Unsupervised Recovery of Hidden Markov Models from Transformers with Evolutionary Algorithms

Dylan Bowman
dylanbowman314@gmail.com

Colin Lu
2colin.lu@gmail.com

Organized by
Adam Shai, Paul Reichers, PIBBSS.

Abstract

[1]Prior work finds that transformer neural networks trained to mimic the output of a Hidden Markov Model (HMM) embed the optimal Bayesian beliefs for the HMM's current state in their residual stream, which can be recovered via linear regression. In this work, we aim to address the problem of extracting information about the underlying HMM using the residual stream, *without* needing to know the MSP already. To do so, we use the R^2 of the linear regression as a reward signal for evolutionary algorithms, which are deployed to search for the parameters that generated the source HMM. We find that for toy scenarios where the HMM is generated by a small set of latent variables, the R^2 reward signal is remarkably smooth and the evolutionary algorithms succeed in approximately recovering the original HMM. We believe this work constitutes a promising first step towards the ultimate goal of extracting information about the underlying predictive and generative structure of sequences, by analyzing transformers in the wild.

1 Introduction

In this section, we describe the computational mechanics background for this work, and the motivation for the task of finding an MSP from a trained transformer's residual stream activations, without needing to know the HMM beforehand. In the following section, we detail the exploratory analysis to build intuitions for MSPs and derive an important metric for distinguishing the correct generating HMM. Afterwards, we present our evolutionary algorithm approach, along with results, for recovering the approximately correct Mess3 HMM from the transformer residual stream. Finally, we discuss future work.

1.1 Background

A critical class of problems in machine learning is **sequence prediction**, motivated especially in recent years by the next-token text prediction task used for training large language models. A wide variety of other tasks also exist under this paradigm, such as world modeling in reinforcement learning. Within this paradigm of sequence prediction, there lies a rich structure describing the processes of generating and predicting sequences. One major aim of **computational mechanics** is to theoretically characterize and provide insights on these processes.

Hidden Markov Models (HMMs) can *generate* arbitrary sequences. Due to computational mechanics, we know that Mixed State Presentations (MSPs) are models which can *predict* arbitrary sequences, and that prediction is harder than generation (predictive models must keep track of probability

[1]Research conducted at the Computational Mechanics For AI Safety Hackathon, 2024 (https://www.apartresearch.com/event/compmech)

distributions over generating states, and understand how to update those distributions). Recent work for belief state geometry in residual stream [Shai et al., 2024] indicates that the information needed to predict the probability of being in each HMM state is readily available: a *linear* probe is capable of finding these probabilities from the residual stream. This corresponds to an MSP state, visualizable in a fascinating diagram showing the MSP geometry. This strongly suggests that, at least in certain cases, the standard training process causes transformers to innately model the MSP: the predictive structure necessary to keep track of the probabilities of each HMM state.

1.2 Motivation

The aforementioned paper gives an exciting insight: in at least certain cases, transformers seem to do more than just finding conditional probabilities for tokens. The MSP geometry represented in the residual stream indicates that transformers understand the underlying HMM states and how to update on their likelihoods. However, according to Wentworth and Lorell [2024], we do not yet have a mechanism to *extract* this HMM and MSP geometry from the transformer's residual stream. We would like to explore how to do so.

1.3 Problem Setting

For tractability and during the short duration of this hackathon sprint, we consider in particular 3 state HMMs. This has additional benefits in that the MSP belief states are visualizable in 2 dimensions (since 3-outcome probabilities lie on a 2-simplex). We consider specifically Mess3(x, α) for different x and α: this is the class of 3 token, 3 state HMMs from which Shai et al. [2024] analyzed.

We remark that the overall goal is being able to determine approximately what HMM corresponds to the data the transformer was trained on (and whose MSP shows up in the transformer's residual stream), without knowing anything about its topology or number of states. Here we tackle the first stepping stone, where we know that the possible HMMs are Mess3 HMMs, but with any possible parameters. We aim to determine the parameters of the Mess3 generating the data.

2 Exploratory Analysis

Before we can attempt to find a data-generating HMM using the transformer's residual stream activations, we have a few major questions to answer:

1. What kinds of intuitions can we build about MSPs, especially for Mess3(x, α) with varying x and α?
2. If we have a set of HMMs and one of them corresponds to the data the transformer was trained on, can we distinguish it from the others? If so, by what metric?

Armed with answers to these, we can proceed to finding the MSP geometry embedded in the transformer's residual stream in a well-motivated manner.

2.1 Building Intuitions for Mess3 MSPs

Since we are considering Mess3 processes, we first aim to visualize the impacts of changing x and α. Figure 5 from Jurgens and Crutchfield [2021] illustrates the MSP geometry for various $x \in [0, 0.5]$ and $\alpha \in [0, 1]$. We further visualize the MSP geometries by plotting them using the "chaos game" technique for plotting fractals; this can be found in the project repository as MSP_fractal_generation_with_chaos_games.ipynb. An alternative method of plotting the MSP (with colors, but without embedding it with the correct scale into the 2-simplex) can be found in the hackathon repository, in simplex_analysis_mess3_0.05_0.85.ipynb.

There are a few major takeaways about the geometry of the MSPs.

1. As expected, the fractal has 3 components, which are self-similar up to certain transformations (primarily rescaling, rotation, repositioning, though there is some nonlinear distortion). This is marked in Figure 5 of Jurgens and Crutchfield [2021] with the red, green, and blue regions.

2. There are two regimes for α: $\alpha > \frac{1}{3}$ and $\alpha < \frac{1}{3}$. For $\alpha > \frac{1}{3}$, the fractal is closer to a triangle. For $\alpha < \frac{1}{3}$ the fractal is closer to a circular shape, and the regions are "mirrored" (red is on the top left, instead of bottom right). For α farther away from $\frac{1}{3}$ in either direction, the belief states are pushed farther towards the edges, and the states collapse towards the center, closer to $\alpha = \frac{1}{3}$.
3. There are three regimes for x. For a given α, there is a value of x where the components are adjacent: neither separated nor overlapping. For example, this is $x = 0.15$ for $\alpha = 0.6$, and $x = 0.07$ for $\alpha = 0.85$. When x is smaller, this is the "overlapping" regime, and when x is larger but still less than $\frac{1}{3}$, this is the "separated" regime. With $x > \frac{1}{3}$ the fractals take on a very different shape, and are "inverted" in a sense.
4. How spread out the belief states are within the simplex can vary heavily depending on x and α. This is relevant later, and is a factor for why MSE is not a good difference metric but R^2 is mostly suitable.

2.2 R^2 is a Distinguishing Metric

With an intuition on Mess3 MSPs established, we seek to lay the groundwork for determining an HMM (and MSP) corresponding to the sequence a transformer is predicting. Overall, we aim to do so by generating MSPs of candidate HMMs (the choice of generating candidates is important, as a discretized grid search is intractable in higher dimensions; we describe our candidate generation approach in the following section). Using the linear probe technique from Shai et al. [2024], we feed in sequences of tokens to the transformer, extract the sequence's corresponding activations in the residual stream, and fit a linear regression model to the belief probabilities determined by the MSP. The question remains on how to decide whether the HMM truly corresponds to the transformer's data.

First, we investigate the most direct metric, MSE. This is what linear regression minimizes. Applying the above technique to the two transformers provided in the hackathon repository (for $x = 0.15, \alpha = 0.6$ and $x = 0.05, \alpha = 0.85$), we apply the above linear probe technique against 4 candidate MSPs.[2]

Mess3 parameters	Transformer 1	Transformer 2
$x = 0.15, \alpha = 0.6$	3×10^{-5}	3×10^{-4}
$x = 0.05, \alpha = 0.85$	2×10^{-3}	4×10^{-4}
$x = 0.5, \alpha = 0.6$	4×10^{-3}	4×10^{-3}
$x = 0.11, \alpha = 0.21$	2×10^{-4}	3×10^{-4}

In the above table, Transformer 1 is trained on the sequence produced by Mess3$(x = 0.15, \alpha = 0.6)$ and Transformer 2 is trained on Mess3$(x = 0.05, \alpha = 0.85)$. For a given MSP, the corresponding transformer of the two does give the lower MSE (see rows 1 and 2), it is not the case that, for a given transformer, the corresponding MSP gives the lowest MSE among all provided MSPs (see column 2).

This motivates a better metric to distinguish the MSPs. As noted above, we suspect one major reason for MSE's lack of "distinguishing power" is that it does not account for how spread out the points are (consider an MSP with points packed very closely together). To account for this, we instead use R^2, which is effectively MSE but rescaled (and higher is better): $R^2 = 1 - \frac{MSE}{Var(y)}$, where y is the ground-truth data (not predicted targets).

To confirm that R^2 is capable of distinguishing the correct MSP reasonably well, we train the linear probe from transformers 1 and 2 each, on a large number of different MSPs. These MSPs form a grid in the Mess3 parameter space, with x ranging from 0.02 to 0.5 with step size 0.02, and α ranging from 0 to 1 with step size 0.05 (note that $x = 0$ produces a degenerate MSP concentrated on one point and thus produces problems for R^2).

We remark that, at least for the two-dimensional parameter space for x, α for Mess3, R^2 is remarkably unimodal and smooth. It would be interesting to observe whether this holds for the full 24 degree of

[2]For those curious, we also display visualizations of the original MSP and their linear probe fit for each of the two transformers, in the appendix. We also include observations further corroborating the idea that the linear probe preserves the geometry of the residual stream activations with reasonable fidelity.

freedom parameter space for 3-state, 3-token HMMs, and especially for HMMs with more states and larger vocabulary size.

Ultimately, it seems that R^2 is sufficiently capable of distinguishing HMMs that are similar to the HMM which generated the sequences which the transformer trained on, and we proceed to the next step.

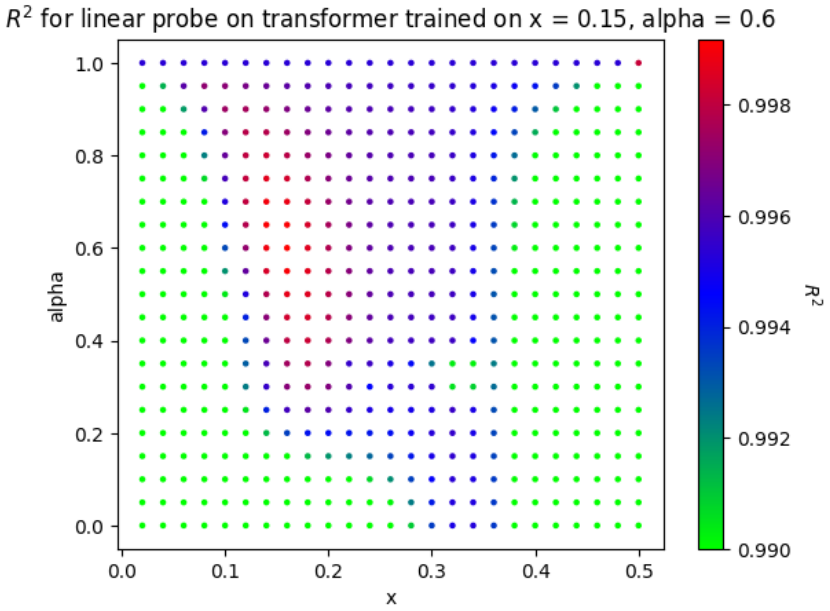

Figure 1: $x = 0.16, \alpha = 0.6$ maximizes R^2 of the linear probe output; it is the closest in the grid to the intended $x = 0.15, \alpha = 0.6$.

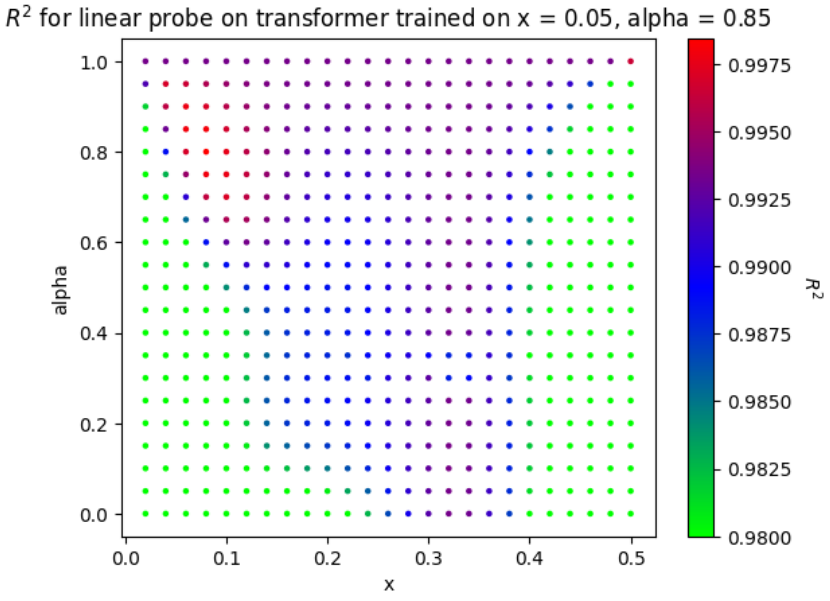

Figure 2: $x = 0.08, \alpha = 0.8$ maximizes R^2 of the linear probe output; it is reasonably close in parameter space (and likely in terms of MSP dynamics and geometry) to the intended $x = 0.05, \alpha = 0.85$. The closest-by $x = 0.06, \alpha = 0.85$ produces the third-largest R^2 value.

3 Evolutionary Algorithms

We use evolutionary algorithms to search the Mess3 parameter space for the pair $(\hat{x}, \hat{\alpha})$ that maximizes R^2 for the regression probe trained with data from Mess3$(\hat{x}, \hat{\alpha})$ on the residual stream from the transformer trained on data from Mess3(x, α). For each new set of parameters, we regenerate the HMM belief states and transformer residual stream tensors, making this process fairly computationally expensive. On an NVIDIA V100 GPU, generating data from 50 new Mess3 processes and training a probe for each process took 5 minutes.

3.1 Search Setup

We wrote a naive evolutionary algorithm that iterates a population of size N through G generations to find the pair of parameters that generates the Mess3 process that is most recoverable from the transformer residual stream. For our analysis, we use $N = 10$, $G = 5$, and crossover_rate $= 0.7$. Each generation, the 2 most fit individuals are carried over to the next. Mutations occur with $P = 0.5$ and sample the mutation from normal distributions with $(\mu = 0, \sigma^2 = \text{range}(x))$ and $(\mu = 0, \sigma^2 = \text{range}(\alpha))$ for x and α, respectively.

Algorithm 1 Naive Evolutionary Algorithm

```
Initialize population 𝒫 with size N
for generation g = 1 to G do
    Evaluate fitness for each individual in 𝒫
    Apply fitness sharing to compute shared fitness
    Select parents 𝒫_parents based on shared fitness using tournament selection
    Initialize offspring population 𝒫_offspring ← {}
    while |𝒫_offspring| < N do
        Select two parents p_1, p_2 ∈ 𝒫_parents
        Perform crossover on p_1 and p_2 to produce children c_1, c_2
        Apply adaptive mutation to c_1 and c_2
        Add c_1 and c_2 to 𝒫_offspring
    end while
    Replace population 𝒫 with 𝒫_offspring using elitism and diversity preservation
end for
Return the best solution in population 𝒫
```

3.2 Results

The evolutionary algorithm was able to find approximately-correct values for x and α using just the R^2 as a reward signal, on both the Mess3$(0.05, 0.85)$ and Mess3$(0.15, 0.6)$ setups. The generated parameters $(\hat{x}, \hat{\alpha})$ had higher R^2 compared to the real parameter values, so the error is due to the lack of granularity with R^2 as a metric, rather than the performance of the evolutionary algorithm.

The computational boost of using an optimization method like our evolutionary algorithm is substantial: in the tables below, we used the grid search over x and α from Section 2.2 to find the values that approximately maximized R^2, which required sampling 525 points (equivalent to 525 iterations). The evolutionary algorithm here was able to find approximately correct parameters with only 50 iterations, marking an 80% improvement in efficiency. The evolutionary algorithm also selected any real number for its parameter iterations rather than multiples of 0.02 and 0.05.

Mess3(0.15, 0.60)	(x, α)	R^2
Parameter found via evo. alg.	$(0.17, 0.53)$	0.9989
Parameter in source HMM	$(0.15, 0.60)$	0.9992
Parameter w/ max. R^2	$(0.16, 0.60)$	0.9992

Mess3(0.05, 0.85)	(x, α)	R^2
Parameter found via evo. alg.	$(0.07, 0.90)$	0.9976
Parameter in source HMM	$(0.05, 0.85)$	0.9966
Parameter w/ max. R^2	$(0.08, 0.8)$	0.9984

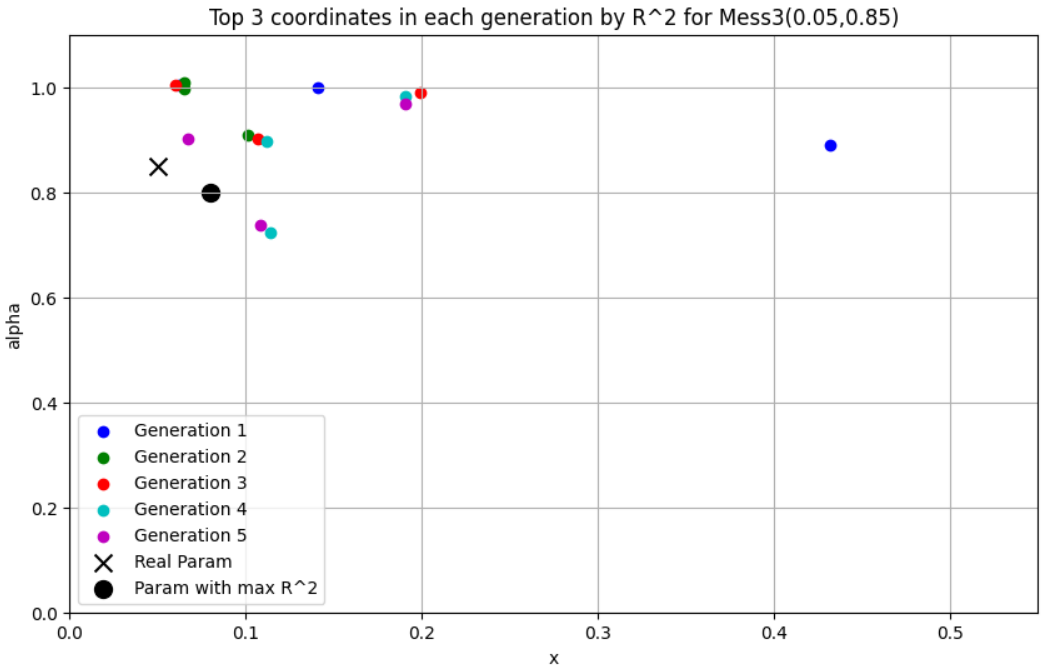

Figure 3: Top 3 points from each evolutionary algorithm generation for Mess3(0.05, 0.85). Later generations trend towards parameters with max R^2.

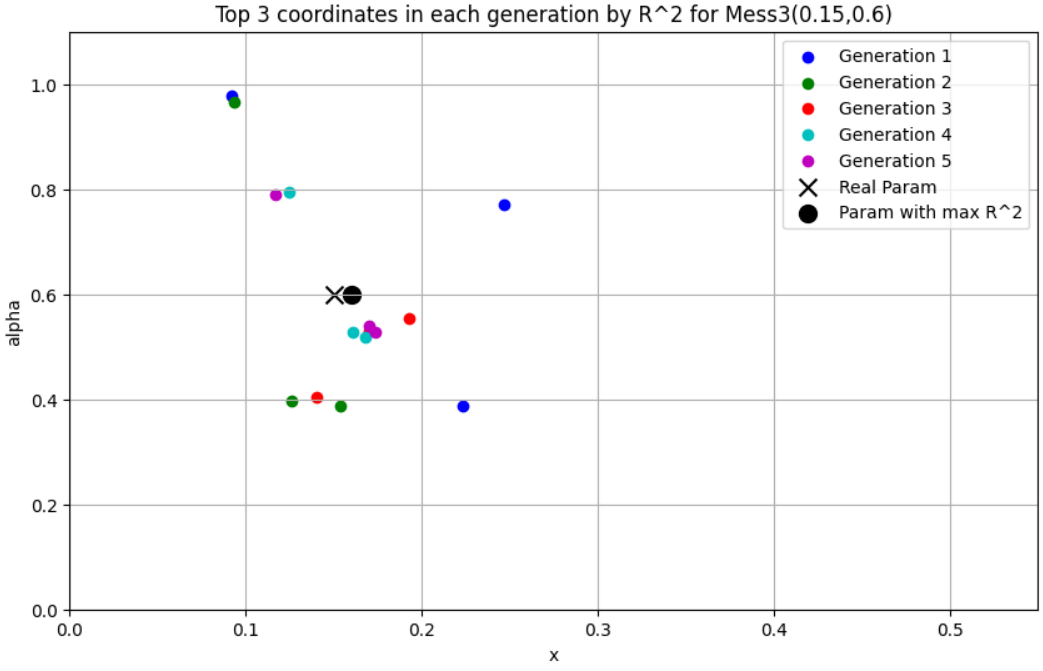

Figure 4: Top 3 points from each evolutionary algorithm generation for Mess3(0.15, 0.6). Later generations trend towards parameters with max R^2.

4 Discussion and Further Work

4.1 Generalize To Unstructured HMMs

One benefit of this analysis was that we knew the HMM was generated by Mess3, so we only had to iterate over the two parameters that generate the HMM. We are unsure but hopeful that our methodology would expand to situations where the structure of the HMM is unknown and the transition/emission probabilities need to be directly searched over. Since the parameter space for unstructured situations would be far greater, the amount of computation required by the search process would also increase. Parallelized algorithms on GPUs would make these experiments more feasible.

4.2 Principled Approaches to Evolutionary Algorithm Search

For this analysis, we deliberately used an out-of-the-box evolutionary algorithm to demonstrate the robustness of the R^2 reward signal. Further analyses could investigate the effectiveness of more specialized algorithms for parameter search.

4.3 Differentiable Reward Signal

In theory, since the optimal Bayesian belief states are a differentiable function of the HMM parameters, and the linear probe training is itself differentiable (e.g., via normal equations), the R^2 reward signal should be differentiable, meaning that we are able to run gradient descent with it. This provides a greater space of optimization algorithms to use for the parameter search process.

References

A. M. Jurgens and J. 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.

A. S. Shai, S. E. Marzen, L. Teixeira, A. G. Oldenziel, and P. M. Riechers. Transformers represent belief state geometry in their residual stream. *arXiv preprint arXiv:2405.15943*, 2024.

J. S. Wentworth and D. Lorell. Why would belief-states have a fractal structure, and why would that matter for interpretability? An explainer. `https://www.lesswrong.com/posts/mBw7nc4ipdyeeEpWs/why-would-belief-states-have-a-fractal-structure-and-why`, 2024. Accessed: 2024-06-03.

Appendix

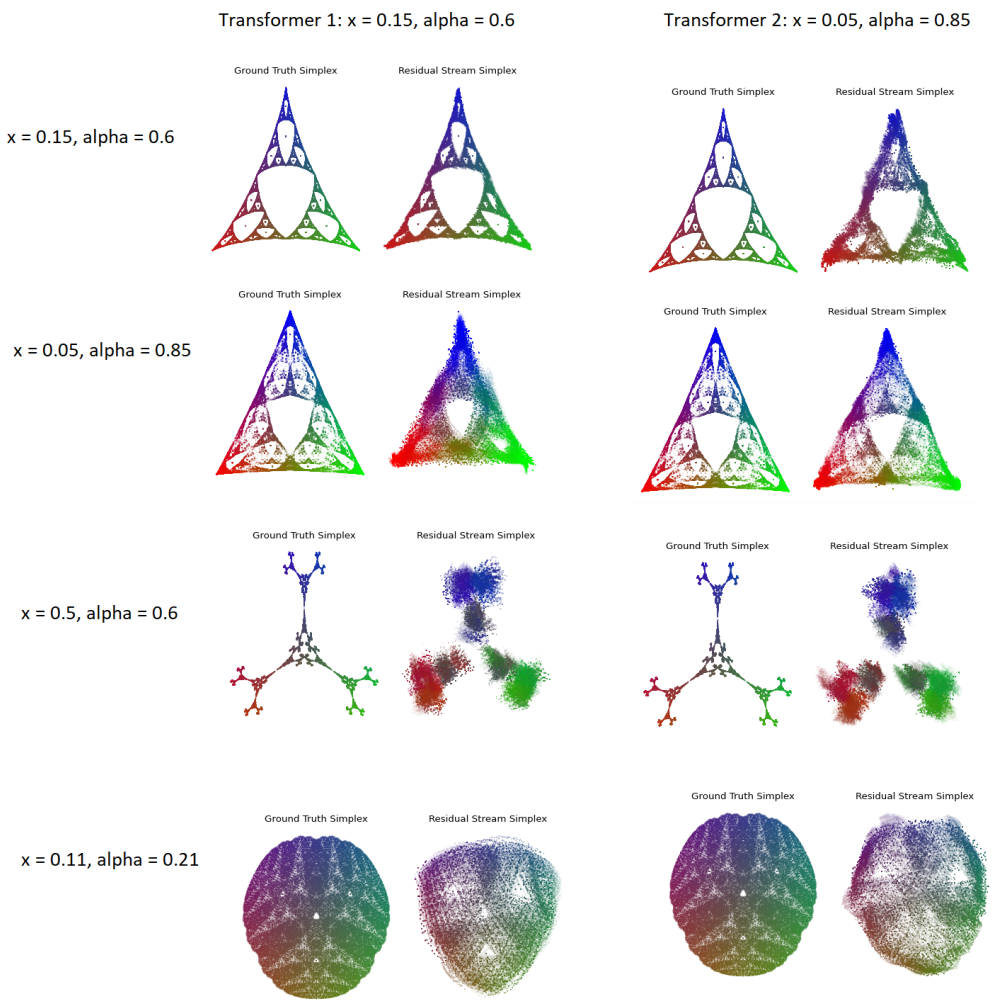

Figure 5: Projections of the residual stream activations of different transformers, when a linear probe is fit to the MSP's belief states. Only column 1, row 1, and column 2, row 2 are mapping the transformer to the MSP of the sequence it was actually trained on. Observe that the geometry looks the most appropriate here (for rows 1 and 2, compare when the transformer is correct versus not). For row 3, the residual stream activations clearly do not contain the appropriate geometries. For the last row, observe especially how there are 4 fairly large holes in transformer 1's projection, which seem to be a part of its residual stream activation geometry which shows up despite the linear probe training.