Transformers Represent Belief State Geometry in their Residual Stream

Adam S. Shai
Simplex
PIBBSS[*]

Sarah E. Marzen
Department of Natural Sciences
Pitzer and Scripps College

Lucas Teixeira
PIBBSS

Alexander Gietelink Oldenziel
University College London
Timaeus

Paul M. Riechers
Simplex
BITS

Abstract

What computational structure are we building into large language models when we train them on next-token prediction? Here, we present evidence that this structure is given by the meta-dynamics of belief updating over hidden states of the data-generating process. Leveraging the theory of optimal prediction, we anticipate and then find that belief states are linearly represented in the residual stream of transformers, even in cases where the predicted belief state geometry has highly nontrivial fractal structure. We investigate cases where the belief state geometry is represented in the final residual stream or distributed across the residual streams of multiple layers, providing a framework to explain these observations. Furthermore we demonstrate that the inferred belief states contain information about the entire future, beyond the local next-token prediction that the transformers are explicitly trained on. Our work provides a general framework connecting the structure of training data to the geometric structure of activations inside transformers.

1 Introduction

In this work, we present a rigorous and concrete theoretical framework that connects the structure of training data to the geometry of activations in trained transformer neural networks. Our framework is grounded in the theory of optimal prediction. It suggests that transformers pretrained on next-token prediction will develop internal structures characterized by the meta-dynamics of belief updating over hidden states of the data-generating process. In other words, pretrained models should learn *more* than the hidden structure of the data generating process—they must also learn how to update their beliefs about the hidden state of the world as they synchronize to it in context.

To test this framework, we conduct well-controlled experiments where we train transformers on data generated from processes with hidden ground truth structure, and then use our theory to make predictions about the geometry of internal activations. Even in cases where the framework predicts highly nontrivial fractal structure, our empirical results confirm these predictions (Figure 1). This provides a comprehensive explanation of how transformers encode information beyond the local next-token predictions they are explicitly trained on.

In Section 2, we review the relevant theory from computational mechanics which motivates our prediction of the geometry of activations in the residual stream. Core to our work is the *mixed-state presentation*, which describes the metadynamic of beliefs in the probability simplex over states of the

[*]Principles of Intelligent Behaviour in Biological and Social Systems

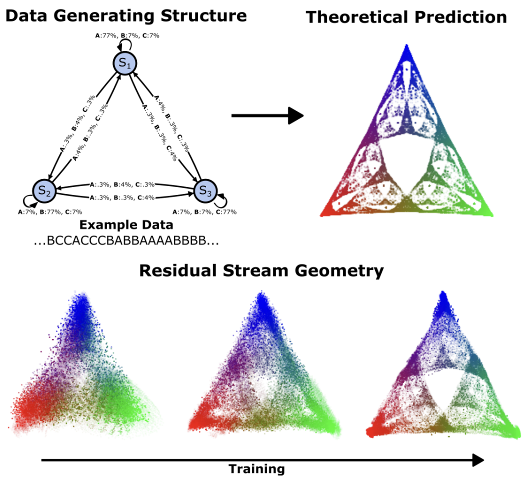

Figure 1: (Top) Given a hidden data-generating structure, our framework predicts a unique belief state geometry in a probability simplex. Often these have highly nontrivial fractal structure as shown in this example. (Bottom) Our main experimental result is that we find that the fractal geometry of optimal beliefs is linearly embedded in the residual stream, and emerges over the course of training.

data generating process. This formalism leads to a geometric structure that forms a natural hypothesis for the internal states of neural networks trained on prediction tasks.

In Section 3, we verify that this geometry is linearly represented in the residual stream of transformers. We implement two main experiments that control for different aspects of the hidden structure of the data generating process and make concrete predictions about the internal states of networks trained on these datasets. In some cases the geometry of belief state updating is found in the final residual stream, while in others it is spread out across multiple layers. We detail how our framework explains this phenomenon. After demonstrating the initial success of this framework, Section 4 discusses the theory and implications in more depth.

2 Theory and methods

2.1 Data generating processes

In this study, we assume that our training data is generated by an edge-emitting hidden Markov model (HMM)[2]. Paths through the HMM produce sequences of tokens from a predefined vocabulary. Tokens ($x \in \mathcal{X}$) are emitted as we transition between hidden states ($s \in \mathcal{S}$). These transitions are governed by token-labeled transition matrices $\{T^{(x)}\}$, where each $T^{(x)}_{i,j} = \Pr(x, s_j | s_i)$ represents the joint probability of emitting token x and transitioning to state s_j, given that the HMM was in state s_i.

As a simple example, consider the HMM shown in Figure 2, that we call the Zero-One-Random (Z1R) process. The Z1R process generates strings of tokens of the form `...01R01R...`, where `R` is a randomly generated `0` or `1`. This HMM has 3 states: $\mathtt{S}_0$, $\mathtt{S}_1$, and $\mathtt{S}_\mathtt{R}$. Arrows of the form $s \xrightarrow{x:p\%} s'$ denote the probability $\Pr(x, s'|s) = p\%$ of moving to state s' and emitting the token x, given that the process was in state s.

2.2 Mixed-state presentations

There are competing intuitions for what transformers should represent. Are they stochastic parrots [2]? Do they build a world model [13]? One natural intuition would be that transformers represent the hidden structure of the data-generating process—a world model—in order to predict the next token well. For instance, Ilya Sutskever has said: "Predicting the next token well means that you understand

[2]For arbitrarily large, possibly non-ergodic HMMs with arbitrary initial distribution over the latent states, this does not limit the sophistication of training data.

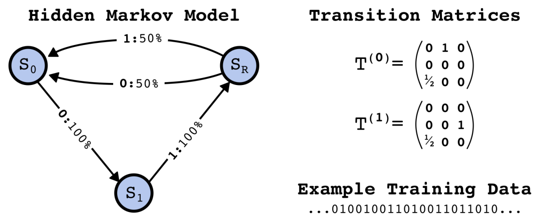

Figure 2: An illustration of a hidden Markov model (HMM) and its components. The left side shows the HMM with states $\mathtt{S}_0$, $\mathtt{S}_1$, and $\mathtt{S}_\mathtt{R}$, and their respective transition probabilities. The right side displays the transition matrices $T^{(0)}$ and $T^{(1)}$ corresponding to token emissions 0 and 1. Example training data is provided at the bottom, demonstrating a sequence generated by the HMM.

the underlying reality that led to the creation of that token [32]." This type of intuition is natural, but not formal.

Computational mechanics is a formalism that was developed in order to study the limits of prediction in chaotic systems, and has since expanded to a deep and rigorous theory of computational structure for any process [4]. One of its contributions is in providing a rigorous answer to what structures are necessary to perform optimal prediction [31].

Interestingly, computational mechanics shows that prediction is substantially more complicated than generation [19, 30]. Informally, the reason for this is that even if an agent knows the underlying hidden generative structure that creates the data they are trying to predict, the agent must still perform computational work in order to infer which state the generative process is in, given finite observations of data. Thus, there are conceptually two distinct inference processes related to prediction: learning the hidden structure of the data generating process, and subsequently inferring which state the data generating process is in given finite observations of data.

Computational mechanics formally captures the computational structure of the latter inference process with the *mixed-state presentation* (MSP). Whereas the HMM of a data generating process describes a generative structure (Figure 3A), the MSP is the structure of prediction (Figure 3B). The MSP answers the question of how an optimal observer updates their beliefs over the states of the data-generating process given finite observations of tokens [27, 15]. If the observer is in a belief state given by a probability distribution $\boldsymbol{\eta}$ (a row vector) over the hidden states of the data generating process, then the update rule for the new belief state $\boldsymbol{\eta}'$ given that the observer sees a new token x is:

$$\boldsymbol{\eta}' = \frac{\boldsymbol{\eta} T^{(x)}}{\boldsymbol{\eta} T^{(x)} \mathbf{1}} \tag{1}$$

where $\mathbf{1}$ is a column vector of ones of appropriate dimension, with the denominator ensuring proper normalization of the updated belief state. In general, starting from the initial belief state $\boldsymbol{\eta}_\varnothing$, we can find the belief state after observing a sequence of tokens $x_0, x_1, \ldots, x_N$:

$$\boldsymbol{\eta} = \frac{\boldsymbol{\eta}_\varnothing T^{(x_0)} T^{(x_1)} \ldots T^{(x_N)}}{\boldsymbol{\eta}_\varnothing T^{(x_0)} T^{(x_1)} \ldots T^{(x_N)} \mathbf{1}} \ . \tag{2}$$

This process of belief updating is itself another HMM, where hidden states are associated with belief states, and paths leading to particular belief states are the observed token sequence. The probability of transitioning from belief state $\boldsymbol{\eta}$ to $\boldsymbol{\eta}' = \boldsymbol{\eta} T^{(x)} / \boldsymbol{\eta} T^{(x)} \mathbf{1}$ of Eq. (1) is simply given by the probability $\boldsymbol{\eta} T^{(x)} \mathbf{1}$ of observing an x from $\boldsymbol{\eta}$. For stationary processes, the optimal initial belief state is given by the stationary distribution $\boldsymbol{\eta}_\varnothing = \boldsymbol{\pi}$ over latent states (the left-eigenvector of the transition matrix $T = \sum_x T^{(x)}$ associated with the eigenvalue of 1).

Each belief state of the MSP is a probability distribution over the states of the data generating process. Belief states thus inherit a natural geometry; we can plot these belief states in a probability simplex, as indicated in Figure 3C. We refer to the geometric arrangement of the points on the simplex as the *belief state geometry*. Note that each belief state—as a distribution over generator states—induces a probability density over all possible futures. Distributions over the entire future clearly carry more than just next-token information. We can see this in our example in Figure 3D. The states $\boldsymbol{\pi}$, $\boldsymbol{\eta}_{10}$, and $\boldsymbol{\eta}_{01}$ all have the same next-token prediction despite being distinct belief states.

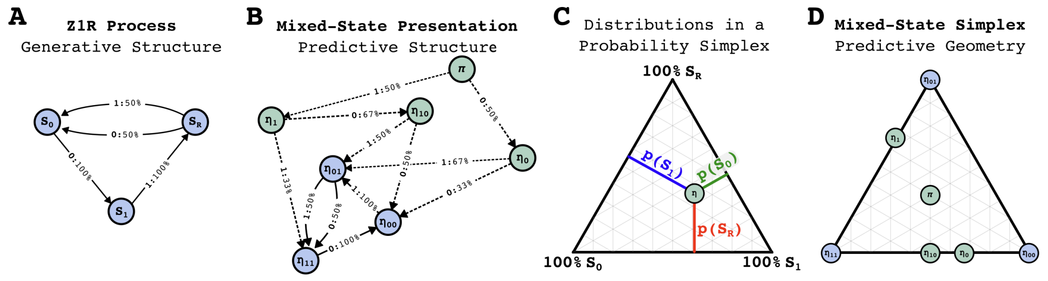

Figure 3: (A) An example generative structure called the zero-one-random process (Z1R), since it generates data of the form `...01R01R01R...` where `R` is a random bit. (B) The generative structure implies a unique metadynamic over belief states as a predictor synchronizes to the hidden state of the world as it observes more context. This predictive structure is called the mixed-state presentation (MSP). We label belief states with $\boldsymbol{\eta}_w$ where w is the shortest string of emissions which leads to that belief state. (C) Belief states are distributions over generator states and can be embedded in a probability simplex. To read off this distribution for a given belief state, $\boldsymbol{\eta}$, one measures the perpendicular distance from the point to each edge of the simplex, shown as blue, green, and red lines in this example. These distances directly give the probabilities for each state. Thus, the vertices represent states of certainty over one generator state, since the perpendicular distance is nonzero to only one of the edges of the simplex. (D) Plotting the belief state distributions in the probability simplex gives the belief state geometry.

2.3 Finding the belief simplex in the residual stream

As described in the previous section, the MSP and belief state geometry are formalizations of the computational structure of the belief updating process an observer must perform in the service of optimal prediction. Consequently, a natural hypothesis would be that transformers, trained on data generated from a given ground-truth HMM, will represent the MSP structure internally. Our procedure for finding the belief states in the residual stream of a pretrained transformer is depicted schematically in Figure 4.

We begin by training a transformer on data generated by a ground-truth HMM. We consider activations of the residual stream (in either a single layer, or a concatenation of layers) induced by all possible input sequences (Figure 4AB), and at all context window positions. Thus our dataset consists of a set of activations, $a \in \mathbb{R}^{d_{\text{resid}}}$, when we consider a single layer at a given position, where d_{resid} is the residual stream dimension. For each input sequence, our framework tells us which belief state the input corresponds to, and the probability distribution over hidden states of the generative process that corresponds to this belief state. These probability distributions can be thought of as points $b \in \mathbb{R}^{|\mathcal{S}|}$, where $|\mathcal{S}|$ denotes the number of hidden states in the generative process.

Since every input has a belief state associated with it[3], we can label (or color, as in Figure 4C) each activation by the associated ground-truth belief state. We then use standard linear regression to find an affine map from a to b, of the form $b \approx Wa + c$, where $W \in \mathbb{R}^{|\mathcal{S}| \times d_{\text{resid}}}$ is a weight matrix and $c \in \mathbb{R}^{|\mathcal{S}|}$ is a bias vector. Parameters W and c are found by minimizing the mean squared error between the predicted belief states and the true belief states. The matrix W projects from the residual stream to (as best as possible) the probability simplex, shown diagrammatically in Figure 4CD.

3 Results

3.1 Belief state geometry is linearly represented in the residual stream

To investigate the computational structure learned by transformers we conducted an experiment using a simple 3-state hidden Markov model (HMM) called the Mess3 Process [20] (Figure 5). We used the data generated by this process (Figure 5) to train a transformer model.

[3] In general, multiple inputs will lead to the same belief state.

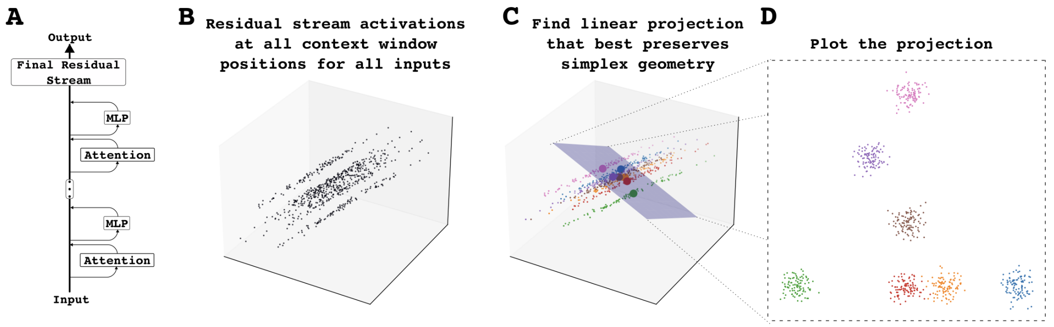
A
Output
Final Residual Stream
MLP
Attention
MLP
Attention
Input
B Residual stream activations at all context window positions for all inputs
C Find linear projection that best preserves simplex geometry
D Plot the projection

Figure 4: To verify if transformers represent belief state geometries in their residual streams, we record (A) residual stream activations at all context window positions over all inputs. (B) These activations live in a high dimensional space. (C) Each input has a ground-truth optimal belief state, which is a probability distribution over states of the data-generating process. In this way we can label, or color, each activation by the ground-truth belief associated with the input. (D) Using linear regression we then find a linear subspace of the activation space that best preserves the belief state geometry of the simplex.

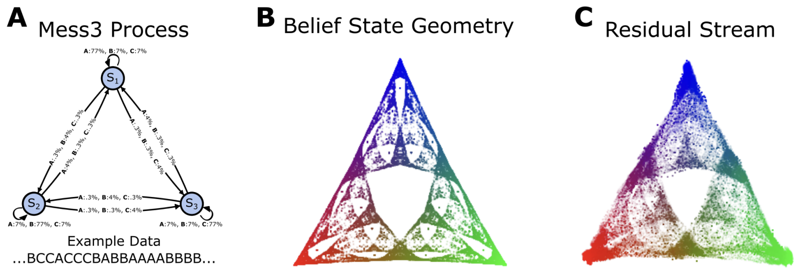

Figure 5: The residual stream of trained transformers linearly represents the belief state geometry of the mixed-state presentation. (A) The Mess3 Process has 3 hidden states and generates sequences in a token vocabulary of $\{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$. (B) The ground truth belief state geometry of the Mess3 Process has intricate fractal structure. Each point in this plot is a belief state—a probability distribution over the hidden states of the Mess3 Process. Points are colored by taking the belief probability distribution and using them as RGB values. (C) We find a linear projection of the final residual stream activations contains a representation of the ground-truth belief geometry. Points are colored according to the ground-truth belief states.

The Mess3 MSP has a fractal structure, shown in Figure 5B, providing a highly non-trivial test of our theory's prediction. Each point in this geometry corresponds to a probability distribution over the hidden states of the data generating process, and thus lies in a 2-simplex.

To test these predictions, we analyzed the final layer of the transformer's residual stream, before the layer norm and unembedding. Using linear regression, we identified a 2D subspace of the 64-dimensional residual activations that best matched the ground-truth belief distributions from the MSP. Remarkably, the geometry of this 2D subspace closely resembled the predicted fractal structure (Figure 5C), providing strong evidence that the transformer had indeed learned to represent the geometry of the belief states in its residual stream[4].

We ran a number of controls, shown in Figure 6, to make sure these results were not artifacts. First, we performed our analysis at multiple points through training and found that the simplex structure emerged gradually (Figure 6A), suggesting that the detailed fractal structure found at the end of training was not a trivial consequence of the model architecture or initialization. Second, quantifying the mean squared error of the regression revealed a decrease in the errors of belief-state geometry representation throughout the course of training (Figure 6D, first 4 bars). In addition, more faithful

[4]These results hold when analyzing the activations after the final layer norm as well, see Figure S1.

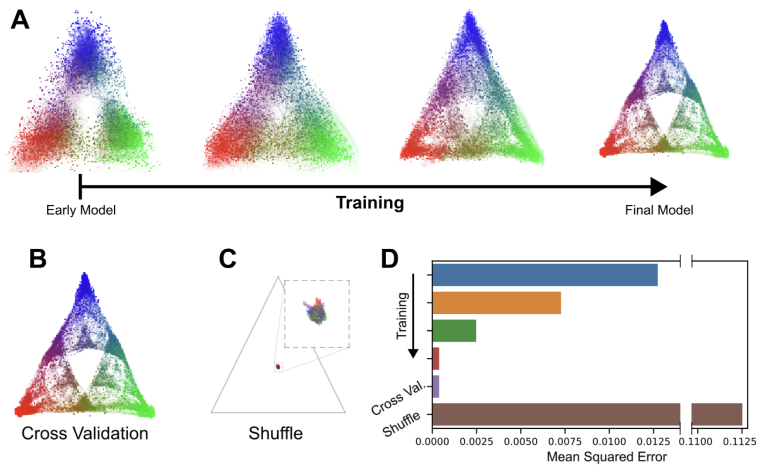

Figure 6: The representation of belief state geometry is nontrivial. (A) Projected activations at different stages of training shows the emergence of belief state geometry. (B) Cross-validation of our main result. (C) We shuffle the belief states in our linear regression procedure, preserving the overall ground-truth fractal shape while getting rid of the associated context. The new projection collapses the data, showing that the fractal's appearance in the residual stream is not an artifact of projecting high-dimensional data to a desired shape. (D) Mean squared error (averaged across input sequences) between (i) the position of projected activations and (ii) the ground-truth position of the corresponding belief state.

representation of the belief state geometry in the residual stream corresponded to lower cross entropy loss (Figure S2). Third, we performed cross-validation, where only 20% of all input-activation pairs were used to train the regression, and then the held out 80% data was used to visualize and analyze the result, and found that the geometry of the belief state was still well represented (Figure 6B) and that the mean squared error was similar to that of the full regression (Figure 6D, red vs. purple bars). Finally, we ran a control that preserved the fractal geometry in the simplex but shuffled the input-point correspondences, resulting in the regression mapping all points to the simplex center due to the lack of discoverable structure (Figure 6C).

These results provide compelling evidence for our central claim: transformers trained on data with hidden generative structure will learn to represent the geometry of belief states in their residual stream. The close match between the predicted fractal structure and the empirical geometry of the residual activations, even for the highly complex Mess3 MSP, suggests that this geometry is a fundamental aspect of how transformers build predictive representations of their input data.

3.2 Belief state geometry represents information beyond next-token prediction

Often, *distinct* belief states will have the *same* next-token prediction associated with them. Our framework suggests that transformers will keep internal distinctions in the representations of these belief states, despite the fact that transformers are trained explicitly on next-token prediction.

The Random–Random–XOR (RRXOR) process, shown in Figure 7A, has these types of degeneracies [28]. The MSP of the RRXOR process has 36 distinct belief states, and can be geometrically represented in a 4-simplex. In Figure 7B we visualize the simplex geometry by projecting down to 2 dimensions. After training, we run our regression analysis to see if the belief state geometry is represented in the residual stream. Unlike our previous results for the Mess3 process, we find that the belief state geometry is not represented in the final residual stream before the unembedding. However, when we take the residual stream activations of all layers and concatenate them, we find a veridical representation (Figure 7C).

The geometry we find is not explainable by next-token predictions. To quantify this, we ask if the pairwise distances between belief state representations in the transformer can be explained by pairwise distances in the next-token predictions, or if they are better explained by ground truth belief state distances. First, we compute the Euclidean distance between pairs of ground truth belief states.

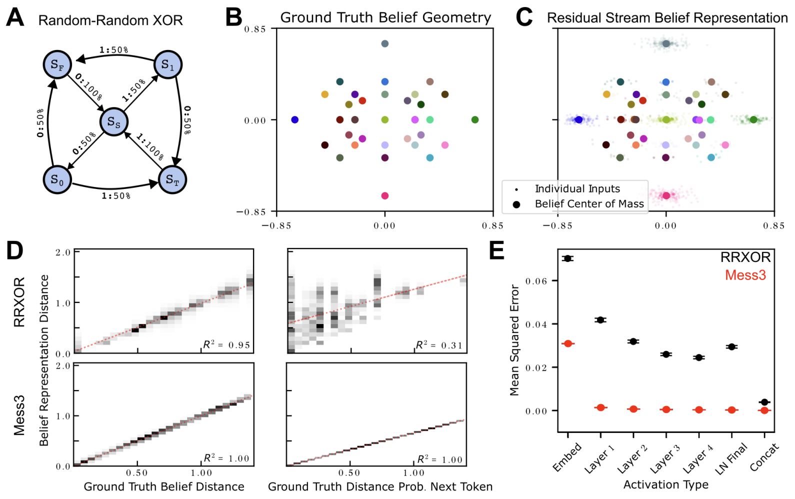

Figure 7: The belief state geometry can be represented across multiple layers when the belief state structure has next-token degeneracies. (A) The Random-Random-`XOR` process has zero pairwise correlation, but has interesting higher-order structure. The MSP of this process has 36 distinct states, with many of them degenerate in terms of their optimal next-token predictions. (B) The belief state geometry lies in a 4-simplex. To visualize we project down to 2-dimensions, with each belief state colored uniquely. (C) The transformer linearly represents this belief state geometry. Small dots correspond to individual activations and are colored according to the ground truth belief state. Large dots correspond to the center of mass of all activations associated with a particular ground-truth belief state. (D)We compare Euclidean distances between pairs of ground truth belief states and see if those distances are preserved in the belief state representation in the transformer (left scatter plots), showing good correspondence for both RRXOR and Mess3 processes. The right scatter plots compare distances in ground truth next-token probabilities to distances in the transformer's belief state representations, showing low correlation for RRXOR ($R^2 = 0.31$) and high correlation for Mess3, indicating that RRXOR belief state geometry is not captured by next-token predictions alone. (E) Mean squared error of the linear regression procedure, applied to individual layers and, at the far right, applied to the concatenation of activations across layers.

For each pair, we also compute distances between the corresponding belief state representations in the transformer. In (Figure 7D Left) we compare the ground truth pairwise distances to the represented pairwise distances, and find that for both the RRXOR and Mess3 process, there is good correspondence.

Next, for every pairwise distances of belief state representations, we compute the corresponding distance in the ground truth next token probabilities. The scatter plots on the right of Figure 7D compares these distances. For the RRXOR process, the correlation with next-token predictions is relatively low ($R^2 = 0.31$) compared to the correlation with the belief state geometry ($R^2 = 0.95$), indicating that belief state geometry captures structure in the residual stream not captured by next-token predictions. For the Mess3 process, the pairwise structure of next-token predictions is preserved in the belief state representation. This difference between the RRXOR and Mess3 results is expected from the theory. Since the RRXOR process contains distinct belief states with the same optimal next-token distribution, the discovered belief state geometry is not well-explained by the next token predictions.

3.3 Belief state geometry can be spread across layers of the residual stream

The framework presented here suggests that belief states should be represented in transformers, but does not say that they must be represented in the final layer. In cases of belief states that are degenerate in their next-token predictions, distinctions can be lost before the unembedding. Thus, in the Mess3 process we expect belief state geometry to persist to the final layer of the residual stream, whereas in the RRXOR trained transformer belief state information can be collapsed before the unembedding.

To quantify this, we report the mean squared error of the belief geometry regression over the different layers of the transformer as well as the concatenation of residual activations from all layers in Figure 7E. We find that the RRXOR belief state geometry is not well represented in the final layer of the transformer. In contrast, the Mess3 belief state geometry is well represented in indidividual layers of the transformer, and persists through to the final layer before the unembedding. Interestingly, the RRXOR belief state geometry is not represented in *any* of the individual layers of the transformer, but is represented in the concatenation of the layers. This suggests that the belief state geometry is spread across multiple layers of the residual stream.

4 Discussion

4.1 Belief state geometry: Why?

During training, models do not directly see the hidden structure generating the data—they only see data. So how is it that the pretrained model infers belief states in the simplex of the generator states? In fact, there are infinitely many distinct HMMs that can generate the same stochastic process [5]. Despite this, any stochastic process has a canonical vector-space representation in the space of probability densities over all possible future token sequences [33]. This corresponds to a minimal HMM representation, and we should expect that transformers learn belief state geometry in the simplex over these minimal generator states. This canonical geometry for a process provides an explanation for why we were able to find the belief state geometry represented in our transformers.

In the jargon of computational mechanics, it is notable that the recurrent belief states map onto the causal states of the process' ϵ-machine [4]. The recurrent states must be distinguished for prediction, even if there exists a generative HMM that is smaller than the ϵ-machine. Typically, data can be generated by a much smaller mechanism than is required to predict it [19, 30]. Accordingly, transformers and other pretrained models learn much more than just an accurate generative model of the world—they also learn how to synchronize to the hidden state of the world through observed context.

We expect our main results do not depend strongly on the particular neural network architecture of transformers, except that they (i) utilize a residual stream and (ii) were pretrained on future-token prediction. Indeed, we expect to find the same belief state geometry in other neural network architectures like Mamba [12], and recent work building on our discovery has shown that recurrent neural networks do represent the belief state geometry for the processes tested in our experiments [26]. The generality of our results is thus a powerful architecture-independent insight for interpretability. Pretraining will generally induce mixed-state geometry. We should then be able to work backwards from this knowledge and the knowledge of the neural network architecture to determine both its world model and how it performs Bayesian updating over its hidden states.

Corollary 2 of [31] implies that, for a recurrent neural network to perform as well as possible on next token prediction, all information about the past necessary to predict the entire future as well as possible must be present in the latent state of the network. The implication for feedforward networks like transformer architectures is much less obvious, but it motivates the hypothesis that to perform as well as possible on next-token prediction requires that all information about the past necessary to predict the entire future as well as possible must be present across the layers of each context window position's residual stream. Our results here support this interpretation.

Intuitively, why would a model learn about the entire future if it is only trained on next-token prediction? The simple answer is that, even if different belief states imply the same probability distribution over the next token, any distinction in probability distributions over the entire future may *eventually*, after further context, imply distinct distributions over the next token at some point in the future. If the initial nuance were discarded, then the ability to distinguish next-token probabilities in the future would be compromised. To minimize loss, pretrained models need to not only nail the next-token probability distribution, but also retain all information necessary to get the correct next-token distributions in the distant future. It turns out that this requires distinguishing all pasts that induce distinct probability distributions over the entire future [31].

It is tempting to think that all of this information will persist at the final layer of the residual stream, but this is not strictly necessary nor borne out in practice as we showed in Figure 7. In general, the belief state is spread across the layers at each position. This is consistent with [25], which found that the predictive accuracy of the residual stream with respect to far-future tokens peaks somewhere in intermediate layers. Performing as well as possible on multi-token prediction, as in [10], would imply the same belief-state structure as next-token prediction, but it should change how this information is spread across the layers and, in practice, performance may be different when loss is not minimized.

It is also interesting to consider the linearity of the belief state representations we found, and if the linearlity is strictly necessary for a network to optimize next-token cross-entropy loss. In fact, recent work has constructed a transformer by hand (i.e. a human being selected the weights) that perfectly solves the RRXOR task, but where the belief state geometry *is not* linearly represented [11]. It is thus a non-trivial empirical finding that transformer architectures trained via stochastic gradient descent find solutions in which the belief state geometry *is* linearly represented.

4.2 Further implications

Our finding of the belief-state simplex in the residual stream strongly suggests that any model capable of minimal loss requires as many residual-stream dimensions as the number of states in a minimal generative model of the stochastic process sampled by the training data. Moreover, we should be able to use this new understanding to determine the minimal loss with fewer residual-stream dimensions. Likewise, using an adaptation of the rate distortion theory applied to the belief-state simplex [20], we anticipate that our framework should be able to make non-trivial predictions about the evolving geometry of activations during training.

To the extent that different users are represented in the training data, they can be thought of as disconnected ergodic components of a non-ergodic stochastic process. Even after the model has learned, it needs to synchronize (in context) to both (i) the ergodic component representing the user and (ii) the current latent state of that component. Each will show up in different ways in the MSP. Using this framework, we should be able to predict rates of adaptations to different users—i.e., rates of convergence in context—as identified by the reduction in next-token entropy as context position increases.

4.3 What are features?

A growing literature in LLM research studies what *features* these systems represent [3]. Importantly, there is no currently agreed upon definition of feature. Consequently, it has been difficult to study this issue in a principled manner in which ground truth features are known. While the MSP provides a ground truth understanding of the computation next-token predictors are asked to accomplish, the relationship between belief states and features is not necessarily one-to-one. A single belief state may encompass multiple features, and the same feature may be represented across multiple belief states. The exact mapping between belief states and features is likely to be complex and dependent on the specific architecture and training data of the transformer model, making it an exciting open problem for future research.

4.4 Limitations and applicability to more realistic settings

Our experimental validation focused on small-scale systems, using HMMs with only 3-5 states and vocabularies of 2-3 tokens. While these systems exhibited meaningful complexity through infinite Markov order in both the RRXOR and Mess3 processes, they represent a significant simplification compared to the sophisticated architectures and massive vocabularies (>50,000 tokens) of modern language models trained on natural language data. This scale limitation raises important questions about how our framework extends to larger and more complex settings.

For realistic systems like board games or natural language, the dimensionality of the belief state simplex would exceed the dimension of the residual stream. This suggests that if transformers represent belief state geometry in these domains, they must do so in a compressed form. Understanding the nature of this compression—what information is preserved and what is discarded—represents a crucial direction for future work.

As the size of the minimal generating structure and the context window length grows, explicitly computing ground-truth belief states becomes intractable. We will need new methodologies for validating our framework in settings where we cannot directly compute the MSP structure.

We limited ourselves to the simpler stationary ergodic setting in this paper to empirically verify important basic features of transformer representations. However, the framework presented in this work will naturally extend to non-stationary [29] and non-ergodic processes [6], which is important for the extension to real-world tasks. For example, in in-context learning the underlying process is inherently nonergodic. In the more general setting, we also expect transformers to represent belief-state geometry, although that geometry will then reflect the more general types of non-stationary and non-ergodic processes representative of real-world data like natural language [7, 8].

In the experiments we ran, we generated training data using HMMs. A natural question is what to expect with training data generated from non-HMM sources. It is important to note that any dataset made of sequences of tokens used to train a transformer can be represented as being generated from an HMM, if one allows for non-ergodicity, non-stationarity, and infinite states. For example, in the case of training data that takes the form of parenthesis matched strings, one would usually conceive of this as being generated by a push-down automaton. However, this can equivalentaly be represented by an HMM consisting of an infinite chain of hidden states. Another interesting point that we did not focus on in our presentation for the sake of simplicity, is that even if one does not assume an HMM for the generating machine, the question of the computational structure of an optimal predictor is naturally an HMM, with hidden states given by distinct distributions over the future (whether the generator is an HMM or a Turing Machine, or something else) [33].

In this work we verified that belief states are linearly represented in the residual stream, even in cases where the belief states form intricate fractal geometries. Although this is quite suggestive that these representations are used to implement some form of Bayesian updating, we did not directly test this. In other words, we do not yet know if these LLMs develop something like a circuit for implementing Bayesian updates, as would be natural for MSP dynamics. So far, we have only established the representation of belief states and their geometry.

Computational mechanics is primarily (but not completely, see Ref. [21]) concerned with optimal prediction, but LLMs in practice are not perfectly optimal. Further work is needed to understand how near-optimality and non-optimality influence belief state representations. Progress in this direction could offer insights into evolving structures during training.

Moving beyond stationary ergodic HMM training data, we still generically expect fractal-like structure in the residual-stream activations. Why? The fractal structure is a result of Bayesian updating applied to the "non-deterministic" computational structure of HMMs [16]. With stationary processes, this same Bayesian map is folded into itself time and again, producing a very clean fractal. For natural language, even in the absence of stationarity, there will nevertheless be resonances of non-deterministic computational structure, which would imply a type of folding beliefs in a self-similar manner. However, a more rigorous version of these statements and the empirical validation is left for future work.

5 Conclusion

In this work, we introduced a theoretical framework that establishes a clear connection between the structure of training data and the geometric properties of activations in trained transformer neural networks. Through experiments, we validated that the geometry of belief states is linearly represented within the residual stream of the transformer architecture. This finding suggests that transformers construct predictive representations that surpass simple next-token prediction. Instead, these representations encode the complex geometry associated with belief updating over hidden states, capturing an inference process over the underlying structure of the data-generating process.

Our work takes a significant step towards concretizing the understanding of the computational structures that drive the behavior of large language models (LLMs) and how these structures relate to the data on which they are trained. This concretization is crucial, as the rapid advancements in LLMs have led to models with increasingly sophisticated behavioral capabilities. However, without a clear understanding of the underlying computational structures and their relationship to the training data, it becomes challenging to interpret, trust, and further improve these models.

References

[1] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

[2] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[3] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[4] J. P. Crutchfield. Between order and chaos. *Nature Physics*, 8(January):17–24, 2012. doi: 10.1038/NPHYS2190.

[5] J. P. Crutchfield, C. J. Ellison, J. R. Mahoney, and R. G. James. Synchronization and control in intrinsic and designed computation: An information-theoretic analysis of competing models of stochastic computation. *CHAOS*, 20(3):037105, 2010. Santa Fe Institute Working Paper 10-08-015; arxiv.org:1007.5354 [cond-mat.stat-mech].

[6] James P Crutchfield and Sarah Marzen. Signatures of infinity: Nonergodicity and resource scaling in prediction, complexity, and learning. *Physical Review E*, 91(5):050106, 2015.

[7] Lukasz Debowski. *Information theory meets power laws: Stochastic processes and language models*. John Wiley & Sons, 2020.

[8] Łukasz Dębowski. A simplistic model of neural scaling laws: Multiperiodic santa fe processes. *arXiv preprint arXiv:2302.09049*, 2023.

[9] Joshua Engels, Isaac Liao, Eric J Michaud, Wes Gurnee, and Max Tegmark. Not all language model features are linear. *arXiv preprint arXiv:2405.14860*, 2024.

[10] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[11] Rick Goldstein. Handcrafting a network to predict next token probabilities for the random-random-xor process. https://apartresearch.com/event/compmech, June 2024. Research submission to the Computational Mechanics Hackathon! research sprint hosted by Apart, PIBBSS, and Simplex.

[12] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[13] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[14] Jiachen Hu, Qinghua Liu, and Chi Jin. On limitation of transformer for learning hmms. *arXiv preprint arXiv:2406.04089*, 2024.

[15] A. M. Jurgens and J. P. Crutchfield. Shannon entropy rate of hidden Markov processes. *Journal of Statistical Physics*, 183(2):32, 2021.

[16] Alexandra M Jurgens and James P Crutchfield. Divergent predictive states: The statistical complexity dimension of stationary, ergodic hidden markov processes. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(8), 2021.

[17] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

[18] Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

[19] Wolfgang Löhr. Predictive models and generative complexity. *Journal of Systems Science and Complexity*, 25:30–45, 2012.

[20] S. E. Marzen and J. P. Crutchfield. Nearly maximally predictive features and their dimensions. *Phys. Rev. E*, 95(5):051301(R), 2017. doi: 10.1103/PhysRevE.95.051301. SFI Working Paper 17-02-007; arxiv.org:1702.08565 [cond-mat.stat-mech].

[21] Sarah E Marzen and James P Crutchfield. Nearly maximally predictive features and their dimensions. *Physical Review E*, 95(5):051301, 2017.

[22] Neel Nanda and Joseph Bloom. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`, 2022.

[23] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[24] Pedro A Ortega, Jane X Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

[25] Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. *arXiv preprint arXiv:2311.04897*, 2023.

[26] Keenan Pepper. RNNs represent belief state geometry in their hidden states. https://apartresearch.com/event/compmech, June 2024. Research submission to the Computational Mechanics Hackathon! research sprint hosted by Apart, PIBBSS, and Simplex.

[27] P. M. Riechers and J. P. Crutchfield. Spectral simplicity of apparent complexity, Part I: The nondiagonalizable metadynamics of prediction. *Chaos*, 28:033115, 2018. doi: 10.1063/1.4985199.

[28] P. M. Riechers and J. P. Crutchfield. Spectral simplicity of apparent complexity, Part II: Exact complexities and complexity spectra. *Chaos*, 28:033116, 2018. doi: 10.1063/1.4986248.

[29] Paul M Riechers and James P Crutchfield. Fraudulent white noise: Flat power spectra belie arbitrarily complex processes. *Physical Review Research*, 3(1):013170, 2021.

[30] Joshua B Ruebeck, Ryan G James, John R Mahoney, and James P Crutchfield. Prediction and generation of binary markov processes: Can a finite-state fox catch a markov mouse? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(1), 2018.

[31] C. R. Shalizi and J. P. Crutchfield. Computational mechanics: Pattern and prediction, structure and simplicity. *J. Stat. Phys.*, 104:817–879, 2001.

[32] Ilya Sutskever. Building AGI, alignment, future models, spies, Microsoft, Taiwan, & enlightenment. Podcast on Dwarkesh Patel Podcast, March 2023. URL `https://www.youtube.com/watch?v=YEUclZdj_Sc`. Accessed: 2024-05-22.

[33] D. R. Upper. *Theory and Algorithms for Hidden Markov Models and Generalized Hidden Markov Models*. PhD thesis, University of California, Berkeley, 1997. Published by University Microfilms Intl, Ann Arbor, Michigan.

A Appendix / supplemental material

A.1 Supplemental Figures

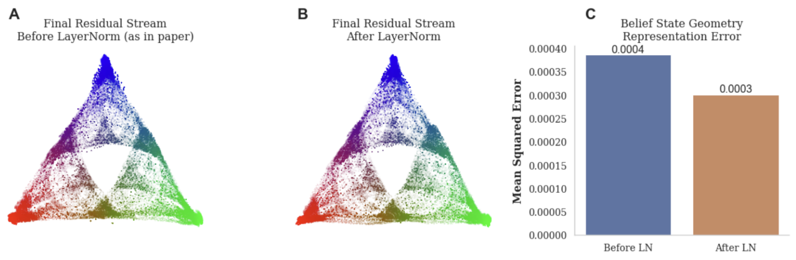

Figure S1: To test the effect of using residual stream activations from before or after the final LayerNorm in our analysis, we compared the belief state geometry representations in both cases. (A) Projection of the final residual stream activations before LayerNorm onto the belief state simplex, as presented in the main paper. (B) The same projection for activations after LayerNorm, showing a qualitatively similar structure. (C) Mean squared error of the linear fit capturing the belief state geometry for both cases. The representation after LayerNorm shows a slightly lower error (0.0003) compared to before LayerNorm (0.0004), indicating that the belief state geometry is preserved and marginally better represented after the LayerNorm operation. These results demonstrate that our findings are robust to the choice of using pre- or post-LayerNorm activations.

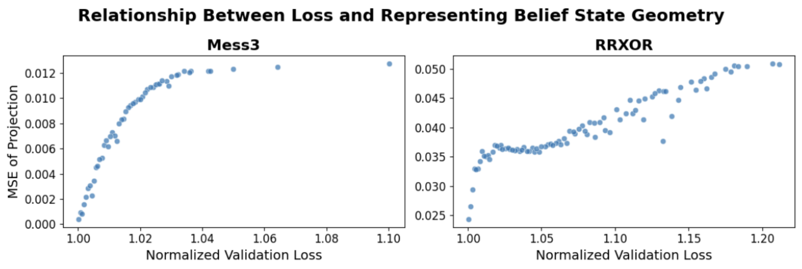

Figure S2: To quantify the relationship between transformer performance on next-token prediction and representing the belief state geometry in the residual stream, we plotted normalized validation loss vs. mean squared error (MSE) of projecting residual stream activations onto the belief state geometry for Mess3 (left) and RRXOR (right) processes. Validation loss is normalized to the theoretical optimal, such that 1.0 characterizes a transformer with optimal loss. As the validation loss decreases (moving left on the x-axis), the MSE of the regression also decreases, indicating that better performance on the next-token prediction task correlates with a more accurate representation of the belief state geometry in the residual stream.

A.2 Relations to Previous Work

A number of connections to previous work in computational mechanics have been mentioned in the Discussion section. Here we will continue the discussion of related works, focusing on connections to neural network research.

Transformer Internal Representations Recent work has significantly advanced our understanding of how and what information is encoded within transformer models. Ref. [25] introduced the "Future Lens" framework, demonstrating that individual hidden states in transformers trained on natural language contain information about multiple future tokens. This finding is consistent with transformers representing belief states, since belief states contain information well beyond the next token. Ref. [9] found that the internal representations of days of the week in LLMs lie in particular geometric arrangements (in a circular pattern). Since our work draws a concrete relation between the structure of data and geometric arrangements inside of transformers, a promising avenue of research would be to explain the days of the week result using our framework. Other work has studied internal representations in the board game Othello, finding a representation of the board state in the residual stream [17, 23]. This finding is interpreted by the authors as the transformers containing a "world model". Conceptually extending our work to such a setting is quite natural. In a sequential game with full knowledge, such as Tic-tac-toe, Othello, and Chess, each board state uniquely determines the future distribution of moves. Thus, the belief states correspond to board states, providing an explanation for why transformers trained on gameplay should contain explicit representations of board states in their residual stream. However, this also makes clear a limitation of our approach. The simplex associated with these belief states would, for any realistic board game, be of larger dimension than the residual stream. If transformers are representing the belief state geometry in those cases, they must be doing so in a compressed fashion. This is an important theoretical problem for future work. Importantly, our work provides a unique model-agnostic point of view to study interpretability.

Behavioral Limitations of Transformers and Chain of Thought A number of papers have studied the limitations of transformers to learn certain algorithmic tasks, such as graph path-finding tasks and HMMs [1, 14]. These papers provide important empirical results that are interesting to think through the perspective of the MSP and belief state geometry. In particular, Ref. [14] provides an upper-bound for the depth of a transformer needed to capture HMM generated data up to a particular context-window length. This makes sense given that transformers are feedforward machines, and the MSP is, in general, an automata containing recurrent components. Thus, what the transformer should be capturing is really an unrolled version of the MSP, which naturally gives an upper bound for how many layers one would need to capture the MSP to a certain depth. This paper also explicitly performs experiments on belief-state inference. In some of their experiments the explicitly train transformers to map sequences of emissions from an HMM to sequences of the associated belief states. In other experiments they train in the normal autoregressive manner solely on the observations, like in our study. Our work provides a nice theoretical connection between the two tasks.

In Ref. [1], the authors test and explain a lack of ability for transformers to perform seemingly simple graph planning tasks. In this work, and others like it that study particular algorithmic tasks with particular formal structures like finite-automata [18], addition, and the like, we think its interesting to think of how our framework would apply. In general, our approach treats the sequential training data as a first class citizen. This makes clear that to relate studies of formal structures (like graph planning) to our framework, one must explicitly consider the particular tokenization used to convert from the formal structure to sequences of tokens. Then the main consideration becomes the computational structure of the sequential nature of the tokens generated by the chosen tokenization scheme (as opposed to the formal structure directly). From our perspective what one learns, and the limitations of being able to perform well on next-token generation given that tokenization, is directly related to the tokenization and not directly to the formal structure that was tokenized. Thus, these results provide important empirical results for our framework to explain, or good tests to find where our framework fails and needs to be amended/expanded.

Meta-learning In Ref. [24], the authors provide a theoretical framework for predictors and agents based on a Bayesian framework which builds state machines of sufficient statistics. This is analogous to the mixed-state presentation we have presented here. Indeed, work in computational mechanics has shown that the states of the MSP correspond not only to beliefs over the generator states, but also to the minimal sufficient statistics for predicting the future based on the past. One exciting area of further research this exposes is the potential for computational mechanics to inform our understanding of artificial RL agents.

A.3 Details of data generating processes used in experiments

The transition matrices for the Mess3 process $T^{(A)}$, $T^{(B)}$, and $T^{(C)}$ are defined as follows:

$$T^{(A)} = \begin{pmatrix} 0.765 & 0.00375 & 0.00375 \\ 0.0425 & 0.0675 & 0.00375 \\ 0.0425 & 0.00375 & 0.0675 \end{pmatrix}, \quad T^{(B)} = \begin{pmatrix} 0.0675 & 0.0425 & 0.00375 \\ 0.00375 & 0.765 & 0.00375 \\ 0.00375 & 0.0425 & 0.0675 \end{pmatrix}, \text{ and}$$

$$T^{(C)} = \begin{pmatrix} 0.0675 & 0.00375 & 0.0425 \\ 0.00375 & 0.0675 & 0.0425 \\ 0.00375 & 0.00375 & 0.765 \end{pmatrix}.$$

The RRXOR process has transition matrices defined as

$$T^{(0)} = \begin{pmatrix} 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad T^{(1)} = \begin{pmatrix} 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & .5 & 0 \\ 0 & 0 & 0 & 0 & .5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where $T^{(0)}$ and $T^{(1)}$ are the transition matrices for emissions 0 and 1, respectively.

A.4 Belief State Simplex

Belief states lie within the probability simplex $\Delta^{|\mathcal{S}|-1}$, defined as:

$$\Delta^{|\mathcal{S}|-1} = \left\{ b \in \mathbb{R}^{|\mathcal{S}|} : \sum_{i=1}^{|\mathcal{S}|} b_i = 1 \text{ and } b_i \geq 0 \text{ for all } i \right\}.$$

A.5 Regression

For the regression procedure used in this paper we used standard linear regression, assuming an affine model for the data and then minimizing:

$$\min_{W,c} \sum_i \|b_i - (Wa_i + c)\|^2$$

where the summation is over all input sequences in the dataset.

A.6 Details of transformer architecture and training

In our experiments, we trained a transformer model using the following hyperparameters and training parameters: The model had a context window size of 10, used ReLU as the activation function, and had a head dimension of 8 and a model dimension of 64. There was 1 attention head in each of 4 layers. The model had MLPs of dimension 256 and used causal attention masking. Layer normalization was applied. For training, we used the Stochastic Gradient Descent (SGD) optimizer with a batch size of 64, running for 1,000,000 epochs, and a learning rate of 0.01 with no weight decay. For each batch we generated 64 sequences from the Mess3 or RRXOR HMM, choosing an initial hidden state from the stationary distribution. We instantiated the networks and performed our analysis using the TransformerLens library [22].

A.7 Details for analysis

For the shuffle and cross validation analyses used in Figure 6, we performed both the shuffle and the cross validation procedure 1000 times over, randomly shuffling or performing the train test split independently 1000 times.

A.8 Reproducibility Instructions

To install the necessary dependencies and the code, follow these steps:

A.8.1 Installation

Navigate to the repository folder and run:

```
pip install -e .
```

A.8.2 Reproducing Figures

To reproduce the figures presented in the paper, step through the provided Jupyter notebooks:

- `Figure6.ipynb`

- `Figure7.ipynb`

These notebooks utilize the saved models located in the `/examples/models` directory.

A.8.3 Training the Models

The models were trained using legacy code found in the following scripts:

- `legacy/epsilon-transformers/run_sweeps_mess3.py`
- `legacy/epsilon-transformers/run_sweeps_RRXOR.py`

These instructions should ensure that the environment is correctly set up and that the figures can be reproduced as described in the paper.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims have to do with the representation of belief state geometry as defined by the mixed-state presentation. Our experiments show that this framework is relevant to transformer neural networks.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include a limitations section in the discussion.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [No]

Justification: Although the paper does contain theory, much of it relies on established results about the structure and limits of prediction. We provide explanations of the theoretical work that allow readers to follow the logic of our experiments and interpret the results. We reference the previous theoretical work that our work is built on.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: We include details for the transformer architecture and training, as well as the HMM structures we used. Additionally we include code that recreates the figures in this work.

Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: We have included code in the submission that reproduces all results. It should be noted that we will continue to work on cleaning up this code for final submission (though the code very much works as is and recreates the figures in the submission).

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We discuss these paramaters in the paper, and also provide code.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: We include explanations of error bars and the statistical methods used.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments Compute Resources**

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

   Answer: [NA]

   Justification: These experiments are on small toy models and thus can be run on any modern hardware.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: I've reviewed the code of ethics and we conform to them.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: Our work is quite theoretical and the experiments are in toy models in highly controlled settings. Thus we don't think our work has implications for societal impact of the kind mentioned in the guidlines.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our work poses no such risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The creators of this work including the code are on this paper. When we use external libraries we cite them.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We include explanations and documentation for the code we submitted.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.