|V|^K$. We thus track a sparse K-gram support that alternates exploration and natural-gradient steps. As a result, we only search low-entropy states such as repetitions. See appendix J for details and alternative optimization approaches.

In the full model, we find repetition fixed points of both kinds described in section 3.5. Apart from stable repetitions, we use our optimization to find an approximately fixed, yet *unstable* repetition loop. The deterministic MTMF flow leaves it without any perturbation, and real-model rollouts launched from it escape to the same neighbouring attractors that the flow predicts (figure 5c; appendix J). This repetition is thus a saddle point of the full model.

4 Discussion

Transformers are deployed autoregressively, suggesting the use of dynamical systems tools, yet the growing KV cache seems to rule out a fixed-size state. Here, we overcome this by splitting the state of a semantic attention head into two parts: a slowly varying distribution π coupled to a fast token-sampling process. We show this process has fixed points satisfying a self-consistency condition, from which we build an algorithm that finds both stable and unstable fixed points. We extend this perspective to general attention heads with the Multi-Timescale Mean-Field (MTMF) model, which combines a fast K-gram state with slow, head-specific context distributions. While timescale separation no longer holds, the fixed-point self-consistency condition remains valid.

Dynamics. Our results reveal a richer landscape than a simple distinction between normal generation and repetitive collapse. We find stable repetition attractors, but show that not all repetitions are fixed or stable. We also find semantic clusters that are asymptotically unstable, yet can persist over

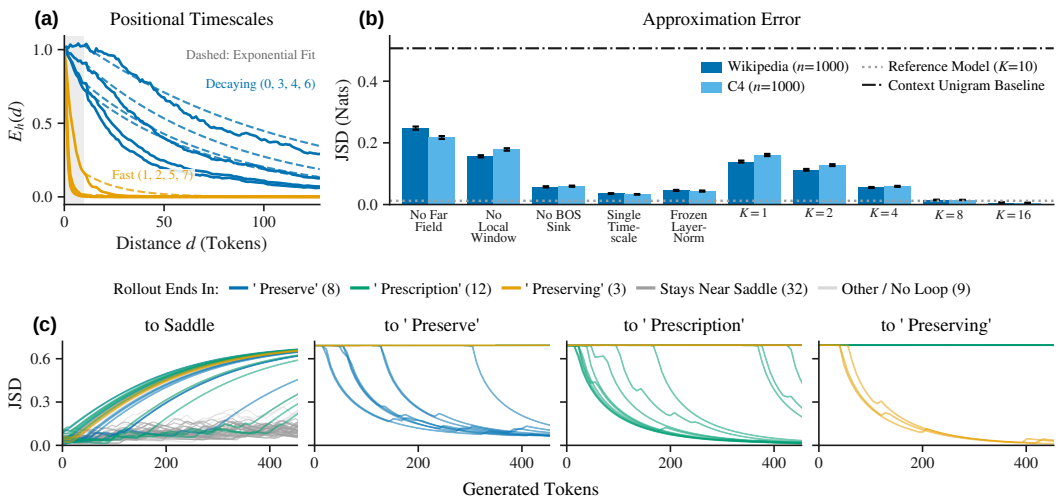

Figure 5: **The multi-timescale mean-field (MTMF) model of the full `attn-only-1l`. (a)** Positional attention factor $E_h(d)$ of every head (solid; geometric mean over 1024 query tokens) and its fitted exponential tail $E_h(K)\,\gamma_h^{d-K}$ (dashed). The grey band is the local window $d < K = 10$. **(b)** Mean JSD between the next-token distributions of the real model and of MTMF with $K = 10$, with one component ablated or the window K changed, on 1000 Wikipedia and 1000 C4 texts (error bars: SEM). Dotted: the non-ablated reference model ($K = 10$); dash-dotted: the context-unigram baseline. **(c)** K-gram JSD between 64 real-model rollouts launched from the unstable " Presley" repetition and four reference loops, coloured by the loop each rollout ends in.

finite generation horizons, highlighting the distinction between asymptotic stability and finite-time behavior. Finally, unstable mixture states act as saddles, from which trajectories diverge toward different neighboring attractors. Thus, unstable structures can organize generation even when the model never converges to them.

Interpretability. This perspective provides a macroscopic description of transformer behavior that complements circuit-level analysis. Repetition attractors characterize specific failure modes, while persistent semantic clusters may relate to fuzzy semantic copying and the model's ability to maintain a topic. Similarly, saddle-like mixture states may describe competition between alternative interpretations or continuations. More speculative is the candidate ghost attractor, which could relate to the concept of heteroclinic orbits (Rabinovich et al., 2008), that can enable LLMs to slowly move from one semantic context to another. These interpretations remain hypotheses, but yield testable predictions about how perturbations near the different states should affect subsequent generation.

Limitations. First, our analysis is restricted to one-layer, attention-only transformers. Extending it to deeper models lifts the state to a probability measure over the outputs of earlier layers, which is generally intractable. Second, the state representation is exact only for position-ablated heads. MTMF relies on an exponential approximation of the positional tail, which is only exact for ALiBi models. Third, our stability analysis holds in the long-context limit, whereas real generation is stochastic and finite. Our rollouts bridge this gap empirically, but a finite-time theory of persistence and escape is still missing. Fourth, the K-gram state space grows as $|V|^K$, restricting our current search in the full model to low-entropy states. Finally, we focus on fixed points, leaving structures such as limit cycles, basin boundaries, and the development of the dynamical landscape during training to future work.

In summary, our work provides a fixed-size state-space perspective on transformer generation and makes both stable and unstable fixed points accessible. This reveals how repetition attractors, persistent semantic clusters, and saddle states can organize autoregressive behavior.

AI USE STATEMENT

We used large language model (LLM) assistants to help implement and debug our methods and experiments, and to write plotting code for the figures. We also used them to assist with writing: editing and polishing the text and mathematical notation, and drafting parts of the appendices from our experiment records and code. For ideation, we used LLMs only at the start of the project, mainly for literature surveys and finding related work. All LLM-generated code was reviewed and tested by the authors, all numbers reported in the paper were produced by our own experiments and checked against the stored results, and all suggested references were verified by the authors. For example, when LLM-generated code produced a candidate fixed point, we verified it in a separate session using only code written by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why Exposure Bias Matters: An Imitation Learning Perspective of Error Accumulation in Language Generation, January 2023. URL http://arxiv.org/abs/2204.01171. arXiv:2204.01171 [cs].

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, October 2017. ISSN 0959-4388. doi: 10.1016/j.conb.2017.06.003. URL http://www.sciencedirect.com/science/article/pii/S0959438817300429.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Matthew Grossauer, Martin Feldhofer, Johannes Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Michel Benaïm. Dynamics of stochastic approximation algorithms. In Jacques Azéma, Michel Émery, Michel Ledoux, and Marc Yor (eds.), *Séminaire de Probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pp. 1–68. Springer, Berlin, Heidelberg, 1999. doi: 10.1007/BFb0096509. URL https://www.numdam.org/item/?id=SPS_1999__33__1_0.

Michel Benaïm and Morris W. Hirsch. Asymptotic pseudotrajectories and chain recurrent flows, with applications. *Journal of Dynamics and Differential Equations*, 8(1):141–176, 1996. doi: 10.1007/BF02218617. URL https://link.springer.com/article/10.1007/BF02218617.

Connor Brennan, Adeeti Aggarwal, Rui Pei, David Sussillo, and Alex Proekt. One dimensional approximations of neuronal dynamics reveal computational strategy. *PLOS Computational Biology*, 19(1):e1010784, January 2023. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010784. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010784.

Kabir Vinay Dabholkar and Omri Barak. Finding separatrices of dynamical flows with Deep Koopman Eigenfunctions. October 2025. URL https://openreview.net/forum?id=4KRERpdVDC.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, and Tom Conerly. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Nicholas Geneva and Nicholas Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022. doi: 10.1016/j.neunet.2021.11.022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *Journal of Machine Learning Research*, 25(123):1–67, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Tatsuya Hiraoka and Kentaro Inui. Repetition neurons: How do language models produce repetitions? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 483–495, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.naacl-short.41/.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration, February 2020. URL http://arxiv.org/abs/1904.09751. arXiv:1904.09751 [cs].

Muhammed Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. From Self-Attention to Markov Models: Unveiling the Dynamics of Generative Transformers. June 2024. URL https://openreview.net/forum?id=72oT4mPLUb.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165. PMLR, November 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Garrett E. Katz and James A. Reggia. Using Directional Fibers to Locate Fixed Points of Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. URL http://ieeexplore.ieee.org/abstract/document/8016349/.

D. Koch, A. Nandan, G. Ramesan, I. Tyukin, A. Gorban, and A. Koseska. Ghost Channels and Ghost Cycles Guiding Long Transients in Dynamical Systems. *Physical Review Letters*, 133(4):047202, July 2024. doi: 10.1103/PhysRevLett.133.047202. URL https://link.aps.org/doi/10.1103/PhysRevLett.133.047202.

Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, November 2013. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature12742. URL http://www.nature.com/articles/nature12742.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Neel Nanda and Joseph Bloom. Transformerlens, 2022. URL https://github.com/TransformerLensOrg/TransformerLens.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher Choquette-Choo, Florian Tramer, and Katherine Lee. Scalable Extraction of Training Data from Aligned, Production Language Models. *International Conference on Learning Representations*, 2025:82363–82435, May 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cce0e917b050208170151f77b497fc71-Abstract-Conference.html.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are Multi-State RNNs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1043. URL https://aclanthology.org/2024.emnlp-main.1043/.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Algorta, Ines Peviand, Joshua Babaei, Hani Balasubramanian, Lian Zhou, Jian Gao, Rui-Yang Qi, et al. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, 2023. URL https://aclanthology.org.

Mikhail I. Rabinovich, Ramón Huerta, Pablo Varona, and Valentin S. Afraimovich. Transient Cognitive Dynamics, Metastability, and Decision Making. *PLOS Computational Biology*, 4(5):e1000072, May 2008. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000072. URL http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000072.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models, October 2025. URL http://arxiv.org/abs/2407.02646. arXiv:2407.02646 [cs.AI].

Steven H. Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

David Sussillo and Omri Barak. Opening the Black Box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. URL http://www.mitpressjournals.org/doi/full/10.1162/NECO_a_00409.

Elia Turner, Kabir V Dabholkar, and Omri Barak. Charting and Navigating the Space of Solutions for Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25320–25333. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d530d454337fb09964237fecb4bea6ce-Abstract.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation Through Neural Population Dynamics. *Annual Review of Neuroscience*, 43(1):249–275, July 2020. ISSN 0147-006X, 1545-4126. doi: 10.1146/annurev-neuro-092619-094115. URL https://www.annualreviews.org/doi/10.1146/annurev-neuro-092619-094115.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wikimedia Foundation. Wikimedia downloads, 2023. URL https://dumps.wikimedia.org. English Wikipedia dump of 2023-11-01, accessed via the Hugging Face dataset wikimedia/wikipedia, config 20231101.en, https://huggingface.co/datasets/wikimedia/wikipedia.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. Learning to Break the Loop: Analyzing and Mitigating Repetitions for Neural Text Generation, October 2022. URL http://arxiv.org/abs/2206.02369. arXiv:2206.02369 [cs].

Junchi Yao, Shu Yang, Jianhua Xu, Lijie Hu, Mengdi Li, and Di Wang. Understanding the repeat curse in large language models from a feature perspective. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7787–7815, Vienna, Austria, 2025. Association for Computational Linguistics. URL https://aclanthology.org.

Oussama Zekri, Ambroise Odonnat, Abdelhakim Benechehab, Linus Bleistein, Nicolas Boullé, and Ievgen Redko. Large Language Models as Markov Chains, February 2025. URL http://arxiv.org/abs/2410.02724. arXiv:2410.02724 [stat].

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency, March 2025. URL http://arxiv.org/abs/2310.01405. arXiv:2310.01405 [cs.LG].

APPENDIX

The appendix is organized as follows:

- **Appendix A** – Extended related work: fixed-memory and recurrent language models, layer-wise dynamics, and transformers trained on dynamical systems.
- **Appendix B** – The model, the treatment of LayerNorm and biases, the semantic (position-ablated) setting, and generation beyond the context window.
- **Appendix C** – Derivations: the semantic head, the averaged ODE and its relation to the one-step residual, and the position-dependent escape time.
- **Appendix D** – Implementation details of the fixed-point finder: objective, natural gradient, hyperparameters and compute.
- **Appendix E** – Initial conditions, repetition mining, control groups and excluded runs of the semantic-head experiments.
- **Appendix F** – Linear stability analysis: the Jacobian, its computation, and a cross-check against the mean-field flow.
- **Appendix G** – Finite-time stability: rollouts, the branching factor, and the finite-sample floor.
- **Appendix H** – The full one-layer model: fast heads, decaying windows, the MTMF forward pass, timescale profiling, and the relation to ALiBi.
- **Appendix I** – The MTMF approximation benchmark, with all ablations.
- **Appendix J** – Fixed-point optimization in the full model and the " Presley" saddle.
- **Appendix K** – Alternative definitions of a state space and why they fall short.
- **Appendix L** – Beyond one-layer models: two-layer models and compositions.

A EXTENDED RELATED WORK

While our work focuses on the analysis aspect of state spaces, other works approach the growing-memory issue of transformers from an implementation efficiency perspective:

Fixed-Memory and Recurrent Language Models Several attempts have been made to design competitive language models with fixed memory using recurrent architectures. Most of them see this as an efficiency issue, while we highlight the importance of having a fixed-size state for analysis purposes. Three main lines of work present different fixed-memory architectures: First, Katharopoulos et al. (2020) show that models that use linear attention can be represented as state-space models, leading to a line of linear-attention based models (Peng et al., 2023; Wang et al., 2020). Second, the SSM (State-Space Models) and Mamba architectures (Gu et al., 2022; Gu & Dao, 2024) are explicitly designed as a state-space model with fixed-size memory. Third, there are several RNN and LSTM-based language models (Hochreiter & Schmidhuber, 1997; Beck et al., 2024). Oren et al. (2024) use a token ejection strategy to maintain a fixed state size. However, all these architectures consistently underperform softmax attention on long-horizon and recall tasks, and are not implemented in most frontier models. We are interested in finding a way to analyze softmax attention transformers with no architecture modifications.

Transformers *on* Dynamical Systems There is a different line of work where Transformers are *trained on dynamical system data* for modeling and prediction (Geneva & Zabaras, 2022), but that is fundamentally different from our goal of taking a pretrained *language*-model and defining it as a dynamical system. They should not be confused.

B MODEL AND IMPLEMENTATION DETAILS

Notation and conventions. Throughout the paper, we use i for the position index, V for the model vocabulary and $|V|$ for its size, $t \in V$ for a token (identified with its one-hot vector in $\mathbb{R}^{|V|}$ when it

enters a sum), and $p_i \in \mathbb{R}^{|V|}$ for the output probability distribution at position i. $q_t^i, k_t^i, v_t^i \in \mathbb{R}^{d_{\text{head}}}$ denote the query, key, and value projections of token t at position i. Distributions over V are denoted by π, and $P_\pi \in \mathbb{R}^{|V|\times|V|}$ is a transition matrix. We analyze attention-only models, using pure softmax sampling (no top-k or top-p filtering) with temperature 1. We drop stateless operations such as LayerNorm, as well as bias terms, for notation clarity.

The model. All experiments use the one-layer, attention-only model `attn-only-1l` of the TransformerLens library (Nanda & Bloom, 2022) (checkpoint `NeelNanda/Attn_Only_1L512W_C4_Code`), a replication of the one-layer model of Elhage et al. (2021) trained on C4 and code. It has $d_{\text{model}} = 512$, $H = 8$ heads with $d_{\text{head}} = 64$, a vocabulary of $|V| = 48{,}262$ tokens, and learned absolute positional embeddings over $n_{\text{ctx}} = 1024$ positions. We load it with the TransformerLens defaults (LayerNorm folding, centred writing weights and centred unembedding), which are function-preserving reparameterizations. All forward passes run in float32, with temperature 1 and no top-k/top-p filtering.

LayerNorm and biases. The notation of the main text omits LayerNorm and bias terms; the implementation keeps all of them, and none of them breaks the state description. In the semantic setting the input LayerNorm acts on the token embedding $e(t)$ alone, so it is a fixed per-token map that can be absorbed into q_t, k_t and v_t. The final LayerNorm acts position-wise on the residual stream $e(t_i) + W_O\, o_i + b_O$, which by equation 4 is a function of (π_i, t_i) only, so $p_i = \mathcal{F}(\pi_i, t_i)$ still holds exactly. The key bias adds the same constant $q_{t_i}^\top b_K$ to every attention score and cancels in the softmax; the value bias passes through the attention unchanged because the weights sum to one; b_O and b_U are constants.

Semantic (position-ablated) heads. To obtain an exact semantic head we zero the positional embeddings (at `hook_pos_embed`) and zero-ablate the outputs of all other heads (at `blocks.0.attn.hook_z`), keeping b_O. The operator $\pi \mapsto \pi P_\pi$ used by our optimizer reproduces this hooked model to within float32 round-off: $\max|\Delta P| \le 2.5 \times 10^{-6}$ for a single head and for groups of up to all eight heads.

Generating beyond the context window. With the positional embeddings zeroed, the model is exactly position-free, and n_{ctx} only limits the size of the positional-embedding table and of the prebuilt causal mask. For the long rollouts of appendix G we therefore replace the positional-embedding table by a zero table with 6000 rows and extend the causal mask accordingly. This changes no output at any position the original table covers, and lets us generate $1500 + 4000$ tokens. Note that the model is run out of its training distribution in this regime, both because it never saw such long sequences and because the position-ablated model is itself off-distribution; the procedure probes the position-free dynamical system, not the capabilities of the original model.

C Derivations

C.1 Semantic Head Derivation

Let us now show this formally. At position i, we have the current token t_i that generates the query vector $q_{t_i} \in \mathbb{R}^{d_{\text{head}}}$. For any other token $t \in V$, its unnormalized attention score ("attention logit") will be $s_{t_i,t} \triangleq q_{t_i}^\top k_t / \sqrt{d_{\text{head}}}$. The normalized attention weight it will receive will be the output of the causal softmax on these logits:

$$\alpha_{t_i,t} = \frac{\exp(s_{t_i,t})}{\sum_{j\le i}\exp(s_{t_i,t_j})}.$$

We now use our assumptions to replace the sum in the denominator to be a sum over all the tokens in the vocabulary. Denoting by $N_i(t)$ the number of times token t appears in the context up to position i, we can write the attention weight as:

$$\alpha_{t_i,t} = \frac{\exp(s_{t_i,t})}{\sum_{j\le i}\exp(s_{t_i,t_j})} = \frac{\exp(s_{t_i,t})}{\sum_{t'\in V} N_i(t')\exp(s_{t_i,t'})}.$$

The output of the attention head at position i is given by the weighted sum of value vectors:

$$o_i = \sum_{j \le i} \alpha_{t_i,t_j}\, v_{t_j} \in \mathbb{R}^{d_{\text{head}}}.$$

Applying the same trick again to replace the summation to be over tokens and plugging in the expression for the attention score, we get equation 4:

$$\begin{aligned} o_i &= \sum_{j \le i} \alpha_{t_i,t_j}\, v_{t_j} \\ &= \sum_{t \in V} N_i(t)\, \alpha_{t_i,t}\, v_t \\ &= \sum_{t \in V} \frac{N_i(t)\exp(s_{t_i,t})}{\sum_{t' \in V} N_i(t')\exp(s_{t_i,t'})}\, v_t \\ &= \sum_{t \in V} \frac{\pi_i(t)\exp(s_{t_i,t})}{\sum_{t' \in V} \pi_i(t')\exp(s_{t_i,t'})}\, v_t, \end{aligned}$$

where we defined $\pi_i(t) \triangleq N_i(t)/i$ as the proportion of token t in the context up to position i.

C.2 The Averaged ODE

We derive equation 7 with standard stochastic-approximation arguments. Write the update equation 6 as

$$\pi_{i+1} = \pi_i + \gamma_{i+1}\big[h(\pi_i) + \xi_{i+1}\big], \qquad \gamma_{i+1} = \frac{1}{i+1}, \quad h(\pi) = \bar{\pi}(\pi) - \pi, \quad \xi_{i+1} = t_{i+1} - \bar{\pi}(\pi_i), \tag{11}$$

where $\bar{\pi}(\pi)$ is the stationary distribution of P_π. The pair (t_i, π_i) is a Markov chain: given the state, t_{i+1} is drawn from row t_i of P_{π_i}. The noise ξ_{i+1} is therefore not a martingale difference; it is *Markovian* noise, whose conditional mean vanishes only on average over the fast chain. Three properties of the semantic head make the standard theory for this setting apply:

1. **Step sizes.** $\sum_i \gamma_i = \infty$ and $\sum_i \gamma_i^2 < \infty$.
2. **Uniform ergodicity.** The attention scores and the logits are bounded functions of (π, t) on the simplex, so every entry of P_π is bounded below by a constant $c > 0$ that does not depend on π. The fast chain is therefore uniformly geometrically ergodic (Doeblin's condition), and for every π the Poisson equation $u_\pi - P_\pi u_\pi = \mathrm{id} - \mathbf{1}\bar{\pi}(\pi)$ has a bounded solution u_π.
3. **Regularity.** $\pi \mapsto P_\pi$ is smooth on the simplex (the attention normalizer $\sum_t \pi(t) e^{s_{t_i,t}}$ is bounded away from zero), so $\bar{\pi}(\pi)$ and u_π are Lipschitz in π.

Using the Poisson solution, ξ_{i+1} splits into a martingale difference, a telescoping term, and a remainder of order γ_i. The interpolated process $\pi(\tau)$, with $\tau_i = \sum_{k \le i} \gamma_k = \ln i + O(1)$, is then an asymptotic pseudotrajectory of the ODE

$$\frac{d\pi}{d\tau} = h(\pi) = \bar{\pi}(\pi) - \pi,$$

which is equation 7 (Benaïm & Hirsch, 1996; Benaïm, 1999). Two standard consequences are relevant for us. First, the limit sets of π_i are internally chain-transitive sets of the flow, and linearly stable fixed points attract the process with positive probability. Second, when the noise is non-degenerate along an unstable direction, the process converges to a linearly unstable fixed point with probability zero. Unstable fixed points are thus invisible to long autoregressive rollouts, which is why we locate them by optimization. The process is closely related to vertex-reinforced random walks and other self-interacting Markov chains.

The one-step residual as a proxy for the drift. Our loss uses the one-step product πP_π rather than the stationary distribution $\bar{\pi}(\pi)$, which is expensive to compute. The two share their zeros, because

Algorithm 1 Fixed-Point Finder (Semantic Head Version)

Require: Model $\mathcal{M}$, initial logits ℓ, temperature β, max iterations $n_{\max}$, step size η

for $n = 1, \ldots, n_{\max}$ **do**
  $\pi \leftarrow \mathrm{softmax}(\ell)$ ▷ logits to probability
  $\pi P_\pi \leftarrow 0$ ▷ initialize accumulator
  for token chunk $T \subset V$ **do**
    $P_\pi[T, :] \leftarrow \mathcal{F}_{\mathcal{M},\beta}(\pi, T)$ ▷ batched forward pass
    $\pi P_\pi \mathrel{+}= \pi[T]\, P_\pi[T, :]$
  end for
  $\mathcal{L} \leftarrow \mathrm{JSD}(\pi, \pi P_\pi)$
  $\tilde{\nabla}_\ell \mathcal{L} \leftarrow \mathrm{NGD}(\nabla_\ell \mathcal{L})$ ▷ apply NGD scaling to gradients
  $\ell \leftarrow \ell - \eta\, \tilde{\nabla}_\ell \mathcal{L}$ ▷ update the logits
end for

P_π has a unique stationary distribution: $\bar{\pi}(\pi) = \pi \iff \pi P_\pi = \pi$. Away from a fixed point, let $r(\pi) = \pi P_\pi - \pi$. Since $\bar{\pi}(I - P_\pi) = 0$ and $(\bar{\pi} - \pi)\mathbf{1} = 0$,

$$\bar{\pi}(\pi) - \pi = r(\pi)\, Z_\pi, \qquad Z_\pi = \big(I - P_\pi + \mathbf{1}\bar{\pi}(\pi)\big)^{-1}, \tag{12}$$

where Z_π is the fundamental matrix of the chain. The residual thus equals the drift up to the factor Z_π, whose non-trivial eigenvalues are $1/(1 - \lambda_k(P_\pi))$. For the fixed points we studied, the second eigenvalue of P_{π^*} is real and negative (e.g. $\lambda_2 = -0.84$ for a head-3 Wikipedia fixed point), the chain mixes within a few tokens, and $\mathrm{JSD}(\pi, \pi P_\pi)$ agrees with $\mathrm{JSD}(\pi, \mu)$, where μ is the stationary distribution of the frozen chain obtained by damped power iteration, to within 8%.

D Fixed-Point Optimization Details

This section details Algorithm 1 as run on single semantic heads. The version for the full model is described in appendix J.

Objective. We minimize $\mathrm{JSD}(\pi, \pi P_\pi)$ (in nats), multiplied by the constant $10^5/|V|$ to keep gradients in a convenient range. All losses we report are unscaled. The product $\pi P_\pi = \sum_u \pi(u) P_\pi[u, :]$ is accumulated over query tokens u without ever forming P_π. During optimization, the sum runs over the 4096 most probable tokens of the current π, processed in micro-batches of 1024 with gradient checkpointing, so that memory is set by the micro-batch rather than by the number of queries. Because a converged π has a heavy tail ($\approx 98\%$ of its mass in its top 4096 tokens, and $\approx 16{,}000$ tokens needed for 99.9%), this truncated loss is biased downwards. We therefore re-evaluate the loss of the final iterate with πP_π summed over *all* $|V|$ query tokens, and report only this full-vocabulary loss. Every π is stored densely: truncating the tail changes the attention normalizer, and thereby P_π itself.

Parameterization and natural gradient. We set $\pi = \mathrm{softmax}(\ell)$, initialized at $\ell = \log \max(\pi_{\text{init}}, 10^{-10})$. The Fisher information of this parameterization is $F = \mathrm{diag}(\pi) - \pi\pi^\top$, and we precondition the gradient with its regularized inverse, computed in $O(|V|)$ by the Sherman–Morrison formula:

$$\tilde{\nabla}_\ell \mathcal{L} = \big(A - \pi\pi^\top\big)^{-1} \nabla_\ell \mathcal{L} = A^{-1} g + \frac{A^{-1}\pi\, (\pi^\top A^{-1} g)}{1 - \pi^\top A^{-1} \pi}, \qquad A = \mathrm{diag}(\pi) + \lambda I, \quad g = \nabla_\ell \mathcal{L}, \tag{13}$$

with $\lambda = 10^{-10}$. The natural gradient is clipped to norm 1, and the logits are updated by plain gradient steps (no momentum) with a cosine learning-rate schedule from 1 to 0.1 over 3000 iterations. The loss plateaus after about 1200 iterations. We keep the best iterate.

Why a single operator application suffices. $\pi = \pi P_\pi$ states exactly that π is stationary for the chain it induces, so the one-step residual is not an approximation of stationarity. Iterating the operator without damping in the loss would be harmful: $\pi P_\pi^n = \pi$ also admits spurious period-n solutions, and indeed the slowest mode of the induced chains we examined is a nearly period-2 oscillation ($\lambda_2 = -0.84$). A damped multi-step residual has the same zero set as the single step, and in our measurements its value agreed with the single-step residual to within 8% (appendix C.2).

Compute. One iteration takes about 0.25 s for a single head on one NVIDIA Quadro RTX 5000 (16 GB), with a peak memory of 1.6 GB. The 360 single-head optimizations of section 3.5 were distributed over eight such GPUs.

E Semantic Head Experiments: Data and Controls

Wikipedia initial conditions. We take the first 100 articles longer than 5000 characters from the English Wikipedia dump (Wikimedia Foundation, 2023) (`wikimedia/wikipedia`, `20231101.en`). The position-ablated model, with a single semantic head active (head 3), continues each article from a 500-token prompt, generating 150 to 500 new tokens at temperature 1. Only the generated continuation is kept, so that the initial conditions are states the semantic dynamics actually visit rather than natural text. For 45 of these continuations, and for each head $h \in \{0, 3, 4, 6\}$, we draw one cut position uniformly from $[150, n_{\text{tokens}})$ (the same position for all heads; mean 231, range 150–448 tokens) and initialize the optimizer at the empirical unigram distribution of the continuation up to that position. This gives $4 \times 45 = 180$ optimizations.

Repetition initial conditions. We mine the repetition loops each head collapses into. Under the semantic hooks, with only head h active, we generate 1400 tokens from 64-token prompts taken from the generated corpus (414 prompts, two seeds), using the long-context procedure of appendix B. We then detect the terminal cycle of each generation: the smallest period $p \leq 48$ whose most frequent p-gram covers at least 60% of the last 400 tokens. Loops are identified up to rotation and ranked by how often the head fell into them. A first set of 20 loops (quotas $8/4/4/4$ for heads $3/0/4/6$, including single-token, two-token and longer quasi-deterministic loops) was extended with further distinct loops, aiming at 45 per head. Loops that do not survive detokenization and retokenization were discarded. Head 6 produced only 7 distinct loops in total and head 0 produced 39, so heads 3 and 4 contribute 67 each: 180 loops overall. Each loop is tiled into a text of 1500 tokens (truncated to the context window) and optimized exactly like the Wikipedia continuations, from the empirical unigram distribution at a random cut position; the "Repetition Start" category of figure 3a is this initial condition.

Controls. All control losses are evaluated with the same full-vocabulary loss, under the head they are assigned to. *Random Tokens*: the empirical unigram distribution of 2500 tokens drawn uniformly from the vocabulary; 45 samples, each evaluated under each of the four heads (180 points). *Unoptimized Repetitions*: for each head, 25 single-token states $\pi = \delta_a$ and 25 two-token states $\pi = (\delta_a + \delta_b)/2$, with a, b drawn uniformly from the 8000 most frequent (non-special) tokens of the Wikipedia corpus (200 points).

Excluded runs. We exclude optimizations whose final loss stays above 1×10^{-2}, and one repetition whose loop text consists of newlines and therefore tokenized to 189 instead of 1024 tokens. Three Wikipedia runs (head 3) did not converge (final losses 0.036–0.43), and one additional repetition ended at the threshold of 1×10^{-2}. This leaves 177 Wikipedia and 178 repetition fixed points.

F Linear Stability Analysis

The flow we linearize. We analyze the stability of the residual flow $d\pi/ds = \pi P_\pi - \pi$, whose Euler discretization $\pi \leftarrow (1-a)\pi + a\,\pi P_\pi$ is the deterministic mean-field iteration we use to cross-check the spectra. It has the same fixed points as equation 7. Its Jacobian at π^* is $J - I$, where

$$J = \left.\frac{\partial(\pi P_\pi)}{\partial \pi}\right|_{\pi=\pi^*} \tag{14}$$

is restricted to the tangent space $\{v : \sum_t v_t = 0\}$ of the simplex. A fixed point is linearly unstable if and only if $\max_k \operatorname{Re}\mu_k > 1$, where μ_k are the eigenvalues of J, and its number of unstable directions is the number of μ_k with $\operatorname{Re}\mu_k > 1$. By equation 12, the linearization of the averaged ODE equation 7 at the same point is $v \mapsto v(J - I)Z_{\pi^*}$. The two coincide when the chain mixes fast ($Z_{\pi^*} \approx I$), and all the verdicts below were cross-checked against rollouts of the real model.

Computation. J is a $|V| \times |V|$ matrix and is never formed. We compute Jacobian–vector products by forward-mode automatic differentiation through the full-vocabulary πP_π (all 48,262 query tokens), centre both v and Jv to restrict to the tangent space, and find the 10 eigenvalues with the largest real part with ARPACK (float64 arithmetic on float32 products, tolerance 10^{-6}). Since πP_π conserves mass, the centring is exact: it removes only the trivial mode $J\pi^* = \pi^*$. About 40 products suffice, 4–15 minutes per fixed point. The count of unstable directions in figure 3c saturates at 10, the number of eigenvalues computed. A top-k version of the Jacobian, with queries restricted to the 1024 most probable tokens of π^*, is not reliable: it leaks mass, is blind to directions that move mass through the excluded tokens, and wrongly classifies 40 of the 180 Wikipedia fixed points as stable.

Cross-check with the mean-field flow. Iterating the mean-field map with $a = 0.05$ from small perturbations of each Wikipedia fixed point agrees with the spectral verdict for 178 of 180 points. The two exceptions are instructive. *(i)* Head 0, index 6 (the arrow in figure 3b) is linearly stable ($\max \mathrm{Re}\, \mu = 0.914$, confirmed with 40 eigenvalues at tolerance 10^{-10}), yet the flow leaves it after about 211 steps (median escape for unstable points: 84). Its loss is the worst of its group, the linearization does not hold at π^* (the correction implied along the slowest mode is $0.69\|\pi^*\|$), and nearby launches first contract onto π^* and then drift out along the slowest mode, from " rem/une/album" towards " book/writer/writing". This is the behaviour of a ghost of a saddle-node bifurcation, which we call a candidate ghost attractor in the main text. *(ii)* Head 4, index 38 has one weakly unstable mode ($\mu = 1.185$) that a 250-step flow did not resolve; run to 600 steps, it escapes as the spectrum predicts.

G Finite-Time Stability: Rollouts and Branching Factor

Rollouts. For each fixed point π^* we sample 20 contexts of 1500 tokens i.i.d. from π^*, and let the position-ablated single-head model generate 2000 further tokens from each at temperature 1 (appendix B). The same configuration is used for all Wikipedia and repetition fixed points.

Branching factor. The final state of a rollout is the empirical unigram distribution of its last 1500 generated tokens, which excludes the transient. We compute the pairwise JSD between the final states of all rollouts, add π^* itself as one extra node, and cluster using DBSCAN with $\varepsilon = 0.5$ and a minimum cluster size of 3. Distances are measured in the optimizer's units ($10^5 \cdot \mathrm{JSD}\,/|V|$), so the threshold corresponds to a JSD of 0.24 nats. The branching factor (BF) is the number of clusters formed by rollouts. π^* can join a cluster but never counts as a branch, and rollouts that belong to no cluster are counted as noise.

The finite-sample floor. An empirical distribution of W tokens sampled from π is at a non-zero JSD from π itself, which grows with the effective support of π. In the trajectory plots of figure 4, the dashed lines show this floor for $W = 750$, the sliding window from which the states in panels (a–b) are estimated. For diffuse fixed points the floor can exceed the clustering threshold (e.g. $0.59 > 0.5$ for the head-0 point above). The rollout test then cannot detect that a trajectory stayed, so BF and asymptotic stability need not agree for such points.

Trajectory geometry (figure 4). For the head-3 saddle (Wikipedia index 34), we track each rollout's state along its generation and measure its JSD to π^* and to the centroids of the two attractor clusters. The loss in panel (c) is evaluated on the cumulative state, estimated from all tokens so far, with πP_π over the full vocabulary.

H Full 1-Layer Model - Extended Details

H.1 Fast Heads - Detailed Derivation

Fast Heads To start with a motivating example, we look at the type of head which is the exact opposite of a pure semantic head – a pure "positional" head, i.e. a head that always attends a certain relative position, regardless of the identity of the current and previous tokens. A famous example of such a head is the so-called "previous-token head" found in the first layer of a simple induction

circuit [cite Anthropic]. This head always attends the previous token[2] and copies it into the residual stream. It is clear that this head does not have a slow component – it operates always at the same timescale of the token sampling process. However, it has a very simple state, which is simply the last two tokens (t_i, t_{i-1}). The output of this attention head can be written as:

$$o_i = v_{t_{i-1}} \triangleq A_{\text{fast}}(t_i, t_{i-1}). \tag{15}$$

This can be interpreted as a *simple Markov Chain* over an *expanded state* with the state being the current *bigram* instead of only the current token (unigram).

Extending this idea, we can see that any head whose attention pattern only depends on the last K tokens, can be represented as simple Markov Chain operating over K-grams. We denote the current K-gram of tokens as $\mathcal{K}_i = (t_{i-K+1}, \ldots, t_i)$. Note that this makes no further assumptions on the specific attention pattern inside this window, aside from the attention being bounded to K tokens. We also call K the *cutoff* distance of the head.

Proposition (fast heads are Markov over K-grams). Let every head of a one-layer model attend only to the last K positions. Then $p_i = \mathcal{F}(\mathcal{K}_i)$ is a deterministic function of the current K-gram, and autoregressive generation is a (time-homogeneous) Markov chain on V^K with transitions

$$\Pr\left[\mathcal{K}_{i+1} = (t_{i-K+2}, \ldots, t_i, t') \,\middle|\, \mathcal{K}_i = (t_{i-K+1}, \ldots, t_i)\right] = \mathcal{F}(\mathcal{K}_i)_{t'}, \tag{16}$$

and probability zero for all other successors.

Lemma Note that any Markov chain operating on the last K states (K-Markovian process) can also be trivially defined as a Markov chain over the last K' states for any $K' \geq K$, because we can always ignore the extra unnecessary states. Similarly, any fast head operating over the last K tokens can also be considered as operating over the last $K' \geq K$ tokens.

This is the standard reduction of a K-th order Markov chain to a first-order chain on K-tuples [cite Markov chain textbook].

Combining Slow and Fast Heads What happens when our attention layer contains both slow and fast heads? Let us consider the output of an attention layer containing a single semantic head, and a single fast head operating over the last K tokens:

$$o_i = A_{\text{sem}}(\pi_i, t_i) + A_{\text{fast}}(\mathcal{K}_i) \triangleq A(\pi_i, \mathcal{K}_i), \tag{17}$$

where we used the fact that the current token t_i is contained within the current K-gram, $t_i \in \mathcal{K}_i$. We can see that we can represent this attention layer as a combination of the two models we discussed so far: a *self-interacting* Markov Chain over K-grams. Note that the slow variable is a distribution over unigrams, while the fast chain is over K-grams.

The same will be true for a layer containing N_{sem} semantic heads $A^{(n)}_{\text{sem}}$ and N_{fast} fast heads $A^{(n)}_{\text{fast}}$ with cutoffs $K_1, \ldots, K_{N_{\text{fast}}}$, by selecting $K = \max\{K_1, K_2, \ldots, K_{N_{\text{fast}}}\}$ as the global cutoff distance:

$$o_i = \sum_{n=1}^{N_{\text{sem}}} A^{(n)}_{\text{sem}}(\pi_i, \mathcal{K}_i) + \sum_{n=1}^{N_{\text{fast}}} A^{(n)}_{\text{fast}}(\mathcal{K}_i) \triangleq A_{\text{MHA}}(\pi_i, \mathcal{K}_i). \tag{18}$$

H.2 Decaying Window Derivations

The final addition we need to make is to take into account the fact that most heads in a model are not purely semantic nor purely positional. Instead, they have some soft dependence on position, mainly through relative distances. They usually have an effective *finite* lookback window. To test this empirically, we profile the positional factor of each head's attention logits, i.e. the multiplicative weight that the positional part of a key at relative distance d contributes, averaged over real query tokens (appendix H.4)[3]. The resulting profiles are shown in figure 5a.

[2] In practice such heads also almost always attend special begin-of-sequence (BOS) tokens. The details for taking this into account in our model are explained in appendix H.3.

[3] This approach has some limitations and might miss "outliers" that could be important for some behaviors.

To develop the formalization for a head with a *finite* lookback window, we begin with the simplest case: a head with a uniform sliding context window over the last W tokens. The explicit update equation for this head is $\pi_{i+1} = \pi_i + \frac{1}{W}(t_{i+1} - t_{i-W+1})$. But this update is not Markovian with respect to π_i and requires an explicit memory of all W last tokens. We can instead approximate this using an Exponential Moving Average (EMA) approximation: This is equivalent to assuming the window is well-mixed, and the dropped token at each step is approximated by the average of the current distribution, which is simply π_i. The update equation then becomes:

$$\pi_{i+1} = \pi_i + \frac{1}{W}(t_{i+1} - \pi_i) = \Big(1 - \frac{1}{W}\Big)\pi_i + \frac{1}{W}t_{i+1} \triangleq \gamma\,\pi_i + (1-\gamma)\,t_{i+1}, \tag{19}$$

where we define the decay factor $\gamma \triangleq 1 - 1/W$, and $1 - \gamma = 1/W$ is the "update rate"[4], which is inversely proportional to the length of the window. Importantly, this equation is again Markovian with respect to π_i (exponential decay is a memoryless kernel). Note that π_i is now the "effective", position-weighted distribution the head sees. Following RL terminology, we can also call it the "discounted distribution".

For an arbitrary positional dependence, the head can be described by a Generalized Langevin Equation (GLE) with a memory kernel κ_i. This is in general not Markovian with respect to π_i. However, we can approximate the head with an exponential decay, by fitting it a decay factor γ. In practice, we approximate all heads as EMA heads, by estimating the decay factor γ_h for each head h individually based on its empirical profiling.

H.3 The MTMF Forward Pass

For a query token t_i, MTMF resolves the context of head h in three blocks, which together partition the context exactly once:

$$\mathcal{Z}_i^h = \underbrace{\sum_{d=0}^{K-1} e^{\,a^h_{i,i-d}}}_{\text{local window (exact)}} + \underbrace{E_h(K)\,Z_h \sum_{t\in V} \pi_i^{(h)}(t)\, e^{\,s^h_{t_i,t}}}_{\text{far field (mean field)}} + \underbrace{e^{\,a^h_{i,0}}}_{\text{BOS sink (exact)}}, \tag{20}$$

where $a^h_{i,j}$ is the full attention logit of head h between query position i and key position j, and the head output is the same three sums taken against the values, divided by $\mathcal{Z}_i^h$.

- **Local window.** The last K tokens are computed without approximation, with their real positional embeddings, the real LayerNorm and all four query–key cross terms between token and position components. The positional attention profile inside the window is therefore produced by the model, not imposed.
- **Far field.** With a linearized LayerNorm, the key of token t at position j splits into a semantic and a positional part, $k(t,j) = k_s(t) + k_p(j)$, so that $e^{q^\top k(t,j)} = e^{q^\top k_s(t)}\, e^{q^\top k_p(j)}$ exactly. The approximation is to treat the two factors as independent across the far field, which turns the sum over far positions into (sum of positional weights) $\times$ ($\pi^{(h)}$-average of semantic weights). With the exponential tail $E_h(d) \approx E_h(K)\gamma_h^{d-K}$, the positional weights sum to $E_h(K)Z_h$ with $Z_h = (1-\gamma_h)^{-1}$. The tail is anchored to the real positional factor $E_h(K)$ at the edge of the window, so the far field starts exactly where the exact block ends. The far-field values additionally include the discount-weighted mean positional embedding. LayerNorm is applied with the joint scale $\sigma(t, j_h)$ of each token at the discount-weighted mean far position j_h, not with a single frozen scalar.
- **BOS sink.** The attention sink at position 0 uses the true key and value of the BOS token, scored with the full query, because attention to the sink depends strongly on position. BOS and padding tokens are excluded from $\pi^{(h)}$, so that the sink is counted exactly once.

The query is placed at a fixed dummy position $t^* = 800$, and the discounted distributions $\pi_i^{(h)}$ of a real prompt are computed by the estimator of equation 10, with the nearest far token weighted by

[4]This is similar to the learning rate or step size in gradient-descent based optimization, which are also stochastic approximation processes.

$\gamma_h^0 = 1$. As an acceptance test, driving MTMF into the semantic regime (zero positional embeddings, one head, $\gamma = 1$, $K = 1$, realized far-field mass) reproduces the real model to JSD $= 1.3 \times 10^{-12}$ nats, independently of $K \in \{1, 2, 8\}$. Each block can be removed on its own, which is how the ablations of figure 5b are defined (appendix I).

H.4 Timescale Profiling

The positional factor of head h at relative distance d is

$$E_h(d) = \exp\Big(q_h(t)^\top k_{p,h}(t^* - d)/\sqrt{d_{\text{head}}}\Big), \tag{21}$$

the multiplicative weight that the positional part of the key at distance d contributes to the attention of query token t at position $t^* = 800$. We evaluate it for $d = 0, \ldots, 512$ for 1024 query tokens sampled from a Wikipedia article (special tokens excluded) and average over queries geometrically, since the profile is a product of exponentials. For each query token we fit $\log E_h(d) = a + d \log \gamma_h$ by least squares over the far field $d \in [K, 256]$, and take the median of γ_h over query tokens. The implied timescale is $L_h = 1/(1 - \gamma_h)$, clamped to $[K + 2, t^*]$ (no head reached the clamp). Table 1 lists the result. The fast heads decay within a few tokens and the tail carries little weight, while the slowly decaying heads keep an approximately exponential tail over tens to hundreds of tokens (figure 5a).

Table 1: Fitted positional timescales of `attn-only-1l` ($K = 10$, $t^* = 800$).

Head h	0	1	2	3	4	5	6	7
γ_h	0.9853	0.9629	0.9474	0.9867	0.9908	0.9620	0.9850	0.9617
L_h (tokens)	68.0	26.9	19.0	75.4	108.9	26.3	66.8	26.1
Group	slow	fast	fast	slow	slow	fast	slow	fast

H.5 ALiBi Equivalence

ALiBi removes explicit positional embeddings, and instead subtracts a head-specific linear penalty from the unnormalized attention score based on relative distance:

$$s_{\text{ALiBi}}(i, j) = q_i^\top k_j - m\,|i - j|, \tag{22}$$

where $m > 0$ is a constant, head-specific scalar, defining the "slope" or decay rate of the head. Because this is applied to the unnormalized scores (before the softmax), the softmax transforms this into an exact exponential decay multiplicative factor $e^{q_i^\top k_j} \cdot e^{-m|i-j|}$. We can define the decay factor $\gamma \triangleq e^{-m}$, and then the effective update rate $1 - \gamma$, to gain the exact same update equation for the discounted distribution as equation 19:

$$\pi_{i+1} = \gamma\,\pi_i + (1 - \gamma)\,t_{i+1}. \tag{23}$$

Thus the EMA formalization describes ALiBi attention heads *exactly* and not approximately. ALiBi models have shown competitive performance on many benchmarks. Our theoretical framework can thus be applied to analyzing these models and the varied capabilities they exhibit analytically.

Note that an important difference in our model is that the fast heads are *not* forced to use exponential decay over the current K-gram, and are thus slightly more expressive than pure ALiBi heads.

I MTMF Approximation Benchmark

Protocol. The ground truth is the untouched model: all eight heads active, positional embeddings on, no hooks. We use 1000 Wikipedia articles (the stream and length filter of appendix E) and 1000 C4 documents (Raffel et al., 2020); the query tokens used for timescale profiling come from the Wikipedia corpus in both cases, so the timescales are identical. Each text is truncated to 1000 tokens, and one query position is drawn uniformly from $[700, 1000)$. At that position, we compare the next-token distribution of the real model, given the full prefix, with that of MTMF, given the

K-token window, the per-head discounted distributions of the rest of the prefix, and the BOS sink. The random positions are seeded per run, so all rows of table 2 are evaluated at the same positions and their differences are paired. The *context-unigram baseline* predicts the next token with the raw unigram histogram of the prefix, without the model.

Rows. Each row changes exactly one ingredient of the reference model ($K = 10$, per-head L_h): *No Far Field* removes the mean-field block (window and sink only), which prices the approximation itself; *No Local Window* removes the exact window, including the query's attention to itself; *No BOS Sink* removes the sink; *Frozen LayerNorm* uses one scalar LayerNorm scale for the far field; *No QK Cross Terms* drops the token–position cross terms inside the window; *No Far Positional Value* drops the mean positional embedding from the far-field values; *True Query Position* uses the prompt's real position instead of t^*; *Single Timescale* gives every head the mean timescale $L = 52.2$; the K rows change the window width.

Table 2: Mean JSD (nats) between the next-token distributions of `attn-only-1l` and of MTMF, $\pm$ SEM over 1000 texts per corpus. The minimal version of this table is figure 5b.

Variant	Wikipedia	C4
Reference model ($K = 10$)	0.0125 ± 0.0005	0.0119 ± 0.0004
No Far Field	0.2479 ± 0.0051	0.2173 ± 0.0047
No Local Window	0.1564 ± 0.0035	0.1786 ± 0.0040
No BOS Sink	0.0574 ± 0.0019	0.0595 ± 0.0018
Frozen LayerNorm	0.0459 ± 0.0017	0.0436 ± 0.0018
No QK Cross Terms	0.0845 ± 0.0020	0.0921 ± 0.0022
No Far Positional Value	0.0812 ± 0.0020	0.0818 ± 0.0019
True Query Position	0.0128 ± 0.0005	0.0119 ± 0.0004
Single Timescale ($L = 52.2$)	0.0357 ± 0.0011	0.0331 ± 0.0010
$K = 1$	0.1394 ± 0.0032	0.1606 ± 0.0036
$K = 2$	0.1126 ± 0.0028	0.1279 ± 0.0030
$K = 3$	0.0761 ± 0.0021	0.0834 ± 0.0022
$K = 4$	0.0553 ± 0.0016	0.0590 ± 0.0016
$K = 5$	0.0390 ± 0.0012	0.0410 ± 0.0012
$K = 6$	0.0292 ± 0.0009	0.0303 ± 0.0010
$K = 7$	0.0221 ± 0.0007	0.0224 ± 0.0007
$K = 8$	0.0152 ± 0.0005	0.0149 ± 0.0005
$K = 12$	0.0076 ± 0.0003	0.0067 ± 0.0002
$K = 16$	0.0065 ± 0.0003	0.0054 ± 0.0002
Context-unigram baseline	0.5061 ± 0.0025	0.5070 ± 0.0027

Reading the table. Both context-bearing blocks are essential. Removing the far field costs $20\times$ the reference error, and removing the local window costs 13–$15\times$, so the model reads both its recent tokens and a discounted bag of older ones. Per-head timescales matter: a single shared timescale triples the error. The error decreases steadily with K. $K = 16$ is the largest window compatible with the fastest head ($L_2 = 19.0$ must exceed $K + 1$).

J MTMF Fixed-Point Optimization

State and loss. The MTMF state is a sparse measure ν over K-grams ($K = 10$). The K-grams select the active queries and carry the fast dynamics (the key shift), while their unigram marginal $\pi = \mathrm{marg}(\nu)$ is the far-field distribution of every head. At a fixed point all $\pi^{(h)}$ coincide (section 3.6), so we use a single π with the per-head timescales entering only through the far-field weights $E_h(K)Z_h$. The loss is the JSD between ν and νP_π on the K-gram space, with BOS and padding excluded from the support on both sides.

Optimization. Each run alternates exploration and gradient steps on the same operator. *Exploration*: a gradient-free power iteration $\nu \leftarrow \nu P_\pi$ proposes new K-grams; successors are ranked after aggregating the flux of all their parents. *Gradient step*: $\nu = \mathrm{softmax}(\ell)$ over the current support. νP_π is computed from the top 4096 K-grams of ν as queries and truncated to the $131{,}072$ largest successors, and the logits are updated with the natural gradient of equation 13. The support is capped at $12{,}065$ K-grams. The truncation of νP_π drops at most 6.4×10^{-4} of its mass at any reference point we report (5.6×10^{-5} at the saddle below). We run 3000 iterations with a cosine learning-rate schedule from 1 to 0.1, with $t^* = 800$, and initialize from the discounted K-gram distribution of a real prompt: the unigram distribution of the slowest head, lifted to K-grams with the same geometric discount. The runs are not fully converged at 3000 iterations; the reported losses are upper bounds at a matched budget.

Candidates. The sparse support restricts this search to low-entropy states, so we seed it with repetition loops that the real model visibly left. We sampled $115{,}200$ trajectories of the untouched model (temperature 1, up to n_{ctx}) from Wikipedia windows, repeated Wikipedia spans and repeated hand-written phrases. A loop detector found $1{,}807$ events in which a loop A was held and then abandoned for a different loop B. For nine selected events (seven candidates and two negative controls) we optimized a state S from the position inside loop A where the initial condition has the lowest loss, and a state for B from the end of the trajectory. We then tested S in two ways: with the deterministic MTMF flow $\pi \leftarrow \pi + \alpha(\pi P_\pi - \pi)$, $\alpha = 1/L_4$ (one step $\approx$ one token), launched from S with and without a nudge towards B; and with 64 real-model rollouts launched from contexts sampled as K-gram chains from S, starting at the absolute position where the model originally sat on the loop. The states for B served as controls: in no candidate did a rollout launched from B leave it, and every B was stable under the flow. Six of the seven candidates are unstable under the flow and are left by 34–100% of the real rollouts. The seventh (“ matrimony”) is stable under the flow although about half of the real rollouts leave it; this is the only disagreement between MTMF and the real model. The two controls (“ Virginia”, “ New York”) are stable in both tests.

The “ Presley” saddle (figure 5c). In the source trajectory the model repeated “ Presley” from token 137 to 607 and then switched abruptly to a “ Preserve” loop. The optimized state S has loss 7.4×10^{-3}, down from 3.0×10^{-2} at initialization, against 4.3×10^{-3} for “ Preserve” and 7.6×10^{-4} and 6.3×10^{-4} for the two further attractors “ Prescription” and “ Preserving”. S is not the pure repetition: 86% of its mass lies on the loop’s own K-grams, and the rest on one-token slips such as “ Presby”. The pure loop has a loss of 2.8×10^{-2}, four times higher, so the slips make the state more self-consistent. The deterministic flow leaves S without any perturbation after about 300 steps and converges to “ Prescription” (loss 6×10^{-5}), whereas “ Preserve” stays put. Of the 64 real-model rollouts launched from S, 32 remain near it after 450 tokens, 12 end in “ Prescription”, 8 in “ Preserve”, 3 in “ Preserving”, and 9 elsewhere; all 32 control rollouts launched from “ Preserve” remain there. The long “ Presley” repetition is therefore a saddle of the full model. It is sustained for hundreds of tokens and escapes along directions that share its “ Pres” prefix token.

Why the search finds only low-entropy states, and alternatives. A fixed point with unigram entropy H_1 and per-token conditional entropy h has a K-gram measure of entropy $H_K \approx H_1 + (K-1)h$, and representing it explicitly requires on the order of e^{H_K} K-grams. For the discounted distribution of a real Wikipedia prompt ($H_1 \approx 4.7$, $h \approx 4.5$ nats), $H_{10} \approx 45$ nats, far beyond any explicit support; only nearly deterministic states, i.e. repetition loops, fit into a support of $\sim 10^4$ K-grams. An alternative that avoids this limit keeps the operator but moves the state to the unigram marginal $\pi \in \Delta(V)$. MTMF reads the far field only through quantities linear in π, so at frozen π the K-gram chain is linear and its stationary measure is slaved to π, leaving a self-consistency condition $\pi = G(\pi)$ on the simplex alone. This can open the door for more tailored approaches, which we leave to future work.

K Discussion of Alternative State-Space Definitions

Unbounded Markov Chain: We could define the process as a Markov Chain between sequences, for example transitioning from the “Tomorrow we are going to the” state to the “Tomorrow we are going to the game” state. However, in doing so we will find that this is an *unbounded/infinite* chain, which

greatly limits the tools and results we can apply. Most importantly, it does not resolve the problem we presented, as this Markov Chain by definition can never visit the same state twice.

Sliding Context Window: In many practical implementations, is it common to limit the length of the context window to some (large) finite number of tokens, and use a sliding window implementation where at each step we eject the oldest token from the window and shift all other tokens one position backwards[5]. We can further force all sequences to be of the same length by left-padding them with "<PAD>" tokens until they fill the entire context window. While a naïve implementation of this is highly inefficient, from a theoretical viewpoint, it will indeed make our model act on a *finite/bounded* Markov Chain, enabling stronger analysis. This was done by Zekri et al. (2025). However, it is not satisfactory for our needs, as it is only a "superficial" fix for our original problem: Under this formalization, the sequences "<PAD><PAD><PAD><PAD><PAD><PAD>Tomorrow we are going to the game" and "<PAD><PAD><PAD>Tomorrow we are going to the game to the game" are still different, and would only formally repeat once the entire context window is exhausted – clearly missing some fundamental aspect of the dynamics. Moreover, since padding tokens are always masked, padding a sequence to the full length of the context window is also a superficial change, because the attention is still computed using only the same un-padded tokens as it would without the padding. Put simply, this only helps us in analyzing the regime of an exhausted context window, while we are interested in the regime before the window is exhausted, as it seems to be the one at which most models operate most of the time.

Simplifying the Architecture: Replacing the softmax attention with a linear attention operation, thus rendering it a stateful block (Katharopoulos et al., 2020). However, this fundamentally changes the architecture and performance of the model.

Learning a Latent Dynamical Space: We could decide to abandon the natural space of the Transformer, and instead train a latent representation model like an Autoencoder. Two promising directions for this are using sentence embedding models that let us project different-length sentences into a global space, thus making it possible to define distances between them. We could then look for fixed points or convergence in this embedding space. A second direction could be to train a new embedding space that will be specifically suited for dynamical analysis, for example using a method from the latent dynamical modeling literature. While both approaches are valid and promising directions for future research, they both make analytical analysis harder by introducing a second "black-box" into our framework. Without having strong, *interpretable* understanding of this embedding block, it will be hard to reach rigorous or interpretable results about its outputs.

L Beyond 1-Layer Models

L.1 2-Layers Model

Can we extend our framework to multi-layered models? We begin with describing the extension of our framework to a 2-layers model.

L.1.1 State Propagation

The main difference between a first layer head and a second layer head lies in the fact that the input to the second layer is no longer raw token embeddings – it is not strictly one of the $|V|$ possible tokens. Instead, it is a continuous residual stream vector. If we denote the residual stream vector at the input of the second layer at position i as r_i, we can examine what it depends upon:

$$r_i = e(t_i) + o_i(\pi_i, \mathcal{K}_i) \triangleq f(\pi_i, \mathcal{K}_i). \tag{24}$$

It is a function of both the current distribution π_i and the current K-gram $\mathcal{K}_i$. Assuming a purely semantic second-layer head for simplicity, we can apply the same reasoning again and conclude that the output depends only on the current residual stream r_i, and the *distribution* of previous residual stream vectors π_i^r. This is in turn a distribution over the input variables to this layer. In the first layer heads, the input was a raw token embeddings, meaning the only input variable was the token identity t_i, making π a distribution over tokens. However, for a second layer head, the input variables to the

[5] In practice more nuanced methods are used in order to preserve things like attention sinks and special tokens, but these details are not important for the claim we are making.

layer are $(\pi_i, \mathcal{K}_i)$. Thus, the distribution is the *joint probability measure* over *pairs* $(\pi, \mathcal{K})$. This is no longer a simple, categorical probability distribution over a finite vocabulary, but instead a joint probability measure over the continuous space $\mathbb{R}^{|V|}$ and the categorical states $\mathcal{K} \in V^K$. Extending this notion to deeper layers, we get a hierarchical probability measure structure. This makes direct analytical analysis of the general case mostly intractable.

L.1.2 State Compositions

However, similar to circuits analysis, in practice there might be cases that simplify this. The main idea to consider is that of Q-, K- and V-compositions, as discussed in detail by Elhage et al. (2021). Even though the full residual stream might depend on the output of the first layer attention heads, the second layer heads would only be affected by this if their QKV projections *read* from the same subspace in the residual stream that the previous attention heads *wrote* to. In practice, pretrained models don't always take advantage of this possibility. For example, in the simple 2-layers model analyzed by Elhage et al. (2021), the authors empirically discovered that most of the second layer heads are not involved in any composition, rendering them effectively equivalent to extra first-layer heads. Thus our first-layer analysis is perfectly valid for them as well.

When heads *are* involved in compositions, how does that affect them? The answer depends both on the specific kind of composition, and whether the involved heads are fast or slow. In broad terms, the dependencies can be described as:

Q-composition vs. K- and V-composition A Q-composition changes the query vector, but not the KV cache vectors it operates against. It can be understood as changing the *function* of the head, but *not its state*. Since the state is related to the KV cache vectors, it is still dependent in this case on the input token and distribution in the same way as a first-layer head. This means that the update equation for π_i remains unchanged. However the output of the head is now dependent also on any extra variables that effect the output of the head we perform the composition with (the writing head). On the other hand, K- or V-compositions alter the KV cache vectors themselves, lifting the distribution variable to a distribution over the variables determining the output of the writing head.

Fast vs. Slow Compositions There is also dependence of whether the writing head is a slow or a fast head. If the writing head is a fast head, its output depends only on the current K-gram, and not on the current unigram distribution. A head that performs K- and V-compositions with this head, will have its distributional variable lifted to a *distribution over K-grams* instead of over unigrams: $\pi^{(K)} \in \mathbb{R}^{|V|^K}$. If the writing head is a slow head, its output depends on its discounted unigram distribution. A head that participates in K- or V-compositions with this head will have its distributional variable lifted to a joint probability measure over pairs of distributions and tokens $(\pi^{(1)}, t)$.

L.1.3 Example: Induction Circuit

Using these insights, we can analyze a simple example of a basic induction circuit, as found in a 2-layer attention-only model (Elhage et al., 2021). In this model, the only significant composition is a *K-composition* between a *fast* first-layer head (the previous-token head) and a *slow* second-layer head (the induction head). Moreover, the previous-token head specifically looks only a single position back (in the simplest case), making its lookback cutoff $K = 2$. Thus, the second-layer induction head is essentially a pure semantic head, but *operating over bigrams* instead of unigrams: $\pi_i^{(2)} \in \mathbb{R}^{|V|^2}$ of this head is a distribution over bigrams. This exemplifies an important role of compositions in deep networks – they can function as a way to *lift/expand the vocabulary space* over which we are operating. It also shows how in practice, deep heads might still operate in a way conceptually similar to the first-layer heads we analyzed – with the same fast-slow structure, and the same distributional dependence – but operating over a lifted or expanded vocabulary space.