What Arranges Features in Activation Space? Non-Classical Predictive Geometry in Next-Token Predictors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Mechanistic interpretability often studies the local features and circuits that implement model computations. What principles govern the arrangement of these features and circuits into geometric structures in activation space? To make this tractable, we study how the computational class of the training-data generator constrains the geometry of predictive states. We show that while the data distribution determines which features are required for prediction, a predictor realizes those features as beliefs about its current latent state, and the generator class determines the geometry of those beliefs. Using this theoretical insight, we design synthetic datasets whose minimal predictive representations fall into different model classes, and test which geometry neural networks learn. In particular, we train transformers, LSTMs, GRUs, and vanilla RNNs on datasets whose predictive geometries are known analytically: a classical HMM process, a quantum-realizable process with no finite-state HMM realization, and a generalized-probabilistic process with no finite-dimensional quantum realization. Across architectures, a single affine map from activations decodes the corresponding predictive representation in each case: HMM beliefs in a latent simplex, Bloch-vector quantum states, or a finite-dimensional generalized predictive vector. These representations emerge during training and fit the compact non-classical geometry far better than finite-order classical Markov baselines. These results suggest that understanding predictive representations requires asking not only which features a network represents, but what geometry organizes those features.

1 Introduction

A productive working picture in mechanistic interpretability decomposes network computation into features, taken to be directions or sparse codes in activation space [Elhage et al., 2022, Park et al., 2024, Bricken et al., 2023], and into local circuits among them [Ameisen et al., 2025, Lindsey et al., 2025]. While a feature dictionary can make computation locally intelligible, it does not by itself answer what principles govern the *global* geometric arrangement of features and circuits. This question is typically studied empirically, after decomposition, and the geometry found can be striking. Days of the week, for instance, are encoded as irreducibly multi-dimensional features arranged on circles, with the geometry directly executing the relevant weekly modular arithmetic [Engels et al., 2025].

It seems that the geometry a network arranges its activations into reflects the structure of the computation it supports. From this point of view, local circuits are themselves slices through a broader task geometry. Methods like attribution graphs expose the feature-level pathways supporting a model's output on a particular prompt [Ameisen et al., 2025], while analyses of, e.g., arithmetic

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

require aggregating across the full input domain to reveal modular, magnitude-sensitive structure among number features [Lindsey et al., 2025, Nikankin et al., 2025]. But what arranges the local circuit graphs into such global geometric structure?

Our starting point is that activation geometry is not merely a byproduct of features; it is part of the computation. A sequence of tokens placed in the context window puts the model at a point whose position determines what it predicts and how it should update after the next token is put into the context window. We will see that this point can be understood from several sides: as a data-defined distinction among token sequences, as a belief about the latent data-generating structure, as a belief about the future, as a predictive vector in a linear update rule, or as a neural activation. One of the main conceptual goals of this paper is to make these descriptions understandable as a single geometry of prediction. Empirically, it tests whether trained sequence models actually learn that geometry in their activations.

We make this concrete for next-token prediction. The natural object is the set of distinctions among token sequences that an optimal predictor must preserve, which we call the *process states*. Process states are fixed by the training data, but their arrangement in a vector space is not. That is instead determined by the class of latent generator, the world model that the predictor assumes produces the data. Different generators in different computational classes can produce the same training distribution while implying qualitatively different geometries for how process states sit in space. Our work suggests a conceptual shift. The features and circuits of a sequence model are organized by an underlying *predictive-state geometry*, which characterizes the reachable beliefs about the latent generator. The *shape* of that geometry is set by the generator's computational class, not by the data alone. The global geometry of activations is then the geometry of belief updating itself.

This view lets us design synthetic datasets whose minimal predictive geometry is known analytically and falls into qualitatively distinct classes: classical (a probability simplex), quantum-realizable (the space of density matrices or, equivalently, the extended Bloch ball) and post-quantum (a richer convex set with no finite classical or quantum realization). We train transformers, LSTMs, GRUs, and vanilla RNNs on these processes, and ask whether a single affine map from activations recovers the analytic geometry.

Across architectures, we find that networks recover the compact non-classical geometry rather than approximating it with a classical geometry, including for processes with no finite classical realization. The geometry emerges during training and preserves pairwise relationships among histories. The answer to "what arranges local circuits into global geometric structure," in the case of sequence prediction, is the predictive-state geometry of the data-generating process. Features encode reachable beliefs about the latent generator, and the convex shape of activation space is set by the generator's computational class.

2 Predictive States Beyond the Simplex

2.1 Prediction defines process states

We first define the prediction-relevant features associated with the sequences of tokens that make up a training dataset. Consider a token alphabet $\mathcal{X}$ and a distribution over token sequences $\Pr(X_{1:L})$. A history of tokens, or context, $w \in \mathcal{X}^*$ induces a conditional distribution over future token sequences $\overrightarrow{X}$. Two histories that induce the same conditional distribution over future tokens are indistinguishable for prediction, in the sense that no optimal future-token predictor can require treating them differently. We call the resulting equivalence classes of histories *process states*, following Upper [1997a]. In this sense, process states are the task-level predictive features: the minimal set of distinctions among histories that an optimal predictor must preserve.

These predictive features are defined by the data distribution alone. They are not yet neural-network features in the usual interpretability sense of directions, neurons, or sparse dictionary elements in activation space. Rather, they are abstract equivalence classes of histories, and can be thought of as the computationally relevant states for prediction over the sequences of tokens that make up the data distribution. A model that does well at prediction has to represent them somehow.

To do this, neural networks embed each context into their activation space, and then use that embedding to predict the future. What determines how the process states are embedded? In our

theory below, each process state ends up at a single point in some finite-dimensional space, which we will call its *predictive vector*; the question is what *shape* the cloud of predictive vectors traces out across all contexts. As the next subsections show, the answer depends on the computational class of the system that generated the data. This is because process states map into vector spaces as beliefs about the latent structure of the generator. Different generators, even ones that generate the exact same training data, thus correspond to different predictive geometries. For instance, classical generators[1] confine the cloud to a probability simplex, finite-dimensional quantum generators arrange it on a Bloch ball, and more general computational classes allow still richer convex shapes (Fig. 1). Importantly, the same dataset of token sequences can be generated by different generators in these different model classes, and will often imply different geometries for the way predictive states embed into a vector space. After presenting these theoretical results, we will use them to construct training datasets to determine the implied computational class that is native to neural networks.

2.2 Generators realize process states as predictive vectors

To make the preceding bridge precise, we use *generalized hidden Markov models* (GHMMs) [Upper, 1997b, Fanizza et al., 2024]. A GHMM is a finite-dimensional latent state generator of token sequences whose latent state is updated by each observed token. It gives a concrete belief state for every sequences of tokens and therefore a concrete geometry for the process states. The framework is broad enough to put classical HMMs, finite-dimensional quantum generators, and finite-dimensional generalized-probabilistic generators into the same linear-algebraic language (Fig. 1). This lets us compare the belief geometries induced by different generator classes without changing the basic next-token-prediction task.

A GHMM represents a generator of data, whatever its underlying physical or computational nature, as three pieces: a token alphabet $\mathcal{X}$, an initial-state row vector $\boldsymbol{\eta}_\varnothing$, and one linear operator $T^{(x)}$ per token. The probability of any observed token sequence $w = x_1 \cdots x_\ell$ is computed by chaining the per-token operators and reading out with a fixed column vector $\mathbf{1}$:

$$\Pr(X_{1:\ell} = w) = \boldsymbol{\eta}_\varnothing \, T^{(w)} \, \mathbf{1}, \qquad T^{(w)} = T^{(x_1)} \cdots T^{(x_\ell)}, \tag{1}$$

with $\boldsymbol{\eta}_\varnothing$ normalized so that $\boldsymbol{\eta}_\varnothing \mathbf{1} = 1$. After observing the history w, the generator's updated, renormalized state is the *predictive vector*

$$\boldsymbol{\eta}_w \coloneqq \frac{\boldsymbol{\eta}_\varnothing \, T^{(w)}}{\boldsymbol{\eta}_\varnothing \, T^{(w)} \, \mathbf{1}}. \tag{2}$$

$\boldsymbol{\eta}_w$ is sufficient for prediction: every conditional probability over any continuation of tokens is determined by it. The map $w \mapsto \boldsymbol{\eta}_w$ thus realizes the abstract process states of §2.1 as points in a finite-dimensional belief space. In a standard HMM this is exactly Bayesian updating over a finite set of latent states; in a general GHMM the same update rule plays the same predictive role, but $\boldsymbol{\eta}_w$ need not be a probability distribution. In this way, one can think about latent state updating in a GHMM as a generalized version of Bayesian updating over the latent states of generator, with $\boldsymbol{\eta}_w$ as the current belief. For more details see App. A.

This representation is linear in a prediction-relevant sense. If $\boldsymbol{\eta}_{w''} = c\,\boldsymbol{\eta}_w + c'\,\boldsymbol{\eta}_{w'}$, then for every future string of tokens u,

$$\Pr(u \mid w'') = c \, \Pr(u \mid w) + c' \, \Pr(u \mid w').$$

Linear relations among predictive vectors are therefore linear relations among conditional future distributions. If a single *linear* map from a network's activations to these predictive vectors fits well across many contexts, it can therefore be interpreted as recovering prediction-relevant linear structure. This fact motivates our use of affine probes in §3. [2]

2.3 Three shapes the predictive-vector cloud could take

The remaining question is what kind of geometry the generator's belief states allow. Different generator classes impose different constraints on the latent belief space and therefore on the reachable

[1] in the sense of the paradigm of classical computation, as opposed to e.g. quantum computing. This will be made precise.

[2] The map $w \mapsto \boldsymbol{\eta}_w$ is identifiable only up to a known ambiguity from non-minimal latent coordinates; we quotient these out so that distinct predictive vectors correspond to distinct process states. See App. A.

Paradigm	States of certainty	Memory basis	General beliefs
Classical	finitely many	fixed	stochastic vectors in a simplex
Quantum	continuum	arbitrary	density matrices / generalized Bloch vectors
Post-quantum	extrema of convex structure	arbitrary	belief vectors

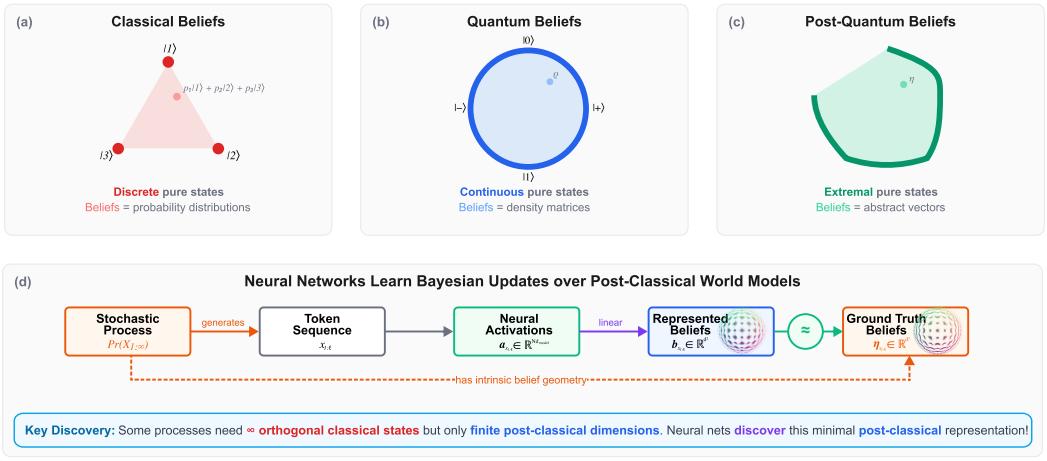

Figure 1: **Generators realize predictive states as reachable belief states whose geometry depends on the generator class. (a)** Classical hidden Markov models constrain predictive states to a probability simplex; pure states are mutually exclusive latents at the corners. **(b)** Finite-dimensional quantum models replace the simplex with the convex set of density matrices, equivalent to a generalized Bloch representation; pure states form a continuous manifold. **(c)** General finite-dimensional probabilistic theories admit broader convex state spaces; pure states are extremal points. **(d)** Experimental setup. A stochastic process generates token sequences. We train a sequence model on next-token prediction and ask whether a single affine map $\mathcal{L}$ from activations recovers the minimal predictive-state geometry of the generator. We test this for processes whose minimal realization falls in each of (a–c).

predictive-vector cloud. We sketch three cases at the level needed to understand the experiments. The important point is that each generator class permits a different convex geometry of belief states [Plávala, 2023, Fanizza et al., 2024]. More detailed explanation is given in App. A.

A **classical belief** is a probability distribution over a finite set of mutually exclusive hidden states. Every reachable predictive vector is a probability vector over the same finite set, so the cloud is confined to a probability simplex with pure states (one-hot distributions) at the corners (Fig. 1a). These generators are mathematically equivalent to HMMs.

A ***quantum* belief** replaces this discrete list with a continuous manifold of latent states. The latent space is the set of density matrices on a d-dimensional Hilbert space, equivalently a generalized Bloch representation; predictive vectors form a convex set whose extremal points trace out a manifold rather than sitting at finitely many corners (Fig. 1b). A ***post-quantum* belief** sits in a still richer convex set: its extremal points need neither form a manifold nor sit at discrete corners; they are simply the extremal points of some finite-dimensional convex body (Fig. 1c). Formal constructions for all three classes, including the density-matrix and generalized Bloch representations, are in App. A, B.

The three classes form a strict hierarchy in expressive power, $\mathcal{C} \subsetneq \mathcal{Q} \subsetneq \mathcal{G}$. There exist classical token sequences with no finite HMM realization but a compact quantum one [Monràs and Winter, 2016], and others with no finite quantum realization but a compact GPT one [Fanizza et al., 2024]. This hierarchy is what makes our experimental design possible: a network whose internal predictive geometry were forced to be a simplex would either spawn unboundedly many corners or settle for a finite-order classical approximation, while a network whose cloud takes the right non-classical shape stays compact. Notably, several existing interpretability findings, such as circular features for days of the week and helical features for modular arithmetic [Engels et al., 2025], already describe non-

simplex predictive geometries observed in trained models on natural data; our controlled processes make the target geometry analytically known.

3 Experimental Design: Probing Predictive Geometry

Our goal is twofold. First, to validate that our theoretical framework accurately captures activation geometry in networks trained on next token prediction, and second to test which class of predictive geometries becomes linearly accessible in neural activations after ordinary next-token training. To make these question well-posed, we use sequence distributions whose minimal predictive-vector geometry is known analytically. For each process, we know the latent generator and can compute the predictive vector $\boldsymbol{\eta}_w$ associated with every context w: the generator's updated belief state after observing that context. We can therefore compare the geometry recovered from neural activations directly to the ground-truth geometry of belief updating over latent generators in different classes.

The networks are not given the generator, its latent states, or the analytic predictive vectors during training. They only see token sequences and are trained with the standard next-token cross-entropy objective. The predictive vectors enter only at analysis time, as probe targets.

We carry out a number of experimental controls in order to rule out alternative explanations. Random-network controls test whether the geometry is an artifact of high-dimensional activations and affine regression, finite-order Markov baselines test whether networks merely learn a bounded-history classical approximation, and a separate one-step-prediction control in §6 tests whether the recovered geometry is exhausted by the current next-token information. Further, cross-architecture comparisons test whether the effect is specific to attention or recurrence.

3.1 Controlled processes with known predictive geometry

We use three main processes, each chosen so that the minimal predictive geometry sits in one of the three classes from §2.3, plus a fourth process used in §6 as a stronger control:

Process	Alphabet	Minimal predictive geometry	Role in experiment
Mess3	3 tokens	Fractal in a 2-simplex (classical)	Sanity check; extends prior belief-state-geometry results to our setting
Bloch Walk	4 tokens	Beliefs in a circle (quantum)	Tests if networks learn a compact quantum geometry rather than a finite classical approximation[3]
Moon	3 tokens	3D generalized-probabilistic geometry (post-quantum)	Tests whether networks learn geometry outside the finite-dimensional quantum class
qRRXOR	2 tokens	Quantum geometry mostly invisible to one-step prediction	Used in §6 as a control that the recovered geometry is not just the current-token logits

For all processes, we use the stationary version of the generator, so the initial predictive vector is the stationary left eigenvector of $T = \sum_x T^{(x)}$. Full details are in App. D.

3.2 Sequence models, training, and analysis

Sequence models and training. We train four standard sequence-model architectures on each process: a 4-layer transformer (implemented in TransformerLens [Nanda and Bloom, 2022]), and 4-layer LSTM, GRU, and vanilla-tanh RNN models. All models are trained on next-token cross-entropy alone. The architectures are intentionally small and matched in width: 8-token context window, 64-dimensional hidden or residual state Optimizer, scheduler, and architecture-specific details are in App. E.

Probes for predictive vectors. For each context w up to the context length, we fit an affine probe $\widehat{\boldsymbol{\eta}}_w = A\,\vec{a}_w + b$ from the concatenated activations $\vec{a}_w \in \mathbb{R}^{N d_{\text{model}}}$ at the final token position to the analytic predictive vector $\boldsymbol{\eta}_w$ by weighted least squares, with weights given by the probability of w under the process. By the linear-superposition property in §2.2, a successful fit recovers prediction-relevant geometry rather than an arbitrary coordinate label. Regularization, cross-validation, and metrics are in App. E.5.

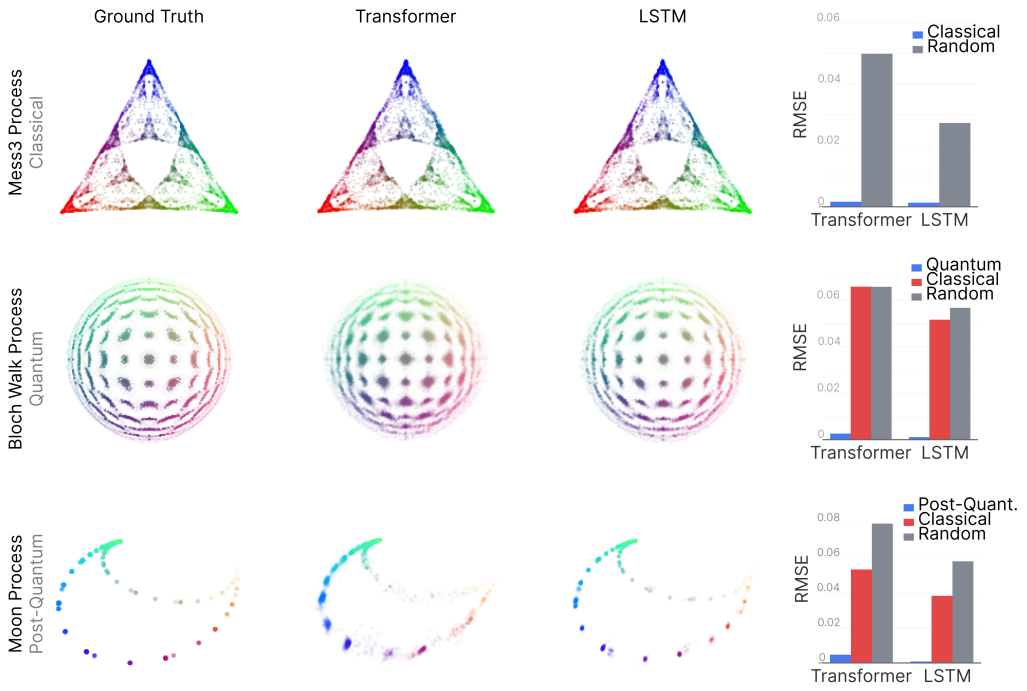

Figure 2: **Neural networks linearly represent the minimal predictive geometry of their training data.** Rows: Mess3 (classical, fractal in 2-simplex), Bloch Walk (quantum, x–z Bloch slice; no finite HMM realization), Moon (post-quantum; no finite-dimensional quantum realization Fanizza et al. [2024]). For each process: ground-truth predictive geometry, affine probe fit from transformer activations, affine probe fit from LSTM activations, and RMSE bars for the minimal generator (blue), Markov-order-3 classical approximation (red), and random-init control (gray). Points are colored by ground-truth position and weighted by occurrence probability.

Baselines. We use two baselines, each designed to rule out a specific alternative explanation. First, we apply the same probing pipeline to randomly initialized models. This tests whether the geometry we recover is a consequence of training, rather than of high-dimensional activations passed through a flexible affine map. Second, for the non-classical processes, we compare the fit to the compact predictive geometry against the fit to the belief states of a finite-order classical Markov approximation. We use order 3 because, for the Bloch Walk process, the resulting classical belief state has 64 dimensions, which is exactly the hidden width of our models. The classical baseline is therefore not disadvantaged by dimensionality. If networks merely learn a finite-history classical approximation, this baseline should fit well; if they learn the compact non-classical predictive geometry, the non-classical target should fit better.

4 Neural Networks Recover Predictive-State Geometry

We now ask which predictive geometry is linearly represented in the activations of trained sequence models. Figure 2 shows the main result. Across classical, quantum-realizable, and generalized-probabilistic processes, neural activations linearly recover the compact predictive-state geometry induced by the minimal generator. The recovered geometries match the ground-truth point clouds, preserve their color gradients, and achieve low RMSE relative to both random networks and finite-order classical baselines.

4.1 Networks recover the compact generator-induced geometry

Across all three processes, a single affine map from concatenated activations recovers the compact predictive geometry of the minimal generator: the fractal in the 2-simplex for Mess3, the x–z Bloch slice for Bloch Walk (which has no finite classical HMM realization Monràs and Winter

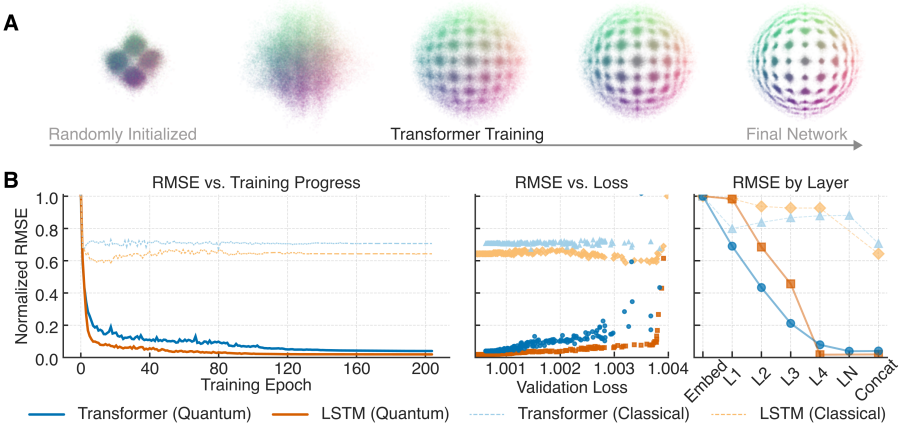

Figure 3: **Predictive-state geometry emerges over training and tracks the minimal quantum generator, not its classical approximation.** (A) Activations-to-Bloch-slice probe fit at successive checkpoints during training of a transformer on Bloch Walk; the affine map is refit at every checkpoint. (B) Normalized RMSE (relative to random-init RMSE) for transformer (blue) and LSTM (orange) against the true quantum belief geometry (solid) and the Markov-order-3 classical approximation (dashed), as a function of training epoch (left), validation loss (center), and network layer (right; L1–L4, LN=LayerNorm, Concat=all layers concatenated).

[2016]), and the curved post-quantum manifold for Moon (which has no finite-dimensional quantum realization Fanizza et al. [2024]). RMSEs are an order of magnitude below random-init baselines (Fig. 2, gray bars), and the matching color gradients show the probe is recovering the organization of predictive states across contexts rather than fitting per-context labels; pairwise geometry preservation is quantified in App. F.

The Markov-order-3 baseline rules out the natural classical alternative. For Bloch Walk this baseline has 64 states, matching the hidden width of the networks; for Moon it is lower-dimensional than the hidden state. In neither case is it disadvantaged by capacity. Yet activations fit the compact non-classical geometry far better than this baseline (Fig. 2, red bars), so what the networks recover is not a finite-history classical approximation that happens to be linearly accessible.

The same pattern holds across all architectures (transformer, LSTM, GRU, vanilla RNN; results in App. G), making attention or any specific recurrent mechanism unlikely to be the cause. The common factor is next-token prediction on a process with a particular predictive geometry.

5 Geometry Emerges During Training

The geometry is not present at initialization. Refitting the affine map at each checkpoint during Bloch Walk training, the mapped activations begin as an unstructured cloud and organize into the target Bloch slice over training (Fig. 3A). Normalized RMSE against the quantum-realizable geometry drops by more than 80% in the first 20–40 epochs and tracks validation loss; the fit to the Markov-order-3 classical baseline does not improve in the same way (Fig. 3B). Training is selectively making the compact non-classical geometry linearly accessible, not making activations more probeable in every coordinate system. The fit also strengthens with depth, and is best at the concatenation of all layers, consistent with later layers exposing the predictive state more cleanly, though our probes do not identify the computational mechanism.

6 Beyond One-Step Prediction

A trained next-token predictor must represent enough information to set the current next-token probabilities, and those probabilities are themselves a linear function of the predictive vector. So an affine probe could in principle look successful just by decoding next-token logits, without representing

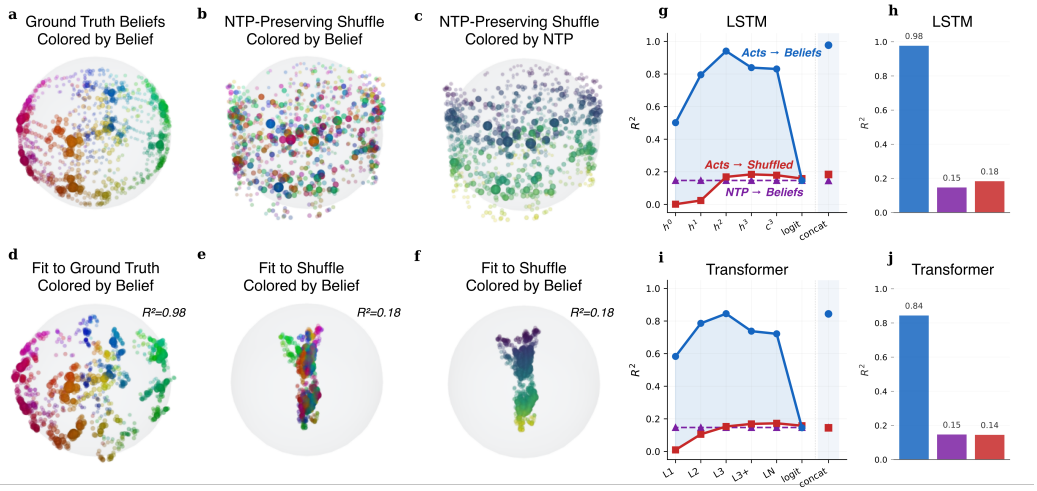

Figure 4: **The learned predictive-state geometry is not exhausted by one-step prediction.** We use the quantum-RRXOR process or which next-token probabilities explain only $R^2 = 0.15$ of the predictive-vector variance; 85% of the geometry is invisible to local next-token readout. **(a–c)** Ground-truth predictive vectors and the next-token-probability (NTP)-preserving shuffle, shown in the same PCA-3 basis fit on the true vectors. Identity colors in (a,b) show that the shuffle scrambles the point geometry, while the coloring in (c) shows that the smooth $p(x_{t+1} = 0)$ gradient is preserved. **(d–f)** LSTM activations recover the true predictive geometry ($R^2 = 0.98$) but not the shuffled orthogonal geometry ($R^2 = 0.17$), aside from the preserved next-token component. **(g,h)** Across LSTM layers, the activation-to-belief fit strongly exceeds both the activation-to-shuffle fit and the next-token-only ceiling; at CONCAT-h, the gap above the ceiling is 0.83, corresponding to roughly 98% of the next-token-invisible belief variance. **(i,j)** The transformer shows the same dissociation: CONCAT-resid activations recover the true predictive vectors ($R^2 = 0.84$), corresponding to roughly 81% of the next-token-invisible belief variance, while the shuffled target remains near the next-token ceiling ($R^2 \approx 0.14$). Points are weighted by the exact prefix probability over all 2^{10} contexts; regressions use the same PCA-plus-affine-probe pipeline as Fig. 2.

any of the predictive structure relevant for future updates. We test this with a process designed so that most of the predictive geometry is invisible to the one-step readout.

The process is qRRXOR (App. D), a binary process for which an affine map from predictive vectors to next-token probabilities explains only $R^2 = 0.15$ of the weighted variance; 85% of the geometry lies in directions that do not affect the current logits.

This lets us build a targeted shuffle. Decompose each predictive vector η into the part determined by its next-token distribution $E^\top \eta$ and the orthogonal residual. The *NTP-preserving shuffle* keeps $E^\top \eta$ fixed for every history and randomly permutes the residuals across histories. The shuffled targets assign exactly the same next-token probabilities as the true predictive vectors, and even preserve the marginal distribution of the residuals; what they destroy is the assignment of residuals to histories. If activations only carry next-token information, the true and shuffled targets should be about equally recoverable, capped near $R^2 = 0.15$. If activations carry the full predictive state, the true targets should fit well above this ceiling and the shuffle should not.

The dissociation is clean (Fig. 4). LSTM activations recover the true qRRXOR geometry with $R^2 = 0.98$ and the shuffled target with $R^2 = 0.17$, near the next-token ceiling. The transformer shows the same effect somewhat more weakly: $R^2 = 0.84$ on the true target, ≈ 0.14 on the shuffle. Roughly 98% (LSTM) and 81% (transformer) of the next-token-invisible belief variance is linearly accessible in activations. Next-token training thus produces representations that linearly carry update-relevant predictive structure beyond what the current logits require.

7 Discussion

We view the central conceptual contribution of this paper as identifying predictive geometry as a global organizing principle for the local features and circuits studied in mechanistic interpretability. Much mechanistic work decomposes activations into features and traces the circuits that compose them. Our results suggest a complementary level of description: prompt-level circuits are local slices through a broader geometry of prediction. They show how the model moves among prediction-relevant representations for one input, while the full geometry describes how such moves are organized across possible contexts.

The points in this geometry are beliefs in a strictly operational sense: the update-relevant information about the latent structure generating the observations, sufficient to predict future tokens and update correctly after the next observation. The GHMM framework makes this precise. The same token-labeled operators define the training distribution, their normalized products define the updated belief after each context, and the constraints placed on latent states and operators distinguish classical, quantum, and more general probabilistic generators. Data distribution, optimal inference, computational class, and activation geometry become different views of the same structure.

What we present here is the starting point of a research plan to understand the nature and structure of activations. In this work we focus on a minimal belief geometry, fixed by the generator's computational class, and traced out by unconstrained Bayesian-style updates. We do not expect that real networks trained on complex data realize this form exactly. Even today, it is already know that predictive states can be assembled compositionally from factored beliefs whose geometries combine multiplicatively [Shai et al., 2026], and that the updates a network actually performs are typically constrained, for instance by the form of the attention mechanism, yielding characteristic distortions of the underlying belief geometry that are themselves informative [Piotrowski et al., 2025]. We expect the road from this platonic form to the geometry of real trained networks to be intricate, and to surface structure we cannot yet anticipate — about how beliefs compose, how computation bends them, and ultimately about what it means for a network to model the world.

The non-classical generators are stress tests for the framework, not claims about the substrate of the network. Ordinary real-valued sequence models trained on token sequences recover analytically specified non-simplex geometries far better than finite-order classical alternatives. This changes how one should think about features and circuits. Features can be useful coordinates, measurements, or local directions on a predictive geometry, but they need not be the whole object. A curved or continuous geometry can be covered by many feature directions without being reducible to a finite list of independent latent cases. Likewise, a circuit should not only be described by which features it reads and writes. It can also be described by what update it implements on the predictive geometry. Each observed token moves the model from one belief to another; attention heads, MLPs, recurrent gates, and residual-stream operations are candidate mechanisms for implementing these moves.

This perspective also changes the interpretation of non-orthogonality and sparse decompositions. Non-orthogonal directions are often treated as evidence of superposition, interference, or imperfect feature separation. Those phenomena are real, but sometimes non-orthogonality is simply the correct geometry of prediction. Similarly, sparse autoencoders and related methods may provide useful local coordinate systems, but a sparse dictionary should not automatically be identified with the model's predictive ontology. A circle, a Bloch slice, or a more general convex geometry can be locally parameterized by many directions while the global object being updated remains invariant.

There are important limitations, and they point directly to the broader program. Affine probes show that the predictive geometry is linearly accessible; they do not prove that the probed coordinates are the unique causal variables used by the model. The natural next step is intervention: move activations along recovered belief coordinates and test whether future predictions change as the theory predicts. Our processes are also small, stationary, and analytically specified, which makes the experiments clean but leaves open how the picture scales to natural language. Finally, we do not yet identify the mechanisms that compute the updates. The goal is therefore not to replace feature and circuit analysis, but to put it in context: to understand not only which features a predictor represents, but what geometry those features live on, what beliefs correspond to points in that geometry, and what circuits move the model through it.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

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. URL https://transformer-circuits.pub/2023/monosemantic-features.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=d63a4AM4hb.

Marco Fanizza, Josep Lumbreras, and Andreas Winter. Quantum theory in finite dimension cannot explain every general process with finite memory. *Communications in Mathematical Physics*, 405 (2):50, 2024. doi: 10.1007/s00220-023-04913-4.

Jerryman A Gyamfi. Fundamentals of quantum mechanics in Liouville space. *European Journal of Physics*, 41(6):063002, 2020.

L. Jakóbczyk and M. Siennicki. Geometry of Bloch vectors in two-qubit system. *Physics Letters A*, 286(6):383–390, 2001.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

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.

Alex Monràs and Andreas Winter. Quantum learning of classical stochastic processes: The completely positive realization problem. *Journal of Mathematical Physics*, 57(1):015219, 01 2016. ISSN 0022-2488. doi: 10.1063/1.4936935. URL https://doi.org/10.1063/1.4936935.

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

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic Without Algorithms: Language Models Solve Math With a Bag of Heuristics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39643–39666. PMLR, 2024.

Mateusz Piotrowski, Paul M. Riechers, Daniel Filan, and Adam S. Shai. Constrained belief updates explain geometric structures in transformer representations. *ICML*, 2025.

Martin Plávala. General probabilistic theories: An introduction. *Physics Reports*, 1033:1–64, 2023. ISSN 0370-1573. doi: 10.1016/j.physrep.2023.09.001.

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.

Paul M. Riechers and Thomas J. Elliott. Identifiability and minimality bounds of quantum and post-quantum models of classical stochastic processes. *arXiv*, 2025.

Paul M. Riechers, Chaitanya Gupta, Artemy Kolchinsky, and Mile Gu. Thermodynamically ideal quantum state inputs to any device. *PRX Quantum*, 5:030318, Jul 2024. doi: 10.1103/PRXQuantum.5.030318. URL `https://link.aps.org/doi/10.1103/PRXQuantum.5.030318`.

Adam Shai, Loren Amdahl-Culleton, Casper L Christensen, Henry R Bigelow, Fernando E Rosas, Alexander B Boyd, Eric A Alt, Kyle J Ray, and Paul M Riechers. Transformers learn factored representations. *arXiv preprint arXiv:2602.02385*, 2026.

Adam S. Shai, Sarah E. Marzen, Lucas Teixeira, Alexander Gietelink Oldenziel, and Paul M. Riechers. Transformers represent belief state geometry in their residual stream. *NeurIPS, arXiv:2405.15943*, 2024.

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

Daniel Ray Upper. *Theory and Algorithms for Hidden Markov Models and Generalized Hidden Markov Models*. Ph.d. dissertation, University of California, Berkeley, Berkeley, CA, February 1997b.

A Classical, Quantum, and Post-Quantum States: Beliefs Beyond the Simplex

While the belief states of a classical model live in a probability simplex (a type of polytope), belief states over quantum and post-quantum states can live in more general cross sections of generalized cones that can't be described by a finite number of vertices. The contrast between classical and quantum memory states offers an easy analogy. A classical bit has only two pure states—zero and one. If we have some uncertainty about its state, then it can be in a probabilistic mixture of these two states—a convex combination that lives on the one-simplex. In contrast, a quantum bit (i.e., a 'qubit') has an uncountably infinite number of distinct pure states $\{|\psi\rangle\langle\psi| : |\psi\rangle = c_0|0\rangle + c_1|1\rangle;\ |c_0|^2 + |c_1|^2 = 1;\ c_0, c_1 \in \mathbb{C};\ \langle\psi| = |\psi\rangle^\dagger\}$, typically represented as the surface of the Bloch sphere—note that while $|\psi\rangle$ is a linear combination of $|0\rangle$ and $|1\rangle$, the state $|\psi\rangle\langle\psi|$ as a rank-1 operator of trace 1 is nevertheless not a convex combination of any other pure states (which are all restricted to rank-1 operators of trace 1). Meanwhile the interior of the Bloch ball corresponds to all possible distinct non-pure mixed states of a qubit—convex combinations of pure states (although any non-pure quantum mixed state has infinitely many different convex combinations that lead to it). More generally, pure states are the extremal states that cannot be obtained by convex combinations of other states.

HMM conventions. The HMM definition we use—where each $T^{(x)}$ encodes a joint transition-and-emission probability $T^{(x)}_{s,s'} = \Pr(s', x \mid s)$—is a Mealy HMM. The more common Moore HMM, where each latent state carries an emission distribution over tokens, is class-equivalent: a process has a finite Mealy HMM if and only if it has a finite Moore HMM Riechers and Crutchfield [2018]. The restriction to a finite number of states is necessary for the classical/quantum/post-quantum distinction to be nontrivial: any one-sided classical stochastic process can be described by an infinite-state HMM.

Minimality and the predictive-vector geometry. Different histories can induce the same predictive vector, in which case they form an equivalence class with no distinguishable effect on the future. For a non-minimal GHMM, the converse can fail in a more subtle way: predictive vectors $\langle\langle\eta|$ and $\langle\langle\eta'|$ may differ as elements of the latent space yet act identically on the span of reachable future operators $\mathcal{F} \coloneqq \mathrm{span}\big(\{T^{(w)}|1\rangle\rangle\}_{w\in\mathcal{X}^*}\big)$, since $(\langle\langle\eta| - \langle\langle\eta'|)|f\rangle\rangle = 0$ for all $|f\rangle\rangle \in \mathcal{F}$. Quotienting out this nullspace yields the minimal set of maximally predictive features—the predictive-state geometry whose recovery from neural activations is the object of our experiments. The metadynamic of prediction lives in the space orthogonal to $|1\rangle\rangle$, since $(\langle\langle\eta| - \langle\langle\eta'|)|1\rangle\rangle = 0$ for any two normalized predictive vectors. This is why the probe results of §4 are non-trivial: a network whose activations linearly recover the minimal predictive geometry is recovering structure forced by the data distribution itself, not an artifact of any particular realization.

B Quantum Representations: From Density Matrices and Kraus Operators to GHMMs

B.1 (Transposed) Liouville-space representation

We obtain the transpose of the standard Liouville-space representation Gyamfi [2020] via the linear invertible ket-flipper map $\mho$, such that $\mho(c|\alpha\rangle\langle\beta|) = c|\alpha\rangle^\top \otimes \langle\beta| = \langle\alpha|^* \otimes c\langle\beta|$ for all $c \in \mathbb{C}$ and for any two vectors in the Hilbert space $|\alpha\rangle, |\beta\rangle \in \mathcal{H}$, where $\langle\alpha| = |\alpha\rangle^\dagger$ and $(\cdot)^*$ denotes complex conjugation. In this case, our GHMM is constructed from the initial vector $\langle\langle\eta^{(\varnothing)}| = \mho(\rho_0)$ and from the transition operators $T^{(x)} = \sum_y K_{x,y}^\top \otimes K_{x,y}^\dagger$. The resulting net transition operator $T = \sum_{x\in\mathcal{X}} T^{(x)}$ has the stationary right eigenstate $|1\rangle\rangle = \sum_\zeta |\zeta\rangle^* \otimes |\zeta\rangle$ for any orthonormal basis $\{|\zeta\rangle\}_\zeta$ such that $\langle\zeta'|\zeta\rangle = \delta_{\zeta',\zeta}$, corresponding to the standard trace functional of quantum mechanics.

B.2 Generalized Bloch representation of density matrices

It is well known that the state of a qubit ρ can be expressed via its Bloch vector $\vec{a}$:

$$\rho = I/2 + \vec{a}\cdot\vec{\sigma}/2\,, \tag{3}$$

where $\vec{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$ is the vector of Pauli matrices. For a quantum system of arbitrary finite dimension—i.e., a qudit ρ acting on a d-dimensional vector space $\mathcal{V}_d$—we achieve something similar via a generalized Bloch decomposition Jakóbczyk and Siennicki [2001], Riechers et al. [2024]. We choose any complete basis $(I/d, \Gamma_1, \Gamma_2, \ldots \Gamma_{d^2-1})$ for linear operators acting on $\mathcal{V}_d$, such that the Hermitian operators Γ_n are all traceless and mutually orthogonal, satisfying

$$\mathrm{tr}(\Gamma_n) = 0, \quad \text{and} \tag{4a}$$

$$\mathrm{tr}(\Gamma_m\Gamma_n) = \xi\,\delta_{m,n}\,, \tag{4b}$$

where we will choose the normalizing constant to be $\xi = \frac{d-1}{d}$. Any density matrix then has a unique decomposition in the operator basis $\vec{\Gamma} = (\Gamma_1, \Gamma_2, \ldots \Gamma_{d^2-1})$, described by the *generalized Bloch vector* $\vec{b} \in \mathbb{R}^{d^2-1}$ via

$$\rho = I/d + \vec{b}\cdot\vec{\Gamma} \tag{5}$$

$$= \Big(\big[1\ \vec{b}\big] \otimes I\Big) \begin{bmatrix} I/d \\ \Gamma_1 \\ \vdots \\ \Gamma_{d^2-1} \end{bmatrix}\,, \tag{6}$$

where each "I" above is the d-dimensional identity matrix. Density matrices are thus a linear function of their d^2-dimensional *extended Bloch vector* $\big[1\ \vec{b}\big]$. This linear function is determined uniquely from the choice of operator basis. Conversely, given any Hermitian operator $M = cI/d + \vec{b}\cdot\vec{\Gamma}$, its extended Bloch vector $\big[c\ \vec{b}\big]$ can be obtained via

$$c = \mathrm{tr}(M) \quad \text{and} \quad \vec{b} = \mathrm{tr}(M\vec{\Gamma})/\xi\,. \tag{7}$$

Since the magnitude of the Bloch vector is $b = \sqrt{\frac{\mathrm{tr}(\rho^2)d-1}{d-1}}$, the density matrix represents a pure state iff the magnitude of its corresponding Bloch vector is one. For $d > 2$, not all points in the Bloch ball correspond to physical states, but the set of all physical states is nevertheless a convex set—the convex hull of the pure states, which all lie on a $2(d-1)$-dimensional submanifold of the (d^2-2)-dimensional surface of the Bloch sphere Jakóbczyk and Siennicki [2001].

We recover the standard Bloch vector representation in the familiar two-dimensional case of a qubit when $\vec{\Gamma} = \vec{\sigma}/2 = (\sigma_x/2,\, \sigma_y/2,\, \sigma_z/2)$ and $\xi = 1/2$.

B.3 Generalized Bloch representation of subchannels

If we marginalize over measurements, the memory goes through a completely positive and trace preserving (CPTP) map. However, because of the measurement, the CPTP map has $|\mathcal{X}|$ subchannels, each a trace-non-preserving superoperator A_x on the density matrix. By linearity, each of these subchannels can be fully determined by measuring the output from d^2 independent inputs to the subchannel. Moreover, each subchannel has a generalized Bloch representation that can be readily constructed from such experiments.

Given access to the subchannels $(A_x)_{x\in\mathcal{X}}$ (either directly or via post-selection), and d^2 linearly independent input memory states $(\rho_{(n)})_{n=1}^{d^2}$, we can build up the Bloch representation of the process itself. Notice that each subchannel A_x is a linear operator on a density matrix, which is in turn a linear function of its extended Bloch vector. Accordingly, each subchannel can just as well be represented as a linear operator $G^{(x)}$ acting on the extended Bloch vector:

$$\left[1\ \vec{b}_n\right] G^{(x)} = \left[c_n\ \vec{b}'_n\right] , \tag{8}$$

where $\left[1\ \vec{b}_n\right]$ is the extended Bloch vector of $\rho_{(n)}$, and $\left[c_n\ \vec{b}'_n\right]$ is the extended Bloch vector of $A_x(\rho_{(n)}) = \sum_y K_{x,y}\rho_{(n)}K_{x,y}^\dagger$, with $c_n = \mathrm{tr}\left[A_x(\rho_{(n)})\right]$ and $\vec{b}'_n = \mathrm{tr}\left[A_x(\rho_{(n)})\vec{\Gamma}\right]/\xi$.

Stacking these equations for the d^2 linearly independent memory inputs

$$\underbrace{\begin{bmatrix} 1 & \vec{b}_1 \\ 1 & \vec{b}_2 \\ \vdots & \vdots \\ 1 & \vec{b}_{d^2} \end{bmatrix}}_{=:B} G^{(x)} = \underbrace{\begin{bmatrix} c_1 & \vec{b}'_1 \\ c_2 & \vec{b}'_2 \\ \vdots & \vdots \\ c_{d^2} & \vec{b}'_{d^2} \end{bmatrix}}_{=:B'} , \tag{9}$$

and recording the new extended Bloch matrices B and B', allows us to directly construct $G^{(x)}$:

$$G^{(x)} = B^{-1}B' . \tag{10}$$

Notice that B is always invertible, since we have insisted on d^2 linearly independent input states (which is a generic outcome of selecting d^2 such matrices at random).

This construction directly yields a d^2-dimensional GHMM-like representation of the stochastic process. We obtain the net transition operator $G = \sum_{x\in\mathcal{X}} G^{(x)}$, and the stationary vectors of the process $\langle\langle\boldsymbol{\pi}| = \langle\langle\boldsymbol{\pi}|G = \left[1\ \vec{b}_{\boldsymbol{\pi}}\right]$ and $|1\rangle\rangle = G|1\rangle\rangle = \left[1\ 0\ \ldots\ 0\right]^\top$.

Just as in Eq. (1), stationary probabilities are obtained via

$$\Pr(X_{1:L} = x_{1:L}) = \langle\langle\boldsymbol{\pi}|G^{(x_{1:L})}|1\rangle\rangle , \tag{11}$$

with $G^{(x_{1:L})} = G^{(x_1)}\cdots G^{(x_L)}$.

C Shared Properties Across Quantum Representations

How unique are the generalized Bloch representations of a process? Indeed, other representations can be easily obtained by changing the Hermitian operator basis. Belief geometry associated with minimal generative models is nevertheless unique up to linear transformation of the space.

More generally, different representations will all have the same spectral properties on their non-zero eigenspaces, so long as those eigenspaces are not in the nullspace of the history and future spaces. The belief geometry will be unique up to a linear transformation—and so the geometric relationship among beliefs will be qualitatively preserved across representations.

When a single Kraus operator is associated with an observation x—i.e., when $\sum_y K_{x,y} \otimes K^*_{x,y} = K_x \otimes K^*_x$—then the spectral properties of the transition operator $T^{(x)}$ are inherited from the spectral properties of the Kraus operator K_x. From the properties of the tensor product, it immediately follows from the Liouville-space representation that the eigenvalues of the transition operators $T^{(x)}$ are all the possible products of the eigenvalues Λ_{K_x} of the corresponding Kraus operator and its complex conjugates: $\Lambda_{T^{(x)}} = \bigcup_{\lambda,\zeta \in \Lambda_{K_x}} \{\lambda^* \zeta\}$. Moreover, the eigenvectors of $T^{(x)}$ can similarly be composed from the tensor products of the K_x left and right eigenvectors with other complex-conjugated left and right eigenvectors of K_x.

D Process Definitions

D.1 Classical: Mess3 process

The Mess3 process Marzen and Crutchfield [2017], Shai et al. [2024], Piotrowski et al. [2025] has three hidden states $\boldsymbol{\mathcal{S}} = \{1, 2, 3\}$, and three observable tokens $\mathcal{X} = \{a, b, c\}$.

The process is defined by two parameters, α and x, with dependent quantities $\beta = (1 - \alpha)/2$ and $y = 1 - 2x$. For experiments we used $x = 0.05, \alpha = 0.85$.

The labeled transition matrices are:

$$T^{(a)} = \begin{bmatrix} \alpha y & \beta x & \beta x \\ \alpha x & \beta y & \beta x \\ \alpha x & \beta x & \beta y \end{bmatrix} \tag{12}$$

$$T^{(b)} = \begin{bmatrix} \beta y & \alpha x & \beta x \\ \beta x & \alpha y & \beta x \\ \beta x & \alpha x & \beta y \end{bmatrix} \tag{13}$$

$$T^{(c)} = \begin{bmatrix} \beta y & \beta x & \alpha x \\ \beta x & \beta y & \alpha x \\ \beta x & \beta x & \alpha y \end{bmatrix} . \tag{14}$$

D.2 Quantum: Bloch Walk process

Here we introduce the Bloch Walk process, a probability density over sequences of tokens $\mathcal{X} = \{0, 1, 2, 3\}$, which can be generated with a single qubit of quantum memory but has no finite HMM representation.

We find that neural networks trained on this classical stochastic process (whether RNNs or transformers) linearly represent the predicted Bloch vector of the qubit state of the minimal quantum memory capable of producing this process. As anticipated, the fractal belief geometry in the Bloch sphere is linearly represented in the activation of the neural networks.

Any y-component of the Bloch vector has no effect on the dynamics, and any such y-component initially present in quantum memory decays exponentially. However, for the stationary process we train on, the optimal belief states always live in the x-z slice of the Bloch ball.

Note that the belief geometry over the minimal quantum representation of this process lives in a two-dimensional subspace of the Bloch sphere, whereas the minimal classical-computational representation of this process would require an infinite number of dimensions.

D.2.1 Kraus operators

The process is parametrized by $\alpha > 0$ and $\beta \in \mathbb{R}$. Let $\gamma = 1/(2\sqrt{\alpha^2 + \beta^2})$. We will take $\alpha = 1$ and $\beta > 1$. For experiments we used $\alpha = 1$, $\beta = \sqrt{51}$.

The classical stochastic process is fully described via the following four observable-indexed Kraus operators that act on the quantum memory:

$$K_0 = \gamma \begin{bmatrix} \alpha+\beta & 0 \\ 0 & \alpha-\beta \end{bmatrix} = 2\gamma\alpha(I/2) + 2\gamma\beta(\sigma_z/2) \tag{15}$$

$$K_1 = \gamma \begin{bmatrix} \alpha-\beta & 0 \\ 0 & \alpha+\beta \end{bmatrix} = 2\gamma\alpha(I/2) - 2\gamma\beta(\sigma_z/2) \tag{16}$$

$$K_2 = \gamma \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \qquad = 2\gamma\alpha(I/2) + 2\gamma\beta(\sigma_x/2) \tag{17}$$

$$K_3 = \gamma \begin{bmatrix} \alpha & -\beta \\ -\beta & \alpha \end{bmatrix} \qquad = 2\gamma\alpha(I/2) - 2\gamma\beta(\sigma_x/2)\,. \tag{18}$$

These Kraus operators satisfy $\sum_{n=0}^{3} K_n^\dagger K_n = I$. Each Kraus operator induces a non-trace-preserving superoperator on the quantum memory state $A_n(\rho) = K_n \rho K_n^\dagger$. In this case, since α and β are real, $K_n^\dagger = K_n$. We also note the linear dependence $K_3 = K_0 + K_1 - K_2$.

From the initial fully mixed state, the belief would move towards $+|z\rangle$, $-|z\rangle$, $+|x\rangle$, or $-|x\rangle$, upon seeing the token 0, 1, 2, or 3, respectively. For this process, the density matrix for quantum memory updates as $\rho_{t+1} = \frac{K_{n_t}\rho_t K_{n_t}^\dagger}{\mathrm{tr}(K_{n_t}\rho_t K_{n_t}^\dagger)}$ given a particular measurement outcome $n_t \in \mathcal{X}$.

D.2.2 Bloch representation

As shown in Sec. B, we can find the GHMM-like extended Bloch representation of this process via the extended Bloch vectors of four linearly independent memory inputs to each subchannel, as well as the extended Bloch vectors of the subsequent outputs.

For our qubit memory, we can represent its state in the Hermitian operator basis $(I/2, \vec{\sigma}/2) = (I/2, \sigma_x/2, \sigma_y/2, \sigma_z/2)$, with the standard Pauli matrices

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{19}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \text{ and} \tag{20}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{21}$$

In fact, we can input the operator basis itself in this case since we can calculate everything analytically. This yields the convenient Bloch matrix $B = I$ for the input operators:

$$\underbrace{\begin{bmatrix} c_1 & \vec{b}_1 \\ c_2 & \vec{b}_2 \\ \vdots & \vdots \\ c_4 & \vec{b}_4 \end{bmatrix}}_{=:B} G^{(n)} = G^{(n)} = \underbrace{\begin{bmatrix} c_1' & \vec{b}_1' \\ c_2' & \vec{b}_2' \\ \vdots & \vdots \\ c_4' & \vec{b}_4' \end{bmatrix}}_{=:B'}. \tag{22}$$

Each output row on the right-hand side of Eq. (22) is the extended Bloch representation of $A_n(I/2)$ (for the first row) or $A_n(\vec{\sigma})$ (for the last three rows). In particular, $c_m' = \mathrm{tr}\big[A_n(\rho_{(m)})\big]$ and $\vec{b}_m' = \mathrm{tr}\big[A_n(\rho_{(m)})\vec{\sigma}\big]$. To calculate these, it will be useful to recall the relevant Cayley table:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix} [\sigma_x \quad \sigma_y \quad \sigma_z] = \begin{bmatrix} I & i\sigma_z & -i\sigma_y \\ -i\sigma_z & I & i\sigma_x \\ i\sigma_y & -i\sigma_x & I \end{bmatrix}. \tag{23}$$

Following through the algebra, $A_n(\rho_{(m)}) = K_n \rho_{(m)} K_n^\dagger = \gamma^2(\alpha I \pm \beta\sigma_{x/z})\rho_{(m)}(\alpha I \pm \beta\sigma_{x/z})$, where $\rho_{(m)} \in (I/2, \vec{\sigma}/2)$.

We find

$$G^{(0)} = \begin{bmatrix} 1/4 & 0 & 0 & 2\alpha\beta\gamma^2 \\ 0 & (\alpha^2-\beta^2)\gamma^2 & 0 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ 2\alpha\beta\gamma^2 & 0 & 0 & 1/4 \end{bmatrix} \tag{24}$$

$$G^{(1)} = \begin{bmatrix} 1/4 & 0 & 0 & -2\alpha\beta\gamma^2 \\ 0 & (\alpha^2-\beta^2)\gamma^2 & 0 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ -2\alpha\beta\gamma^2 & 0 & 0 & 1/4 \end{bmatrix} \tag{25}$$

$$G^{(2)} = \begin{bmatrix} 1/4 & 2\alpha\beta\gamma^2 & 0 & 0 \\ 2\alpha\beta\gamma^2 & 1/4 & 0 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ 0 & 0 & 0 & (\alpha^2-\beta^2)\gamma^2 \end{bmatrix}, \text{ and} \tag{26}$$

$$G^{(3)} = \begin{bmatrix} 1/4 & -2\alpha\beta\gamma^2 & 0 & 0 \\ -2\alpha\beta\gamma^2 & 1/4 & 0 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ 0 & 0 & 0 & (\alpha^2-\beta^2)\gamma^2 \end{bmatrix}. \tag{27}$$

(Note that $G^{(3)} \neq G^{(0)} + G^{(1)} - G^{(2)}$, even though $K_3 = K_0 + K_1 - K_2$.)

The net transition operator

$$G = \sum_{n\in\mathcal{X}} G^{(n)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 + 2(\alpha^2-\beta^2)\gamma^2 & 0 & 0 \\ 0 & 0 & 4(\alpha^2-\beta^2)\gamma^2 & 0 \\ 0 & 0 & 0 & 1/2 + 2(\alpha^2-\beta^2)\gamma^2 \end{bmatrix} \tag{28}$$

has the stationary extended Bloch vector $\langle\langle\boldsymbol{\pi}| = \langle\langle\boldsymbol{\pi}|G = [1 \quad 0 \quad 0 \quad 0]$ associated with the fully mixed quantum state $I/2$. Starting from this initial belief over the quantum memory, the Bayesian-updated extended Bloch vector (upon sequential observations) remains in the subspace spanned by $\{I, \sigma_x, \sigma_z\}$. Accordingly the belief metadynamics of the stationary stochastic process exists in a three-dimensional linear subspace and moreover, due to the normalization of quantum-state density matrices, in only a two-dimensional affine subspace corresponding to the x-z plane of the Bloch sphere.

The minimal three-dimensional GHMM for this process is thus easily obtained by projecting out the non-utilized y-component of the Bloch sphere:

$$T^{(0)} = \begin{bmatrix} 1/4 & 0 & 2\alpha\beta\gamma^2 \\ 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ 2\alpha\beta\gamma^2 & 0 & 1/4 \end{bmatrix} \tag{29}$$

$$T^{(1)} = \begin{bmatrix} 1/4 & 0 & -2\alpha\beta\gamma^2 \\ 0 & (\alpha^2-\beta^2)\gamma^2 & 0 \\ -2\alpha\beta\gamma^2 & 0 & 1/4 \end{bmatrix} \tag{30}$$

$$T^{(2)} = \begin{bmatrix} 1/4 & 2\alpha\beta\gamma^2 & 0 \\ 2\alpha\beta\gamma^2 & 1/4 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 \end{bmatrix}, \text{ and} \tag{31}$$

$$T^{(3)} = \begin{bmatrix} 1/4 & -2\alpha\beta\gamma^2 & 0 \\ -2\alpha\beta\gamma^2 & 1/4 & 0 \\ 0 & 0 & (\alpha^2-\beta^2)\gamma^2 \end{bmatrix}, \tag{32}$$

which can be interpreted as acting from the right on the coefficients $[c, b_x, b_z]$ of the ordered operator basis $(I/2,\ \sigma_x/2,\ \sigma_z/2)$.

The net transition operator

$$T = \sum_{n\in\mathcal{X}} T^{(n)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 + 2(\alpha^2-\beta^2)\gamma^2 & 0 \\ 0 & 0 & 1/2 + 2(\alpha^2-\beta^2)\gamma^2 \end{bmatrix} \tag{33}$$

has the stationary vectors $\langle\langle\boldsymbol{\pi}| = \langle\langle\boldsymbol{\pi}|T = [1 \quad 0 \quad 0]$ and $|1\rangle\rangle = T|1\rangle\rangle = [1 \quad 0 \quad 0]^\top$.

D.3 Quantum: FRDN

Despite not having any finite HMM generator, the FRDN process can be generated by a single qutrit—a quantum system with a three-dimensional Hilbert space Fanizza et al. [2024]. We find that neural networks trained on this classical stochastic process intrinsically learn the finite-dimensional quantum generative mechanism, and represent Bayesian updates over the quantum state of this post-classical generator as the neural network observes more tokens during inference. The observable alphabet only has two tokens $\mathcal{X} = \{a, b\}$, so the linear map from the latent space to the next-token distribution is non-invertible.

Riechers et al. Riechers and Elliott [2025] show that every stochastic process with a finite d-dimensional quantum representation has a GHMM representation with d^2 dimensions. The minimal GHMM representation can be smaller, and can be obtained from any GHMM representation via the algorithm presented in Ref. Upper [1997a]. In this case, as pointed out in Ref. Fanizza et al. [2024], the FRDN has a 4-state GHMM representation. In terms of parameters $\alpha \in \mathbb{R}$ and $0 < \lambda \leq 1/2$, we can write the matrices $T^{(a)}$ and $T^{(b)}$ as

$$T^{(a)} = |\omega\rangle\!\rangle\langle\!\langle \pi_0| \tag{34}$$

and

$$T^{(b)} = \lambda \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\alpha & -\sin\alpha \\ 0 & 0 & \sin\alpha & \cos\alpha \end{bmatrix} , \tag{35}$$

where $|\omega\rangle\!\rangle^\top = \big[1,\, 1-\lambda,\, 1+\lambda(\sin\alpha - \cos\alpha),\, 1-\lambda(\sin\alpha + \cos\alpha)\big]$ and $\langle\!\langle \pi_0| = \big[1 - \frac{1}{2(1-\lambda)} + \frac{1}{4}(c_+ + c_-),\, \frac{1}{2(1-\lambda)},\, -c_+/4,\, -c_-/4\big]$ with $c_\pm := \frac{1-\lambda\cos\alpha \pm \lambda\sin\alpha}{(1-\lambda\cos\alpha)^2 + \lambda^2\sin^2\alpha}$.[4]

When π/α is irrational, there is no finite-dimensional HMM realization of the process. When π/α is rational, there can still be a significant dimensional advantage. For the experiments shown in the appendix using the FRDN process, we used $\alpha = 2000$, $\lambda = 0.49$.

D.4 Post-quantum: Moon process

A minimal example of a post-quantum process—a classical stochastic process with a finite-dimensional GHMM generator, yet no finite HMM and no finite-dimensional quantum generator—has a simple three-dimensional representation, and an observable alphabet of three symbols $\mathcal{X} = \{a, b, c\}$. Following Ref. Fanizza et al. [2024], the linear maps generating the process are

$$T^{(a)} = \nu |m_0\rangle\!\rangle\langle\!\langle \mu_0| \,, \quad T^{(b)} = \nu \begin{bmatrix} \alpha & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \ln\alpha & 1 \end{bmatrix} , \text{ and } T^{(c)} = \nu \begin{bmatrix} \beta & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \ln\beta & 1 \end{bmatrix} , \tag{36}$$

with $\alpha, \beta, \nu \in \mathbb{R}$, such that $\alpha > 1 > \beta > 0$, $\alpha + \beta \neq 2$, $\ln(\alpha)/\ln(\beta) \in \mathbb{R} \setminus \mathbb{Q}$. The scalar value ν is chosen to make T have a maximal eigenvalue of 1.

Following Fanizza et al. Fanizza et al. [2024], in our experiments we use a parametrization of the post-quantum process with $\alpha = e$, $\beta = 1/2$, $|m_0\rangle\!\rangle = [1 \quad 1 \quad 0]^\top$, and $\langle\!\langle \mu_0| = [1 \quad -1 \quad -1]$.

D.5 Quantum: qRRXOR process

The quantum Random–Random–XOR (qRRXOR) process is a binary-alphabet ($\mathcal{X} = \{0, 1\}$) quantum generalization of the classical RRXOR process. It is generated by a single qubit of quantum memory with no finite HMM realization for generic parameters, and was designed for the experiments of Section 6 as a stress test that *decouples* predictive geometry from one-step next-token information: its minimal predictive vectors live in a three-dimensional subspace of the generalized Bloch ball, of which the next-token-readout subspace $\mathrm{col}(E)$ is only one-dimensional, leaving a two-dimensional "next-token-invisible" geometry that any successful affine probe must recover from neural activations.

The process is defined by three parameters (ϕ, θ, ϵ). For all experiments in the main text we use $\phi = \pi/2$, $\theta = \pi/4$, $\epsilon = 0.02$, chosen so that the single-step readout $\mathrm{col}(E)$ explains $\sim 15\%$ of the weighted predictive-vector variance, leaving $\sim 85\%$ in the orthogonal $\ker(E^\top)$ subspace.

[4] Note that there is a sine sign error in Eq. (27) of Ref. Fanizza et al. [2024] that we fix here.

D.5.1 Quantum circuit and Kraus operators

The qRRXOR process is the output of a repeating two-qubit circuit acting on a persistent memory qubit and an ancilla that is reset and measured each step:

$$\begin{array}{c} |\psi\rangle_{\text{mem}} \;\text{——}\, R_y(\phi) \text{——} \bullet \text{————} |\psi'\rangle_{\text{mem}} \quad \text{(persistent)} \\ \qquad\qquad\quad \big| \\ |0\rangle_{\text{anc}} \text{————} \oplus \text{-} R_x(\theta) \text{-} M \text{——} x_t \end{array} \tag{37}$$

Each step applies a y-rotation $R_y(\phi)$ to the memory, a CNOT with the memory as control and the ancilla as target, an x-rotation $R_x(\theta)$ on the ancilla, and a computational-basis measurement of the ancilla yielding the emitted token $x_t \in \{0, 1\}$. The ancilla is then re-initialized in $|0\rangle$ for the next step.

This circuit yields the two observable-indexed Kraus operators $K_x = D_x\, R_y(\phi)$ acting on the memory qubit, with

$$R_y(\phi) = \begin{bmatrix} \cos(\phi/2) & -\sin(\phi/2) \\ \sin(\phi/2) & \cos(\phi/2) \end{bmatrix}, \tag{38}$$

$$D_0 = \operatorname{diag}\big(\cos(\theta/2),\, -i\sin(\theta/2)\big) \quad \text{and} \quad D_1 = \operatorname{diag}\big(-i\sin(\theta/2),\, \cos(\theta/2)\big)\,. \tag{39}$$

The pair $\{K_0, K_1\}$ is Kraus-complete, $K_0^\dagger K_0 + K_1^\dagger K_1 = I$, so the induced superoperator $\mathcal{T}(\rho) = \sum_x K_x \rho K_x^\dagger$ is trace-preserving. Given the memory density matrix ρ_t, the joint distribution over the next token and post-measurement memory is

$$\Pr(x_t = x \mid \rho_t) = \operatorname{tr}\big(K_x\, \rho_t\, K_x^\dagger\big)\,, \qquad \rho_{t+1} = \frac{K_{x_t}\, \rho_t\, K_{x_t}^\dagger}{\operatorname{tr}\big(K_{x_t}\, \rho_t\, K_{x_t}^\dagger\big)}\,. \tag{40}$$

Parameter intuition. The angle ϕ controls how strongly the memory state is rotated each step before the CNOT entangles it with the ancilla. When ϕ/π is irrational, the orbits of the memory qubit under repeated application of $R_y(\phi)$ are dense in the x–z plane of the Bloch sphere, and the resulting process has no finite HMM realization. The angle θ controls how informatively the ancilla is read out: $\theta \to 0$ measures the ancilla in the same basis it was prepared in (uninformative), while $\theta \to \pi$ makes the measurement fully classical with respect to the CNOT-induced correlations. Intermediate θ leaves a controlled fraction of the memory's quantum coherence unmeasured, which is what makes the predictive geometry rich beyond what the next-token marginal sees.

D.5.2 Generalized Bloch (GHMM) representation

Following Sec. B, we represent the memory state in the generalized Bloch basis $\big(I/2,\, \sigma_x/2,\, \sigma_y/2,\, \sigma_z/2\big)$, yielding a 4-dimensional GHMM. The labeled transition matrices $T^{(x)}$ are obtained from the Kraus operators by

$$T^{(x)}_{ij} = 2\operatorname{tr}\big(B_j\, K_x\, B_i\, K_x^\dagger\big)\,, \qquad B \in \big(I/2,\, \sigma_x/2,\, \sigma_y/2,\, \sigma_z/2\big)\,, \tag{41}$$

acting from the right on row predictive vectors $\langle\!\langle \eta| = [c \quad b_x \quad b_y \quad b_z]$ whose trailing three components are the Bloch coordinates of the memory's density matrix $\rho = \frac{1}{2}(I + b_x\sigma_x + b_y\sigma_y + b_z\sigma_z)$.

The summed transition operator $T = T^{(0)} + T^{(1)}$ has $\langle\!\langle \boldsymbol{\pi}| = [1 \quad 0 \quad 0 \quad 0]$ as left eigenvector with eigenvalue 1 (the fully mixed memory state $\rho_\star = I/2$ is the stationary belief) and $|1\rangle\!\rangle = [1 \quad 0 \quad 0 \quad 0]^\top$ as right eigenvector with eigenvalue 1 (trace preservation).

Optional leakage. The leakage parameter $\epsilon \in [0, 1]$ mixes each transition with a re-set to the stationary state,

$$T^{(x)}_\epsilon = (1 - \epsilon)\, T^{(x)} + \frac{\epsilon}{|\mathcal{X}|}\, |1\rangle\!\rangle\langle\!\langle \boldsymbol{\pi}|\,, \tag{42}$$

which keeps $T^{(0)}_\epsilon + T^{(1)}_\epsilon$ row- and column-stochastic under the GHMM convention. We use a small $\epsilon = 0.02$ to regularize the dynamics: it ensures that the predictive-vector distribution has full support on its three-dimensional reachable subspace and that no row of any reachable transition is exactly degenerate, while leaving the qualitative geometry essentially unchanged.

D.5.3 Predictive geometry and the next-token-invisible subspace

The emission matrix relating predictive vectors to next-token probabilities is $E \in \mathbb{R}^{4\times 2}$, with $E[:, x] = T^{(x)}|1\rangle\!\rangle$, so that $\Pr(x_{t+1} = x \mid \langle\!\langle \eta_t|) = \langle\!\langle \eta_t|E[:, x]$. For the parameters $(\phi, \theta, \epsilon) = (\pi/2, \pi/4, 0.02)$, $\mathrm{rank}(E) = 2$, so

$$\dim \mathrm{col}(E) = 2, \qquad \dim \ker(E^\top) = 2\,. \tag{43}$$

The predictive vectors $\langle\!\langle \eta_w|$ induced by histories w explore a 3-dimensional submanifold of $\mathbb{R}^4$ (the affine constraint $\langle\!\langle \eta_w||1\rangle\!\rangle = 1$ removes one dimension). Of those three dimensions, only one lies along $\mathrm{col}(E)$ —the direction along which $p(x_{t+1} = 0)$ varies—while two lie in $\ker(E^\top)$ and have *no effect* on the next-token marginal. Concretely, an affine map from predictive vectors to next-token probabilities, $\langle\!\langle \eta_w| \mapsto \langle\!\langle \eta_w|E$, explains

$$R^2_{\text{NTP-ceiling}} \;=\; \frac{\mathrm{Var}_w\big(P_E\,\langle\!\langle \eta_w|\big)}{\mathrm{Var}_w\big(\langle\!\langle \eta_w|\big)} \;\approx\; 0.15 \tag{44}$$

of the (history-weighted) predictive-vector variance. The remaining $\sim 85\%$ lies in directions to which the one-step readout is structurally blind. This is the property the qRRXOR process is engineered to have.

D.5.4 The NTP-preserving shuffle

The control target used in Section 6 is a geometric shuffle of predictive vectors that preserves $\mathrm{col}(E)$ (and hence the next-token marginal) exactly, while randomizing the orthogonal $\ker(E^\top)$ component across histories. Let

$$P_E = E(E^\top E)^{-1}E^\top, \qquad P_{E^\perp} = I - P_E \tag{45}$$

be the orthogonal projectors onto $\mathrm{col}(E)$ and $\ker(E^\top)$ respectively, and let $\bar\eta$ denote the empirical weighted mean of predictive vectors over histories. Decompose each predictive vector as

$$\langle\!\langle \eta_w| = \bar\eta + (\langle\!\langle \eta_w| - \bar\eta)P_E + (\langle\!\langle \eta_w| - \bar\eta)P_{E^\perp}\,, \tag{46}$$

and let σ be a uniformly random permutation over the set of histories. The NTP-preserving shuffle is then

$$\langle\!\langle \eta^{\text{shuf}}_w| \coloneqq \bar\eta + (\langle\!\langle \eta_w| - \bar\eta)P_E + (\langle\!\langle \eta_{\sigma(w)}| - \bar\eta)P_{E^\perp}\,. \tag{47}$$

By construction, $\langle\!\langle \eta^{\text{shuf}}_w|E = \langle\!\langle \eta_w|E$ for every history w: the shuffled targets assign *identical* next-token probabilities to every history as the true predictive vectors. The marginal distribution of the orthogonal residuals across histories is also preserved (only their assignment to histories is randomized). The only thing the shuffle destroys is the joint structure—which residual goes with which next-token distribution, and consequently the geometry of how the predictive state will be updated by future tokens.

Some notes about this control. Any affine probe of activations that recovers the true predictive vectors $\langle\!\langle \eta_w|$ *must* also be able to recover the next-token-relevant component, since it is a linear function of the target. So a high R^2 to the true target could in principle be explained entirely by the model representing next-token probabilities and the probe reading them off. The NTP-preserving shuffle separates these two hypotheses: a probe that only recovers $E^\top\eta_w$ (next-token info) will achieve the same R^2 on shuffled and true targets—both bounded above by $R^2_{\text{NTP-ceiling}} \approx 0.15$. A probe that recovers the full predictive geometry will fit the true target far above this ceiling and the shuffled target at or below it. The dissociation between the two R^2 values is what we report in Section 6.

D.5.5 Parameters and dataset details

We use $\phi = \pi/2$, $\theta = \pi/4$, $\epsilon = 0.02$ for the experiments in Section 6 and Figure 4. Models are trained on next-token cross-entropy alone, with context length $n_{\text{ctx}} = 10$. For the probing analysis we exhaustively enumerate all $2^{n_{\text{ctx}}} = 1024$ histories of length n_{ctx}, computing each one's analytic predictive vector by sequentially applying the Kraus operators to the stationary memory state

$\rho_\star = I/2$, and weighting each history by its exact prefix probability under the process. All R^2 values reported are weighted-R^2,

$$R^2_w \;=\; 1 - \frac{\sum_w p(w)\,\|\boldsymbol{\eta}_w - \widehat{\boldsymbol{\eta}}_w\|^2}{\sum_w p(w)\,\|\boldsymbol{\eta}_w - \bar{\boldsymbol{\eta}}\|^2}\;, \tag{48}$$

with $\bar{\boldsymbol{\eta}}$ the weighted mean. The probe pipeline is identical to that used for the main results (§3.2, App. E.5).

Architecture choice. For the LSTM (Fig. 4 d–h) we use a 4-layer LSTM with hidden width $h = 64$ trained at batch size $b = 64$ for 20,000 epochs with the default PyTorch initialization. The transformer (Fig. 4 i–j) is a 4-layer muP transformer with $d_{\text{model}} = 64$, $n_{\text{heads}} = 4$, $d_{\text{head}} = 16$, $d_{\text{mlp}} = 256$, trained at batch size $b = 16$ for 20,000 epochs under standard GPT-2-style initialization.

Reachable geometry. The predictive vectors over the 1024 enumerated length-10 histories trace out the reachable set of the GHMM dynamics on a 3-dimensional submanifold of $\mathbb{R}^4$. PCA-3 of the predictive vectors captures 100% of their variance, which is why the 3-D scatter panels of Fig. 4(a–f) are exact—no information is lost in the projection. The HSV-sphere coloring of panels (a, b, d, e) assigns each point a unique hue from its azimuthal angle and a brightness from its polar height in the PCA-3 basis, so that identity correspondences across panels (beliefs → shuffled → probe predictions) are visually trackable. Panels (c, f) recolor the same shuffled positions by next-token probability $p(x_{t+1} = 0)$ using the perceptually uniform `viridis` colormap; the smooth gradient surviving the shuffle visualizes the preservation of $\mathrm{col}(E)$.

Relation to classical RRXOR. The qRRXOR construction is a quantum lift of the classical RRXOR process Riechers and Crutchfield [2018]: classical RRXOR is a 5-state HMM that emits two random bits ($r_1, r_2 \in \{0, 1\}$) followed by their XOR ($r_1 \oplus r_2$), repeating forever. In the classical setting the predictive states form a discrete simplex of finite cardinality. Replacing the classical memory with a single qubit and the deterministic XOR with a coherent rotation yields qRRXOR; the resulting process retains the basic "two random tokens then a structured third" flavor but lifts the predictive geometry off any finite-state simplex into the Bloch ball, parameterized continuously by (ϕ, θ). The choice $\theta = \pi/4$ configuration places the readout exactly between the uninformative and classical limits, maximizing the fraction of the predictive geometry that is invisible to the one-step marginal.

E Training and Probing Details

We performed standard self-supervised pretraining on next-token-prediction cross-entropy loss. Given the parameter vector of weights and biases $\boldsymbol{\theta}$, and a given context $x_{1:\ell-1}$, a neural-network sequence model produces logits for the next token $-\log \Pr_{\boldsymbol{\theta}}(X_\ell | X_{1:\ell-1} = x_{1:\ell-1})$. Given an input sequence $x_{1:L}$, the pretraining loss is $-\sum_{\ell=1}^{L-1} \log \Pr_{\boldsymbol{\theta}}(x_{\ell+1} | x_{1:\ell})$.

E.1 Experimental design

We conduct a comprehensive evaluation of four neural network architectures on four distinct stochastic processes, resulting in 16 experimental configurations. The architectures include transformers, LSTMs, GRUs, and vanilla RNNs, while the processes consist of Mess3 (classical), FRDN (quantum), Bloch Walk (quantum), and the Moon Process (post-quantum)—each representing different computational complexity classes as described in Appendix D.

E.2 Model architectures

For the Transformer architecture, we employ a 4-layer model implemented using the TransformerLens framework [Nanda and Bloom, 2022]. The model uses multi-head attention with 4 heads (dimension 16 per head), 64-dimensional embeddings, and a 256-dimensional feed-forward network with ReLU activation. Layer normalization is applied before each sub-layer, and the model processes fixed sequences of 8 tokens with learned positional embeddings.

We compare three RNN variants, each configured with identical hyperparameters to ensure fair comparison: 4 recurrent layers with 64 hidden units per layer, unidirectional processing, one-hot

input encoding, and a linear output projection. The variants differ only in their gating mechanisms: LSTM uses forget, input, and output gates; GRU employs reset and update gates; and the vanilla RNN uses simple tanh activation without gating.

E.3 Training methodology

Training data is generated from each stochastic process with the following parameters (see Appendix D for process definitions):

- **Mess3**: $\alpha = 0.85$, $x = 0.05$
- **Bloch Walk**: $\alpha = 1$, $\beta = \sqrt{51}$
- **FRDN**: $\alpha = 2000$, $\lambda = 0.49$
- **Moon Process**: $\alpha = e$, $\beta = 1/2$

Each training example consists of an 8-token input sequence with corresponding next-token targets for each position. All experiments use consistent random seeding (seed=42) for reproducibility. We train using standard cross-entropy loss. Validation is performed every epoch on the full dataset, with model checkpoints saved every 100 epochs (201 total) and comprehensive metric logging via Weights & Biases.

E.3.1 Training hyperparameters

Table 1 shows the core hyperparameters used across all experiments. All models are trained for 20,000 epochs using the Adam optimizer with learning rate 1×10^{-4}.

Table 1: Core hyperparameters used across all experiments.

Hyperparameter	**Value**
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
Learning rate	1×10^{-4}
Weight decay	None
Gradient clipping	None
Epochs	20,000
Validation frequency	Every epoch
Checkpoint frequency	Every 100 epochs
Random seed	42

The majority of experiments use the following configuration:

Table 2: Standard configuration used by most experiments.

Configuration	**Standard Setting**
Batch size	128
Batches per epoch	200
LR scheduler	ReduceLROnPlateau*
Total checkpoints	201

*ReduceLROnPlateau parameters: factor=0.5, patience=1000, cooldown=200, threshold=10^{-6}.

While all RNN variants (LSTM, GRU, RNN) use the standard configuration above for all processes, we found that certain transformer experiments trained better with modified settings:

Table 3: Experiment-specific variations from standard configuration.

Experiment	Batch Size	Batches/Epoch	LR Scheduler
Transformer-FRDN	16	20	None
Transformer-Moon	16	20	ReduceLROnPlateau

E.4 Implementation details

All experiments are implemented in PyTorch 2.0 with CUDA acceleration, using FP32 precision throughout. Training is distributed across multiple GPUs, with specific GPU assignments managed through a parallel execution framework. To ensure reproducibility, we use fixed random seeds.

We maintain 4 layers across all architectures to ensure fair comparison of inductive biases rather than capacity differences. The high checkpoint frequency (every 100 epochs) enables detailed analysis of learning dynamics and convergence behavior. Finally, we deliberately avoid dropout, weight decay, or other regularization techniques to study the pure learning dynamics of each architecture on these processes.

Compute resources. Training was performed on NVIDIA H100 GPUs. Each individual training run (4-layer model, hidden width 64, 8-token context, 20,000 epochs) took approximately one GPU-hour. The 16 main configurations (4 architectures $\times$ 4 processes) reported in Fig. 2 therefore required roughly 16 GPU-hours; the qRRXOR experiments in §6 and the training-dynamics analysis in §5 added on the order of tens of additional GPU-hours. The full research project including exploratory and discarded runs required substantially more compute than the experiments reported in the paper.

E.5 General approach to probing belief geometry

Our main analysis quantifies whether neural network activations encode belief states through an affine transformation of their internal activations. Given a neural activation vector $\vec{a}_w \in \mathbb{R}^d$ (with e.g., activations from a particular position and a single layer $d = d_{\text{model}}$, or activations from a particular position across all layers $d = N d_{\text{model}}$) induced by a token sequence $w \in \mathcal{X}^*$ and a proposed belief state $\boldsymbol{\eta}^{(w)} \in \mathbb{R}^{d_{\text{g}}}$, we seek an affine transformation:

$$\boldsymbol{\eta}^{(w)} \approx \vec{a}_w L + \boldsymbol{b} = \hat{\boldsymbol{\eta}}^{(w)} \tag{49}$$

where $L \in \mathbb{R}^{d \times d_{\text{g}}}$ is a linear map and $\boldsymbol{b} \in \mathbb{R}^{d_{\text{g}}}$ is a bias vector. We express this compactly using augmented notation:

$$\boldsymbol{\eta}^{(w)} \approx [1 \quad \vec{a}_w]\, \mathcal{L} = \hat{\boldsymbol{\eta}}^{(w)} \tag{50}$$

where $\mathcal{L} = \begin{bmatrix} \boldsymbol{b} \\ L \end{bmatrix} \in \mathbb{R}^{(1+d) \times d_{\text{g}}}$.

We assemble a dataset from anchor sequences $\mathcal{A} = \{w_n\}_{n=1}^{|\mathcal{A}|} \subset \mathcal{X}^*$. In our analysis, we take the set of anchor points to be all sequences generated by the process up to the context window length of the network. Let $A \in \mathbb{R}^{|\mathcal{A}| \times (1+d)}$ be the matrix of augmented activation vectors with n-th row $[1, \vec{a}_{w_n}]$, and $\Gamma \in \mathbb{R}^{|\mathcal{A}| \times d_{\text{g}}}$ be the matrix with corresponding belief vector $\boldsymbol{\eta}^{(w_n)}$ in row n.

To reflect the process's true statistics, we weight each sequence by its probability $p_n = \frac{\langle\langle \eta^{(\varnothing)} | T^{(w_n)} | 1 \rangle\rangle}{\sum_{w \in \mathcal{A}} \langle\langle \eta^{(\varnothing)} | T^{(w)} | 1 \rangle\rangle}$ derived from the ground-truth generative model. In those cases where the anchor sequences consist of all allowable words up to some context length, i.e., $\mathcal{A} = \bigcup_{\ell=1}^{\ell_{\text{context}}} \{w \in \mathcal{X}^\ell : \langle\langle \eta^{(\varnothing)} | T^{(w)} | 1 \rangle\rangle > 0\}$, we will have $p_n = \langle\langle \eta^{(\varnothing)} | T^{(w_n)} | 1 \rangle\rangle / \ell_{\text{context}}$. The optimal transformation minimizes:

$$\mathcal{L}^* = \text{argmin}_{\mathcal{L}} \left\langle \left\| [1 \quad \vec{a}_{w_n}]\, \mathcal{L} - \boldsymbol{\eta}^{(w_n)} \right\|_2^2 \right\rangle_n . \tag{51}$$

Following the standard approach for weighted linear regression, we note that the solution via weighted least squares is:

$$\mathcal{L}^* = (P^{1/2} A)^+ P^{1/2} \Gamma \,, \tag{52}$$

where P is a diagonal matrix with $P_{nn} = p_n$, and M^+ denotes the regularized Moore–Penrose pseudoinverse of M.

E.6 Implementation details for the probe

E.6.1 Deduplication

Before regression, we identify and aggregate duplicate token prefixes. For each unique prefix, we retain the activation vector from its first occurrence, and then we sum the probabilities across all occurrences of the same prefix. This deduplication reduces computational cost while preserving the correct probability weighting.

E.6.2 Computing the pseudoinverse

For computational efficiency when evaluating multiple regularization parameters r, we compute the SVD once:

$$P^{1/2}A = U\Sigma V^T \,, \tag{53}$$

where Σ is a diagonal matrix whose diagonal elements are the singular values in descending order $\Sigma_{n+1,n+1} \leq \Sigma_{n,n}$ for $n \geq 0$. The regularized pseudoinverse for any $r > 0$ is then

$$(P^{1/2}A)^+_r = V\Sigma^+_r U^T \,, \tag{54}$$

where Σ^+_r is diagonal with entries

$$(\Sigma^+_r)_{n,n} = \begin{cases} \frac{1}{\Sigma_{n,n}} & \text{if } \Sigma_{n,n} > r\Sigma_{1,1} \\ 0 & \text{otherwise .} \end{cases} \tag{55}$$

E.6.3 Cross-validation and model selection

We use 10-fold cross-validation to select the optimal regularization parameter r from a set combining $\{10^{-15}, 10^{-10}, 10^{-5}\}$ with 50 logarithmically-spaced values between 10^{-8} and 10^{-3}.

For each fold and each candidate r:

1. Partition data into training (90%) and validation (10%) sets.
2. Fit the weighted regression on the training set.
3. Evaluate weighted error on the validation set: $\sum_i p_i \|\boldsymbol{\eta}^{(w_i)} - \hat{\boldsymbol{\eta}}^{(w_i)}\|_2$.

The r minimizing average validation error across folds is selected for the final model trained on all data.

E.6.4 Evaluation metrics

To quantify how well the belief states were represented in network activations through an affine map, we compute the root mean squared error, RMSE, of the fit, by first computing the mean squared error, MSE, according to:

$$\text{MSE} = \sum_i p_i \left\| \boldsymbol{\eta}^{(w_i)} - \hat{\boldsymbol{\eta}}^{(w_i)} \right\|_2^2 \tag{56}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \,. \tag{57}$$

E.6.5 Cosine similarity analysis

To assess whether the learned representations preserve the geometric relationships between belief states, we analyze pairwise cosine similarities. For a set of belief states $\{\boldsymbol{\eta}^{(w_i)}\}_{i=1}^{|\mathcal{A}|}$, we compute the cosine similarity matrix:

$$S_{ij} = \frac{\boldsymbol{\eta}^{(w_i)} \cdot \boldsymbol{\eta}^{(w_j)}}{\|\boldsymbol{\eta}^{(w_i)}\| \|\boldsymbol{\eta}^{(w_j)}\|} \,. \tag{58}$$

We extract the upper triangle of this matrix (excluding the diagonal) to obtain a distribution of pairwise similarities. This analysis is performed for:

1. Ground truth belief states $\boldsymbol{\eta}^{(w_i)}$.

2. Predicted belief states $\hat{\boldsymbol{\eta}}^{(w_i)}$ from the regression.

By comparing these geometric relationships, we evaluate whether the linear probe preserves the relative orientations between belief vectors. This provides a complementary view to the distance-based metrics, focusing on angular rather than Euclidean-distance relationships. The analysis is conducted separately for both Markov-order-3 approximations of the processes and full generator belief geometries.

E.7 Control experiments

To verify that the learned representations are not artifacts of the architecture alone, we compare against networks with randomly initialized weights (i.e., before any training (backpropagation) has happened). This control undergoes the same regression analysis, allowing us to quantify how much structure arises from training versus architecture.

We also apply the same regression pipeline to classical Markov models of order 3, mapping their belief states to neural activations. We chose the Markov order to be 3 since in our main experimental condition for a post-classical process (the Bloch Walk Process), the Markov-order-3 approximation of the process has 64 states, and thus the corresponding 64-dimensional belief states should exactly fit into the hidden-state dimensionality of our neural networks. This provides a baseline for assessing whether neural networks learn representations aligned with classical approximations of the post-classical processes, or if they align more closely with the compact post-classical predictive-state geometry.

F Pairwise Geometry Preservation

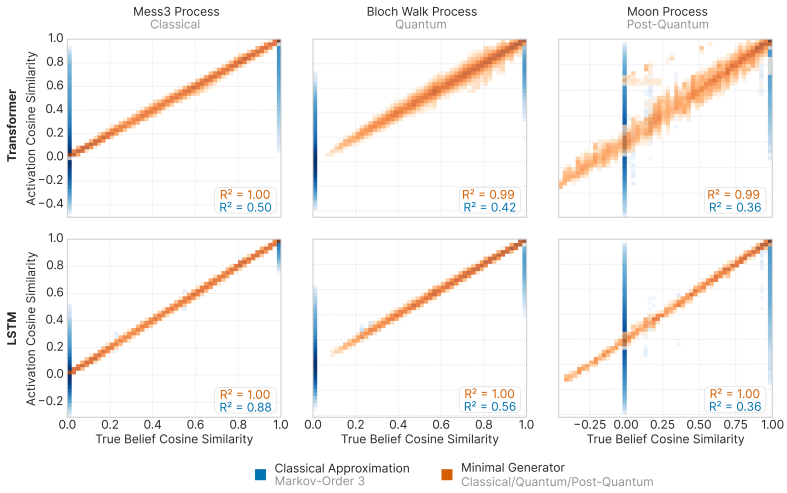

Figure 5: **Neural networks preserve geometric relationships between belief states across classical, quantum, and post-quantum processes.** Pairwise cosine similarity between belief states as represented in neural network activations (y-axis) versus the true theoretical belief geometry (x-axis). Cosine similarity measures the angular relationship between belief vectors, with 1.0 indicating parallel vectors and 0.0 indicating orthogonal vectors. Each point represents a pair of belief states induced by different context sequences, computed over all possible contexts up to the context window length. Orange points show the relationship when beliefs are taken from the minimal generator (classical for Mess3, quantum for Bloch Walk, post-quantum for Moon), while blue points show relationships for a classical Markov-order-3 approximation. Perfect preservation of geometric relationships would yield points along the diagonal. Both transformer (top row) and LSTM (bottom row) architectures show near-perfect preservation of the minimal generator geometry ($R^2 \in [0.99, 1.00]$, orange) across all three process types, while poorly representing classical approximations (blue) for the classical ($R^2 \in [0.5, 0.88]$), quantum ($R^2 \in [0.42, 0.56]$), and post-quantum ($R^2 = 0.36$) processes. The tight clustering along the diagonal for minimal generators demonstrates that networks learn not just individual belief states but the complete geometric structure that governs how different observation histories relate to each other in belief space. This preservation of pairwise relationships provides strong evidence that neural networks are discovering the true underlying geometry rather than learning a distorted approximation.

G Extended Architecture Results

Our central claim of universality—that the discovery of minimal belief geometry is independent of neural network architecture—is supported by a comprehensive set of experiments. While the main text highlights results for Transformers and LSTMs (Fig. 2), we performed identical analyses on vanilla Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs). The results, presented in Figs. 6, 7, 8, and 9 below, show that all tested architectures successfully learn to linearly represent the correct classical, quantum, and post-quantum belief geometries of their respective training data. The affine-mapped activation geometries and their corresponding RMSE plots consistently demonstrate a strong preference for the minimal generator over classical approximations and random baselines, reinforcing that this phenomenon is a fundamental outcome of the training process rather than an artifact of a specific architecture.

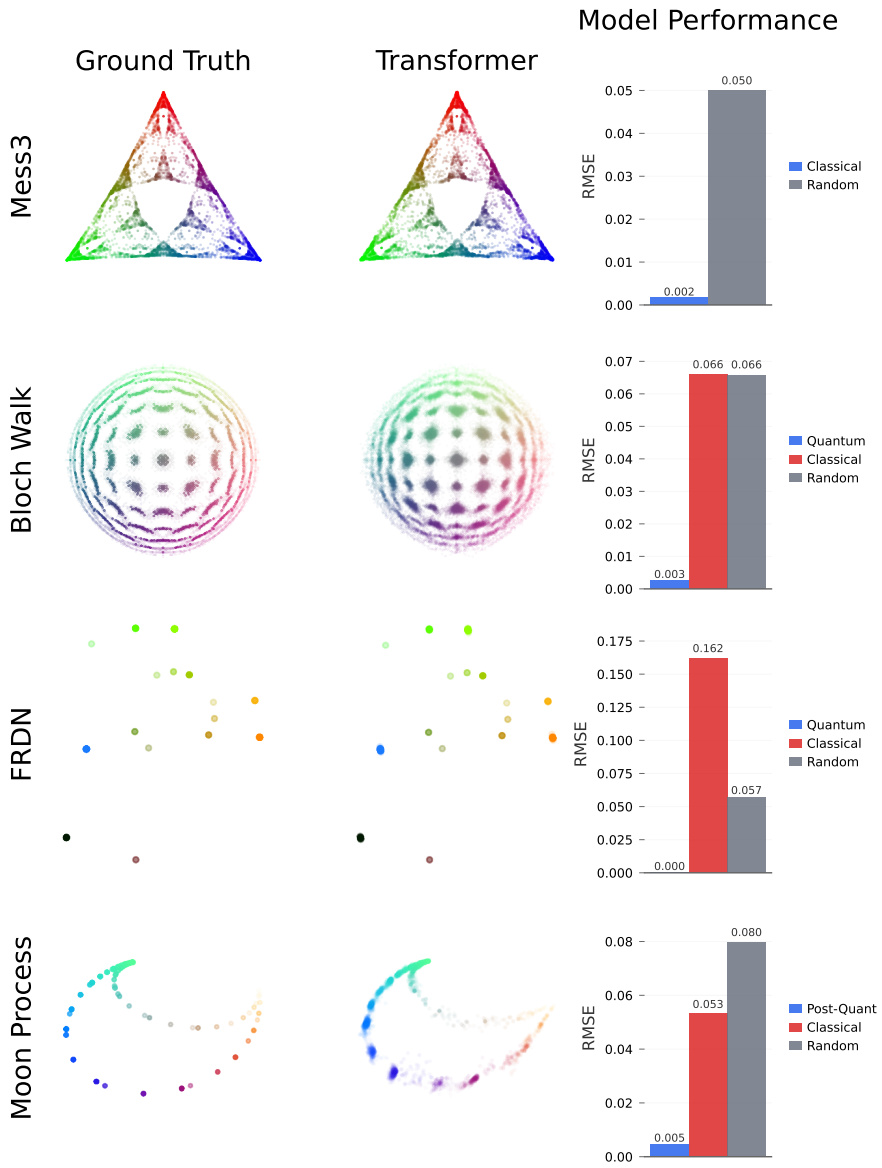

Figure 6: **Transformers learn the minimal belief geometry across all process classes.** Comparison of ground truth belief geometries (left column) with the affine-mapped activation geometries of a trained 4-layer Transformer (center column). The four rows correspond to the Mess3 (Classical), Bloch Walk (Quantum), FRDN (Quantum), and Moon (Post-Quantum) processes. Points are colored by their position in the ground truth space to visualize the preservation of geometric relationships. Bar plots (right column) show the root mean square error (RMSE) for fits to the true minimal generator (blue/purple), a classical Markov-order-3 approximation (red), and a randomly initialized network (gray). The Transformer consistently achieves low RMSE for the minimal generator, demonstrating its ability to learn the most compact representation, whether classical, quantum, or post-quantum.

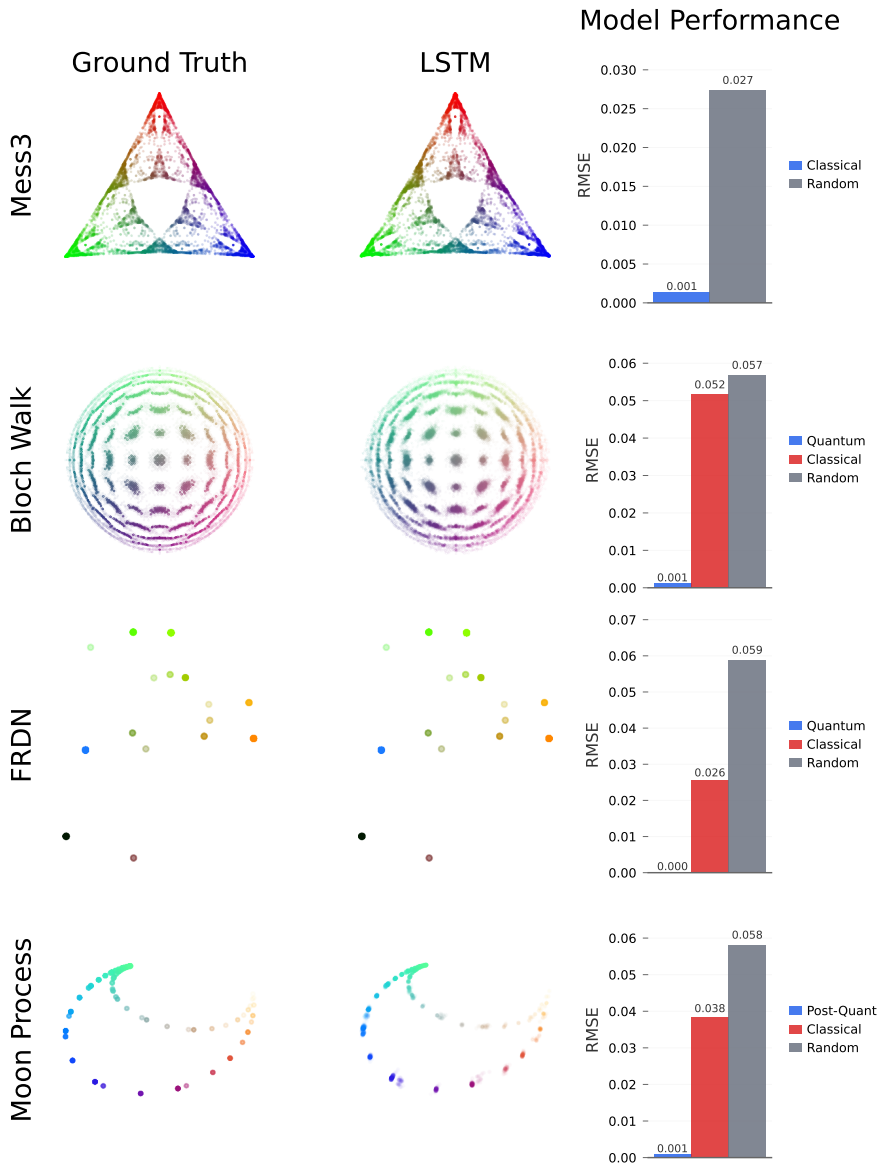

Figure 7: **LSTM architecture successfully discovers minimal belief geometries.** Comparison of ground truth belief geometries (left column) with the affine-mapped activation geometries from a trained 4-layer LSTM (center column). Each row presents a different stochastic process: Mess3 (Classical), Bloch Walk (Quantum), FRDN (Quantum), and Moon (Post-Quantum). The color correspondence illustrates that the LSTM preserves the relative structure of the belief space. The bar plots (right column) quantify the fit's accuracy, comparing the RMSE for the minimal generator (blue/purple) against a classical approximation (red) and a random network baseline (gray). The results demonstrate that, like Transformers, LSTMs effectively learn the correct classical, quantum, and post-quantum representations.

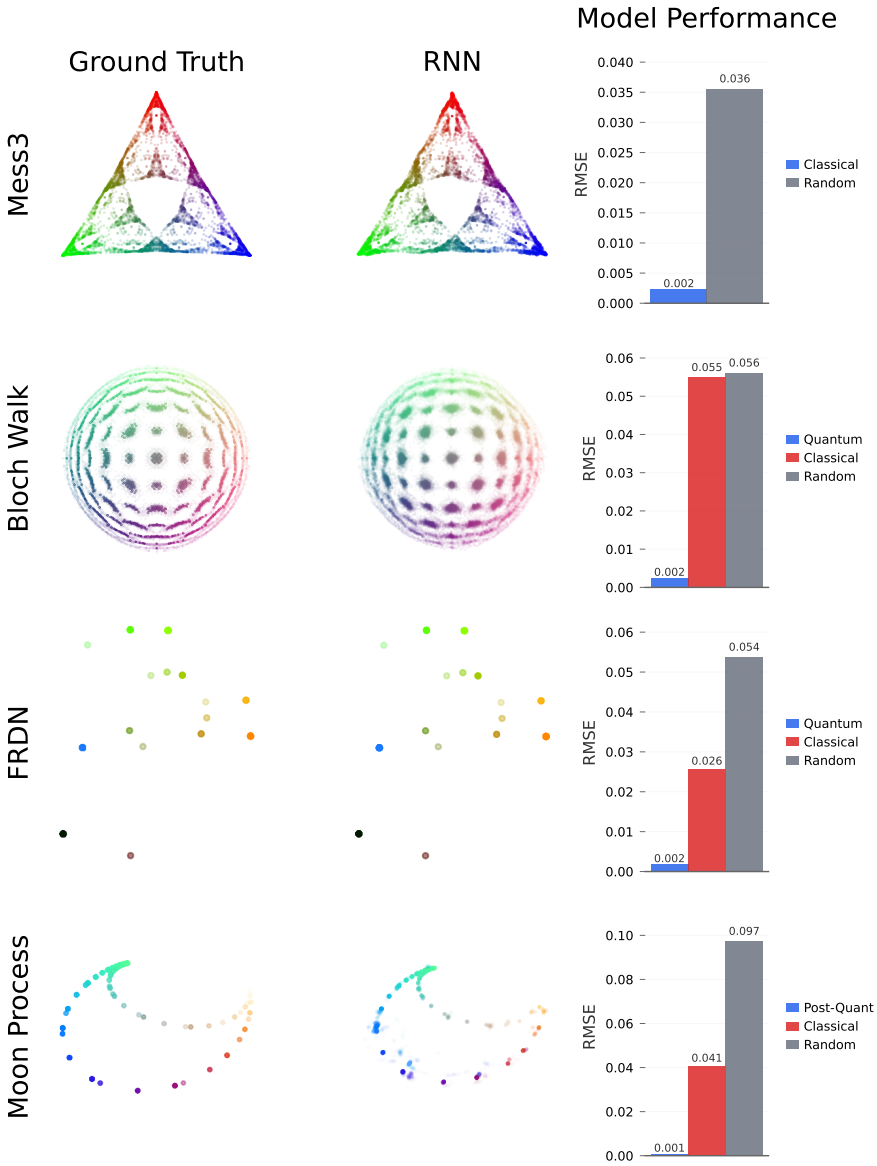

Figure 8: **Vanilla RNNs captures the correct minimal belief geometry.** Comparison of ground truth belief geometries (left column) with those learned by a trained 4-layer vanilla Recurrent Neural Network (RNN) (center column), for the Mess3, Bloch Walk, FRDN, and Moon processes. Despite its simpler architecture lacking gating mechanisms, the vanilla RNN learns to structure its activations in a way that linearly maps to the correct minimal belief geometry for classical, quantum, and post-quantum processes. The RMSE plots (right column) confirm a significantly better fit to the true generator (blue/purple) than to classical approximations (red) or random baselines (gray). This reinforces the universality of this phenomenon.

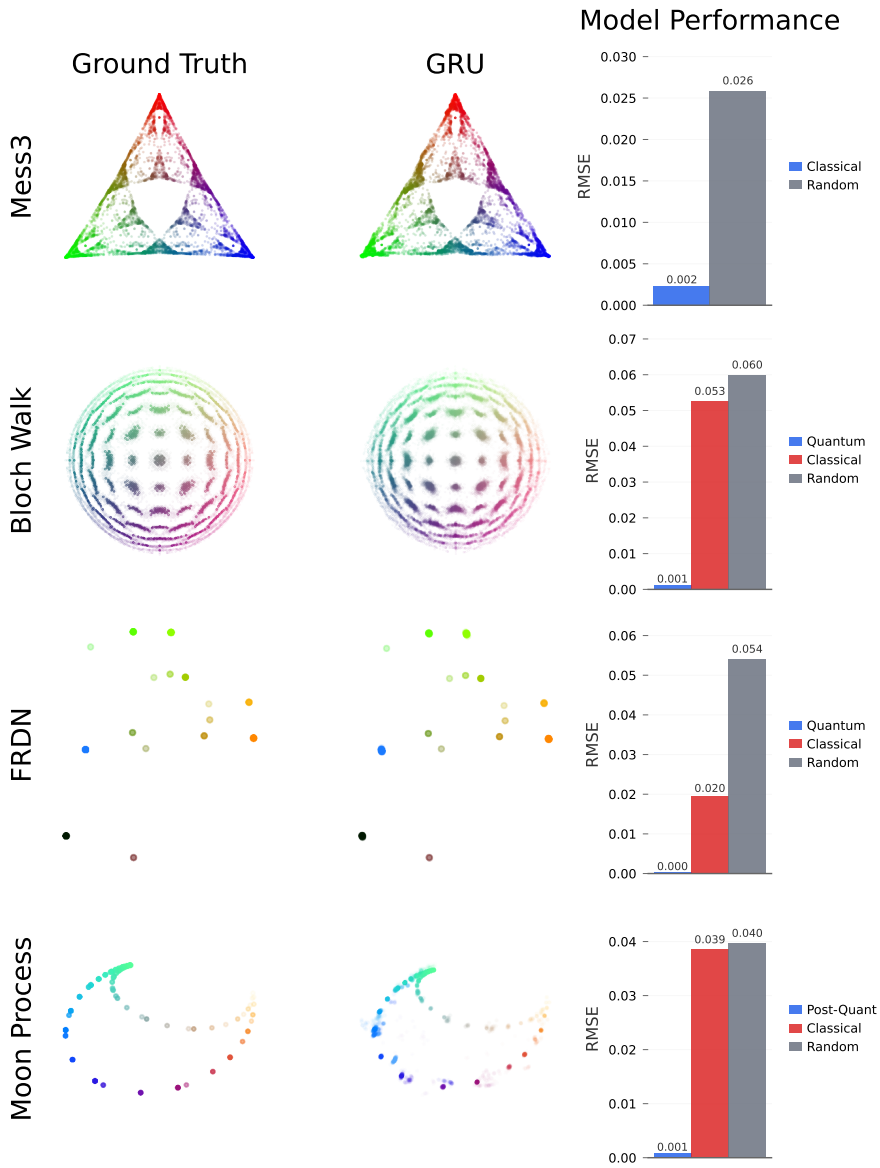

Figure 9: **GRU network performance mirrors other architectures in learning belief geometries.** Comparison of ground truth belief geometries (left column) with the affine-mapped activation geometries from a trained 4-layer Gated Recurrent Unit (GRU) network (center column) for all four process types. The visual and quantitative results are consistent with those from Transformer, LSTM, and vanilla RNN architectures. The GRU successfully identifies and represents the minimal belief geometry in its activations, as shown by the low RMSE values for the correct generator (blue/purple) compared to classical approximations (red) and random networks (gray). This further strengthens the claim that the discovery of minimal belief geometries is a universal feature of training recurrent-style networks on next-token prediction.

H Reproducibility Details

In the interest of reproducibility and completeness, we provide the codebase used to train networks, run analysis, and create the figures in this manuscript. The repository link is *[redacted for double-blind review; code will be released upon publication]*.

H.1 Code repository

The code repository URL is *[redacted for double-blind review; code will be released upon publication]*. The accompanying README provides instructions on how to recreate all training and figure generation in this manuscript.

The repository contains:

- Training scripts for all architectures (Transformer, LSTM, GRU, RNN).
- Activation analysis pipeline (`scripts/activation_analysis/run_regressions_analysis.py`).
- Figure generation scripts (`Fig2.py`, `Fig3.py`, `Fig4.py`, `FigAppendix.py`).
- Process definitions and generators (`epsilon_transformers/process/`).
- Experiment configuration files (`scripts/experiment_config_*.yaml`).

H.2 Hugging Face dataset

The complete dataset of trained model checkpoints and pre-computed analysis results is publicly available at *[Hugging Face URL redacted for double-blind review]*. The dataset can additionally be regenerated from the process definitions in App. D. The dataset contains:

- **16 trained neural network models** (4 architectures $\times$ 4 processes).
- **Pre-computed belief state regression analysis results** for all models.
- **Model checkpoints** at multiple training stages (201 checkpoints per model).
- **Training configurations and loss curves**.

A README accompanying the released dataset (*URL redacted for double-blind review*) provides further details.

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: Abstract and introduction preview the conceptual claim and the empirical results, delivered in §4, §5, and §6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in the final paragraph of §7.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not introduce new theoretical results. The framework (§2) and process definitions (App. D) restate established results from cited prior work.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Process definitions are given analytically in App. D, training details and hyperparameters in App. E, and the probe pipeline in App. E.5.

Guidelines:

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

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

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

Answer: [No]

Justification: Code and data URLs are redacted for double-blind review and will be released upon publication (§H). The processes are defined analytically (App. D) and training and probe details are in App. E, so results can be reproduced from the paper alone.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Optimizer, learning rate, training duration, seed, and architecture details are in App. E.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: Main-result figures report single-seed runs without error bars. The reported effects are large in magnitude (R^2 gaps of 0.5–0.8 above NTP and Markov-order-3 baselines), and three control experiments (random-init networks, Markov-order-3 classical approximation, NTP-preserving shuffle) provide effect-size information.

Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: GPU type and per-run / total compute estimates are given in the *Compute resources* paragraph of App. E.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The paper studies synthetic stochastic processes and small neural networks. No human subjects, scraped data, or deployed systems are involved.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The paper is foundational interpretability research on synthetic data with no direct societal application.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The released models and data pose no misuse risk. Models are small (4 layers, hidden 64) and trained on synthetic stochastic processes.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The TransformerLens framework [Nanda and Bloom, 2022] is cited at first use (§3.2). No external datasets are used. All data are generated from the analytic process definitions in App. D.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets are released with the submission. Code and a checkpoint/analysis dataset will be released upon publication.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not a component of the core methodology of this paper.

    Guidelines:

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