HIDDEN STATES AS STATES: DISCRETIZING HIDDEN-STATE GEOMETRY IN LANGUAGE MODEL COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

While large language models (LLMs) operate on discrete tokens, their internal computation unfolds in a continuous, high-dimensional representation space. We introduce **Hidden States as States (HSS)**, a training free and label free framework that discretizes hidden representations at a chosen granularity into a cross-layer vocabulary of geometric states. HSS clusters representations within each layer and aligns cluster centroids across adjacent layers, converting each forward pass into a layer-indexed discrete state sequence. The resulting maps are stable under clustering perturbations, expanding in intermediate layers and contracting later in model-specific patterns. Conditioning on behavioral labels further reveals distinct state occupancy patterns, with failures collapsing into a few persistent sink states that emerge in early layers. Simple count-based predictors over these discrete state sequences outperform closed-form continuous baselines, match trained continuous probes on capability and safety benchmarks, and support online failure warning during generation. Reconstruction and activation editing show that, despite their coarse granularity, these states retain functionally relevant structure for downstream computation. HSS thus converts continuous hidden dynamics into an actionable discrete structure.

1 INTRODUCTION

Modern language models (LMs) (Vaswani et al., 2017; Achiam et al., 2023; Bai et al., 2023; Liu et al., 2024) compute by transforming hidden representations across layers. These hidden states are the internal substrate through which information is processed before being projected to output tokens. Despite major gains in model performance, the global organization of hidden representations, and its relation to model behavior, remains poorly understood.

Existing tools examine hidden states through external or task-specific readouts: logit-based lenses (Belrose et al., 2023; nostalgebraist, 2020) project representations toward the output vocabulary, supervised probes (Hewitt & Liang, 2019; Belinkov, 2022; Marks & Tegmark, 2023) require labels and give property-specific views, and feature-level methods such as sparse autoencoders (Templeton et al., 2024; Bricken et al., 2023) and circuit analyses (Ameisen et al., 2025) focus on narrow behaviors or require additional training. As a result, it remains challenging to obtain a holistic view of the geometric space induced by language model computation.

Such a geometric view is important because model behavior depends on how hidden vectors are positioned relative to representations from other tokens, contexts, and samples, requiring population-level analysis rather than single-vector inspection (Li et al., 2025). Directly analyzing continuous high-dimensional states, however, is noisy and difficult to compare across samples, layers, and models.

We introduce a representation-centric abstraction, **Hidden States as States (HSS)**, that discretizes continuous hidden representations into consistent geometric states. The key idea is to cluster hidden representations into finite regions of representation space and analyze computation as movement through these states across layers. As shown in Figure 1, directly tracking continuous hidden-state trajectories is noisy and hard to interpret, whereas the discrete state view exposes large-scale geometric patterns such as persistence, transition, emergence, and disappearance.

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

The HSS state map is constructed without labels or interventions. We collect hidden representations from pretrained LMs, cluster them independently within each layer, and align clusters across layers to form a shared discrete state vocabulary. Each input is then represented as a layer-wise sequence of global state IDs. After this label-free map is fixed, downstream labels are used only to analyze or predict how behaviors distribute over the induced states.

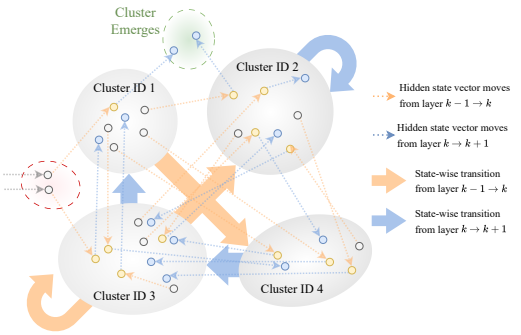


Figure 1: Discretizing hidden states into geometric clusters exposes large-scale patterns. Arrows denote hidden-state movements: vectors may switch clusters, self-loop, or enter a newly emerged cluster (green, layer $k{+}1$); a cluster vanishes when all vectors leave (red, layer k).

Experiments across models from different families and scales reveal consistent geometric patterns in hidden-state organization: intermediate layers often show greater dispersion and finer partitioning while later layers consolidate into fewer active states, and these patterns are stable under clustering seeds, dataset subsampling, and alternative construction choices. Correctness and safety labels concentrate non-uniformly over the induced states, and lightweight count-based predictors over HSS state sequences achieve competitive performance compared to trained probes and support sentence-level early warning during generation. Taken together, our results show that HSS provides a practical, label-free tool for uncovering the geometric structure of hidden representations, characterizing how model internal computations are organized across layers, and converting continuous hidden dynamics into an interpretable discrete form.

2 Related Work

Observation scales for hidden-state analysis. Interpretability methods expose Transformer computation at different scales: readout lenses map intermediate activations to vocabulary-level predictions (nostalgebraist, 2020; Belrose et al., 2023), supervised probes test whether specific properties are encoded in activations (Hewitt & Liang, 2019; Belinkov, 2022), and sparse autoencoders and circuit analyses expose sparse directions or component-level interactions (Cunningham et al., 2023; Conmy et al., 2023; Ameisen et al., 2025). HSS targets a different scale: for a chosen token, span, or response representation, it assigns one geometry-induced state per layer and tracks it across depth, giving a persistent view of where representations lie in the model's own population geometry.

Discrete structure of hidden states. Prior work has shown that hidden representations admit discrete structure: hidden-state clustering reveals task-identity organization (Wu & Varshney, 2025), trajectory-based verifiers separate correct from incorrect reasoning under outcome labels (Liang et al., 2025), and supervised probes find discrete internal states for intermediate results of multi-step arithmetic (Chen et al., 2024). Vector-quantized models and codebook features learn discrete codes by training, the latter changing the network so that it computes through them (Van Den Oord et al., 2017; Tamkin et al., 2023), and mixtures of factor analyzers decompose activations into local regions with their own subspaces (Shafran et al., 2026). HSS needs neither a target variable nor training: it partitions each layer of a frozen model, aligns the partitions into a shared vocabulary, and represents each forward pass as a trajectory over it, making persistence, emergence, and behavior-conditioned occupancy natural objects of analysis; Section 6.2 tests whether its regions carry directions the model's computation relies on. A broader discussion is in Appendix F.

3 From Hidden Vectors to Discrete States

This section describes how HSS converts continuous hidden representations into a finite, layer-spanning vocabulary of geometric states (Figure 2). The construction proceeds in three stages: collecting layer-wise hidden representations at a chosen granularity (§3.1), discretizing them into geometric clusters (§3.2), and aligning clusters across layers into a shared state vocabulary (§3.3). We

then instantiate this pipeline on a concrete testbed and examine descriptive properties of the resulting state sequences (§3.5).

3.1 Hidden-State Collection and Granularity

Consider a Transformer-based language model f_θ with L layers. Given an input prompt x, the model generates a text continuation. Under chain-of-thought settings (Wei et al., 2022), this continuation contains intermediate reasoning before the final answer.[1]

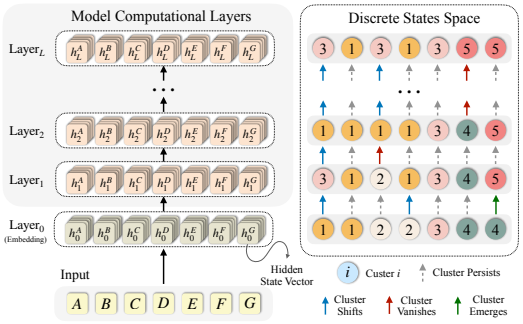


Figure 2: Hidden-state discretization inside a Transformer LM. Left: standard high-dimensional hidden representations. Right: discrete geometric states obtained by unsupervised clustering, tracked globally across layers.

At each token position t, the model produces a hidden representation $h_{l,t}(x) \in \mathbb{R}^d$ at layer $l \in \{0, 1, \ldots, L\}$, where l=0 is the embedding layer. For population-level analysis, we summarize token-level hidden states into one layer-wise representation per sample. Given a dataset $\{x^{(n)}\}_{n=1}^N$, this yields, for each layer l, a representation set $\mathcal{H}_l = \{h_l^{(n)}\}_{n=1}^N$. Here $h_l^{(n)}$ can be a fixed token position such as the last prompt token before generation (Fay et al., 2025), an average over all generated tokens (Ren et al., 2022; Wang et al., 2024), an average over the generated prefix up to each sentence boundary (Bogdan et al., 2025), or a single generated token. This choice sets the scale of the view: prompt-last states capture the pre-generation state, response averages summarize completed generations, and prefix averages support online monitoring. Clustering and alignment are the same for every choice; only the unit of a sample changes (a prompt, a response, a prefix, or a token). Coarser units lose token-level semantic detail but match the level at which behavior is labeled, while finer units stay closer to what the model computes with; each analysis uses one granularity, chosen by its question (Table 4).

3.2 Discretizing Hidden-State Geometry

We discretize hidden representations by clustering them according to geometric similarity. For each layer l, we apply a clustering procedure to $\mathcal{H}_l$, yielding K_l clusters $\mathcal{C}_l = \mathrm{Cluster}(\mathcal{H}_l)$, each occupying a distinct region of the representation space with centroid $\boldsymbol{\mu}_{l,k} \in \mathbb{R}^d$ for $k \in \{1, \ldots, K_l\}$. Each hidden representation is assigned to its nearest cluster center, replacing the continuous vector with a discrete state identifier. This discretization can be viewed as vector quantization (Van Den Oord et al., 2017; Esser et al., 2021) of hidden representations.

The number of clusters K_l may vary across layers depending on the clustering criterion. We cluster each layer independently rather than jointly across layers, allowing each layer to maintain its own geometric partition and enabling analysis of how clusters persist, split, or vanish across depth. Appendix B discusses this design choice.

3.3 State Alignment across Layers

Because discretization is performed independently at each layer, the resulting clusters are layer-local and not directly comparable. However, clusters at different layers may correspond to geometrically similar regions. We align them into global state identities based on centroid similarity.

We begin at layer l=0 by assigning each of the K_0 clusters a unique global state identifier, initializing $S = \{0, 1, \ldots, K_0{-}1\}$. For each subsequent layer, we compute pairwise cosine similarities $\cos(\boldsymbol{\mu}_{l-1,i}, \boldsymbol{\mu}_{l,j})$ between adjacent-layer centroids and apply the Hungarian algorithm (Kuhn, 1955) to find the optimal one-to-one matching. For each matched pair (i, j): if $\cos(\boldsymbol{\mu}_{l-1,i}, \boldsymbol{\mu}_{l,j}) \geq \eta$ (a similarity threshold), cluster j inherits the global ID from cluster i; otherwise, it receives a new ID.

[1] If there is no intermediate sequence, it can be treated as empty.

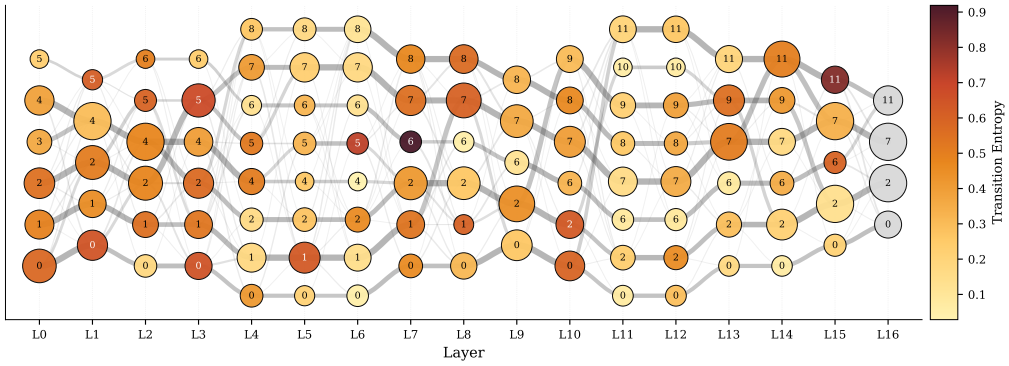


Figure 3: **HSS state map** (Llama-3.2-1B-Instruct, MATH, 5K samples). Columns are layers; nodes are aligned discrete states. Node size indicates sample count, edge opacity transition frequency, and node color outgoing transition entropy.

Unmatched clusters also receive new IDs, expanding S accordingly. This process repeats layer by layer until all clusters are processed; pseudocode is given in Algorithm 1.

The result is a global state set $S = \{0, 1, \ldots, K-1\}$ and a mapping $g(\boldsymbol{\mu}_{l,k}) \mapsto s \in S$ associating each layer-local cluster with a global state identifier. This alignment produces a shared, layer-spanning discrete state vocabulary.

3.4 HSS State Map and State Sequences

The fitted global state vocabulary S, together with the layer-local cluster structure $\{\mathcal{C}_l\}_{l=0}^{L}$ and the alignment mapping g, constitutes the HSS **state map**. Given this state map, each $h_l^{(n)}$ is assigned to its nearest layer-local cluster and then mapped to a global state identifier via g. For a forward pass on input $x^{(n)}$, the layer-wise representations $(h_0^{(n)}, \ldots, h_L^{(n)})$ become a discrete **state sequence** $\mathbf{s}^{(n)} = (s_0^{(n)}, \ldots, s_L^{(n)})$, where each $s_l^{(n)} \in S$. This sequence is analogous to a token sequence over the vocabulary S, except that its positions index layers rather than text positions. The state map is the population-level object used to analyze hidden-state geometry; state sequences are per-input representations used to relate geometric trajectories to generation properties.

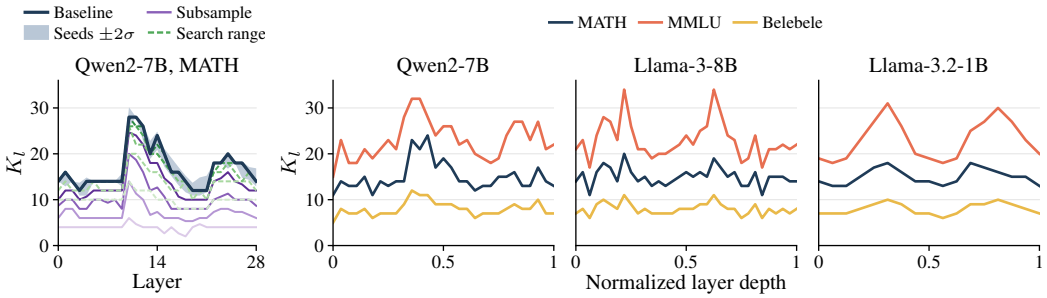
3.5 Instantiation and State Map Properties

We now instantiate the above pipeline on a concrete testbed. Unless otherwise noted, all subsequent experiments use this configuration; alternative choices are evaluated in Section 4 and Appendix B.

Configuration. We use Llama-3.2-1B-Instruct (Grattafiori et al., 2024) on MATH (Hendrycks et al., 2021) with N=5,000 samples, averaging hidden states over all generated tokens to obtain one response-level vector $\mathbf{h}_l^{(n)} \in \mathbb{R}^d$ per sample per layer. At each layer, we fit a diagonal-covariance GMM (Reynolds et al., 2009); the component count K_l is selected via the ICL criterion (Biernacki et al., 2002), $\mathrm{ICL}_l(k) = \mathrm{BIC}_l(k) + 2\,H(\Gamma_l^{(k)})$, which penalizes uncertain assignments; we retain the smallest k within 6% of the minimum ICL (Appendix A.3). Adjacent layers are aligned by cosine similarity between cluster centers via Hungarian matching (η=0.6).[2]

State map. Figure 3 shows the resulting state map: HSS maps 5,000 MATH samples into sequences over 12 global states. Two patterns emerge. First, many states persist across layers with concentrated outgoing transitions, suggesting that samples remain in nearby geometric regions across depth. Second, the number of active states varies non-monotonically, reaching 8 at layers 4–6 and 11–12 and staying between 4 and 6 elsewhere, indicating that some layers partition the hidden space more finely. Section 4 analyzes these layer-wise differences systematically.

[2]Results are stable for $\eta \in [0.3, 0.7]$



(a) Stability under refits

(b) Number of components across models and datasets

Figure 4: **Reliability and generality of the HSS state map** (generation-averaged; K_l: number of clusters at layer l). *(a)* Qwen2-7B-Instruct / MATH: baseline, $\pm 2\sigma$ over seeds, subsamples (f_{data}=0.2–0.8, solid) and search ranges ($k_{\max}$=20–70, dashed), lighter meaning less data or a smaller range; the peak stays at the same layer. *(b)* Three models on MATH, MMLU, and BELEBELE, with K_l selected at a 5% relative-ICL tolerance: profiles are stable within a model and differ across families. Center reproducibility and separation are in Appendix Figure 6.

Sequence statistics. Figure 10 (Appendix C.1) provides three complementary views. *(a)* The five most frequent states account for just over half of all visits, indicating uneven usage without collapse. *(b)* Hierarchical clustering (Ward Jr, 1963) of state sequences by Hamming similarity reveals block structure, indicating shared transition patterns across sample subsets. *(c)* Self-transition probabilities range from about 0.55 to 0.9, so aligned states persist across depth.

Together, these statistics show that the induced state space is structured, and that state sequences provide a compact description of layer-wise hidden space geometry. Section 4 validates this structure under perturbations; Sections 5–6 demonstrate its utility for prediction and behavioral analysis.

4 Reliability and Generality

Before drawing conclusions from the state map, we verify that its structure is a stable property of hidden-state geometry rather than an artifact of the clustering procedure, and test whether it generalizes across models and datasets. All checks use the pipeline of Section 3.5 without task labels, on Qwen2-7B-Instruct (Bai et al., 2023) with MATH.

4.1 Reliability under Procedural Perturbations

Stability of the State Count Trend. We first check whether the layer-wise number of clusters K_l is stable under perturbations of the clustering procedure. We rerun the full pipeline with different initialization seeds, sample sizes (20%–80% of the full dataset), and maximum candidate values of k in the search grid. As shown in Figure 4(a), the shape of the K_l curve is preserved across all settings: the same layer ranges consistently use more clusters, and the peak remains at layer 10. The mean absolute deviation across seeds is less than two clusters per layer, so the trend is not an artifact of a particular seed, sample subset, or search-grid range.

Reproducibility of Cluster Centers. A stable K_l trend alone does not guarantee that the same geometric regions are recovered. We fix each K_l to the default value and refit the GMM under the same perturbations. For each pair of refits, we match cluster centers via Hungarian matching on cosine similarity and compute the average cosine distance between matched pairs. We compare these distances with a random baseline obtained by uniformly assigning samples to clusters. Figure 6(a) shows that refit distances are much smaller than the random baseline (mean cosine distance ≈ 0.1 vs. ≈ 0.65). Thus, the clustering procedure recovers clusters with similar centers across refits, not just a similar number of clusters.

Cluster Separation. We next examine whether the fitted clusters are well separated in the representation space and whether samples within each cluster are concentrated around their fitted component.

Table 1: **Prediction before generation.** Binary prediction of eventual correctness or safety from prompt-last representations. HSS-NB uses discrete state sequences across layers; continuous baselines concatenate prompt-last vectors across all layers. AUROC / Accuracy (%). ⋆count-based discrete; †trained continuous; ‡analytic continuous. Best non-gradient result in **bold**; overall best underlined.

	Reasoning / Capability						Safety	
Method	Qwen2 MATH	Qwen2 THMQA	Qwen2 MMLU	L3 MATH	L3 THMQA	L3 MMLU	L2 JBB	L2 HARMB.
HSS-NB⋆	**77.1** / 70.2	**75.3** / 74.3	**80.6** / **74.9**	**75.1** / 70.8	**74.6** / **76.8**	**79.4** / **73.8**	**97.7** / 97.1	**96.4** / **95.2**
LDA‡	62.9 / 60.6	68.3 / 69.5	70.9 / 70.7	63.3 / 65.7	67.4 / 75.4	70.2 / 69.4	95.8 / 96.5	94.2 / 94.6
Gaussian NB‡	73.7 / **70.5**	71.6 / **75.0**	76.1 / 73.1	68.9 / **72.9**	70.8 / 76.1	75.4 / 72.6	92.9 / 93.6	91.8 / 91.5
MLP†	79.0 / 71.6	78.1 / 73.8	83.1 / 76.0	79.3 / 71.8	77.0 / 76.5	82.0 / 74.9	98.5 / 96.9	97.8 / 95.8
Logistic Reg.†	71.8 / 66.3	74.0 / 73.8	78.1 / 74.2	73.4 / 74.4	73.2 / 76.8	77.3 / 73.6	98.4 / 96.5	97.0 / 95.1
Linear SVM†	72.2 / 66.0	74.4 / 75.8	78.4 / 74.6	72.7 / 76.7	73.6 / 77.2	77.8 / 74.2	98.0 / 93.6	96.7 / 94.4

For each layer l, the fitted GMM represents cluster j as $q_{l,j} = \mathcal{N}(\boldsymbol{\mu}_{l,j}, \boldsymbol{\Sigma}_{l,j})$, where $\boldsymbol{\mu}_{l,j}$ is the cluster center and $\boldsymbol{\Sigma}_{l,j}$ the covariance. From the assigned samples we also estimate $\hat{q}_{l,j} = \mathcal{N}(\hat{\boldsymbol{\mu}}_{l,j}, \hat{\boldsymbol{\Sigma}}_{l,j})$. We compare two symmetric KL quantities: the between-cluster KL, computed between different fitted components in the same layer, and the within-cluster fit KL, computed between $q_{l,j}$ and $\hat{q}_{l,j}$. As shown in Figure 6(b), between-cluster KL remains consistently larger than within-cluster fit KL, indicating that fitted clusters are well separated relative to their internal variation.

4.2 Generalization across Models and Datasets

Section 3.5 showed that K_l varies non-monotonically across layers. We now test whether this pattern is an artifact of one configuration or a general property of hidden-state geometry. We apply the same HSS pipeline to three models: QWEN2-7B-INSTRUCT (Bai et al., 2023), LLAMA-3-8B-INSTRUCT (Grattafiori et al., 2024), and LLAMA-3.2-1B-INSTRUCT (Grattafiori et al., 2024), across three datasets: MATH (Hendrycks et al., 2021) (N=5,000), MMLU (Hendrycks et al., 2020) (N=14,000), and BELEBELE (Bandarkar et al., 2024) (N=2,100, pooling English, German, and Chinese splits). All representations are generation-averaged hidden states with greedy decoding.

Figure 4(b) shows the resulting K_l profiles, with layer depth normalized to $[0, 1]$ for cross-architecture comparison. Within each model, the shape of the K_l curve is broadly stable across datasets despite differences in domain, language, and sample size, indicating that the non-monotonic pattern reflects intrinsic hidden-state geometry. Across model families, the profiles differ: Llama models tend toward two expansion regions in the early-middle and late-middle layers, while Qwen2 concentrates most of its expansion in a single mid-layer region. These results show that different model families organize their internal computation at different layers, and that HSS reliably distinguishes these family-level patterns without supervision.

5 Prediction and Monitoring from State Sequences

We now ask whether state sequences carry information about generation properties, first from the prompt before generation (§5.1) and then online from partial generations (§5.2).

5.1 Prediction Before Generation

This experiment evaluates whether a prompt-induced state sequence, obtained before generation begins, can support lightweight count-based prediction of eventual response properties.

Datasets, labels, and models. We evaluate on datasets spanning two behavioral domains. For reasoning and capability: MATH, THEOREMQA (Chen et al., 2023), and MMLU (Hendrycks et al., 2020). For safety: JAILBREAKBENCH (Chao et al., 2024) and HARMBENCH (Mazeika et al., 2024). Each prompt receives a binary label: correct vs. incorrect (by answer matching against ground truth) or safe vs. unsafe (by Llama Guard (Inan et al., 2023)). We use QWEN2-7B-INSTRUCT (Bai et al.,

2023), LLAMA-3-8B-INSTRUCT (Grattafiori et al., 2024), and LLAMA-2-7B-CHAT (Touvron et al., 2023); all generations use greedy decoding with a zero-shot chain-of-thought prompt.

Hidden-state extraction and HSS construction. For prediction before generation, we use the hidden state at the final prompt token of every layer (prompt-last; §3.1). The map is built as in §3.5 on the training split and frozen; each prompt's state sequence $\mathbf{s}^{(n)} = (s_0^{(n)}, \ldots, s_L^{(n)})$ is the only input to the classifier.

Predictor: HSS-NB. Once the state map is fixed, every training example is represented as a state sequence $\mathbf{s} = (s_0, \ldots, s_L)$. For each layer ℓ and label y, we estimate $p(s_\ell \mid y)$ from state counts using Laplace smoothing ($\alpha = 1$) over the frozen state vocabulary $\mathcal{S}$; $p(y)$ is the training-set label frequency. Treating layers as conditionally independent given y, we compute $a_y(\mathbf{s}) = \log p(y) + \sum_{\ell=0}^{L} \log p(s_\ell \mid y)$. We predict $\hat{y} = \arg\max_y a_y(\mathbf{s})$ and use the posterior log odds $\Delta(\mathbf{s}) = a_1(\mathbf{s}) - a_0(\mathbf{s})$ for AUROC. Prediction requires only count estimation and no gradient-based optimization.

Baselines and evaluation. We compare HSS-NB against supervised continuous baselines fit with the same training labels. LDA and Gaussian NB are analytic baselines; MLP, logistic regression, and linear SVM are optimization-based. All use the concatenation of prompt-last hidden vectors across all layers and the same stratified 40%/60% train/test split. We report test AUROC and classification accuracy using each classifier's standard decision rule.

Results. Table 1 shows that HSS-NB consistently outperforms all closed-form continuous baselines in AUROC despite using only discrete state IDs and count estimates. For example, on Qwen2-MATH it reaches 77.1 AUROC, improving over Gaussian NB (73.7) and logistic regression (71.8) by a clear margin, and falling only 1.9 points below the gradient-trained MLP. On JAILBREAKBENCH, HSS-NB reaches 97.7, within 0.8 points of the MLP.

These results indicate that correctness and safety are reflected in hidden-state geometry. Although the HSS state map is built entirely without labels, examples with different outcomes occupy the induced states differently across layers. A simple predictor that counts how often each state is associated with each label can therefore approach the performance of trained probes, showing that HSS preserves behaviorally relevant structure in a compact discrete form.

5.2 SENTENCE-LEVEL EARLY FAILURE WARNING

We next ask whether HSS can serve as an online monitor: warning, before a response is complete, that the final answer will fail.

Task. We evaluate on MATH with QWEN2-7B-INSTRUCT, treating an incorrect final answer as a failure. Each response is segmented into sentences. At the end of each sentence boundary, the monitor observes only the generated prefix so far and outputs a danger score.

State construction. For each sentence boundary and layer, we average hidden states over all generated prefix tokens, yielding one vector per response, layer, and boundary. We first make a stratified 40/20/40 response-level train/validation/test split and then fit the HSS map only on training prefixes.

Table 2: **Sentence-level early warning** on MATH (QWEN2-7B-INSTRUCT). Thresholds target 10% validation FAR; @50% uses the last sentence boundary at or before 50% token progress. Best AUROC, detection, and savings in **bold**.

	AUROC (%)		Detected	FAR	Saved
Method	**@50%**	**@Final**	**(%)**	**(%)**	**(%)**
Prefix entropy	62.1	66.8	27.4	8.6	18.5
Prefix logprob	59.8	65.1	24.1	7.9	16.2
Continuous probe	**77.6**	**80.4**	40.5	9.9	28.1
HSS-NB	75.9	78.3	**43.3**	10.2	**30.3**

Scoring and evaluation. We reuse the count-based predictor from §5.1, fit on training-prefix state sequences. At each boundary, the posterior log odds of failure versus success serve as the danger

score; its threshold is calibrated to 10% response-level FAR on validation responses. A successful response is a false alarm if any non-final boundary crosses the threshold; a failed response is detected if any non-final boundary does so, with the first crossing defining detection time. Saved is the mean avoidable-token fraction over all failed responses, assigning zero to undetected failures. We report test FAR, detection, saved tokens, and AUROC at the last boundary at or before 50% token progress and at the final boundary.

Baselines. Prefix entropy and prefix log-probability are mean token-level scores over the observed prefix. A regularized logistic-regression probe uses the concatenation of the same prefix-average hidden vectors across all layers, providing a continuous counterpart to HSS-NB.

Results. As shown in Table 2, at a nearly identical realized test FAR (10.2% vs. 9.9%), HSS-NB detects 43.3% of failed responses before completion and saves 30.3% of generated tokens, compared with 40.5% detected and 28.1% saved by the continuous prefix probe. Both hidden-state methods detect substantially more failures than prefix entropy (27.4%) or prefix log-probability (24.1%). For HSS-NB, AUROC increases from 75.9 by 50% progress to 78.3 at the final boundary, indicating that failure-relevant information becomes increasingly readable as generation proceeds. Thus, the frozen state map supports early failure warning using only count-based scoring.

6 Characterizing and Intervening on States

We first use external labels to characterize what the states capture (§6.1), and then intervene on hidden states to test whether region-specific directions affect the model's downstream computation (§6.2–6.3), using Qwen2-7B-Instruct.

6.1 States Mix Problem Type, Difficulty, and Outcome

Each state of the generation-averaged MATH map (Appendix Figure 12) is a coarse region of response-level hidden space, so one state can reflect several properties of a generation at once. Problem type, difficulty, and correctness are all associated with the state assignment at every layer (chi-square, $p < 10^{-10}$), with Cramér's V largest for problem type, then correctness, then difficulty (Appendix Figure 13).

Type and difficulty. Problem type is the strongest axis but does not determine the state: responses of the same type are divided among several states according to difficulty, and these divisions change across layers.

A sink state. One state appears in early layers and persists to the last layer with very low accuracy, so its members can be tracked across depth. Its responses span several problem types, but about 80% of them contain no `\boxed` answer (Appendix C.4), suggesting a shared failure mode in which generation ends without the required final answer. Failures that share this mode, rather than a topic, are thus gathered into one region. These associations are correlational. We next test whether the geometry of such regions matters to the model's computation.

6.2 Reconstructing and Removing Region-Specific Directions

The generation-averaged map above summarizes many token activations into one response-level vector, which suits describing behavior but is not a representation the model computes with. To test whether regions matter to the model's computation, we therefore apply the same construction to single tokens and edit them directly. We fit a map with $K{=}64$ components at layer 14 of Qwen2-7B-Instruct on the first 16 generated tokens of 3,011 MATH training problems. For each region s with center $\boldsymbol{\mu}_s$, *Local8* denotes the 8 directions U_s along which its member tokens vary most around the center. We then take 64 GSM8K problems that play no part in the fit, edit the layer-14 states of their first 16 generated tokens once, let the model run, and compare it with the unedited model: KL is the divergence of the next-token distribution at position 17, and ΔNLL the change in per-token negative log-likelihood of the original continuation (Appendix D).

Keeping region-specific directions. Replacing each token by its region center keeps only its region identity, and the output changes substantially (KL 5.40). We then add back 8 coordinates of the

Table 3: **Keeping and removing 8 directions of layer-14 token states** (Qwen2-7B-Instruct, 64 GSM8K problems; KL of the next-token distribution and ΔNLL of the continuation, both relative to the unedited model). *Left:* each token is replaced by its region center plus 8 coordinates along B; lower is closer to the original model. *Right:* 8 directions are removed from the intact token with equal edit size; higher means the output is more sensitive to them.

Keep (center + 8 coordinates) Directions B	KL ↓	ΔNLL ↓	KL vs. center
None (center only)	5.40	0.103	–
Wrong-local8	5.27	0.100	−2%
Shared8	4.30	0.094	−20%
Local8	**2.89**	**0.065**	**−46%**

Remove (equal edit size) Directions	KL ↑
Random8	0.022
Local8	**0.332**
Local8 / Random8	15×

residual, $\hat{\mathbf{h}} = \boldsymbol{\mu}_s + BB^\top(\mathbf{h} - \boldsymbol{\mu}_s)$, and compare three choices of the 8 directions B, all with the same center and the same number of coordinates: the region's own directions (Local8), 8 directions shared by all regions and obtained from the pooled residuals (Shared8), and the directions of a different region (Wrong-local8). Relative to the center alone, Local8 reduces the KL by 46% and ΔNLL by 37%, more than twice the reduction of Shared8 (20% and 9%), whereas Wrong-local8 reduces them by only 2–3% (Table 3, left). The useful directions therefore depend on the region the token is in.

Removing region-specific directions. Keeping more of a vector brings the output closer in any case, so the advantage of Local8 could simply reflect a better compression. We therefore reverse the test on intact activations: we subtract only the Local8 component, $\mathbf{h}' = \mathbf{h} - U_s U_s^\top(\mathbf{h} - \boldsymbol{\mu}_s)$, and compare with removing a random 8-dimensional subspace scaled at every token to the same edit size (*Random8*). Removing Local8 changes the next-token distribution 15 times as much as Random8 (KL 0.332 vs. 0.022; Table 3, right), with a reliably positive paired difference (Appendix D), yet 98.3% of tokens stay in their region. Region identity thus points to the directions to which the downstream computation is sensitive, while the coordinates along these directions carry information that the identity alone does not.

6.3 Exploratory Steering Along a Sink Direction

Finally, we test a direction derived from the clearest sink state. On the generation-averaged map at middle layer (layer 14), we take the unit direction $\mathbf{v}$ from the mean of other responses to the mean of the sink-state responses (325 responses, 9 correct), and a threshold τ, the 75th percentile of single-token projections in other responses. At every generated token, the part of the projection above τ is removed, $\mathbf{h}' = \mathbf{h} - \max(\mathbf{h}^\top\mathbf{v} - \tau,\, 0)\,\mathbf{v}$, and tokens below τ are left unchanged; no correctness label enters $\mathbf{v}$ or τ. The clamp acts selectively. When sink-state responses not used to construct $\mathbf{v}$ are fed back to the model, it lowers the likelihood of their continuations 2.8 times as much as ten orthogonal edits of the same size, and it leaves correct responses from other states essentially unchanged (Appendix E).

In free generation with greedy decoding on 202 problems, 39 responses gain a complete `\boxed` answer under the clamp and 11 lose one, and 10 incorrect answers become correct while 4 correct ones become incorrect. This is a proof of concept that the sink state corresponds to a direction the model's computation responds to: moving away from it helps the model complete an answer. Since the sink state gathers failures to complete rather than specific reasoning errors, effects on correctness need further study.

7 Conclusion

We introduced Hidden States as States (HSS), a training-free and label-free framework that discretizes hidden representations into a vocabulary of geometric states shared across layers. HSS maps are stable under clustering perturbations, show model-specific layer profiles, and organize samples by problem type and outcome, with failures gathered into persistent sink states. Count-based predictors over state sequences approach trained probes and support early failure warning, and token-level tests show that a region identifies directions to which downstream computation is sensitive. Limitations are discussed in Appendix G.

AI USE STATEMENT

We used large language models (LLMs) in three ways. *Writing:* LLMs aided and polished the writing, including rephrasing, restructuring, and LaTeX formatting; the authors revised and approved all text. *Retrieval:* LLMs helped search for related work; every cited paper was read and checked by the authors. *Research execution and ideation:* LLM-based coding assistants helped implement and run parts of the analysis and experiment code, including the token-level tests of Section 6, and re-plot figures from existing result files, following protocols specified by the authors; LLMs were also used to discuss experimental controls and presentation. The authors designed the method and the experiments, reviewed all AI-generated code and text, verified every reported number against the experiment outputs, and take full responsibility for the paper, its claims, and its accompanying materials.

ETHICS STATEMENT

The study analyzes publicly available language models and benchmark datasets and does not involve new human-subject research or the collection of private participant data. Safety-related experiments use existing benchmarks and automated safety labels.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A specify the state-map construction, including hidden-state extraction, ICL selection, and the alignment procedure (Algorithm 1); Table 4 lists the model, data, split, and granularity of every experiment. Appendix B reports the robustness checks, and Appendices D and E give the protocols, controls, and statistics of the token-level tests. All models and datasets are publicly available, and we will release the code in the final version.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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. `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 749–775, 2024.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725, 2002.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter?, 2025. URL https://arxiv.org/abs/2506.19143.

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. https://transformer-circuits.pub/2023/monosemantic-features/index.html, 2023.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Junhao Chen, Shengding Hu, Zhiyuan Liu, and Maosong Sun. States hidden in hidden states: Llms emerge discrete state representations implicitly. *arXiv preprint arXiv:2407.11421*, 2024.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, 2023.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 16318–16352, 2023.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. https://transformer-circuits.pub/2021/framework/index.html, 2021.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12873–12883, 2021.

Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

Aideen Fay, Inés García-Redondo, Qiquan Wang, Haim Dubossarsky, and Anthea Monod. The shape of adversarial influence: Characterizing llm latent spaces with persistent homology. *arXiv preprint arXiv:2505.20435*, 2025.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Yuri Gardinazzi, Karthik Viswanathan, Giada Panerai, Alessio Ansuini, Alberto Cazzaniga, and Matteo Biagetti. Persistent topological features in large language models. *arXiv preprint arXiv:2410.11042*, 2024.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Yanhong Li, Ming Li, Kar en Livescu, and Jiawei Zhou. On the predictive power of representation dispersion in language models. *arXiv preprint arXiv:2506.24106*, 2025.

Zhenwen Liang, Ruosen Li, Yujun Zhou, Linfeng Song, Dian Yu, Xinya Du, Haitao Mi, and Dong Yu. Clue: Non-parametric verification from experience via hidden-state clustering. *arXiv preprint arXiv:2510.01591*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Alexander Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities. *Advances in Neural Information Processing Systems*, 37:8901–8929, 2024.

nostalgebraist. Interpreting GPT: The logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`, 2020. LessWrong.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. Out-of-distribution detection and selective generation for conditional language models. *arXiv preprint arXiv:2209.15558*, 2022.

Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663):3, 2009.

Or Shafran, Shaked Ronen, Omri Fahn, Shauli Ravfogel, Atticus Geiger, and Mor Geva. From directions to regions: Decomposing activations in language models via local geometry. *arXiv preprint arXiv:2602.02464*, 2026.

Alex Tamkin, Mohammad Taufeeque, and Noah D Goodman. Codebook features: Sparse and discrete interpretability for neural networks. *arXiv preprint arXiv:2310.17230*, 2023.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 51234–51252, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. Latent space chain-of-embedding enables output-free LLM self-evaluation. *arXiv preprint arXiv:2410.13640*, 2024.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 24824–24837, 2022.

Xinbo Wu and Lav R Varshney. Transformer-based causal language models perform clustering. In *Findings of the Association for Computational Linguistics: NAACL*, pp. 5347–5372, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

A Implementation and Reproducibility Details

This appendix specifies how state maps are built and evaluated. Every map is fit without labels; labels are attached only after a map is frozen, either to characterize states or to train count-based predictors.

A.1 Models, datasets, and experimental roles

Table 4 lists the model, data, fitting set, and granularity of every experiment. Descriptive maps (Sections 3.5, 4, and 6.1) are fit on all samples of a dataset. Maps used for prediction, monitoring, and reconstruction and removal are fit on a training split and frozen before any evaluation sample is assigned; the steering map of Section 6.3 is fit on all responses.

Table 4: **Experimental configurations.** "Fit on" is the set of representations on which the state map is fit.

Role	Model(s)	Data (N)	Fit on	Granularity
Default map (§3.5)	Llama-3.2-1B-Instruct	MATH (5,000)	all samples	Generation-avg.
Reliability (§4.1)	Qwen2-7B-Instruct	MATH (5,000)	all samples; seeds, subsamples	Generation-avg.
Cross-model (§4.2)	Qwen2-7B-Instruct, Llama-3-8B-Instruct, Llama-3.2-1B-Instruct	MATH (5,000), MMLU (14,000), BELEBELE (2,100)	all samples, one map per pair	Generation-avg.
Prediction (§5.1)	Qwen2-7B-Instruct, Llama-3-8B-Instruct, Llama-2-7B-Chat	MATH, THEOREMQA, MMLU, JAILBREAKBENCH, HARMBENCH	40% train split	Prompt-last
Monitoring (§5.2)	Qwen2-7B-Instruct	MATH	40% train split, all boundaries	Prefix-avg.
Characterization (§6.1)	Qwen2-7B-Instruct	MATH (5,000)	all samples	Generation-avg.
Reconstruction, removal (§6.2)	Qwen2-7B-Instruct, layer 14	MATH train (3,011); GSM8K (64)	MATH train tokens	Single-token
Steering (§6.3)	Qwen2-7B-Instruct, layer 14	MATH (5,000); 202 regenerated problems	all samples	Generation-avg. map; token-level edit

Models. Llama-3.2-1B-Instruct (L=16, d=2048), Llama-3-8B-Instruct (L=32, d=4096), Llama-2-7B-Chat (L=32, d=4096), and Qwen2-7B-Instruct (L=28, d=3584), loaded from HuggingFace. Including the embedding layer, a model with L blocks yields $L+1$ layers of hidden states.

Datasets and labels. BELEBELE pools the English, German, and Chinese splits. For reasoning and capability tasks, the final answer is extracted from the response (the last `\boxed` expression for MATH and THEOREMQA, the selected option for MMLU) and compared with the ground truth after normalization; a response without an extractable answer is labeled incorrect. For safety tasks, each response is labeled safe or unsafe by Llama Guard (Inan et al., 2023). In prediction and monitoring, the positive class is the incorrect or unsafe response. The 64 GSM8K problems of Section 6.2 are drawn from the official training split by a fixed hash before any generation and are not used in any fit.

Prompts and decoding. Reasoning and capability prompts are zero-shot chain-of-thought instructions in the model's chat template, asking the model to reason step by step and to state its final answer

(in `\boxed` for MATH and THEOREMQA); safety prompts are the benchmark prompts. Responses are generated with greedy decoding and repetition penalty 1.05 until the end-of-sequence token or the length limit. The free-generation runs of Section 6.3 regenerate responses with repetition penalty 1.0 in every condition and at most 2,048 new tokens.

Compute. Experiments run on a single NVIDIA A10 GPU (24 GB) with 32 CPU cores and 128 GB of memory. Extracting generation-averaged hidden states for MATH (N=5,000) takes about 3 hours for Qwen2-7B-Instruct and 5 hours for Llama-3-8B-Instruct. No multi-GPU setup is required.

A.2 HIDDEN-STATE EXTRACTION

Let $h_{l,t}^{(n)} \in \mathbb{R}^d$ be the hidden representation of sample n at layer l and token position t. Layer l is the output of the l-th Transformer block and l=0 is the token embedding (`hidden_states[l]` in HuggingFace Transformers). Section 3.1 uses four granularities.

Prompt-last. The hidden state at the final prompt token, $h_l^{(n)} = h_{l,t_{\text{prompt end}}}^{(n)}$, available before any response token is generated. Because the final prompt token is the same template token for every prompt, all samples share one state at layer 0.

Generation-averaged. The mean over the generated tokens T_n of a response,

$$h_l^{(n)} = \frac{1}{|T_n|} \sum_{t \in T_n} h_{l,t}^{(n)}.$$

At layer 0, this is the mean token embedding of the response.

Prefix-average. For sentence boundary b, the mean over the generated prefix $T_{n,b}$ up to b,

$$h_{l,b}^{(n)} = \frac{1}{|T_{n,b}|} \sum_{t \in T_{n,b}} h_{l,t}^{(n)},$$

so the monitor uses only tokens generated before the boundary it scores.

Single-token. For the tests of Section 6.2, each of the first 16 generated tokens of a response is one sample, $h_{14,t}^{(n)}$ for $t = 1, \ldots, 16$.

A.3 PER-LAYER CLUSTERING AND K SELECTION

For each layer l, HSS fits an independent diagonal-covariance GMM to the raw representations $\mathcal{H}_l = \{h_l^{(n)}\}_{n=1}^N$ by expectation–maximization, initialized with k-means and a fixed random seed. The number of clusters K_l is selected per layer by the integrated completed likelihood (Biernacki et al., 2002),

$$\mathrm{ICL}_l(k) = \mathrm{BIC}_l(k) + 2H(\Gamma_l^{(k)}), \qquad \mathrm{BIC}_l(k) = -2\log \hat{\mathcal{L}}_l(k) + p_k \log N,$$
$$H(\Gamma_l^{(k)}) = -\textstyle\sum_{n,j} \gamma_{nj}^{(k)} \log \gamma_{nj}^{(k)},$$

where $\hat{\mathcal{L}}_l(k)$ is the maximized likelihood, $\gamma_{nj}^{(k)}$ the posterior probability of cluster j for sample n, and $p_k = k - 1 + 2kd$ the number of free parameters of a k-component diagonal GMM; the entropy term penalizes uncertain assignments. Candidates range over a grid of k up to $k_{\max}$ ($k_{\max}$=80 for the baseline of Figure 4(a)). Among near-optimal candidates we keep the smallest, namely the smallest k whose relative ICL,

$$\text{rel-ICL}_l(k) = \frac{\mathrm{ICL}_l(k) - \min_{k'} \mathrm{ICL}_l(k')}{|\min_{k'} \mathrm{ICL}_l(k')|},$$

is at most a tolerance τ_{ICL}. The default configuration of Section 3.5 (Figure 3) uses τ_{ICL}=6%, which applies unless stated otherwise; Figure 4(b) uses 5% and the steering map of Section 6.3 uses 2%.

The ICL surface is flat near its minimum (Figure 9). Refits of the same configuration therefore agree on where K_l expands but can differ by several clusters per layer, and the level of K_l grows with N (Figure 4(a)); we compare K_l profiles by their shape.

A.4 CROSS-LAYER STATE ALIGNMENT

Layer-local clusters are aligned into a shared vocabulary by matching centers of adjacent layers with cosine similarity and the Hungarian algorithm (Algorithm 1). A cluster at layer l inherits the global ID of its matched predecessor only if their cosine similarity is at least η=0.6; otherwise it starts a new global state. Each Transformer block adds its output to the residual stream (Elhage et al., 2021), so adjacent layers share one coordinate system and center similarity compares regions in the same basis; cosine similarity is used because representation norms change with depth.

The matching is one-to-one, so a state persists, splits, or vanishes as follows. A state persists when its cluster is matched to a cluster at the next layer with similarity at least η. A state splits when its members move to two or more clusters at the next layer: the matched child inherits the ID and the other children start new IDs, and the transition frequencies from the parent to each child record the split. A state vanishes when no cluster at the next layer inherits its ID. Conversely, a cluster that receives the members of two predecessors inherits at most one ID, and its transition frequencies record both incoming flows.

Algorithm 1 Greedy layer-wise alignment via Hungarian matching

Require: Centers $\{\boldsymbol{\mu}_{l-1,i}\}_{i=1}^{K_{l-1}}$, $\{\boldsymbol{\mu}_{l,j}\}_{j=1}^{K_l}$; threshold η; map g_{l-1}; counter G.
Ensure: Map g_l; updated G.

```
a_ij ← cos(μ_{l−1,i}, μ_{l,j}) for all i, j
M ← HUNGARIAN(−[a_ij]) {max-weight matching}
for j = 1 to K_l do
    if ∃ i : (i, j) ∈ M and a_ij ≥ η then
        g_l(j) ← g_{l−1}(i) {inherit}
    else
        g_l(j) ← G;    G ← G + 1 {emerge}
    end if
end for
```

A.5 COUNT-BASED PREDICTOR AND BASELINES

HSS-NB. After the map is frozen, each example is a state sequence $\mathbf{s} = (s_0, \ldots, s_L)$. For each layer ℓ and label y, $p(s_\ell \mid y)$ is estimated from training counts with Laplace smoothing (α=1) over the frozen state vocabulary $\mathcal{S}$, so a test state never observed with label y receives the smoothed probability; $p(y)$ is the training label frequency. The score is $a_y(\mathbf{s}) = \log p(y) + \sum_{\ell=0}^{L} \log p(s_\ell \mid y)$, the prediction is $\arg\max_y a_y(\mathbf{s})$, and AUROC uses the log odds $a_1(\mathbf{s}) - a_0(\mathbf{s})$. Because the clusters of each layer map one-to-one to global IDs, HSS-NB depends only on the per-layer partitions and not on the alignment threshold η.

Continuous baselines. LDA, Gaussian NB, logistic regression, linear SVM, and an MLP are trained on the concatenated prompt-last vectors of all layers, with the same labels and the same stratified 40%/60% split as HSS-NB. Features are standardized with training statistics; logistic regression and the linear SVM use ℓ_2 regularization, and the MLP has one hidden layer and is trained with Adam and early stopping on a held-out part of the training split. Accuracy uses each classifier's standard decision rule.

A.6 SENTENCE-LEVEL MONITORING PROTOCOL

Responses are segmented into sentences at sentence-final punctuation followed by whitespace and at line breaks; the final boundary is the end of the response. Responses are split 40/20/40 into training, validation, and test sets, stratified by outcome. One map is fit on the prefix-average representations of all boundaries of the training responses, because an online monitor must score a prefix at any point of generation, and HSS-NB is trained on these boundary state sequences with the label of their response. The danger score of a boundary is the posterior log odds of failure; its threshold is set to give 10% response-level false alarms on validation responses. On test responses, a successful response is a false alarm if any non-final boundary crosses the threshold, and a failed response is

detected if any non-final boundary does, the first crossing defining the detection time. Saved is the fraction of generated tokens after the detection time, averaged over all failed responses with zero for undetected failures. AUROC is computed at the last boundary at or before 50% of the response tokens and at the final boundary. The continuous probe is a regularized logistic regression on the concatenated prefix-average vectors of all layers, trained on the same boundaries; prefix entropy and prefix log-probability are mean token-level scores over the observed prefix.

B Robustness and Construction Controls

Section 4 shows that the K_l profile is stable under clustering seeds, subsampling, and the search range. This appendix discusses the two construction choices of Section 3, per-layer clustering and the alignment threshold, and reports the remaining checks. Unless stated otherwise, all checks use Qwen2-7B-Instruct on MATH with generation-averaged representations (N=5,000).

B.1 Per-layer versus global clustering

HSS clusters each layer separately and links the clusters by alignment (Section 3.2). The alternative is to stack the representations of all 29 layers into one $[N(L{+}1) \times d]$ matrix and fit a single mixture. Figure 5 shows the result with 44 components: for each global cluster c, the fraction $p(l \mid c)$ of its members that come from layer l. Every cluster is confined to a contiguous block of layers, and about two thirds of the clusters draw almost all of their members from a single layer. The components are also spent unevenly across depth: layers 0–18 share only seven clusters, so the within-layer structure of the early and middle layers, including the mid-layer expansion of Figure 4(a), is lost, while the last layer alone receives ten. Representation norms and directions change systematically with depth, so a single mixture mainly separates layers and describes depth rather than how samples partition within a layer. Per-layer clustering instead models the partition of each layer directly, adapts K_l to each layer, and leaves the cross-layer link to the alignment, whose only role is to assign persistent IDs.

B.2 Alignment threshold

The threshold η decides whether a matched cluster inherits the ID of its predecessor or starts a new state. It changes only which clusters share an ID across adjacent layers: the per-layer partitions, and hence K_l, the per-layer label associations, and HSS-NB (Appendix A.5), are the same for every η. A lower threshold links more clusters into persistent states, and a higher one starts more new states. All results in the paper use η=0.6.

B.3 Center reproducibility and cluster separation

Figure 6 shows the two checks of Section 4.1. *(a)* With each K_l fixed to its default value, centers refit under different seeds and subsamples are matched by Hungarian matching on cosine similarity, and the mean cosine distance of matched pairs is compared with centers obtained from random assignments of samples to clusters. *(b)* For each layer, the fitted component $q_{l,j} = \mathcal{N}(\boldsymbol{\mu}_{l,j}, \boldsymbol{\Sigma}_{l,j})$ is compared with $\hat{q}_{l,j} = \mathcal{N}(\hat{\boldsymbol{\mu}}_{l,j}, \hat{\boldsymbol{\Sigma}}_{l,j})$ estimated from its assigned samples. Between-cluster symmetric KL, computed between different components of a layer, exceeds within-cluster fit KL, computed between $q_{l,j}$ and $\hat{q}_{l,j}$, at every layer.

B.4 Clustering-algorithm robustness

To test whether the layer profile is specific to GMMs, we repeat the construction with MiniBatch KMeans, selecting K per layer by the silhouette score. Silhouette selects two to three clusters in most layers and many more at layers 10 and 20 (Figure 7); layer 10 is also where the GMM profile peaks (Figure 4(a)). Clusters whose accuracy deviates most from the average appear from these middle layers onward (Figure 8). The mid-layer expansion is therefore not specific to GMMs, although silhouette selects much coarser partitions than ICL in the other layers.

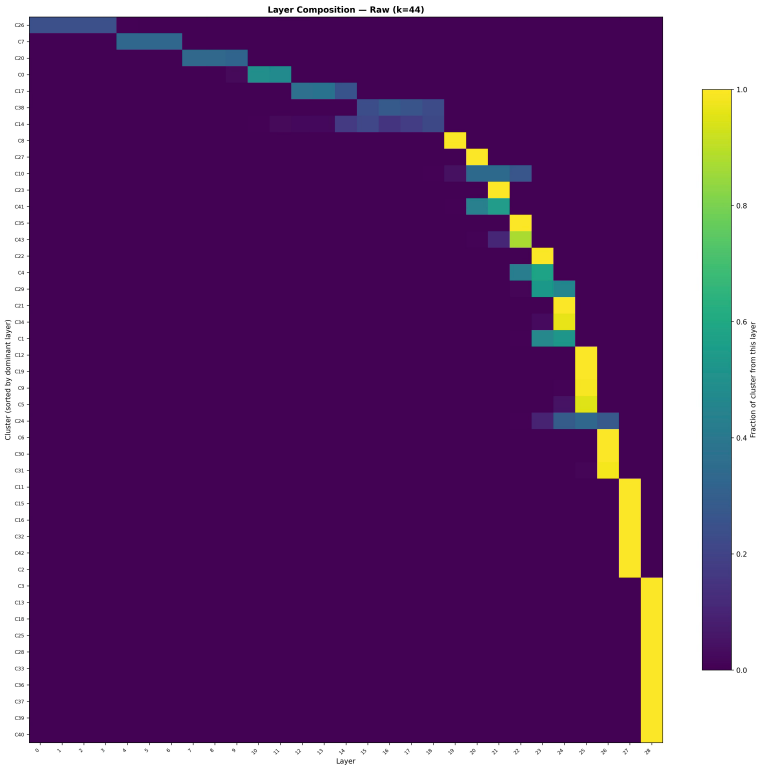
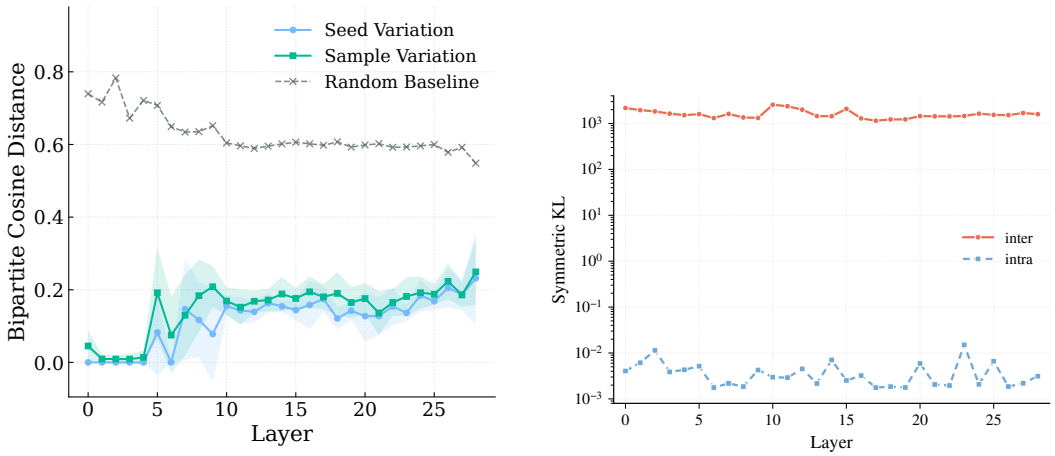


Figure 5: **Global clustering of all layers** (Qwen2-7B-Instruct / MATH, generation-averaged, raw representations; one mixture with 44 components fit on the stacked representations of layers 0–28). Each row is a global cluster, sorted by its dominant layer; color is the fraction of the cluster's members that come from each layer. Clusters follow depth, and the last layers receive most of the components.



(a) Center reproducibility

(b) Cluster separation

Figure 6: **Center reproducibility and separation** (Qwen2-7B-Instruct / MATH, generation-averaged). *(a)* Mean cosine distance between matched centers of refits under seed and sample variation, and between centers of random assignments. *(b)* Between-cluster and within-cluster symmetric KL per layer (log scale).

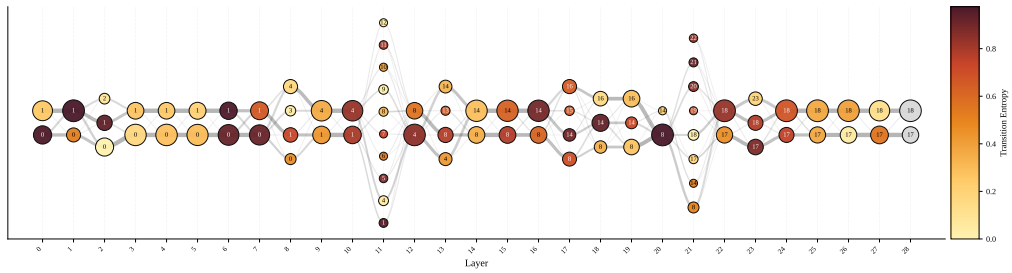


Figure 7: **KMeans state map** (Qwen2-7B-Instruct / MATH, generation-averaged, K_l selected by silhouette). Node size: number of samples; color: outgoing transition entropy.

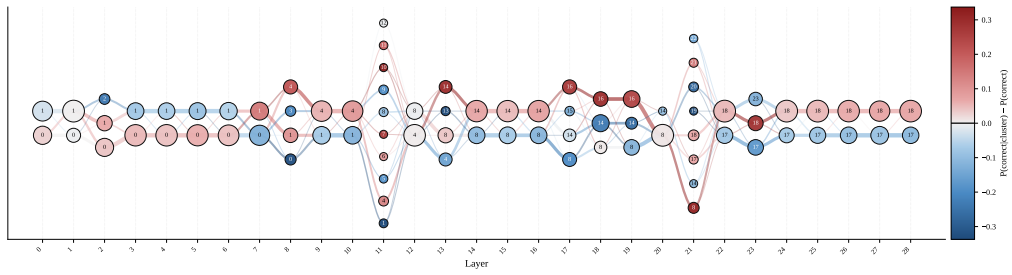


Figure 8: **KMeans state map with correctness.** The map of Figure 7; node color is $\Delta_l(s) = P(\text{correct} \mid s_l{=}s) - P(\text{correct})$.

B.5 ICL SURFACE

Figure 9 shows the relative ICL (Appendix A.3) over candidate k and layer. In the layers where K_l expands, a broad range of k lies within a few percent of the minimum, so the expansion is supported by many candidates rather than by one unstable minimum. For the same reason, the absolute K_l depends on the tolerance τ_{ICL}, whereas the layers at which K_l expands do not.

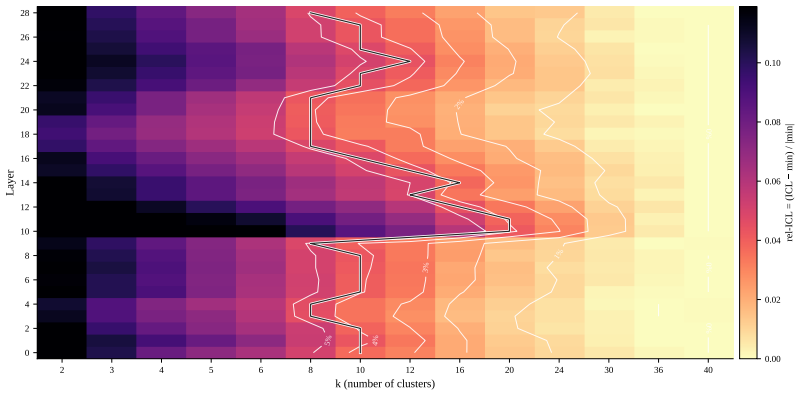


Figure 9: **Relative ICL surface** (Qwen2-7B-Instruct / MATH, generation-averaged). Color: $\text{rel-ICL}_l(k)$; white lines: contours from 0% to 5%; black line: selected K_l.

B.6 COARSENESS OF THE STATE ABSTRACTION

Any coarse abstraction maps distinct vectors to the same state. The level of detail is set by two explicit choices. The first is the number of clusters, which grows with the number of samples and with a smaller tolerance τ_{ICL} (Figure 4(a)). The second is the granularity: a response-level map summarizes whole responses, whereas the token-level map of Section 6.2 has 64 regions at one layer, and there a region with 8 local coordinates preserves the next-token distribution better than 8 shared

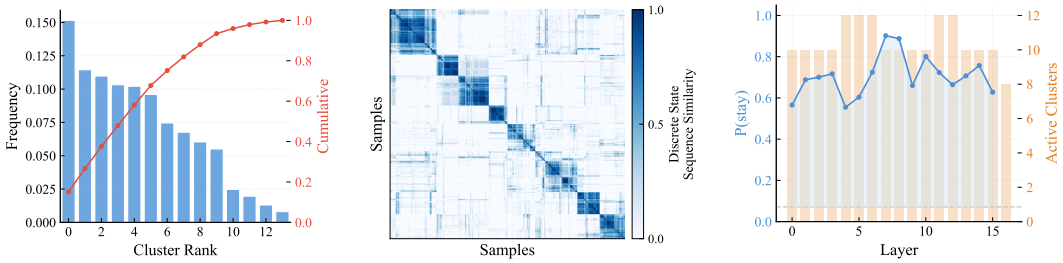
coordinates (Table 3). HSS is therefore a family of maps at adjustable resolution rather than a unique discretization, and each experiment states the resolution it uses (Table 4).

C State Sequences and State Characterization

This appendix supports Sections 3.5 and 6.1: it shows the sequence statistics of the default configuration, the prompt-last and generation-averaged maps of Qwen2-7B-Instruct, the label associations, and the sink state.

C.1 State sequence statistics

Figure 10 summarizes the state sequences $\mathbf{s}^{(n)} = (s_0^{(n)}, \ldots, s_L^{(n)})$ of Llama-3.2-1B-Instruct on MATH (Section 3.5). The statistics come from a separate fit of this configuration, so per-layer state counts can differ from those of Figure 3. *(a)* Usage is uneven across states but not collapsed onto a few of them. *(b)* Ordering samples by hierarchical clustering (Ward Jr, 1963) of the Hamming similarity between their sequences reveals blocks of samples that share most of their trajectory. *(c)* The self-transition probability, the fraction of samples that keep their state ID from one layer to the next, stays high across depth, so aligned states persist.



(a) Marginal state frequency (b) Trajectory similarity (c) Persistence across depth

Figure 10: **State sequence statistics** (Llama-3.2-1B-Instruct / MATH, generation-averaged). **(a)** Frequency of each state over all (sample, layer) assignments (bars) and cumulative mass (line). **(b)** Pairwise Hamming similarity of state sequences, samples ordered by hierarchical clustering. **(c)** Self-transition probability per layer (line) and number of states per layer (bars).

C.2 Prompt-last and generation-averaged state maps

Prompt-last states exist before generation and serve prediction (Section 5.1); generation-averaged states summarize the completed response and serve the characterization of Section 6.1. The two maps are fit separately, so their state IDs are unrelated. Outcome-associated states separate more clearly in the generation-averaged map (Figure 12) than in the prompt-last map (Figure 11), whose layer-0 states coincide because every prompt ends with the same template token.

C.3 External labels and Cramér's V

Three labels are attached to each MATH response after the map is frozen: problem type (the seven subjects of the dataset), difficulty (levels 1–5), and correctness (Appendix A.1). None is used to fit or align the map. For layer l and label Y, Cramér's V between the state assignment S_l and Y is

$$V(S_l, Y) = \sqrt{\frac{\chi^2(S_l, Y)}{N \, \min(|\mathcal{S}_l| - 1, \, |\mathcal{Y}| - 1)}},$$

where $\chi^2(S_l, Y)$ is the Pearson chi-square statistic of the state–label contingency table and $|\mathcal{S}_l|$ the number of states at layer l. Figure 13 reports V for every layer; the chi-square test rejects independence at every layer and for every label ($p < 10^{-10}$).

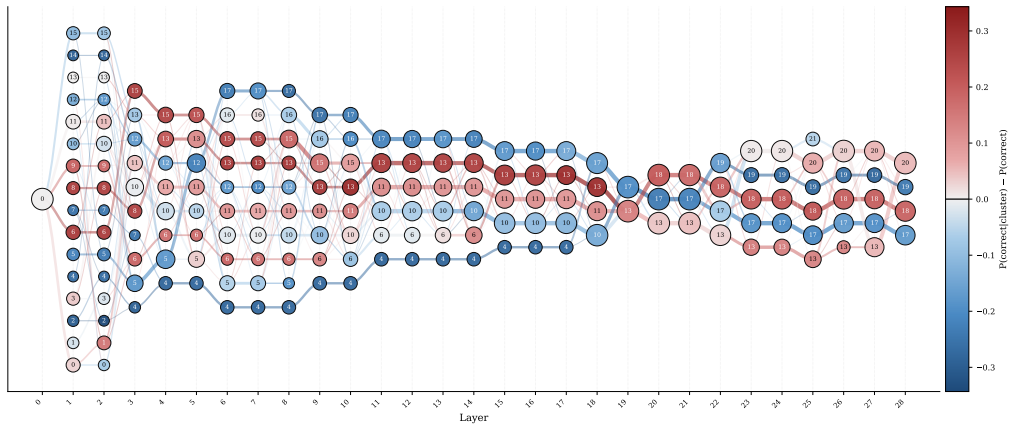


Figure 11: **Prompt-last state map** (Qwen2-7B-Instruct / MATH). Node size: number of samples; color: correctness deviation $\Delta_l(s) = P(\text{correct} \mid s_l{=}s) - P(\text{correct})$.

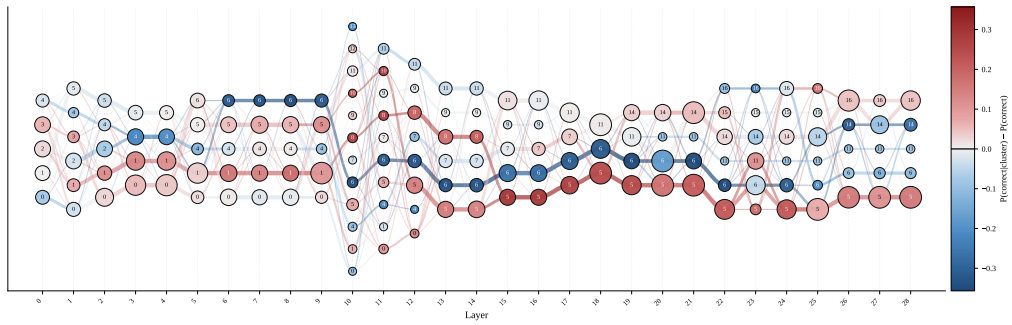


Figure 12: **Generation-averaged state map** (Qwen2-7B-Instruct / MATH, $N{=}5{,}000$), the map characterized in Section 6.1. Node size: number of samples; color: $\Delta_l(s)$.

C.4 The sink state

A sink state is a low-accuracy state that emerges early and persists to the last layer. Figure 14 shows the correctness deviation of every state at every layer: one state (the dark column) has much lower accuracy than average from the first layers to the last, while other low-accuracy states appear later or vanish.

About 80% of the sink state's responses contain no `\boxed` answer. Because correctness is assigned by matching the final answer, a response without a final answer counts as incorrect, so the state gathers a failure to *complete* an answer across problem types rather than wrong answers to one type of problem. In the finer map used in Section 6.3 ($K{=}33$ at layer 14), the clearest sink cluster contains 325 responses, 9 of them correct; its responses mix missing final answers, answers without `\boxed`, and a few complete solutions.

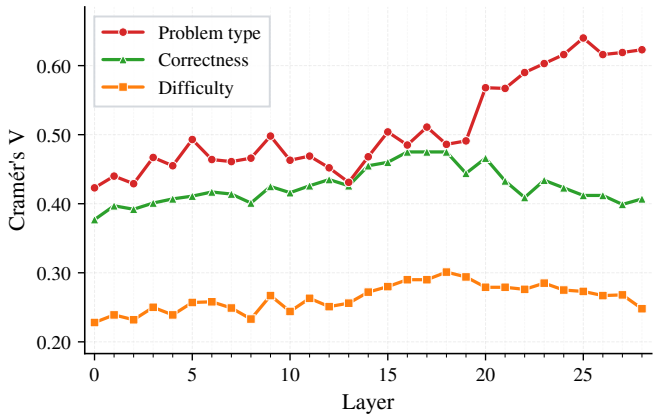


Figure 13: **Cramér's V between states and labels across layers** (generation-averaged map of Figure 12). Problem type is associated most strongly with the states, then correctness, then difficulty.

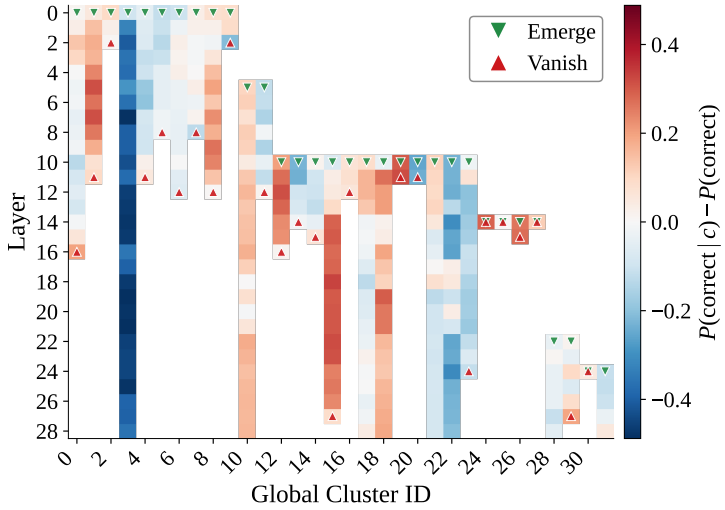


Figure 14: **Correctness deviation by state and layer** (Qwen2-7B-Instruct / MATH, generation-averaged; a fit with more states per layer than Figure 12). Color: $\Delta_l(s)$ for each global state s (columns) and layer l (rows); green triangles mark where a state emerges and red triangles where it vanishes. The sink state is the dark column (state 4), present from layer 0 to the last layer.

D Reconstruction and Removal: Details

This appendix specifies the tests of Section 6.2. All edits are applied to Qwen2-7B-Instruct at layer 14 (`hidden_states[14]`, the output of the 14th of 28 blocks); the model's weights are unchanged and an edited activation is passed through layers 15–28.

Map and directions. A diagonal-covariance GMM with $K{=}64$ components is fit on single-token representations: the first 16 generated tokens of 3,011 MATH training problems (48,176 vectors). A token is assigned to its nearest center $\boldsymbol{\mu}_s$. For each region s, the columns of $U_s \in \mathbb{R}^{d\times 8}$ (Local8) are the top-8 left singular vectors of the $d \times n_s$ matrix whose columns are the residuals $\mathbf{h} - \boldsymbol{\mu}_s$ of its n_s member tokens around the fixed center. Shared bases with 8, 64, and 512 directions are computed in the same way from the pooled residuals of all regions. Wrong-local8 uses three fixed permutations of the regions in which every region receives the basis of a different region; results are averaged within each problem. The map and bases are frozen before evaluation.

Evaluation. For each of the 64 GSM8K problems, the unedited model first generates a reference response. We then rerun the prompt with the first 16 generated tokens, replace their layer-14 states once, and let the remaining layers run. KL is $\mathrm{KL}(p \,\|\, \hat{p})$ between the unedited and edited next-token distributions at position 17. ΔNLL is the change in mean per-token negative log-likelihood (nats) of the reference continuation from position 17 to the end-of-sequence token under teacher forcing. Problems are the units of analysis, and confidence intervals use 2,000 paired bootstrap resamples of problems.

Results. Table 5 lists all variants. Local8 lowers KL relative to Shared8 by 1.41 (95% CI $[0.75, 2.13]$) and ΔNLL by 0.028 ($[0.021, 0.037]$), and relative to Wrong-local8 by 2.37 ($[1.54, 3.27]$). Local8 and Shared8 both use 8 coefficients per token, but Local8 stores 512 basis vectors in total; shared bases of that size preserve the output better, so the comparison concerns which directions are kept at a fixed per-token budget, not storage.

Table 5: **Reconstruction variants** (layer 14, 64 GSM8K problems). Coefficients: continuous values kept per token. Basis vectors: stored d-dimensional directions. All variants also use the region ID and the 64 centers.

Replacement	Coefficients / token	Basis vectors	KL
Center only	0	0	5.402
Center + Wrong-local8	8	512	5.267
Center + Shared8	8	8	4.302
Center + Local8	8	512 (64×8)	2.893
Center + Shared64	64	64	1.530
Center + Shared512	512	512	0.108

Removal. At the same positions we remove the local component $\mathbf{u} = U_s U_s^\top (\mathbf{h} - \boldsymbol{\mu}_s)$. The Random8 control removes a random 8-dimensional subspace, rescaled at every token to match $\|\mathbf{u}\|$ and the component of $\mathbf{u}$ along $\mathbf{h}$ (three seeds, averaged within problems). Removing Local8 gives KL 0.332 against 0.022 for Random8 (difference 0.310, 95% CI $[0.106, 0.592]$; ΔNLL difference 0.014, $[0.011, 0.017]$). Removing only 25% or 50% of $\mathbf{u}$ gives KL 0.003 and 0.012. After full removal, 98.3% of tokens keep their region assignment.

E Steering Along a Sink Direction: Details

This appendix specifies the test of Section 6.3. The protocol, controls, and endpoints were fixed before the runs. The map is fit on all responses and the problems had been examined in earlier analyses, so the results are exploratory.

Sink cluster, direction, and threshold. We use the generation-averaged MATH map of Qwen2-7B-Instruct at layer 14, selected at $\tau_{\mathrm{ICL}}{=}2\%$ ($K{=}33$). A rule fixed in advance selects, among clusters with at least 50 training members, the one with the lowest rate of complete `\boxed` answers: 325 responses, 9 of them correct (Appendix C.4). Responses are split into a construction set A and an evaluation set B. The unit direction $\mathbf{v}$ points from the mean of the other responses in A to the mean of the sink-state responses in A, using generation-averaged representations. The threshold τ is the 75th percentile of single-token projections $\mathbf{h}^\top \mathbf{v}$ over 300 other responses in A (at most 200 tokens each). The clamp $\mathbf{h}' = \mathbf{h} - \max(\mathbf{h}^\top \mathbf{v} - \tau, 0)\,\mathbf{v}$ is applied at every generated position of layer 14 and leaves tokens below τ unchanged.

Orthogonal controls. Ten directions orthogonal to $\mathbf{v}$, five random and five differences between other cluster centers, receive at every token an edit of exactly the clamp's norm at that token, and no edit where the clamp is inactive. After bfloat16 rounding, the realized edit sizes of the clamp and the controls differ by at most 0.022% per response and no edit where the clamp is inactive; these controls are used in the teacher-forcing comparison.

Teacher forcing. We feed back 96 sink-state responses from B (one of the 97 has no scorable suffix under the pre-set rule) and 100 correct, boxed responses from other states, and measure the change

in per-token negative log-likelihood (nats) of their continuations. Sink-state responses are scored from the first repeated sentence if one is detected and otherwise from the sentence start nearest the midpoint; the other responses are scored in full. Confidence intervals use 100,000 paired bootstrap resamples of problems, at 98.75% for the four pre-specified comparisons (Bonferroni). Table 6 gives the results: on sink-state continuations the clamp's effect is 2.8 times that of the controls, and on other correct responses it is negligible, because the clamp rarely activates there.

Table 6: **Teacher-forced effect of the clamp** (Qwen2-7B-Instruct, layer 14; change in per-token NLL in nats, 98.75% CIs).

	Sink-state responses (n=96)	Other correct responses (n=100)
Clamp	$+0.087$ $[0.073, 0.100]$	-0.0004 $[-0.0008, 0.0001]$
Orthogonal controls (mean of 10)	$+0.031$ $[0.026, 0.035]$	$+0.0003$ $[0.0000, 0.0005]$
Clamp − controls	$+0.056$ $[0.046, 0.066]$	-0.0006 $[-0.0010, -0.0003]$
Tokens with active clamp	66.6%	21.2%

The clamp exceeds each of the ten controls individually, including the strongest (difference $+0.036$, $[0.027, 0.045]$). Pairing each sink-state response with a response outside the sink state of the same problem type and difficulty (96 exact pairs, both scored from the midpoint), the clamp changes NLL by 0.092 and 0.005 respectively (difference 0.087, $[0.076, 0.098]$), although it is also active on more tokens of sink-state responses (67% vs. 35%). With the direction, threshold, and controls recomputed at other layers, the clamp exceeds its controls at layers 4, 8, 12, 16, and 20 (by 0.015 to 0.115 nats) but not at layer 24 (0.004, $[-0.003, 0.011]$). Directions toward the four largest other clusters, chosen by size before any edit, have smaller effects within their clusters (0.001 to 0.013 nats) and none outside them.

Free generation. We regenerate responses with greedy decoding (repetition penalty 1.0, at most 2,048 new tokens) for 202 problems, 152 whose original responses fell in the sink state (the 97 sink-state problems of B and 55 held out from both A and B) and 50 others, without editing and with the clamp. Each pair of responses is checked by reading them, for whether the response ends with a complete `\boxed` answer and whether that answer is correct. Relative to no edit, the clamp makes 39 responses gain a complete boxed answer and 11 lose one, and turns 10 incorrect answers into correct ones and 4 correct answers into incorrect ones.

F Extended Related Work

This appendix expands the related-work discussion of Section 2.

Probing and closed-form discriminants. A classical instance of the probe paradigm is Linear Discriminant Analysis (Fisher, 1936), which projects representations onto label-defined axes. Probe-based readouts require labels and answer one property at a time; HSS first partitions the representation space without labels and only afterward examines how labels distribute over the partition, so one map can carry several label axes.

Discrete codes and codebooks. Vector-quantized models learn codebooks for latent representations (Van Den Oord et al., 2017; Esser et al., 2021), and Codebook Features train sparse discrete codes inside a network as interpretable bottlenecks for reconstruction and intervention (Tamkin et al., 2023). Codebooks also appear in model compression (Kim et al., 2023; Egiazarian et al., 2024). These methods learn codes by training, and Codebook Features change the network so that it computes through its codes. HSS fits regions to the hidden states of a frozen model and selects their number per layer. Section 6.2 applies the output-fidelity check used for learned codes to these regions without training.

Discrete states and local subspaces. Chen et al. (2024) identify discrete internal states for intermediate results of multi-step arithmetic with supervised probes for a task-defined variable on synthetic data; HSS asks, without specifying the variable, which regions the representation population occupies on natural data and how samples move between them. Shafran et al. (2026) decompose activations

with a mixture of factor analyzers into local regions with their own subspaces and use them for steering; Section 6.2 applies a related idea to HSS regions, testing whether region-specific directions matter to the model's output.

Hidden-state geometry and topology. Prior work has analyzed representation geometry through anisotropy (Ethayarajh, 2019), intrinsic dimension (Valeriani et al., 2023), dispersion and effective rank (Li et al., 2025), and latent-space shape under behavioral or adversarial variation (Fay et al., 2025). Persistent-homology methods summarize the topology of the whole point cloud at each layer and identify similar or redundant layers (Gardinazzi et al., 2024). HSS instead assigns every sample to a region at every layer, so the same map applies to new inputs and can be combined with labels, making persistence, emergence, transitions, and behavior-conditioned occupancy directly measurable.

Behavioral readouts and latent monitoring. Unsupervised latent-knowledge methods search for behavior-relevant directions without direct label supervision (Burns et al., 2022), output-free self-evaluation uses hidden-state trajectories to estimate generation quality (Wang et al., 2024), and Kernel Language Entropy quantifies semantic uncertainty from similarities among generated outputs (Nikitin et al., 2024). HSS shares the goal of reading behavior from model internals but introduces labels only after a label-free state vocabulary has been built, so correctness, safety, problem type, and difficulty can be analyzed over the same kind of map.

Activation patching and circuit-level analysis. Mechanistic interpretability studies behavior through features, components, paths, and circuits, using activation patching (Heimersheim & Nanda, 2024), path patching (Goldowsky-Dill et al., 2023), causal tracing (Zhang & Nanda, 2023), and frameworks for circuit structure (Elhage et al., 2021); see also Bereska & Gavves (2024). HSS operates one level above these objects. The edits of Sections 6.2 and 6.3 are chosen by region rather than by component, and the state map can serve as an interface for selecting layers, regions, and example subsets for circuit-level study.

G Limitations

Scope. Our experiments cover dense decoder-only models from two families (Llama and Qwen2) with 1B to 8B parameters; larger models and other architectures are not evaluated.

Resolution. States are coarse by design (Appendix B.6). K_l grows with the number of samples and depends on the selection tolerance, so absolute counts are comparable only under the same N and selection, and we compare profiles by their shape. A map describes the part of representation space populated by its data; an input far from all centers is still assigned to the nearest one.

Functional evidence. The token-level tests use one model and one layer. Reconstruction and removal use 64 test problems. The steering test uses one sink state, reuses problems examined in earlier analyses, and is exploratory.

Data. Stable partitions require enough samples per layer. The peak location of the K_l profile is preserved down to 20% of MATH (Figure 4(a)), but much smaller datasets may yield unreliable maps.

H Broader Impacts

HSS discretizes hidden-state geometry into a reusable state map and enables lightweight prediction and monitoring without training additional models. It can help explain how language models organize their internal computation, sentence-level early warning could reduce serving cost by stopping likely failures early, and a training-free signal from frozen internals lowers the barrier to reliability diagnostics.

The same capabilities carry risks. Knowing which regions correspond to failures could help design prompts that steer a model into failure-prone states, and region-derived directions that change completion could be used to steer a model, which calls for the same care as other activation-steering

methods. Lightweight monitoring may also invite over-reliance on imperfect warnings, especially without calibration. A state map is derived from a particular model and dataset and should not be treated as a universal diagnostic. We recommend using HSS alongside other safety and reliability measures and validating maps and predictors on deployment-relevant data before use.