WHAT DOES IT TAKE TO FIND A FEATURE MANIFOLD?

Anonymous authors
Paper under double-blind review

ABSTRACT

There is a growing interest in identifying "feature manifolds" or "multidimensional features" in neural network activations, where computational variables are encoded with some organized *geometry* in low-dimensional subspaces of activation space. However, the craft of discovering and characterizing these structures in practice remains nascent. In this work, we perform a deep case study of the representation of "character count" in language models, building on the work of Gurnee et al. (2025). We use this setting to **explore a variety of questions about the characterization and automatic discovery of feature manifolds.** In particular, we describe in detail how sparse autoencoders "tile" the character count manifold and then test whether methods that cluster SAE latents like Bhalla et al. (2026a) can re-discover this structure. We find that clustering methods can correctly group together the latents that tile the character count manifold, but using these latents to then visualize raw model activations does not surface clean structure. We then apply these lessons to **discover a novel representation of character count**, that appears on a different data distribution from Gurnee et al. (2025), finding that models represent character count modulo 8, 4, and 2 on strings of hexadecimal characters. We investigate the functional role of these representations and their universality across models. We lastly consider whether these "character count" representations are part of a higher-dimensional joint representation of other computational variables, which leads to some new ways of visualizing model computation. Overall, we hope that our work provides the community with useful lessons for the identification of feature manifolds and a calibrated picture of the difficulty of this task.

1 INTRODUCTION

How do neural networks organize their internal representations? This question, broadly construed, underlies many parallel efforts to understand both artificial neural networks and biological brains. Neural representations can be characterized at different levels of granularity. At a *macroscopic* level, the activity of entire populations of neurons, across a broad distribution of inputs, can be described with summary statistics (Ansuini et al., 2019; Stringer et al., 2019) and compared between networks (Kornblith et al., 2019; Huh et al., 2024; Caucheteux & King, 2022; Kriegeskorte et al., 2008). At a *microscopic* level, individual neurons can be understood in terms of how they respond to specific network inputs (Olah et al., 2020; Cammarata et al., 2020; Hubel & Wiesel, 1962; Quiroga et al., 2005).

There are many reasons why it is desirable to decompose neural network representations into microscopic *units* (like neurons), and to build our understanding from the bottom up (Von Der Malsburg, 1986; Sharkey et al., 2025). For one, this kind of reduction can enable a *mechanistic* (Saphra & Wiegreffe, 2024) explanation of network computation: to understand how a representation is computed, we can study how its constituent units are computed from those in earlier layers, forming circuits across the network (Olah et al., 2020; Cammarata et al., 2021; Conmy et al., 2023; Arora et al., 2026). We can also hope to explain macroscopic properties of representations with reference to facts about their constituent parts. For instance, different networks might induce similar representations of inputs because they possess a set of neurons that respond to those inputs in similar ways (Dravid et al., 2023).

What units should be the basis of such low-level, mechanistic analysis? (Mueller et al., 2026; Sharkey et al., 2025) While individual neurons, or perhaps general *directions* (Nanda et al., 2023b;

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.

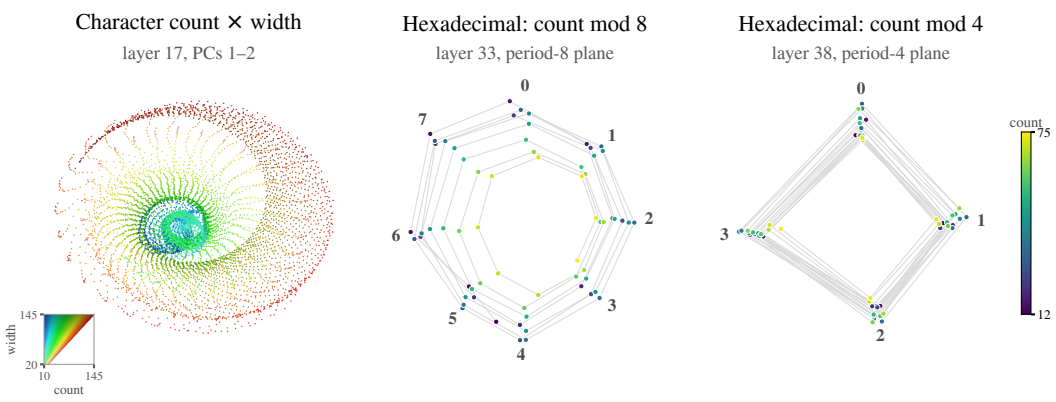

Figure 1: **Summary of new structures we identify, here in Gemma-4-31b.** We discover (left) that computing *centroids* by character count as in Gurnee et al. (2025), but conditioned on the *document width*, reveals additional elegant structure in language model activations. We also discover (middle, right) novel periodic representations of character count on strings of hexadecimal characters. See Section 3 and Section 4.

Marks & Tegmark, 2023; Arditi et al., 2024) in activation space can be studied, networks appear to represent some quantities via the *geometry* of their activations within *subspaces of dimension greater than one*. For instance, Engels et al. (2024) found subspaces where language models organize their activations on inputs referring to "days of the week" or "months of the year" along a circle, so that representations of "Sunday" are next to "Monday". A large number of works have likewise documented nontrivial geometry within subspaces of activation space in both toy (Liu et al., 2022; Nanda et al., 2023a; Shai et al., 2024; Riechers et al., 2025; Ray et al., 2026) and large-scale neural networks (Zhou et al., 2024; Kantamneni & Tegmark, 2025; Modell et al., 2025; Gurnee et al., 2025; Bhalla et al., 2026a; Feucht et al., 2026). The discovery of one of these structures requires, at minimum, the identification of a *subspace* of activation space and a *distribution of network inputs* such that network activations on that distribution, when projected onto that subspace, organize with some interesting geometry. These structures have been variously referred to as "multi-dimensional features" (Engels et al., 2024; Olah, 2024), "feature manifolds" (Bricken et al., 2023; Olah & Batson, 2023; Michaud et al., 2025; Gurnee et al., 2025), "representation manifolds" (Modell et al., 2025) and "concept manifolds" (Bhalla et al., 2026a). In this work, we use the terms "multidimensional feature" and "feature manifold" somewhat interchangeably, but prefer "multidimensional feature" when activation geometry is not necessarily smooth.

These structures are attractive objects of study. They are richer than individual neurons or directions, but, as units of analysis, are simpler and narrower than the entire representation space of which they are a part. They are often aesthetically beautiful, and their geometry calls out for theoretical explanation (Karkada et al., 2026). While they can be difficult to first discover, they can usually be communicated and reproduced with relatively *simple recipes*. The research community therefore seems poised to identify an increasing number of such structures over the coming years. Special attention was recently placed on this agenda by Bhalla et al. (2026a;b), who reported several new instances of such geometry in language models, proposed an improved method for automatically discovering such structures, and emphasized the opportunity for further developing such methods.

However, the craft of discovering and characterizing these structures is still nascent. As we will show, a variety of subtle choices can change the quality of results. It is also not yet clear what standard of evidence should be met by new reports of these structures. Altogether, this ambiguity is a barrier to those attempting to discover new multidimensional features and grow this literature. Motivated by these concerns, in this work we perform a case study on identifying feature manifolds that we hope will provide useful lessons and a calibrated picture of the difficulty of this task. We begin by studying the "character count" manifold, which was discovered by Gurnee et al. (2025) in Claude 3.5 Haiku, arguably the cleanest feature manifold reported thus far. Because this structure is so clean, it provides an excellent setting to test *methods* for discovering feature manifolds and to understand the *choices* that influence the clarity of results. Pulling on these threads leads us to a variety of discoveries, including the discovery of a novel multidimensional representation of character count with different geometry from Gurnee et al. (2025). Our contributions include:

- **We support recent findings that sparse autoencoders "tile" feature manifolds in unintuitive ways.** We first report how *sparse autoencoders (SAEs)*, a common technique for decomposing neural network representations into features, reconstruct the character count manifold from Gurnee et al. (2025). Our results add detail and further support to a growing literature on how sparse autoencoders "tile" feature manifolds. We broadly find results consistent with Bhalla et al. (2026a), who observe similar tilings on other manifolds. We additionally investigate how these tilings vary with SAE dictionary size and sparsity.
- **We test methods for discovering feature manifolds with sparse autoencoders.** We next investigate the difficulty of re-discovering the character count manifold using techniques that cluster sparse autoencoder latents, such as Bhalla et al. (2026a). We find that clustering methods can correctly group together the latents we had identified manually, but that visualizing raw activations with these methods does not yield clean structure, emphasizing the difficulty of applying these methods in an entirely unsupervised way.
- **We discover a novel multidimensional feature that tracks character count, with different representational geometry to Gurnee et al. (2025).** Using methods similar to Bhalla et al. (2026a), we also discover a novel character count representation that appears on strings of digits and hexadecimal characters, and that represents character count modulo 8, 4, and 2. We document this structure in detail, investigate its causal role in model computation, and also document its nuanced universality across models.
- **We investigate whether the character count representation is in fact part of an intrinsically higher-dimensional object.** Returning to the original manifold from Gurnee et al. (2025), we find that computing centroids jointly by character count and document width allows us to directly visualize the model's computation, where a character count representation is transformed into a characters remaining one. We additionally study this in toy models.

Throughout our main text of this paper, we assume familiarity with many of the working hypothesis and techniques in the modern interpretability literature. We review these in Section A. We discuss additional related work in Section B.

2 A CASE STUDY ON THE CHARACTER COUNT MANIFOLD

In this section, we perform a case study of the "character count" manifold from Gurnee et al. (2025). Our contributions include (1) reporting how the manifold changes across layers, (2) describing in detail how sparse autoencoders "tile" this manifold, and (3) testing whether popular methods for identifying feature manifolds can re-discover this manifold. We find that (1) the manifold changes substantially across layers, emphasizing the importance of looking for structure across all model layers, (2) that SAEs tile the manifold in unintuitive ways, depending on the SAE sparsity and (3) that we can find the latents that tile the character count manifold automatically via clustering, but that using these latents to directly visualize model activations does not yield clean structure. We think these lessons will be of interest to those searching for other similar structures, and we successfully apply them in Section 3 to discover a new multidimensional feature.

2.1 LOCATING THE MANIFOLD, AND CAUTIONS ABOUT CENTROIDS

Implicit in the next-token prediction objective of language model pretraining are a huge variety of computational tasks (Michaud et al., 2023). In natural text corpora, many documents contain text that is *broken*, where newline characters are inserted periodically so that each line of text doesn't exceed a given number of characters. We refer to this maximum number of characters as the "document width", "width", or w. Accordingly, on documents with this structure, language models are incentivized to compute a count of the number of characters seen on each line of text—the "character count"—and to compare this count with the inferred document width so as to predict the newline token with high probability as the character count approaches that width. Gurnee et al. (2025) discovered and described this newline prediction circuit in detail in Claude 3.5 Haiku, and found that the model learned to represent "character count" as a point along a spiral in the residual stream at an early layer.

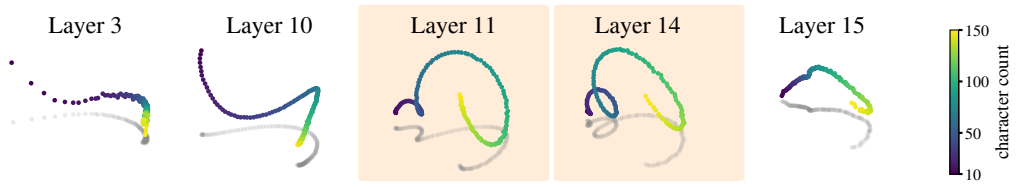

Figure 2: **Representations of character count in Gemma-2-9B, across layers.** Across a synthetic corpus of documents, we compute mean residual stream vectors across all tokens that have the same "character count" in their line, for character count 10-150, and perform PCA on these *centroids*. We find that although centroids are organized continuously by character count across all layers, it is only in layers 11-14 where character counts have particularly distinct and interesting geometry, forming a spiral shape. From layer 15 onward, most of the variance of these centroids lies in the first PC (see Figure 12). We show these centroids for all 42 model layers in Figure 11.

The simplest recipe given by Gurnee et al. (2025) to visualize this structure is to compute *centroids*[1]: across a corpus of line-broken documents, compute the mean residual stream activation at all token positions which are the same number of characters into their line of text, where the current token's characters contribute towards that count. For our replication[2], we take paragraphs from Project Gutenberg books (agentlans, 2025; Project Gutenberg) and insert newlines so that the text is wrapped with a fixed width. We use widths $w \in \{20, 21, \ldots, 160\}$, and create 20 documents per width. Documents each contain between 10-50 lines. We refer to this as our "synthetic prose corpus" and compute centroids by character count on these documents[3]. We visualize centroids for character counts 10-150 for select model layers in Figure 2, and all model layers in Figure 11.

From Figure 2 and Figure 11, we observe that (1) across all layers, centroids appear organized geometrically along a line, roughly in order of character count, but (2) there is a narrow band of layers—layers 11 to 14—where centroids have a visually distinct and beautiful spiral geometry. From these observations, we note a couple of simple takeaways for practitioners interested in discovering and characterizing feature manifolds. First, representations change substantially across layers, and so one should make sure to visualize representations across all model layers, whenever possible, since a variable can be present at every layer while having rich geometry only in a few. Second, even at layers where model representations are not particularly rich, the *centroids* are still arranged continuously by character count in the top PCs. One therefore has to be cautious: just because activations (especially centroids) appear somewhat organized, that observation doesn't necessarily pin down the model's true representation of some feature (see Figure 14). We will return to some of the challenges with centroids again in Section 4.

2.2 SAE TILINGS CAN BE UNINTUITIVE, BUT REFLECT THE GEOMETRY OF ACTIVATIONS

Having located the character count manifold, we now study how sparse autoencoders reconstruct this structure. Our results here set up our study of automated methods in the next section, (Section 2.3). We use the Gemma Scope SAEs (Lieberum et al., 2024). Recall that SAEs reconstruct activations as a sparse, nonnegative combination of learned latent directions, with weights given by latent activations z_i. Since each SAE latent corresponds to a single direction in activation space, a single latent cannot, in general, capture a manifold, (Michaud et al., 2025; Gurnee et al., 2025; Bhalla et al., 2026a). Gurnee et al. (2025) documented how crosscoders tile the character count manifold in Claude 3.5 Haiku, and found that their crosscoder latents respond to a specific and contiguous range of character counts, giving a sensible "featurization". We find a somewhat more nuanced story for SAEs. We select a Gemma Scope SAE for layer 11 of Gemma-2-9b, with 16k latents and average L0 118 (the mean number of latents active per token), and apply a method we describe in Section C.1 to identify a set of 20 of these latents that together reconstruct the character count manifold. We present these latents in Figure 3.

[1] We follow Bhalla et al. (2026a) in using the term *centroid*.

[2] Sinii & Balagansky (2026) also replicate the character count manifold in open LLMs.

[3] We exclude tokens within the first three lines from the centroid computation, since early in the document the model may not yet "realize" that it needs to track character count.

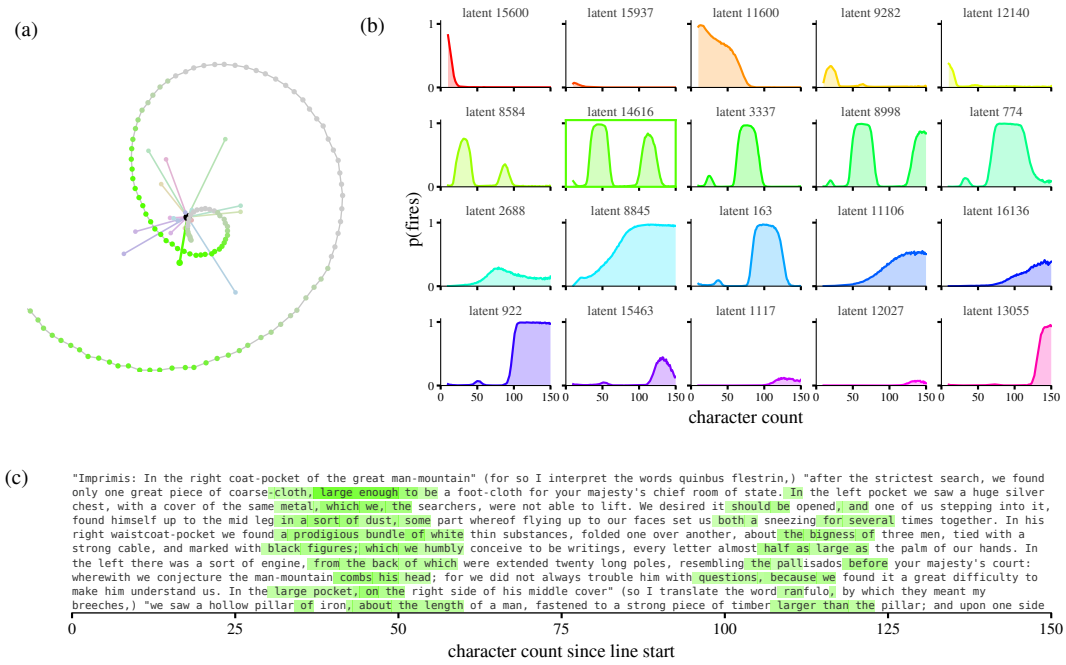

Figure 3: **SAE latents that tile the character count manifold.** We show: (a) the *geometry* of these latents' decoder directions relative to the character count centroids, plotting the SAE decoder directions emanating from the decoder bias. We highlight centroids by the probability that latent 14616 activates on that character count on our synthetic corpus. (b) Tuning curves of these SAE latents as a function of character count. For some latents, we see bimodal tuning curves, reflecting the periodic geometry of the character count helix. (c) The tokens where latent 14616 activates in a particular document.

In Figure 3(b), we plot the probability that each latent fires ($z_i > 0$) as a function of character count across our synthetic prose corpus. Observing these "tuning curves", we find results broadly consistent with what Bhalla et al. (2026a) describe as "dilution", where multiple SAE latents co-activate to reconstruct activations along different parts of the manifold. We also observe that some of these latents activate on multiple separated intervals of character counts, which is ultimately a consequence of the fact that the character count is represented on a helix that doubles back on itself along some axes, which we attempt to illustrate in Figure 3(a). These patterns of activation would make the task of "interpreting" such latents challenging, and also likely make them unsuitable units for understanding model computation. SAE latent activations instead reflect the particular geometry of the model activations that represent character count. In Section C.2, we report that this behavior depends strongly on the SAE sparsity, but not on the SAE dictionary size, bearing on the question of whether these tilings are in some sense "canonical" (Gurnee et al., 2025).

2.3 Re-discovering the character count manifold

The most prominent proposals for discovering multidimensional features involve clustering SAE latents to attempt to identify the subset of latents that "tile" such features (Engels et al., 2024; Bhalla et al., 2026a). Having located the character count manifold and having identified the SAE latents that tile it, we now ask whether unsupervised clustering methods could have automatically surfaced this precise set of latents. We find that this provides a nice benchmark for methods that purport to discover feature manifolds via clustering in this manner.

Engels et al. (2024) cluster SAE latents according to the cosine similarity of latent decoder directions. Bhalla et al. (2026a) instead suggest clustering SAE latents by co-activation statistics, propose a measure of co-activation based on the Ising model, and use network community detection algorithms for the clustering step. To test these and related choices, we construct a graph over the 16k latents in the layer-11 SAE from the previous section, and assign edge weights between latents ac-

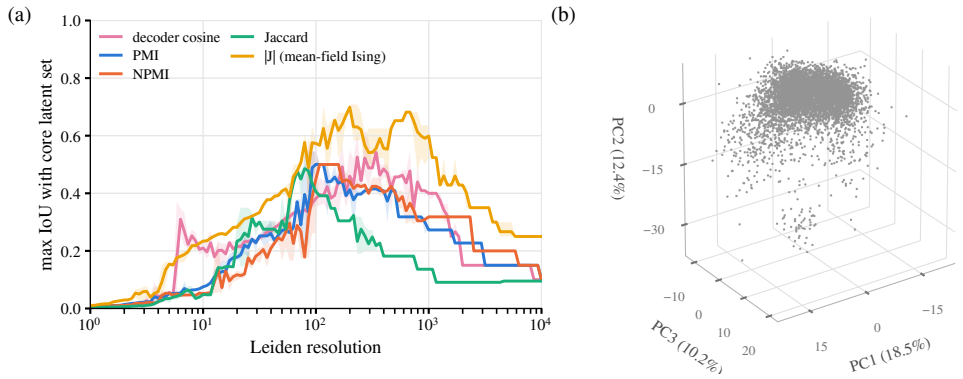

Figure 4: **Evaluation of methods for re-discovering the character count manifold with SAEs.** We run the Leiden communities algorithm across "resolution" parameters for different notions of edge weight between latents, then report the maximum intersection over union (IoU) of the resulting communities with the set of latents from Figure 3. We find that a method similar to Bhalla et al. (2026a) (gold line) performs best. However, visualizing raw model activations on the subspace spanned by these latents (right) does not reveal clear structure.

cording to either decoder cosine similarity, or one of four co-activation statistics computed over ∼1B tokens of The Pile (PMI, NPMI, Jaccard similarity, and Ising couplings, which we estimate with a mean-field approximation). We keep the strongest 3M edges in this graph and apply the Leiden community detection algorithm across a range of "resolution" parameters, which control the typical community size. For each resulting partition, we find the community with the maximum intersection over union (IoU) with the set of latents we had manually identified in Section 2.2. We give full details in Section C.4.

Our replication of the Ising couplings approach of Bhalla et al. (2026a) recovers the reference set best (Figure 4a): at resolution ≈200, the best community has 21 latents, 17 of which are in the reference set (IoU 0.71), including all of the latents that respond most strongly to character count (Figure 15). However, clusters like these are then meant to be used to visualize activations directly: across a corpus, collect the activations on which any latent in the cluster fires, project them onto the span of the cluster's decoder directions, and run PCA (Engels et al., 2024; Bhalla et al., 2026a)[4]. We apply this to the highest-IoU community of latents, on documents from The Pile in Figure 4 (right), and find that **projection does not reveal a clean structure**. We show point clouds using other projection methods and data distributions in Figure 16, and generally find that raw model activations appear dramatically less structured than centroids.

Overall, we have found that (1) we can cluster SAE latents to re-discover the set of latents that tile the character count manifold, but (2) that using these latents to visualize raw model activations does not reveal interpretable structure. We think this last result is intriguing and important. As a user of these clustering methods, one finds that the point clouds of model activations produced by most clusters tend not to be interpretable. However, this may not mean that there is no structure there, as we've seen with this cluster.

3 A NOVEL PERIODIC CHARACTER COUNT REPRESENTATION

We now apply the lessons from the previous section to discover a new multidimensional feature. This feature also tracks character count, but appears on a different distribution of text documents from the manifold studied in Gurnee et al. (2025), and has distinct representational geometry. We first describe how this structure can be found, further contributing to the methodological points from the previous section. We then answer a variety of object-level questions about its structure, universality across models, and causal role, which we hope will be of independent interest.

[4]Engels et al. (2024) instead reconstruct activations using the cluster's latents alone then run PCA on the resulting point cloud.

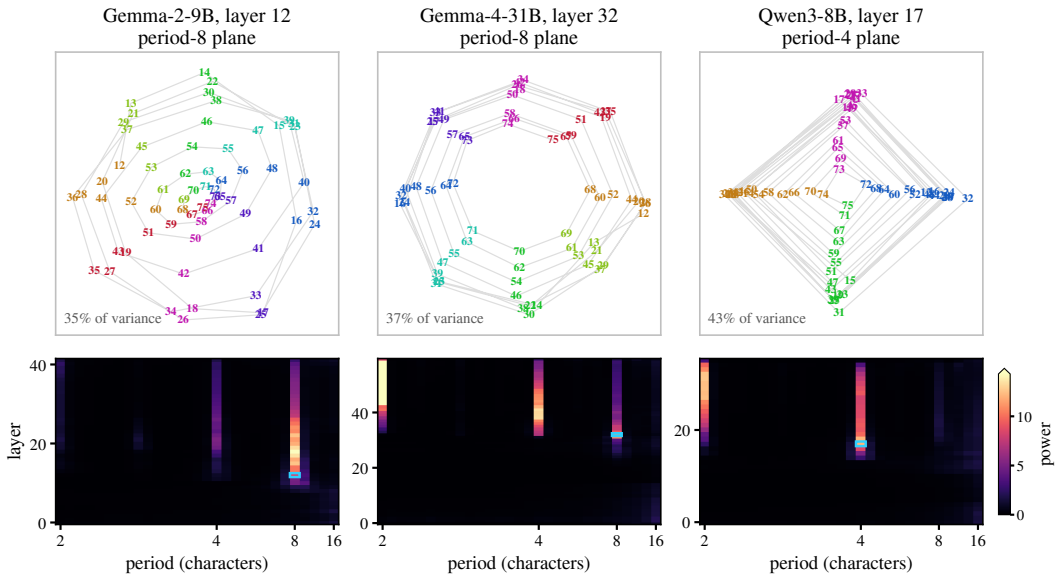

Figure 5: **Models represent character count within strings of hexadecimal characters via a periodic multidimensional feature.** We find subspaces of the residual stream where character count representations arrange into octagons and squares. We can also detect this periodic structure with Fourier analysis, and find that the universality of this structure across models is subtle, with different models representing different frequencies more or less clearly.

In the previous section, we re-discovered the latents that tile the character count manifold from Gurnee et al. (2025) using a method similar to Bhalla et al. (2026a). Looking across other communities in that same SAE, we find a different community containing 14 latents[5] that have a curious pattern of activation. Across a variety of documents from The Pile, some of which we show in Figure 19, we find that the latents in this cluster activate periodically, roughly every 8 characters, on strings of decimal or hexadecimal digits[6]. Based on our experience in the previous section, we therefore expect a character count representation on these strings, perhaps a helix, that wraps around with period 8.

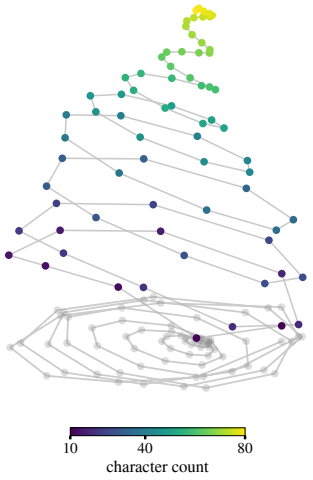

Figure 6: PCA of character-count centroids of Gemma-2-9B at layer 12 on hex documents with fixed line widths.

Centroids. To visualize this structure, we use the following recipe: we create a new "synthetic hex corpus" of fixed-width documents containing lines of random hexadecimal strings separated by newlines, where each line has exactly the same width w. Widths range from 20-80 across documents. As in Section 2, we harvest model activations and compute centroids according to each token's character count into its current line. We look at these centroids across all layers in Figure 20, and show specifically layer 12 in Figure 6. We find that, unlike in the smooth "prose" character count representation, these centroids are spread out in discrete points. After some analysis, we understand these centroids to have the following structure: on these strings (unbroken strings of hex, digits, etc.) the model computes the character count representation from Gurnee et al. (2025) as before, but *additionally* represents character count modulo 8, 4 and 2 in subspaces nearly orthogonal to the prose character count manifold (Figure 17). Principal components of these centroids in general mix these subspaces, but we can isolate them with Fourier analysis.

[5]At resolution=200, where IoU was highest for recovering the character count manifold latents in Figure 4, this community is broken into 4 separate communities. It is one community at lower resolution=116.

[6]They also activate on strings of DNA base pairs.

Fourier analysis and universality. Using the centroids for character counts 12–75 (64 counts), we take a discrete Fourier transform across character count for each residual-stream dimension, after centering each layer's centroids and normalizing them by their Frobenius norm. Summing the squared magnitudes across dimensions gives a "Fourier power" at each period $64/k$ characters ($k = 1, \ldots, 32$). We compute this at every layer for a variety of open-weight models. In Figure 5, we find that models differ in which periods they represent: Gemma-2-9B represents character count much more strongly with period 8 than with period 4, while Qwen3-8B strongly represents period 4 but barely period 8. We show these heatmaps for a broader range of open-weight models in Figure 21, and show in Figure 18 that the same analysis on the centroids from our synthetic prose corpus finds no comparable periodic structure. Figure 5 also shows projections of the centroids onto the plane of the period P component. We fit the centered centroids by least squares to $\cos(2\pi c/P)$ and $\sin(2\pi c/P)$, orthonormalize the two coefficient vectors, and project the centroids onto the plane they span. The centroids form octagons for period 8 and squares for period 4.

Ablating this representation. What is the functional role of these representations? Unlike in prose, where each line varies in width based on the particular words that occur at the end of the line or get placed onto the next, on our hex corpus all lines are of precisely equal length. This mirrors the structure of some real-world documents we observe in Figure 19. Accordingly, the structure of these documents allows for models to predict the position of the newline tokens at the end of their lines extremely precisely, and indeed we find models succeed in doing this on our synthetic corpus of exact-width hex strings. In Figure 7, we see that Gemma-2-9B predicts the newline token with probability $\approx$0 except at the last character of the line, where it predicts newline with probability 0.99. If character count were only represented along a smooth manifold like Gurnee et al. (2025), then it would be difficult for the model to distinguish between adjacent character counts to make such sharp predictions. This periodic structure allows the model to "spread out" its representations of character count in activation space. In order to enable this error correction, it should be sufficient to represent character count modulo N, since the model already has the "prose" character count representation. To test this, we ablate activations in both the period-4 and period-8 planes from layers 6-36 in Gemma-2-9B (orthogonal to the prose character count representation; details in Section D.2), and observe that the model's confidence in predicting the newline token degrades: when the next token is a newline, it outputs newline with probability 0.84 and at the previous token position outputs newline with probability ≈ 0.12 (versus 0 without the ablation). We find that ablating random 4d subspaces of the residual stream has no effect on the model's newline prediction on these documents, suggesting that our periodic representations are meaningfully implicated in this task.

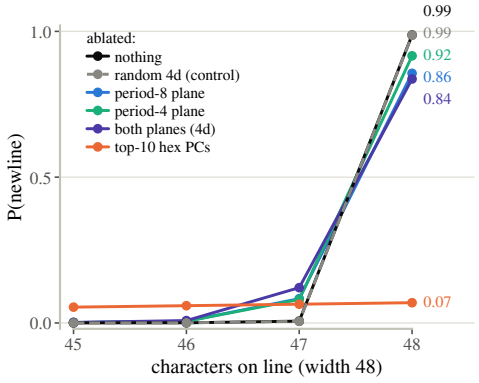

Figure 7: **Ablating periodic character count representations degrades the model's confidence in the newline position**.

4 Higher-dimensional structure

We have found the use of *centroids* to be an essential tool for visualizing model activation geometry. Before closing, we note one additional concern with using centroids. Pulling on this thread leads us to a variety of new visualizations and analysis, most of which is in Section E.

The basic issue with centroids is that they are computed over a *distribution* of inputs and a *partition* of those inputs that are both manually specified. They can therefore mix the geometry of multiple computational variables if one is not careful. Returning to the original setting of Gurnee et al. (2025), we illustrate this point by computing centroids for character count *separately* across documents of different widths. We see from Figure 8 that this leads to substantially different activation geometry than we saw in Figure 2, where we averaged only by character count. We understand this geometry as arising from the model computing both a character count representation and a *characters remaining* representation that becomes dominant at later layers in the model at token positions where the

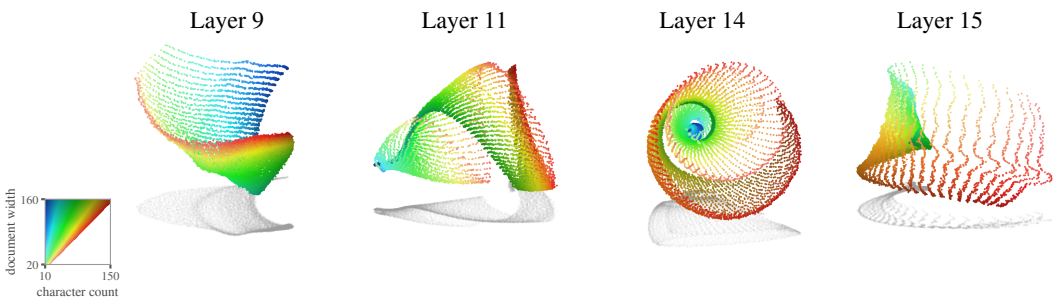

Figure 8: **Joint character count × document width centroids in Gemma-2-9B.**

character count approaches the document width. We note that in the original work of Gurnee et al. (2025), on the distribution of synthetic documents they considered, centroids for character count 150 came only from documents of width 150. There is then a chance that the centroids on the upper end of their range mixed in the model's characters remaining representation, which would alter their arrangement in activation space. Since these two manifolds co-occur, SAEs can sometimes mix them together, treating them like a single higher-dimensional manifold to reconstruct. Indeed, in Figure 13 we find that several SAE latents respond to both character count and characters remaining.

In order to understand where this geometry comes from, we turn to theory, presented in Section E. We find that in an idealized version of the line-breaking task, once the width is known, the optimal prediction of the next newline depends only on the number of characters remaining. It is thus reasonable to expect that a model should reorganize its representations of character count and width around characters remaining as it moves through the layers toward predicting the newline token (Section E.3). When we train small transformers on such synthetic data, we find that they show this reorganization, with activations arranged by character count and width in early layers and by characters remaining in later layers (Figure 22, Figure 25). Gemma-2-9B shows the same reorganization, with activations organized primarily by character count up to layer 11 and characters remaining in layers 12 through 15 (Figure 27). Moreover, the model uses the joint count × width structure; steering activations at layers 10-14 toward the centroid for a given character count *and* width makes the model predict a newline at the end of the injected line, even in documents of a different width, whereas steering toward width-agnostic centroids, as in Gurnee et al. (2025), produces a newline only at the document's actual width (Figure 28).

5 Discussion

Lessons for feature manifold hunters. From our case study, we take away the following lessons about the craft of identifying feature manifolds: **(1)** we find that model representations can change substantially, and abruptly, across model layers, suggesting that representations should be characterized across all model layers, when possible (Section 2.1). **(2)** as in Bhalla et al. (2026a), we find SAE latents can tile feature manifolds in somewhat unintuitive and inconsistent ways, complicating their interpretation as standalone units of model representation (Section 2.2). **(3)** methods that cluster SAE latents, like Bhalla et al. (2026a), can identify groups of SAE latents that together reconstruct feature manifolds (Section 2.3, Section 3), however **(4)** raw model activations can be extremely noisy, and this noise can obscure beautiful structure that indeed underlies model representations (Section 2.3). Lastly, **(5)** this noise can be eliminated by instead visualizing activation *centroids*, but this requires manually defining the set of inputs on which centroids are computed, which can easily mix together potentially distinct model representations.

Object-level findings. Our results contribute to an emerging *taxonomy of multidimensional features* (Lindsey, 2026). Of the structures identified so far in language models, many seem to either have a continuous "rippling" (Olah, 2024; Modell et al., 2025; Gurnee et al., 2025) structure or discrete periodic structure (Kantamneni & Tegmark, 2025; Feucht et al., 2026). We have now established that character count representations in language models exhibit *both* kinds of representations. We wonder whether a larger taxonomy will be discovered in the future, and encourage work to identify more of these structures over the coming years.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods: a large fraction of the code for running experiments and producing figures was written with AI coding assistants, particularly for extending experiments we had first implemented by hand. The synthetic corpora (wrapped Project Gutenberg text and random hexadecimal strings) were generated by this code, not by a generative model. We also used AI tools to design and give feedback on analyses (for example, latent-selection criteria) and to help interpret results to some extent. We used generative AI to give feedback on and develop some of the mathematical claims in Section F, but not to develop the theoretical framework, or to validate any mathematical proofs or claims. We did not use generative AI to propose the paper's central hypotheses. Translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Additionally, AI tools drafted some appendix sections and some passages and figure captions in the main text, which the authors then revised. The authors reviewed some AI-generated code and inspected every figure. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

agentlans. Project Gutenberg Top 100. https://huggingface.co/datasets/agentlans/project-gutenberg-top100, 2025. Paragraph-level text from the 100 most downloaded Project Gutenberg eBooks, snapshot of May 11, 2025. English config (`en`) used.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.11717.

Aryaman Arora, Zhengxuan Wu, Jacob Steinhardt, and Sarah Schwettmann. Language model circuits are sparse in the neuron basis. *arXiv preprint arXiv:2601.22594*, 2026.

Michael F. Barnsley, Stephen G. Demko, John H. Elton, and Jeffrey S. Geronimo. Invariant measures for Markov processes arising from iterated function systems with place-dependent probabilities. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 24(3):367–394, 1988.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 24(3):179–195, 1975.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Owen Lewis, Thomas McGrath, Jack Merullo, Atticus Geiger, and Ekdeep Singh Lubana. Do sparse autoencoders capture concept manifolds? *arXiv preprint arXiv:2604.28119*, 2026a.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Owen Lewis, Thomas McGrath, Jack Merullo, Atticus Geiger, and Ekdeep Singh Lubana. Can SAEs capture neural geometry? Goodfire Research Blog, 2026b. URL https://www.goodfire.ai/research/can-saes-capture-neural-geometry.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference*, pp. 31–40, 2009.

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

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. doi: 10.23915/distill.00024.003. https://distill.pub/2020/circuits/curve-detectors.

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. doi: 10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

Jack W. Carlyle and Azaria Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40, 1971.

Charlotte Caucheteux and Jean-Rémi King. Brains and algorithms partially converge in natural language processing. *Communications biology*, 5(1):134, 2022.

David Chanin and Adrià Garriga-Alonso. Sparse but wrong: Incorrect L0 leads to incorrect features in sparse autoencoders, 2025. URL https://arxiv.org/abs/2508.16560.

David Chanin, Tomáš Dulka, and Adrià Garriga-Alonso. Feature hedging: Correlated features break narrow sparse autoencoders, 2025a. URL https://arxiv.org/abs/2505.11756.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://openreview.net/forum?id=R73ybUciQF.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Matthew A. Clarke, Hardik Bhatnagar, and Joseph Bloom. Compositionality and ambiguity: Latent co-occurrence and interpretable subspaces. https://www.lesswrong.com/posts/WNoqEivcCSg8gJe5h/compositionality-and-ambiguity-latent-co-occurrence-and, December 2024. LessWrong. Produced as part of the PIBBSS Summer 2024 cohort.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Róbert Csordás, Christopher Potts, Christopher D. Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, 2024. URL https://arxiv.org/abs/2408.10920.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.08600.

Pablo Diego-Simón, Stéphane D'Ascoli, Emmanuel Chemla, Yair Lakretz, and Jean-Rémi King. A polar coordinate system represents syntax in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1934–1943. IEEE, 2023.

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. 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. *arXiv preprint arXiv:2405.14860*, 2024. Published at ICLR 2025.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.17191.

Sheridan Feucht, Tal Haklay, Usha Bhalla, Daniel Wurgaft, Can Rager, Raphaël Sarfati, Jack Merullo, Thomas McGrath, Owen Lewis, Ekdeep Singh Lubana, Thomas Fel, and Atticus Geiger. Arithmetic in the wild: Llama uses base-10 addition to reason about cyclic concepts. *arXiv preprint arXiv:2605.01148*, 2026.

Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. Counter machines and counter languages. *Mathematical Systems Theory*, 2(3):265–283, 1968.

Michel Fliess. Matrices de hankel. *J. Math. pure et appl.*, 53:197, 1974.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Liv Gorton. The missing curve detectors of InceptionV1: Applying sparse autoencoders to InceptionV1 early vision. *arXiv preprint arXiv:2406.03662*, 2024.

Liv Gorton. What would non-linear features actually look like? https://livgorton.com/non-linear-feature-reps/, January 2026. Blog post. Accessed: 2026-08-27.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.09124.

Sai Sumedh R Hindupur, Ekdeep S Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *Advances in Neural Information Processing Systems*, 38:11649–11699, 2025.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Hilbert J. Kappen and Francisco B. Rodríguez. Efficient learning in Boltzmann machines using linear response theory. *Neural Computation*, 10(5):1137–1156, 1998.

Dhruva Karkada, Daniel J Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. Symmetry in language statistics shapes the geometry of model representations. *arXiv preprint arXiv:2602.15029*, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

Joachim Lambek. How to program an infinite abacus. *Canadian Mathematical Bulletin*, 4(3):295–302, 1961.

Yuxiao Li, Eric J Michaud, David D Baek, Joshua Engels, Xiaoqing Sun, and Max Tegmark. The geometry of concepts: Sparse autoencoder feature structure. *arXiv preprint arXiv:2410.19750*, 2024.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Jack Lindsey. Here are the questions that currently seem most important to me. Post on X (formerly Twitter), September 2026. URL https://x.com/Jack_W_Lindsey/status/2100143082167832816. @Jack_W_Lindsey.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

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.

Eric J Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.

Eric J. Michaud, Liv Gorton, and Tom McGrath. Understanding sparse autoencoder scaling in the presence of feature manifolds. In *Mechanistic Interpretability Workshop at NeurIPS*, 2025. URL https://arxiv.org/abs/2509.02565.

Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

Alexander Modell. Probing for representation manifolds in superposition. *arXiv preprint arXiv:2605.18537*, 2026.

Alexander Modell, Patrick Rubin-Delanchy, and Nick Whiteley. The origins of representation manifolds in large language models. *arXiv preprint arXiv:2505.18235*, 2025.

Monology. Pile uncopyrighted. https://huggingface.co/datasets/monology/pile-uncopyrighted, 2023.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: Surveying mechanistic interpretability for NLP through the lens of causal mediation analysis. *Computational Linguistics*, 52(1):331–378, March 2026. doi: 10.1162/COLI.a.572. URL https://aclanthology.org/2026.cl-1.10/.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=9XFSbDPmdW.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, 2023b. Association for Computational Linguistics. URL https://arxiv.org/abs/2309.00941.

H. Chau Nguyen, Riccardo Zecchina, and Johannes Berg. Inverse statistical problems: from the inverse Ising problem to data science. *Advances in Physics*, 66(3):197–261, 2017.

Chris Olah. What is a linear representation? what is a multidimensional feature? *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/july-update/index.html#linear-representations.

Chris Olah and Joshua Batson. Feature manifold toy model. *Transformer Circuits Thread*, May 2023. URL https://transformer-circuits.pub/2023/may-update/index.html#feature-manifolds. Section of "Circuits Updates — May 2023".

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. ICLR: In-context learning of representations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2501.00070.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Project Gutenberg. Project Gutenberg. https://www.gutenberg.org. Accessed 2025.

R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

Pradeep Ravikumar, Martin J. Wainwright, and John D. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

Kyle J. Ray, Paul M. Riechers, and Adam S. Shai. The geometry of nonergodic composition. *Belief Updates (Simplex Blog)*, 2026. URL https://belief-updates.pub/nonergodic-geometry/.

Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

Paul M Riechers, Thomas J Elliott, and Adam S Shai. Neural networks leverage nominally quantum and post-quantum representations. *arXiv preprint arXiv:2507.07432*, 2025.

Naomi Saphra and Sarah Wiegreffe. Mechanistic? In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 480–498, 2024.

Raphaël Sarfati, Eric Bigelow, Daniel Wurgaft, Siddharth Boppana, Jack Merullo, Atticus Geiger, Owen Lewis, Tom McGrath, and Ekdeep Singh Lubana. The shape of beliefs: Geometry, dynamics, and interventions along representation manifolds of language models' posteriors. *arXiv preprint arXiv:2602.02315*, 2026.

Adam Shai, Lucas Teixeira, Alexander Oldenziel, Sarah Marzen, and Paul Riechers. Transformers represent belief state geometry in their residual stream. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15943.

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. In *Forty-third International Conference on Machine Learning*, 2026.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Mary Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, Eric J Michaud, Stephen Casper, Max Tegmark, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Thomas McGrath. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=91H76m9Z94. Survey Certification.

Viacheslav Sinii and Nikita Balagansky. Chasing the counting manifold in open llms, 2026. URL https://huggingface.co/spaces/t-tech/manifolds.

Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

Toshiyuki Tanaka. Mean-field theory of Boltzmann machine learning. *Physical Review E*, 58(2): 2302, 1998.

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. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.

Claudia D. van Borkulo, Denny Borsboom, Sacha Epskamp, Tessa F. Blanken, Lynn Boschloo, Robert A. Schoevers, and Lourens J. Waldorp. A new method for constructing networks from binary data. *Scientific Reports*, 4(1):5918, 2014.

Christoph Von Der Malsburg. Am I thinking assemblies? In *Brain Theory: Proceedings of the First Trieste Meeting on Brain Theory, October 1–4, 1984*, pp. 161–176. Springer, 1986.

Martin Wattenberg and Fernanda B. Viégas. Relational composition in neural networks: A survey and call to action. In *ICML 2024 Workshop on Mechanistic Interpretability*, Vienna, Austria, 2024. URL https://arxiv.org/abs/2407.14662.

Daniel Wurgaft, Can Rager, Matthew Kowal, Vasudev Shyam, Sheridan Feucht, Usha Bhalla, Tal Haklay, Eric Bigelow, Raphaël Sarfati, Thomas McGrath, Owen Lewis, Jack Merullo, Noah D. Goodman, Thomas Fel, Atticus Geiger, and Ekdeep Singh Lubana. Manifold steering reveals the shared geometry of neural network representation and behavior. *arXiv preprint arXiv:2605.05115*, 2026.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition. *Advances in Neural Information Processing Systems*, 37: 25120–25151, 2024.

APPENDIX CONTENTS

A Background: superposition, SAEs, feature manifolds **17**

B Related work **18**

C Additional results on the prose character count manifold **19**

C.1 Method for identifying latents that tile a manifold 19

C.2 Tilings are canonical across dictionary size but not across sparsity 19

C.3 Prior character count manifold lengths were arbitrary 20

C.4 Additional details on clustering methods 21

C.5 Additional figures 24

D Additional results on periodic character count representation **28**

D.1 Overlap between the periodic planes and the prose representation 28

D.2 Details on ablation experiments 29

D.3 Additional figures 30

E The count $\times$ width manifold: theory, toy models, and tests in Gemma-2-9B **33**

E.1 Predictive Generator Analysis 33

E.2 Do models represent the Sufficient Statistic? 34

E.3 What determines the centroids arrangement in activation space? 36

E.4 Tracing and steering c, r, and w 40

F Technical details of the predictive-generator analysis and toy models of the count $\times$ width manifold **45**

F.1 Predictive generators 45

F.2 Fully specifying the stochastic counter Machine 46

F.3 Minimal linear dimension of the line-wrap process 48

F.4 Proof that the forecast embedding is independent of content token probabilities . . 49

F.5 Toy model data and training 50

A BACKGROUND: SUPERPOSITION, SAEs, FEATURE MANIFOLDS

Much work on the interpretability of neural network activations follows from two hypotheses. The first is the Linear Representation Hypothesis (Elhage et al., 2022; Park et al., 2023; Gorton, 2026), which states that networks *linearly* encode the value of their internal computational variables $f_i, f_j, \ldots$, called *features*, so that the value of these features can be "read off" linearly $f_i \approx \boldsymbol{a} \cdot \hat{\boldsymbol{e}}_i$ from activations $\boldsymbol{a}$. The second is the Superposition Hypothesis (Elhage et al., 2022), which states that networks seek to represent a greater number of such features than there are dimensions in representation space, requiring features to be represented nonorthogonally ("in superposition"). Together, they suggest a model of activations as:

$$\boldsymbol{a}(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x})\, \hat{\boldsymbol{e}}_i$$

where $\boldsymbol{a}(\boldsymbol{x}) \in \mathbb{R}^d$ is the vector of network activations, at a given layer, when the network input is $\boldsymbol{x}$, $f_i(\boldsymbol{x})$ is the *feature value* of feature i on input $\boldsymbol{x}$, and $\hat{\boldsymbol{e}}_i$ is its *feature direction*. Features are assumed to be *sparse*, meaning $f_i(\boldsymbol{x}) = 0$ for most inputs $\boldsymbol{x}$. When feature directions are approximately orthogonal and features are sparse, then *interference* between features is low, and a feature's value can be approximately read off, linearly, from its direction: $f_i(\boldsymbol{x}) \approx \hat{\boldsymbol{e}}_i \cdot \boldsymbol{a}(\boldsymbol{x})$.

If representations have this structure, then we may automatically recover the feature directions and feature values with *sparse dictionary learning*. A simple and popular method is the sparse autoencoder (SAE) (Cunningham et al., 2024; Bricken et al., 2023; Templeton et al., 2024). An SAE consists of an encoder $\boldsymbol{z} = \mathrm{Enc}(\boldsymbol{a}) = \sigma(\boldsymbol{W}_e \boldsymbol{a} + \boldsymbol{b}_e)$ and a decoder $\boldsymbol{a'} = \mathrm{Dec}(\boldsymbol{z}) = \boldsymbol{W}_d \boldsymbol{z} + \boldsymbol{b}_d$, where $\boldsymbol{W}_e, \boldsymbol{b}_e, \boldsymbol{W}_d, \boldsymbol{b}_d$ are learned and σ is a nonlinearity. SAEs are trained with gradient descent to minimize a loss like $\|\boldsymbol{a} - \boldsymbol{a'}\|_2^2 + \lambda \|\boldsymbol{z}\|_0$, which induces the SAE to reconstruct activations as a sparse linear combination of the columns of $\boldsymbol{W}_d$, the SAE's *latent directions* (also called *decoder directions).* The hope is that these latent directions recover the network's underlying feature directions $\hat{\boldsymbol{e}}_i$ and the latent activations z_i track the feature values f_i.

Engels et al. (2024) argue that some features may be irreducibly *multidimensional*, taking values $\boldsymbol{f}_i(\boldsymbol{x}) \in \mathbb{R}^{d_i}$ within d_i-dimensional subspaces of activation space, and suggest a *multidimensional superposition hypothesis*[7], where neural network activations are modeled as:

$$\boldsymbol{a}(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i(\boldsymbol{x})\, \boldsymbol{V}_i$$

where $\boldsymbol{V}_i \in \mathbb{R}^{d_i \times d}$ are matrices whose rows provide a basis for the subspace in which feature i is represented. When there exist such multidimensional features, multiple SAE latents must participate together in reconstructing the value of one of these features $\boldsymbol{f}_i$. The way that SAE latents reconstruct multidimensional features has since been referred to as "tiling" (Michaud et al., 2025; Gurnee et al., 2025; Bhalla et al., 2026a). Engels et al. (2024) suggest a pipeline for *discovering* multidimensional features by clustering SAE latents and then reconstructing activations with the SAE but only with the latents in that cluster. For this to work, the correct subset of SAE latents that reconstruct the same multidimensional feature must be chosen. Engels et al. (2024) cluster SAE latents based on the cosine similarity of their decoder directions. Recently, Bhalla et al. (2026a) propose a more sophisticated approach based on clustering SAE latents by their co-activation statistics.[8] One of our contributions in this work is to evaluate this new method. We find that it is a meaningful improvement, but also caution against its use in an entirely unsupervised way.

In this work, we operate within the linear representation hypothesis and the multidimensional superposition hypothesis. However, it remains a deep and open question whether neural networks encode meaning in their activations in ways that ultimately go beyond this model (Wattenberg & Viégas, 2024; Hernandez et al., 2024; Feng & Steinhardt, 2024).

[7] This model is equivalent to the one-dimensional superposition hypothesis but where groups of 1d features $f_j, f_k, \ldots$ co-activate to express a multidimensional feature's value $\boldsymbol{f}_i$. Engels et al. (2024) therefore attempt to define a criteria for when multidimensional features are *irreducible* to independent linear directions. These criteria and definitions should be taken as provisional. Importantly, they are merely statistical, concerning the distribution of values that $\boldsymbol{f}_i$ take, but a notion of independence that makes reference to model computation (via interventions on these feature values) may ultimately be more sound (Wurgaft et al., 2026).

[8] Clarke et al. (2024) also explored clustering SAE latents by co-activation to find multidimensional features.

B Related work

Character counting. Our work builds directly on Gurnee et al. (2025), who studied character counting and line breaking in Claude 3.5 Haiku. In that work, they established the helical geometric structure of the character count manifold and related that manifold to the computations used for line breaking. Sinii & Balagansky (2026) subsequently reproduced counting manifolds in several open-weight language models. We use this setting to investigate how representation manifolds can be discovered and characterized.

Neural network activation geometry. A great variety of works have identified feature manifolds in neural network activations. These include work on vision models (Cammarata et al., 2020; Gorton, 2024) and language models (Engels et al., 2024; Kantamneni & Tegmark, 2025; Modell et al., 2025; Modell, 2026; Gurnee et al., 2025; Bhalla et al., 2026a; Feucht et al., 2026). A variety of other kinds of structure have been found in neural network activations in response to different kinds of structure in data (Li et al., 2024; Diego-Simón et al., 2024; Csordás et al., 2024; Shai et al., 2024; Sarfati et al., 2026). For instance, Park et al. (2025) study how the geometry of representations is shaped by structure only present in-context.

Tiling and other sparse autoencoder failure modes A variety of works have investigated the consequences of mismatches between the assumptions that SAEs make and the actual structure of activations. One productive line of work has studied how SAEs behave when there exist feature manifolds in activations. Engels et al. (2024) study how SAEs reconstruct some simple toy data (a sparse linear combination of circles) to motivate their SAE clustering approach. Michaud et al. (2025) study how feature manifolds may effect SAE scaling, if SAEs allocate a large number of latents towards tiling common feature manifolds. Gurnee et al. (2025) study how crosscoders tile the character count manifold in Claude 3.5 Haiku. Bhalla et al. (2026a) perform a much more exhaustive study of toy data than Engels et al. (2024) and characterize the different ways that SAEs may tile feature manifolds. Other works have investigated SAE failure modes for reasons unrelated to feature manifolds, such as when features in data co-occur hierarchically, (Chanin et al., 2025b), or when SAE dictionary size (Chanin et al., 2025a) or sparsity (Chanin & Garriga-Alonso, 2025) is set incorrectly. The SAE activation function also imposes assumptions on activations (Hindupur et al., 2025).

Periodic representations and computation in transformers. Periodic structure has been found in both toy networks and pretrained language models. Nanda et al. (2023a) established that transformers trained on modular addition use Fourier representations and trigonometric identities to implement the task. Kantamneni & Tegmark (2025) found generalized helical representations of numbers in pretrained language models and investigated how these representations carry out the addition. Notably, in the line breaking task, the current character count is continually added to until the newline, at which point the sum starts back over at 0. Gurnee et al. (2025) connected this literature to character counting, describing a helical representation and the computations that use it to predict line breaks. Our study identifies an additional periodic representation on hexadecimal inputs and investigates its contribution to newline prediction alongside the previously described character-count representation.

Explanations of representation geometry. Several approaches connect representation geometry to the structure of training data and the demands of prediction. Modell et al. (2025) described conditions under which representation manifolds preserve the intrinsic geometry of continuous features. Karkada et al. (2026) derived geometric structure from symmetries in language co-occurrence statistics. Shai et al. (2024) instead start from a known generative process and find representations of the resulting predictive belief states in trained transformers. In Section E.1, our predictive-generator analysis follows this last approach for a simplified line-breaking process. It distinguishes the information sufficient for prediction from the geometry obtained by comparing future newline forecasts, and uses that distinction to interpret aspects of the observed count–width 2D manifold. In our work, the comparison is qualitative; it does not establish a unique geometric encoding or recover the language model's inference algorithm.

C Additional results on the prose character count manifold

C.1 Method for identifying latents that tile a manifold

We use the following method to "manually" identify the latents that tile the character count manifold. Our method ranks latents along three separate metrics, then selects latents that simultaneously are in the top K of each ranking. We first use our "synthetic prose corpus" from Section 2, containing Project Gutenberg paragraphs wrapped to widths 20-160, and compute centroids for character counts 10–150 at the layer corresponding to our SAE (layer 11 of Gemma-2-9B, in this work). We also record SAE latent activations across all tokens in this synthetic corpus, and use this data to rank SAE latents along the following three metrics:

1. **Firing frequency.** We record the total number of times each latent activates $z_i > 0$ across our synthetic corpus, and keep the top $K_f = 5000$ latents. Since character count is represented at most token positions across these documents (except perhaps in the first line or two), the latents that reconstruct the character count manifold should be relatively commonly activating.
2. **Decoder direction geometry.** We perform PCA on our set of centroids and take the top 6 principal components, giving us a 6d subspace of the residual stream. We then project all SAE latent decoder directions onto this 6d subspace, and rank latents by the norm of their decoder direction after this projection. We keep the top $K_g = 500$ latents under this metric. SAE latents can only participate in reconstructing a feature manifold if their decoder directions live in the subspace containing the manifold.
3. **Mutual information.** We lastly compute the mutual information between latent activations z_i and the character count c of the tokens these latents activate on. In particular, we binarize SAE latent activations ($F = \mathbf{1}_{z_j>0}$) and compute $I(F;c)/H(F)$ across our synthetic corpus, where c ranges from 10–150. We keep the top $K_h = 1000$ latents under this metric. This metric excludes latents that fire independently of character count (their tuning curves will be flat as a function of character count).

We find that these criteria produce a set of latents that is somewhat insensitive to the particular K_f, K_g, K_h used, and that the set of latents that meet all three of these criteria is much smaller (20 latents) than each of K_f, K_g, K_h. We note one potential concern, which is our mutual information criterion could exclude latents that always activate when character count is represented, no matter the character count (for SAEs with high L0, such latents could perhaps function as a feature-specific bias term that shift activations in a fixed direction, which may help with SAE reconstruction in some cases). We note that when we evaluated the community detection algorithm of Bhalla et al. (2026a) in Section 2.3, we found in Figure 15 that the auto-discovered community contains a latent that fires uniformly across character count, but nevertheless across The Pile co-activates often with the latents that more clearly tile the character count manifold.

C.2 Tilings are canonical across dictionary size but not across sparsity

We report how SAE "tilings" of the character count manifold vary with SAE dictionary size and sparsity. Across several SAEs in the Gemma Scope (Lieberum et al., 2024) suite, we apply the same criteria as we did in Section 2.2 and described in detail above in Section C.1 to identify SAE latents that tile the character count manifold. The SAE sparsity is reported as e.g. "L0 = 60", where L0 is the average number of SAE latents with nonzero activation, as reported in the Gemma Scope release.

We plot what we've been calling "tuning curves", the probability that latent i fires given that the input token has a given "character count" in its line, across our synthetic corpus of documents below:

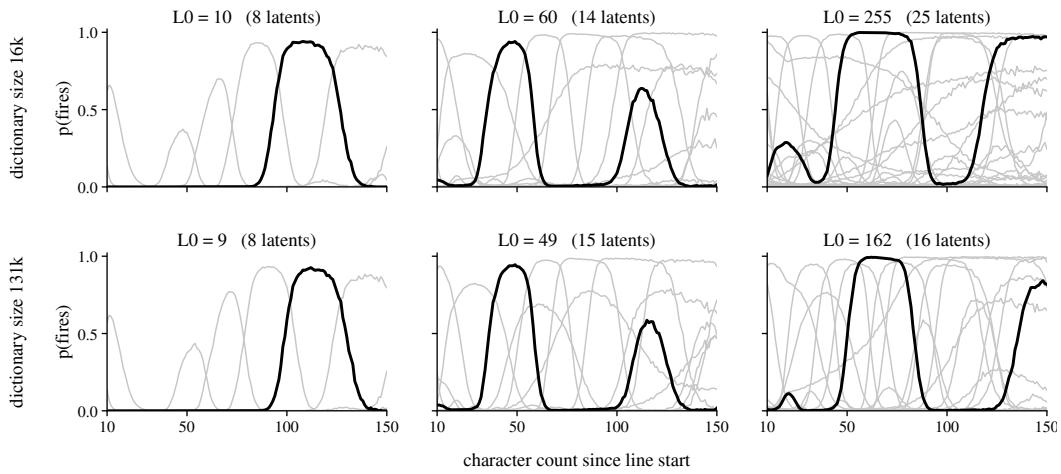

Figure 9: **SAE sparsity determines tiling, not dictionary size.** For a variety of Gemma Scope SAEs (Gemma-2-9B, layer 11), we identify latents that respond to character count and plot their tuning curves. Comparing subplots within columns, we find that SAEs of different dictionary size, but similar sparsity, allocate a similar number of latents to reconstructing the character count manifold, with similar tuning curves. However, comparing subplots within rows, we see that SAEs of different sparsity tile the manifold with a qualitatively different number of latents having different tuning curves. This picture somewhat complicates the claim from Gurnee et al. (2025) that "featurizations" of the character count manifold are canonical. We find that they are canonical across dictionary size, but not sparsity. Note that SAE sparsities are less well-matched in the right column, and, accordingly, we find that the subplots in that column are less alike than other columns.

Overall, we observe that for SAEs with similar sparsity, a similar number of latents tile the character count manifold, and that these latents have qualitatively similar tuning curves. However, we find that SAEs with different sparsity have quite different tilings.

We find these observations noteworthy for a couple of reasons. For one, Michaud et al. (2025) expressed concern that as SAEs are scaled to larger dictionary size, they may pathologically allocate a large number of latents to "tile" common feature manifolds, since doing so can improve the sparsity at which an SAE reconstructs a given manifold (at least in some of the toy settings they studied). Our observation that SAE tilings are consistent across SAE dictionary size is evidence against this being a practical concern, though the range of SAE dictionary sizes we study here is still relatively narrow, and we cannot rule out that much larger SAEs might exhibit different behavior. Gurnee et al. (2025) similarly found that in crosscoders of different scale (1m, 3m, and 10m latents), an identical number of latents appeared to tile the character count manifold and that these latents had almost identical tuning curves. They suggest that the "featurization" (tiling) their crosscoders provide are "canonical". We refine this picture, and find that these tilings are canonical across dictionary size, but vary quite substantially across SAEs with different sparsity.

C.3 Prior character count manifold lengths were arbitrary

Both Gurnee et al. (2025) and Sinii & Balagansky (2026) study the character count manifold in the range of 20-150 characters. However, we find that Gemma-2-9B is able to predict newlines at the correct position at the end of lines in documents of width at least 300 characters. In Section 2 and Figure 2 we studied 10-150 as the range of character counts. In the figure below, we visualize centroids and find the latents that tile this longer manifold, using the same criteria as Section C.1, but applied to character counts from 1-300 instead of 10-150.

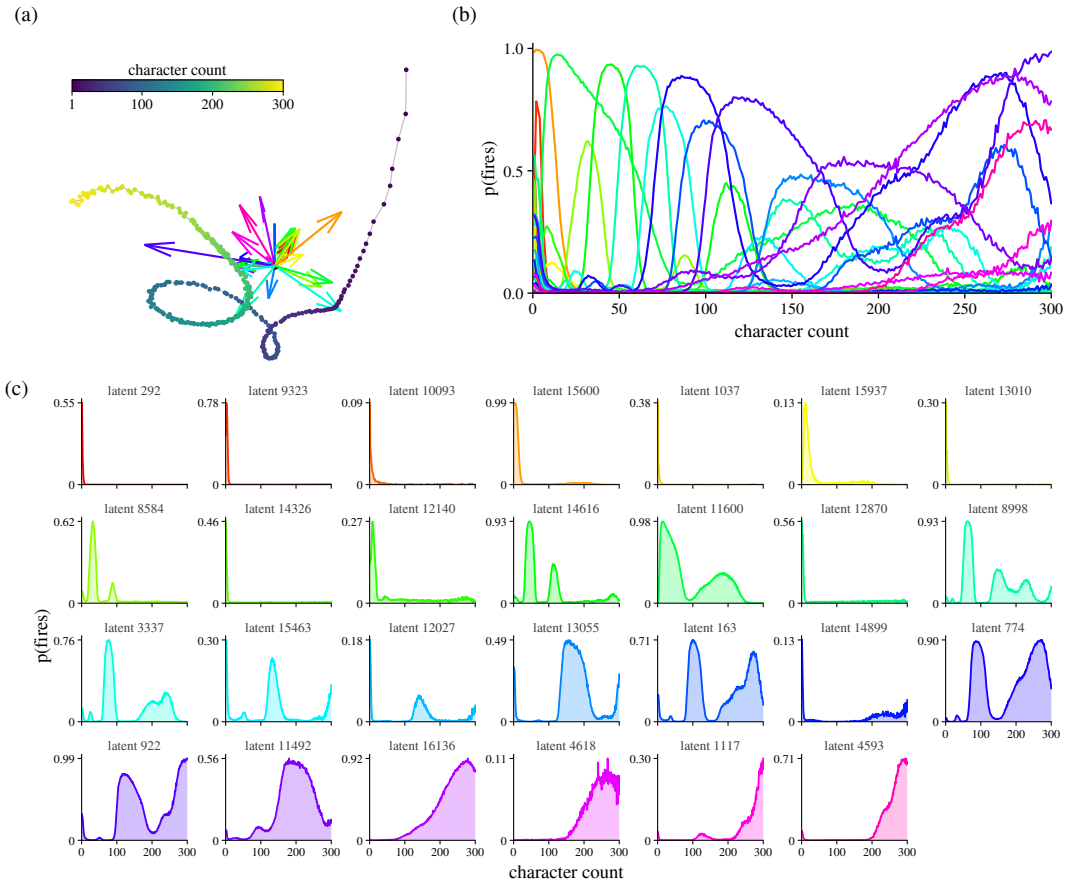

Figure 10: Centroids and SAE tuning curves for the character count manifold in Gemma-2-9B, where we consider a wider set of documents and character counts 1-300 instead of 10-150 as was used in Figure 3. We note that this extended set of centroids is less beautiful but more faithful to the model, which can predict newline tokens at the correct position even in documents of width 300, suggesting the model represents character count up to at least count 300. When we consider this longer range, we also find a wider set of SAE latents that tile the manifold.

We note a couple of observations. First, the manifold is less visually striking, as visualized by the PCA of character count centroids, than it appeared earlier. The choice of the range of centroids (and accordingly the distribution of documents) are a choice that influences the aesthetic quality of the plot. Secondly, a wider set of latents appear to tile the manifold than we found before, though the new latents are largely only responding at very early character counts.

C.4 Additional details on clustering methods

In Section 2.3, we studied whether we could recover the subset of latents that we had manually found to tile the character count manifold, but using automated methods that cluster SAE latents instead. Here we describe the clustering procedure whose results we showed in that section, in Figure 4. Our overall approach is to apply network community detection algorithms, following Bhalla et al. (2026a). To apply these algorithms, we construct a graph where each SAE latent is a node, and weighted edges between latents are given by different notions of similarity. We test five notions of edge weight. Most of these are based on co-activation statistics of SAE latents, so we first describe how we compute these co-activation statistics.

Co-activation statistics on The Pile. We stream the training split of `pile-uncopyrighted` (Gao et al., 2020; Monology, 2023) and truncate each document to 1024 tokens. We run Gemma-2-9B in bfloat16 and encode the layer-11 residual stream with the Gemma Scope SAE from Section 2.2

(16k latents, average L0 118). We exclude BOS, padding, and other special tokens, and keep all remaining token positions. For latent i at token t we define the binary firing indicator $F_i(t) = \mathbf{1}[z_i(t) > 0]$. We accumulate the firing count $n_i = \sum_t F_i(t)$ and the same-token co-firing count $n_{ij} = \sum_t F_i(t)F_j(t)$ for all 16384^2 pairs (the SAE we consider has 16k latents). The harvest covers $N \approx 10^9$ tokens. All co-activation metrics we describe below use the empirical probabilities $p_i = n_i/N$ and $p_{ij} = n_{ij}/N$.

We then compute five notions of edge weight:

- **Decoder cosine similarity** (Engels et al., 2024): $\max\big(0,\, \boldsymbol{d}_i^\top \boldsymbol{d}_j/(\|\boldsymbol{d}_i\|\|\boldsymbol{d}_j\|)\big)$, where $\boldsymbol{d}_i$ is the decoder direction of latent i. This is the only weight that does not use the corpus.
- **Pointwise mutual information (PMI)** (Church & Hanks, 1990): $\mathrm{PMI}_{ij} = \log \frac{p_{ij}}{p_i\, p_j}$, the log of how many times more often the two latents fire together than they would if they fired independently. PMI is unbounded, and very large for pairs of rare latents that co-fire a handful of times.
- **Normalized pointwise mutual information (NPMI)** (Bouma, 2009): $\mathrm{NPMI}_{ij} = \mathrm{PMI}_{ij}/(-\log p_{ij})$. NPMI lies in $[-1, 1]$, equaling 1 when the two latents only ever fire together and 0 when they fire independently. Dividing by $-\log p_{ij}$ removes most of PMI's bias toward rare pairs.
- **Jaccard similarity** (Jaccard, 1912; Clarke et al., 2024): $n_{ij}/(n_i + n_j - n_{ij})$, the number of tokens where both latents fire divided by the number of tokens where either fires.
- **Mean-field Ising couplings** $|J_{ij}|$, described in the next paragraph.

For PMI and NPMI we set negative values to zero. For PMI, NPMI, and Jaccard we set the weight to zero for pairs that co-fire on fewer than 10 tokens ($n_{ij} < 10$), since estimates from so few events are unreliable.

Ising couplings. Bhalla et al. (2026a) convert each latent's firing into a spin $s_i = 2F_i - 1 \in \{-1, +1\}$ and model the joint distribution of spins over tokens with a pairwise Ising model,

$$p(\boldsymbol{s}) \propto \exp\Big(\sum_{i<j} J_{ij} s_i s_j + \sum_i h_i s_i\Big). \tag{1}$$

The fields h_i absorb each latent's firing rate, and the couplings J_{ij} describe the interaction between latents i and j after conditioning on all other latents. Latents that fire on most tokens get large $|h_i|$ but can have small $|J_{ij}|$, and two latents that are correlated only through a third latent can have $J_{ij} = 0$. They then run community detection on the graph with edge weights $|J_{ij}|$, so that strong excitatory and strong inhibitory couplings both count as strong edges.

Bhalla et al. (2026a) fit J by maximizing the pseudolikelihood (Besag, 1975; Ravikumar et al., 2010). This requires every token's full firing pattern. Our co-activation harvest stores only first and second moments (n_i and n_{ij}), which is enough to compute over a billion tokens but not enough for pseudolikelihood. We therefore use the naive mean-field approximation to the inverse Ising problem (Kappen & Rodríguez, 1998; Tanaka, 1998; Nguyen et al., 2017), which estimates the couplings from the inverse of the spin covariance matrix:

$$J_{ij}^{\mathrm{MF}} = -\big(\boldsymbol{C}^{-1}\big)_{ij} \quad (i \neq j), \qquad \text{where} \quad C_{ij} = \mathrm{Cov}(s_i, s_j) = \begin{cases} 4\,(p_{ij} - p_i\, p_j) & i \neq j, \\ 4\,p_i(1 - p_i) & i = j. \end{cases} \tag{2}$$

We restrict the inversion to latents that fire on at least 100 tokens (all but five), add a small ridge term to the diagonal of $\boldsymbol{C}$ before inverting, and use $|J_{ij}^{\mathrm{MF}}|$ as the edge weight. Up to the scaling of each row and column, $-\boldsymbol{C}^{-1}$ is the matrix of partial correlations between spins. The mean-field estimate is therefore the negative precision matrix of the spins, i.e. the quantity that a Gaussian graphical model would use, which Bhalla et al. (2026a) (their Appendix F) describe as the Gaussian analogue of the Ising couplings. Our results suggest that this approximation is adequate for grouping latents, even though, as they note, binarized SAE codes are far from Gaussian.

Our pipeline differs from Bhalla et al. (2026a) in several further ways, some of which reflect differences between their manuscript and their released code:

- **Estimator and regularization.** The manuscript describes pseudolikelihood maximization with L-BFGS, symmetry of J enforced during optimization, and regularization strength chosen by the extended Bayesian information criterion with $\gamma = 0.5$, following IsingFit (van Borkulo et al., 2014). The released code instead fits one L1-regularized logistic regression per latent (predicting that latent's firing from all other latents' firing, with a fixed penalty $\alpha = 0.01$) on 500,000 tokens (by default from C4; the manuscript does not state what data the couplings were fit on), and symmetrizes the coefficients afterwards. We use mean-field inversion on 1B tokens with a small ridge penalty.
- **Community detection.** The manuscript reports Louvain (Blondel et al., 2008); the released code uses Leiden (Traag et al., 2019). We use Leiden, which recovered the reference set substantially better than Louvain in our experiments.
- **Sparsification.** The L1 penalty in their estimator makes J sparse, and their code additionally drops edges with $|J_{ij}| \leq 0.005$ by default. The mean-field J is dense, so we keep only the top 3M pairs by $|J_{ij}|$ as edges (about 2% of all pairs). We apply the same edge budget to every weight in Figure 4, so that all graphs have the same number of edges.
- **Model and SAE.** Their language model experiments use a BatchTopK SAE on Llama-3.1-8B; we use a JumpReLU Gemma Scope SAE on Gemma-2-9B.

We also implemented the estimator from their released code (per-latent L1 logistic regression with liblinear, fit on a random sample of 500,000 Pile tokens with the same tokenization as above) and ran it through the same Leiden sweep. With their penalty $\alpha = 0.01$, the peak IoU at 3M edges was 0.52, compared with 0.70 for mean-field couplings. At $\alpha = 0.01$, the penalty is negligible compared to the log-likelihood summed over 500,000 tokens, and rare pairs of latents whose firing is nearly perfectly predictive of each other receive very large couplings, which crowd the reference latents' couplings out of the top edges. Raising the penalty to $\alpha = 100$ raised the peak IoU to 0.68 (at this penalty J has fewer than 3M nonzero pairs, so the graph uses all of them). We chose $\alpha = 100$ from a handful of values using this same benchmark, so the 0.68 is an optimistic estimate.

Community detection and scoring. The graph has one node per SAE latent (16,384 nodes) and one weighted edge for each of the top 3M pairs. We partition it with the Leiden algorithm (Traag et al., 2019) as implemented in `leidenalg`, maximizing the modularity with a resolution parameter γ (Reichardt & Bornholdt, 2006). Larger γ gives smaller communities. We run Leiden to convergence at 121 values of γ spaced logarithmically between 1 and 10^4, with three random seeds each. For each partition, we compute the max IoU, $\max_S |S \cap R|/|S \cup R|$, where S ranges over the communities of the partition and R is the reference set of 20 latents from Figure 3. Figure 4(a) shows the mean of max IoU over the three seeds at each resolution, with the range over seeds shaded. Mean-field Ising couplings reach a peak mean IoU of 0.70 (at $\gamma \approx 200$), while the other weights peak between 0.49 and 0.54. The best community from the best single partition has 21 latents, 17 of which are in the reference set; Figure 15 compares the two sets.

Sensitivity to the edge budget. The ranking of the weights depends on how many edges we keep. With a budget of 100k edges instead of 3M, Jaccard similarity does best (peak mean IoU 0.63) and mean-field Ising couplings do poorly (0.18). The regularized pseudolikelihood estimator ($\alpha = 100$) was the only weight that scored well at both budgets. The mean-field couplings need a dense graph: the top 100k pairs by $|J_{ij}|$ include only 3 of the 190 pairs within the reference set, compared with 85 of the top 3M.

Projecting activations onto a community. For Figure 4(b), we follow the visualization procedure of Bhalla et al. (2026a) as closely as their manuscript and code allow, using the best community S described above. We stream 3M further tokens of `pile-uncopyrighted` with the same tokenization and record the layer-11 residual stream at tokens where any latent in S fires, keeping a random sample of 60,000 such tokens. Following their conventions, a latent only counts if it fires on at least 30 of these tokens, and an activation only counts if it is at least the 10th percentile of that latent's nonzero activations. We subtract the mean activation over these tokens, project onto the span of the community's 21 decoder directions, run PCA on the projected coordinates, and plot a random subset of tokens on the top three principal components. The span of the decoder directions contains only 1.6% of the variance of these activations.

C.5 ADDITIONAL FIGURES

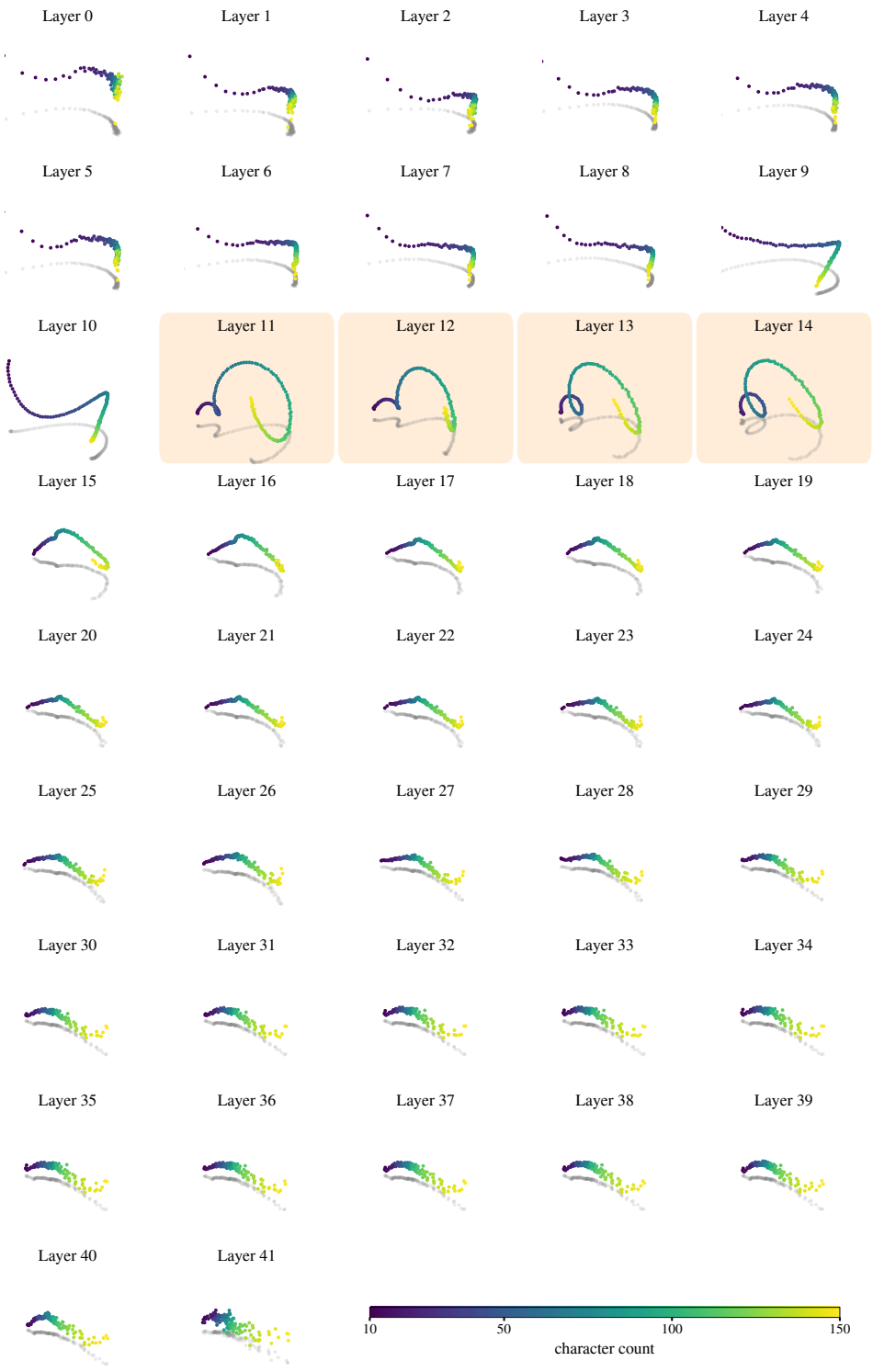

Figure 11: Character count centroids across all layers of the residual stream of Gemma-2-9B. We find that there is a relatively narrow band of layers where centroids organize into their helix shape.

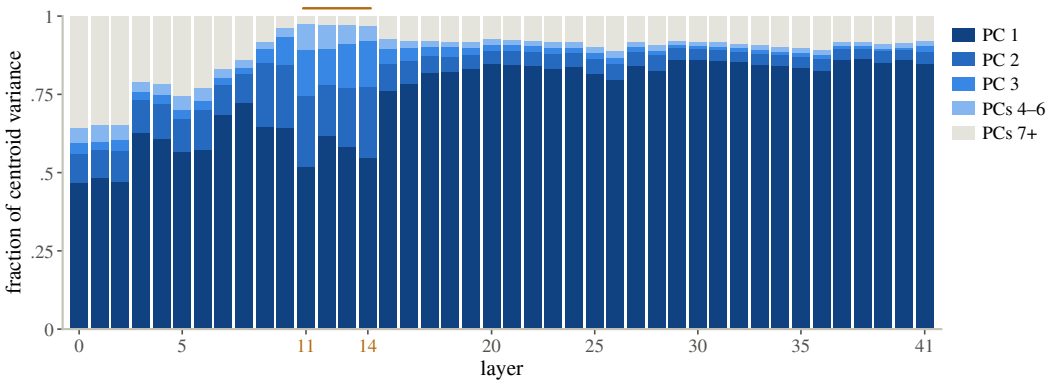

Figure 12: **Distribution of centroid variance across principal components, by layer.** For the character count centroids of Gemma-2-9B (character counts 10-150), we show the fraction of centroid variance captured by PC 1, PC 2, PC 3, PCs 4–6, and all remaining PCs. In layers 11-14, variance is shared across the top several PCs (PC 1 captures only 52–62%), and PCs 1–6 together capture 97–98%. In later layers PC 1 dominates (76–86%).

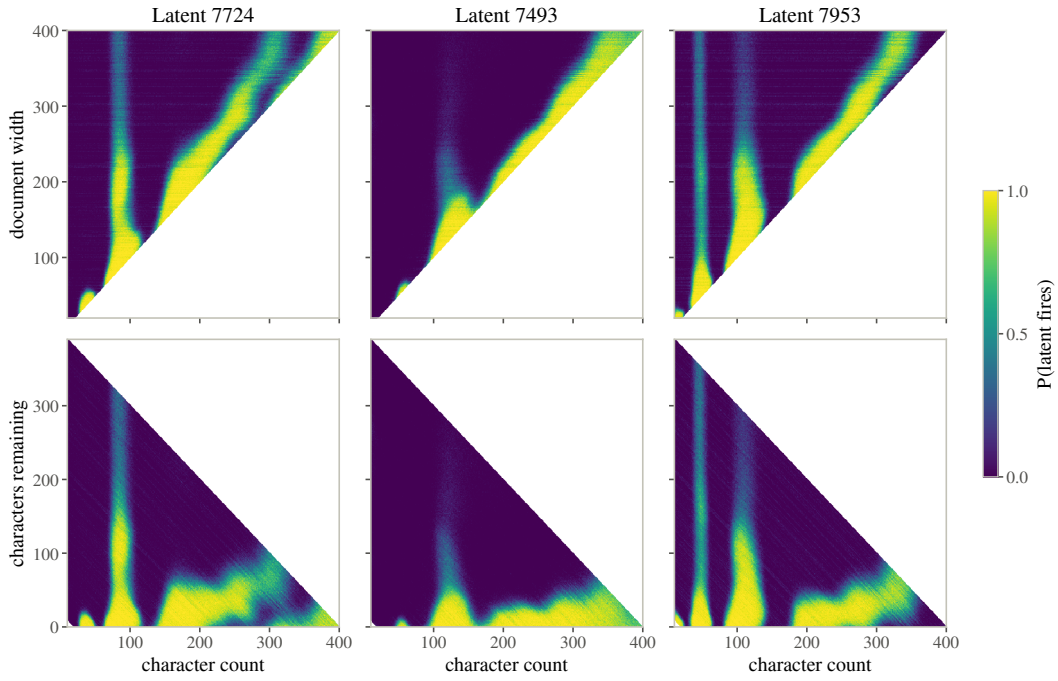

Figure 13: **SAE latents respond to both character count and document width.** For three latents of the Gemma Scope SAE at layer 14 of Gemma-2-9B (16k latents, average L0 137), we show the probability that the latent fires as a function of character count and document width (top) and of character count and characters remaining on the line (width − character count; bottom), on our synthetic prose corpus extended to widths 20–400. All three latents are among those that tile the character count manifold at this layer. On narrow documents, each fires in a band of character counts (vertical stripes in both rows). On wide documents, it instead fires a fixed number of characters before the end of the line (horizontal bands in the bottom row).

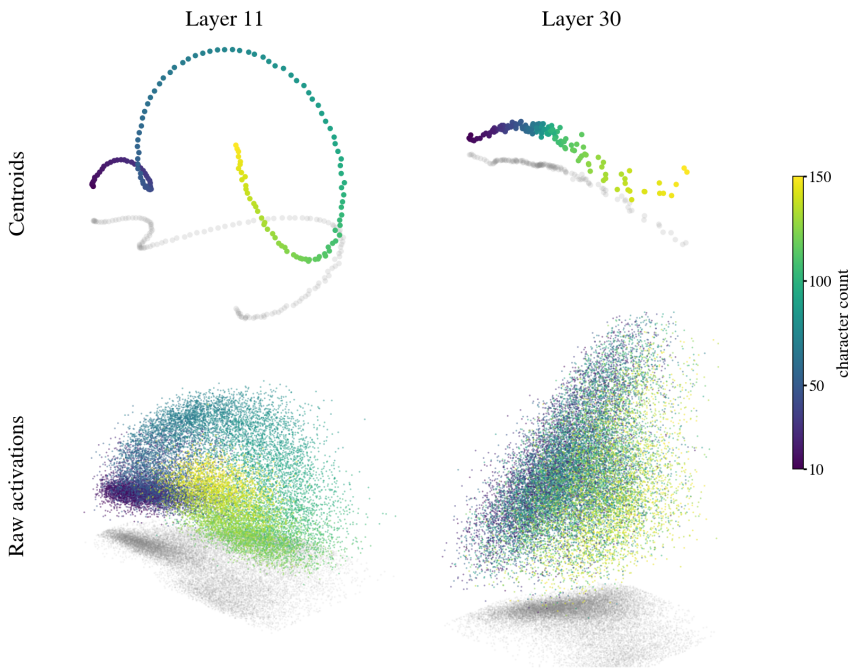

Figure 14: **Centroids versus raw activations.** Top: character count centroids of Gemma-2-9B at layers 11 and 30 (character counts 10-150), shown in their top three PCs as in Figure 2. Bottom: residual stream activations of individual tokens from the same synthetic prose corpus (140 tokens per character count), projected onto the same three PCs. Points are colored by character count, and each panel has its own scale. At layer 11, raw activations are still loosely ordered by character count. At layer 30, the centroids lie along a curve ordered by character count, but the raw activations are largely mixed.

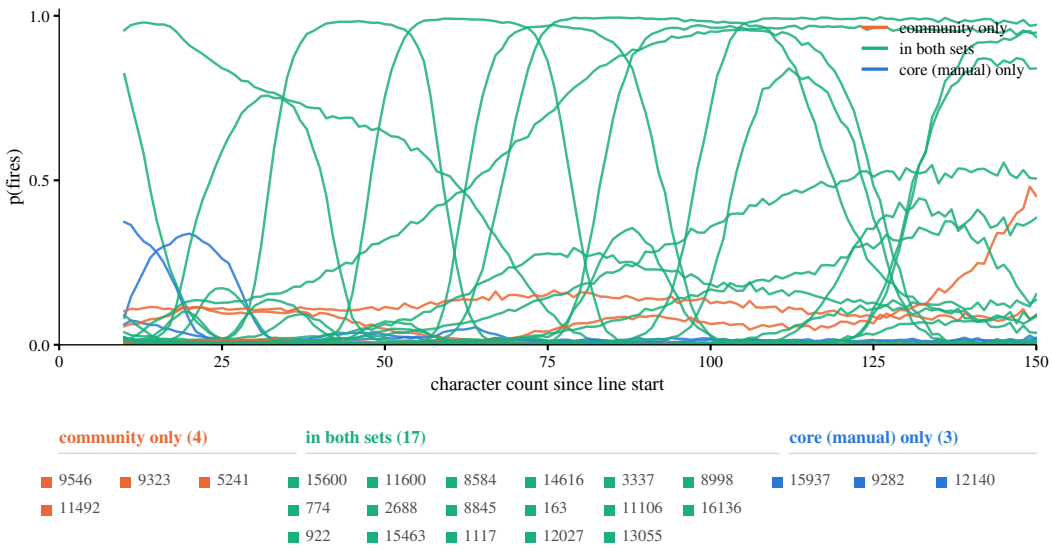

Figure 15: **Manually identified latents versus the best-matching Leiden community.** We show tuning curves $p(\text{latent fires} \mid \text{character count})$ on our synthetic prose corpus for latents that are in our manually identified set of 20 latents from Figure 3 and also latents that are in the Leiden community with the highest IoU in Figure 4 (mean-field Ising $|J|$, resolution ≈ 200, IoU 0.71). We color latents according to whether they are in the manually-identified versus Leiden-community set. Green: latents in both sets; orange: community only; blue: core set only. The Leiden community contains all of the strongly tuned latents.

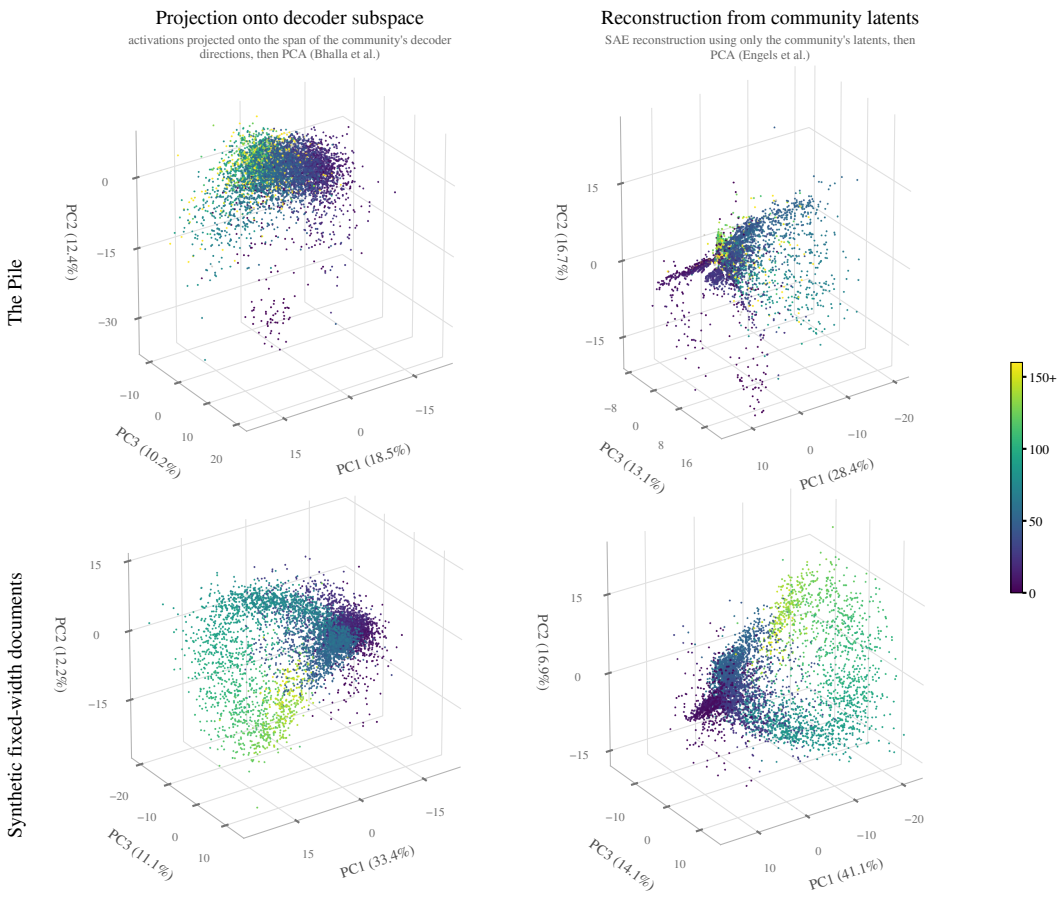

Figure 16: **Raw activations viewed through the Leiden community of Figure 15.** Rows: activations from The Pile and from our synthetic fixed-width documents, restricted to tokens on which a community latent fires. Columns: projection onto the span of the community's decoder directions (Bhalla et al., 2026a) and reconstruction from the community's latents alone (Engels et al., 2024), each followed by PCA. Points are colored by character count.

D Additional results on periodic character count representation

D.1 Overlap between the periodic planes and the prose representation

We measure how much the period-8 and period-4 planes from Section 3 overlap with the prose character count representation. We take the prose representation at each layer to be the span of the top 10 principal components of the character count centroids on our synthetic prose corpus (line widths 20–160, character counts 10–150). Figure 17 reports two measures: (a) the share of each plane that lies inside this prose subspace (the mean squared cosine of the principal angles between them), and (b) the share of the prose centroids' variance across character counts that lies in each plane. From layer 11 onward, where the periodic representations are clearly present, about 4% or less of each plane lies in the prose subspace and each plane contains less than 1% of the prose centroid variance, except at the final layer (8% and 2%). At layers 0–10 the overlap is much larger (up to 22% and 16%). The periodic structure is weak at these layers, so the least-squares fit mostly picks up the smooth character count representation instead. For this reason we project the prose subspace out of the planes in our ablation experiments (Section D.2).

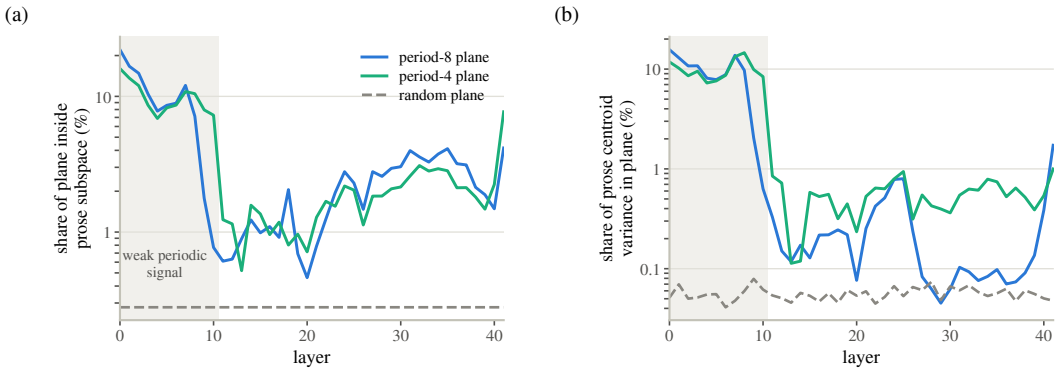

Figure 17: **The periodic planes are nearly orthogonal to the prose character count representation at the layers where they appear.** For each layer of Gemma-2-9B, (a) the share of the period-8 and period-4 planes (fit from hex centroids) lying inside the span of the top 10 principal components of the prose centroids, and (b) the share of the prose centroids' variance lying in each plane. Dashed lines show the same quantities for random planes. In the shaded layers (0–10), the hex centroids have little periodic structure, and the fitted planes mostly capture the smooth character count representation.

D.2 Details on ablation experiments

Here we describe the experiment in Figure 7, in which we remove the periodic parts of Gemma-2-9B's character count representation on hex strings and measure how well the model still predicts the newline.

Subspaces. We ablate the period-8 and period-4 planes described in Section 3 and shown in Figure 5, fit from the same hex centroids that we use for the Fourier analysis (character counts 12–75, on documents with line widths 20–99). At each layer, we subtract the mean $\boldsymbol{\mu}_L$ of the centroids over character counts. For a period P, we fit the centered centroids by least squares to $\cos(2\pi c/P)$ and $\sin(2\pi c/P)$, which gives two directions in the residual stream; orthonormalizing them gives the *period-P plane*. For "both planes", we orthonormalize the four directions from periods 8 and 4 together, giving a 4-dimensional subspace.

Because the planes are fit by least squares, they can partly overlap with the smooth character count representation that the model also uses on prose, especially at early layers (Section D.1). To make sure the ablation does not touch that representation, we compute character count centroids on our synthetic prose corpus from Section 2 (line widths 20–160, character counts 10–150), take their top 10 principal components at each layer, project these directions out of each plane, and re-orthonormalize. The hex centroids' projection onto the resulting planes keeps at least 86% of its root-mean-square size at every layer (over 97% from layer 11 on), so this step removes little of the periodic signal. Without it, the ablation has a larger effect (ablating both planes lowers the probability of predicting the newline at the end of the line to 0.72 instead of 0.84), so part of the effect of the unprojected planes comes from the prose representation. We compare against two other subspaces:

- **Top-10 hex PCs.** The top 10 principal components of the hex centroids at that layer. These contain the smooth character count representation as well as the periodic one, so ablating them shows the effect of removing character count information almost entirely.
- **Random 4d.** A random 4-dimensional subspace, drawn separately at each layer and projected off the prose subspace in the same way. We also tried a random 2-dimensional subspace and a single random 4-dimensional subspace shared by all layers. None of these random controls changed newline prediction.

Ablation. For a subspace with orthonormal basis $\boldsymbol{V}_L$ (one row per direction), we replace the residual stream $\boldsymbol{h}$ at layer L by

$$\boldsymbol{h} - \boldsymbol{V}_L^\top \boldsymbol{V}_L(\boldsymbol{h} - \boldsymbol{\mu}_L), \tag{3}$$

which sets the component of $\boldsymbol{h}$ in the subspace to its value at the centroid mean. We apply this at every token position and at every layer from 6 to 36 in the same forward pass, each layer using its own subspace.

Measuring newline prediction. We generate 40 new documents, each containing 30 lines of exactly 48 random lowercase hexadecimal characters. At each token, we compute the probability that the next token is a newline, summing over all tokens in the vocabulary that contain a newline character. We group positions by the character count of the line after the current token, and only use positions from the third line onward, since the model is still inferring the line width during the first two lines. The position with character count 48 is the last token before the newline, where the model should predict it. Figure 7 shows the mean probability at the last four positions of the line, averaged over about 1,000 positions for each character count.

Robustness. We repeated the experiment on documents with line widths 30–80 (in steps of 5), aligning positions by the number of characters remaining in the line, and found the same ordering of effects. In particular, ablating both planes reduced the probability of predicting the newline at the end of the line from 0.96 to 0.81.

D.3 Additional figures

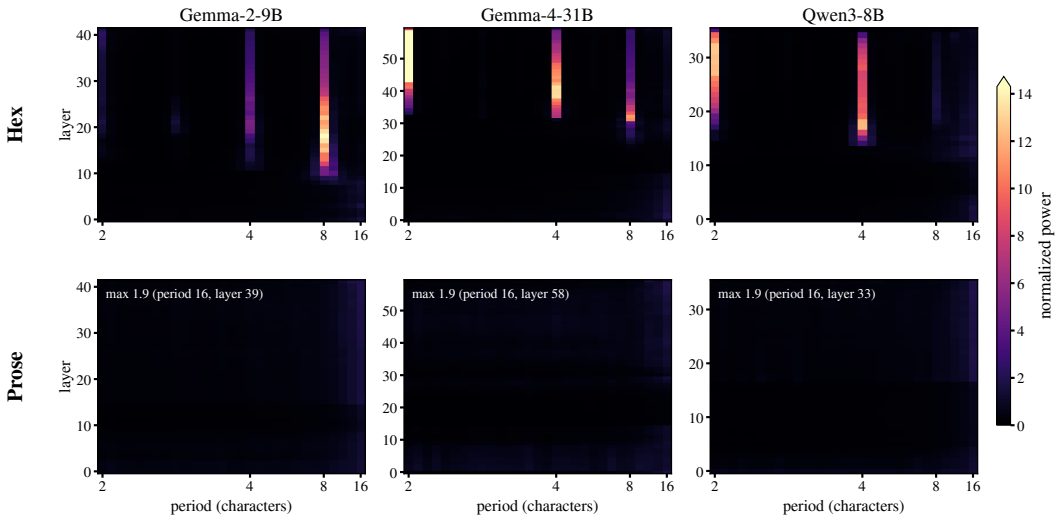

Figure 18: **The periodic structure appears on hex strings but not on prose.** Fourier power of character count centroids across layers and periods, for hex documents (top) and for Project Gutenberg prose wrapped to fixed widths (bottom), in Gemma-2-9B, Gemma-4-31B, and Qwen3-8B. Both corpora use line widths 20–99 and character counts 12–75, and all six panels share one color scale. On this scale the prose panels are nearly black, so we label each with its maximum value, which is at period 16 for all three models.

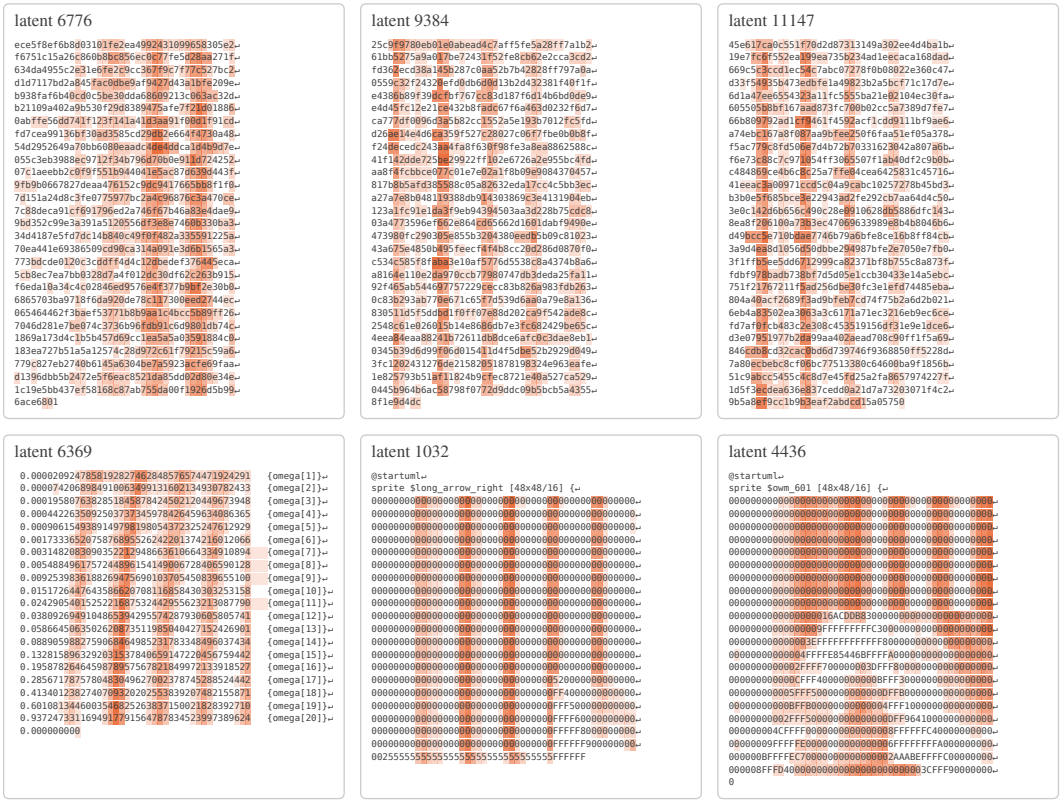

Figure 19: For a subset of the latents in the cluster discussed in Section 3, we highlight tokens by latent activation (z_i) across a selection of documents from The Pile. We find that the latents in the cluster respond periodically, with a period of roughly 8 characters, on strings of digits or hex. For each latent, we considered the top 50 documents on a subset of the Pile (of about 2 million documents) where that latent activated on the most number of tokens ($z_i > 0$). We chose these documents among the top 50 documents because the pattern is especially clear here, but note that not all documents suggest some periodic structure as clearly as these do. Nevertheless, seeing documents like these made us suspect that there was an underlying periodic structure here.

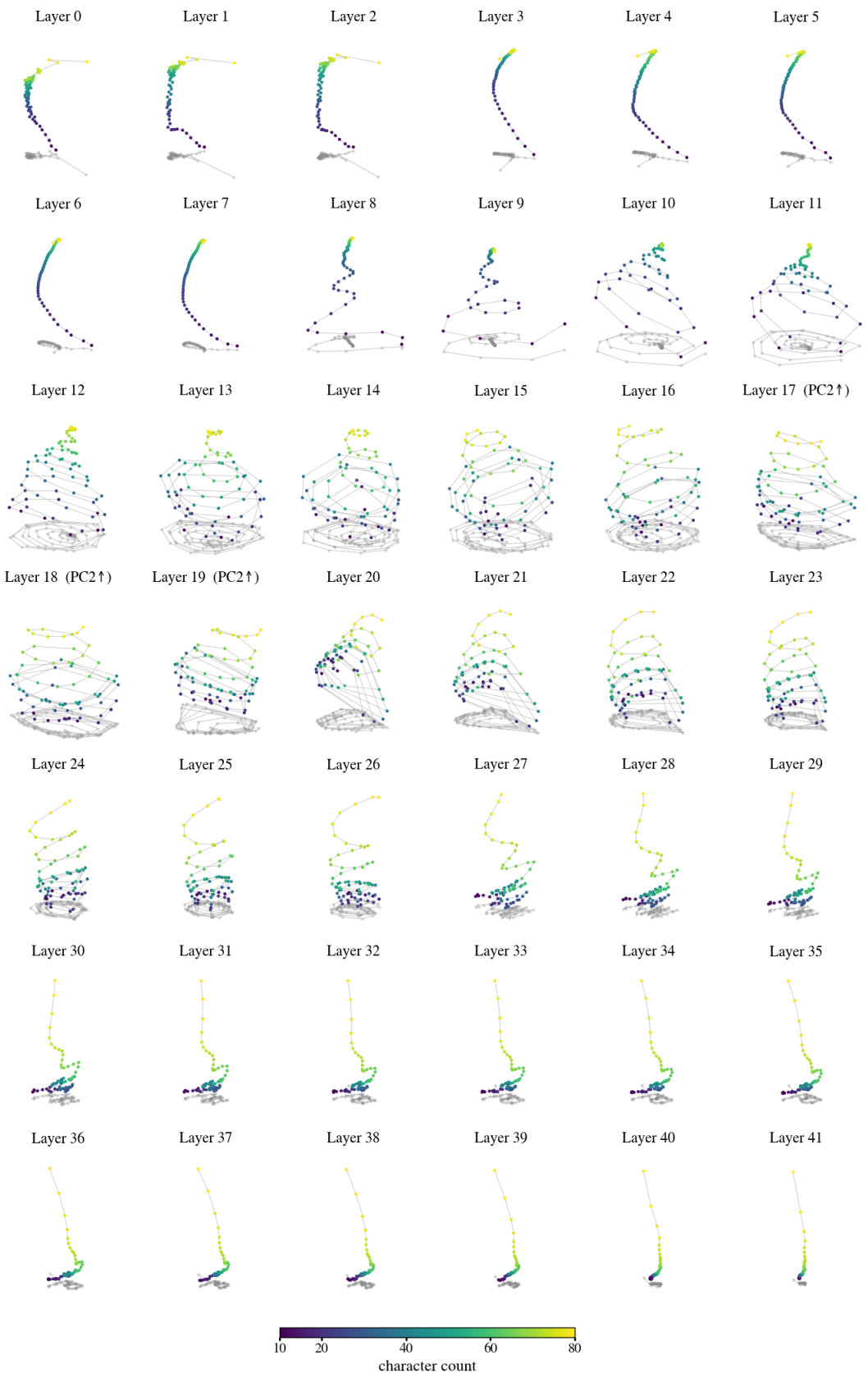

Figure 20: PCA of character count centroids on documents where each line contains random hexadecimal strings of a fixed length.

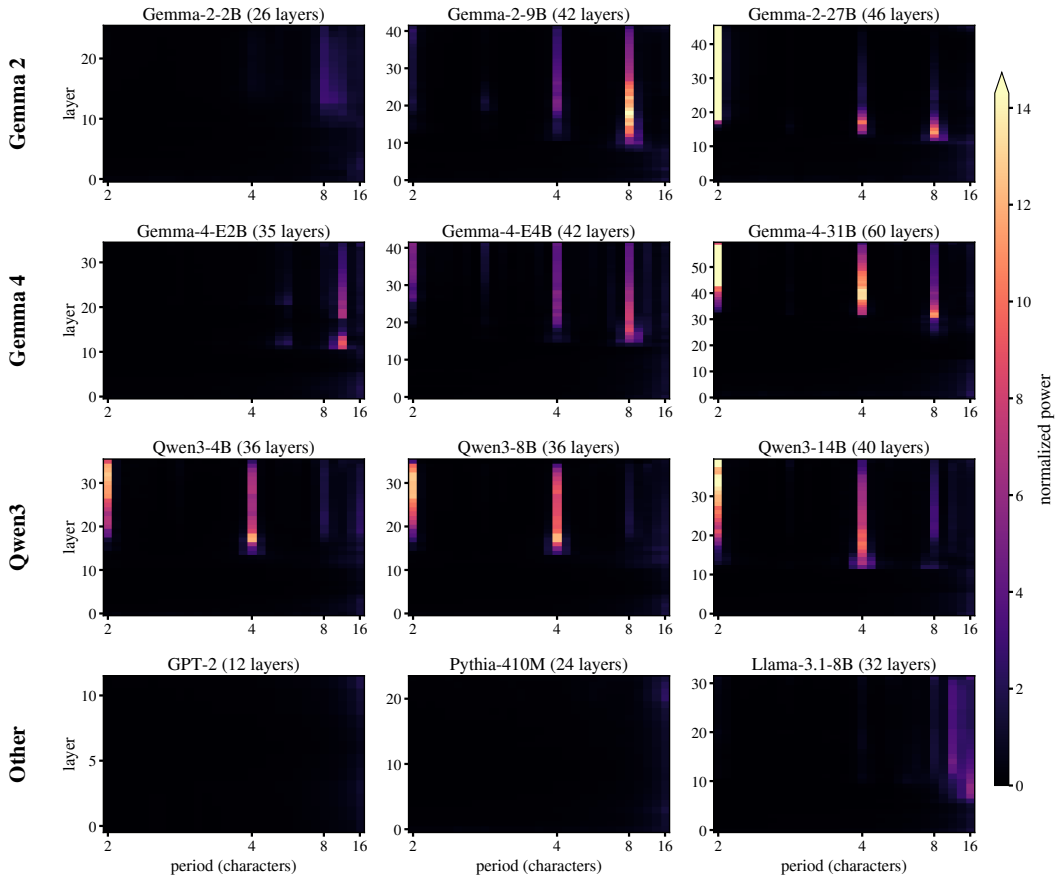

Figure 21: **Fourier power of hex character count centroids across open-weight models.** This is the same analysis as Figure 5 but for a larger set of models.

E THE COUNT × WIDTH MANIFOLD: THEORY, TOY MODELS, AND TESTS IN GEMMA-2-9B

In this section we discuss a variety of points related to the apparent 2d structure we visualize in Figure 8. We demonstrate that a toy model trained on the line breaking task only displays two-dimensional sheets that begin relatively flat, and then fold into shapes that resemble the shape we observe at layer 15 of Gemma-2-9B. By analyzing a compact predictive generator of the task, we argue that the newline prediction task itself asks that the model to organize its activations primarily by characters remaining in the current line– which gives a computational ansatz for what drives the evolving structures we see between layers 9 and 15. The ground truth embedding geometry that we propose captures the general structure of the activations both in Gemma-2-9B and the toy models. We then proceed to validate the reorganization quantitatively in Gemma specifically with linear probes and variance decomposition. We end by investigating to what extent the geometry is causal with a suite of steering and ablation tests.

E.1 PREDICTIVE GENERATOR ANALYSIS

This appendix connects the character counting geometry of the main text to a growing body of work that treats the geometry of a network's representations as something inherited from the *prediction task itself*. Shai et al. (2024) showed that transformers trained on data from a known generative process linearly represent, in their residual stream, the geometry of the *belief states* of an optimal predictor of that process. Subsequent work extended the picture to processes composed of indepen-

dent factors (Shai et al., 2026), and to mixtures of processes (Ray et al., 2026). The data we study in the present work is itself a mixture of documents of different fixed-widths, an instance of the latter.

The methodology in this line of work has three steps. First, write down a generative process for the training data. Second, derive the state an optimal predictor must carry to track predictions across contexts and the geometry that state inherits from the predictions it supports. Third, look for that geometry in the residual stream.

We therefore start by coming up with a *predictive generator*. For our current purposes, we can simply think of this as a machine that can both generate data consistent with fixed width linebreaking, and track predictions over future tokens (For a short technical overview, refer to Section F.1.) If we focus solely on the subtask that determines when newlines are emitted, a model of the data generating process is the following:

- at the beginning of each rollout, a width w is selected uniformly from a range of widths, and c is set to 0
- at each position one of k tokens of length c_k is sampled from an IID distribution p_k.
- if $c + c_k \leq w$, increment c to $c + c_k$ and sample again
- if $c + c_k > w$, emit a newline followed by token k; set $c = c_k$, then sample again.

The data itself does not reveal the selection of w, and so it must be learned through inference from the observed data.

There exists no sufficiently compact linear presentation that generates data from the distribution above (see Section F.3). A compact sufficient statistic however that can be tracked using a *probabilistic counter machine* (Minsky, 1961; Fischer et al., 1968; Rabin, 1963) with four registers (fully detailed in Section F.2). Of these four, only three are relevant at the content tokens: c the current character count into the line, L a lower bound on w, and U an upper bound on w. These three numbers can be used to calculate all future token distributions, and are updated iteratively at each new context position. Both the probability calculations and the state updates are defined by piecewise affine maps. c is fully observable and is updated through observed tokens, resetting at each newline. Every emitted token that isn't a newline proves $w \geq c$ (raising L, the lower bound on w); every newline proves $w < c + c_k$ with c_k being the overflowing token's character count (lowering U, the upper bound on w). A network that tracks these registers should carry them as a low-dimensional manifold in its activations, one point per reachable register value.

E.2 Do models represent the Sufficient Statistic?

We train a toy model transformer on only the line counting data described above, over a mixture of fixed widths (20-150) with tokens of lengths 2-8. Figure 22 shows activations from that model after the second attention block. We see a clear manifold, even in the raw activations, that resembles an embedding of the sufficient statistic. In harvesting these activations, we neglect context positions before the first 3 lines breaks– which is why $U \approx L$.

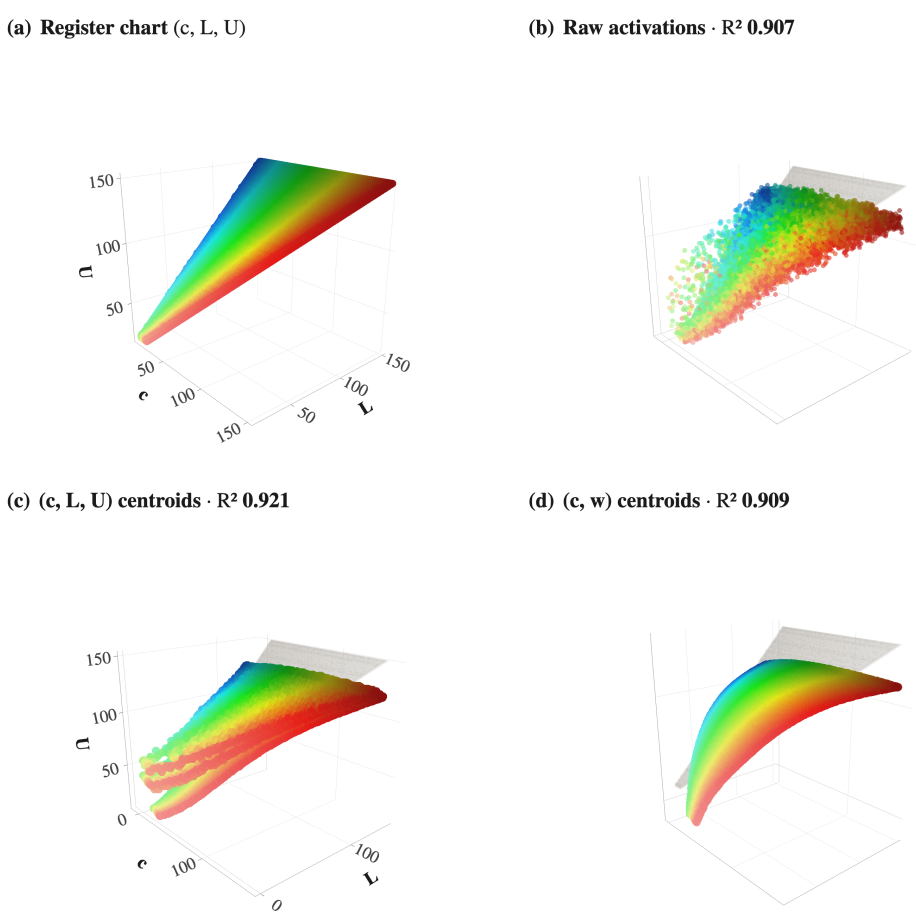

Figure 22: Early in these toy models, we find that the raw activations (b) are recognizably related to the predicted register values (a). All four panels share the (c, L, U) chart space; color gives progress along the line (hue) and width (lightness). (a) The reachable register values of the predictive generator. (b) Raw toy model layer-1 activations mapped onto the chart by a count-weighted rigid Procrustes fit (rotation, uniform scale and translation) on the top-6 PCs of the (c, L, U) centroids. (c) The (c, L, U) centroids under the same fit. (d) The (c, w) centroids under the same rigid fit. Full linear maps reach $R^2 \geq 0.99$ for any of the groupings.

The raw activation PCA already reveals a recognizable manifold, but we can also look at centroids where activations that fall into the same (c,L,U) bin are averaged together. The centroids are then used as the basis for calculating the principal components. This averaging reveals the structure more cleanly than the raw activations.

The careful reader will notice the appearance of several distinct sheets when looking at the c, L, U centroids. This is due to a coupling between another variable the model represents: the total number of newlines seen. Because (c, L, U) cells with larger $U - L$ happen disproportionately early in a document– we find that the models representation of the number of newlines seen thus far leaks into the structure. This effect is more pronounced for narrower documents because they move through newlines more quickly in context. This is why the effect appears stronger at the narrower end of the register.

These network only know about one simple task, so both the raw activations and the centroids show the same general structure. Language models should not be expected to work this way. The raw PC's for the activation stream when Gemma is doing the analogous task contains more information than just predicting the next newline. Task specific centroids, however, average out the information not related to the task at hand. This motivates using the centroids when investigating activations for this task in Gemma-2-9B.

Rather than use (c, L, U), we use (c, w) centroids. The use of U, L as a well motivated object depends entirely on an assumption about the prior over the width task. Rather than the curated dataset we created for the toy models, Gemma's effective prior on the task comes from pretraining and cannot be deduced a priori. By averaging over (c, w) bins, we throw away early, transient information. This is exactly the kind of information we don't have about Gemma's inference algorithm; it's the right information to throw away.

While we don't expect Gemma to have the same prior as our toy, it may be reasonable to expect that the recurrent part of the mechanism is shared. As any predictor gathers evidence, it will refine its belief about the true width — and so all reasonable estimators converge on the same true width eventually. For example, while $U - L$ technically can range from 0 to the maximum width in the dataset — nearly all of this freedom is gone by the time a model has seen the first newline token. In the case of Gemma, character counts in emitted tokens are uniform across the context window — but not across the character counts themselves. The distribution of character counts per token peaks between 4 and 5 characters, which means no more than 3 viable widths typically survive past the first 3 lines. In short: The lion's share of inference in both Gemma-2-9B and the toy models are under conditions for which $U \approx L$.

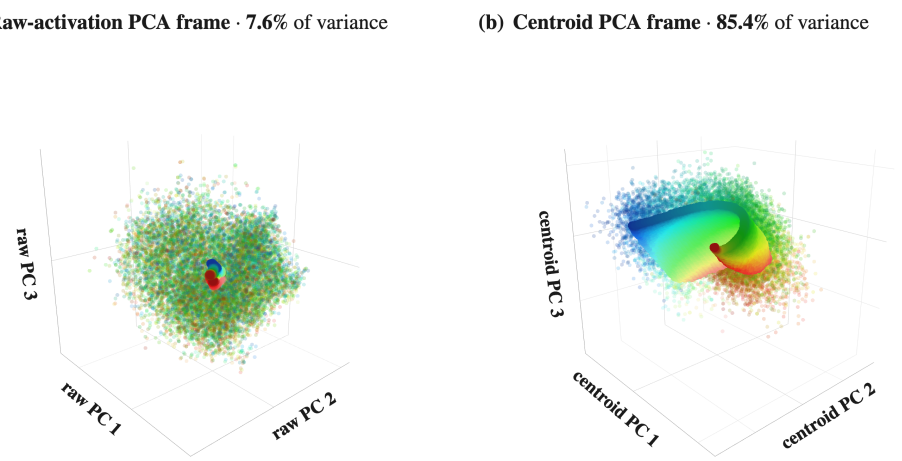

Figure 23: Gemma's raw activations are not as friendly as the toy model: they only reveal the target structure when we look at the task-specific centroid PCA. The same (c, w) centroids (solid) and a sample of raw activations are shown in (a) the top-3 PCA frame of the raw activations and (b) the top-3 PCA frame of the centroids; Each panel gives the share of that point set's variance the three axes hold. Color gives progress along the line (hue) and width (lightness). In the raw frame the centroids collapse to a knot at the center of the cloud; in their own frame they form a curved sheet that resembles the predicted register.

Figure 23 shows the geometric arrangement of Gemma's centroids at L10. A linear map between Gemma's centroids and the sufficient statistic is not as strong as for the toy models, but it is worth pointing out that they appear to organize information in the same way. The visible curve in all of these manifolds indicates a *nonlinear* embedding of the sufficient statistic, which may both explain the failure of the linear map as well as providing the ground work for a linear readout of next-token probabilities later in the model. We investigate this possibility in the next section, by looking at some potential ways the model could embed the w, c manifold in service of the newline prediction task.

E.3 What determines the centroids arrangement in activation space?

The pictures above place each (c, w) cell at its register values: the position on the current line, and the width of the document it came from. Let's call this the register embedding. It comes with a

sense of distance — two states are close when their registers are close. All our plots have used that distance implicitly. It is not the only choice.

The registers are only worth tracking because they determine the predictions (that is what it means for them to be a sufficient statistic). However, while the registers are enough to keep track of optimal prediction, they do not place states *by* what they predict. This would be a second sense of distance: two states are close when they assign similar probabilities to future tokens. A few more characters typed towards the end of a line barely move the registers but can raise the probability of an immediate newline from zero to one. This is where the curvature noted above comes from: the registers can be read off the activations linearly, but the probabilities are a nonlinear function of the registers, so a linear readout can only produce them from a curved embedding. Early in the model we see evidence of a more generic register embedding. As the model turns representations into predictions, we will see the centroids reorganize to respect the prediction task.

Take a fixed horizon of D future steps, where a step is one emitted symbol (a content token or a newline). We now assign a vector of length D to each (c, w) state whose h-th entry is the probability that the symbol h steps ahead is a newline:

$$v(c, w) \;=\; \Big(P(x_h = \text{newline} \mid c, w)\Big)_{h=1}^{D}$$

This length D vector is the basis for what we will call a *forecast embedding*. To see a manifold where distances are related to Euclidean distances between forecasts, we project these vectors onto their directions of largest variance (the principal components). This embedding only cares about newlines, not the full token distribution, which is the appropriate consideration for centroids that also only care about newline (Section F.4 proves that future distributions of content tokens drops out of the embedding except at the newline position itself).

Figure 24 shows two statistics for these vectors: the marginal newline probabilities above, and the *next-newline* only, whose h-th entry is the probability that the current line ends exactly h steps ahead, $P(\text{first newline at step } h \mid c, w)$ for $h = 1, \ldots, D$. This is also a length D vector, but will only have nonzero entries within the current line.

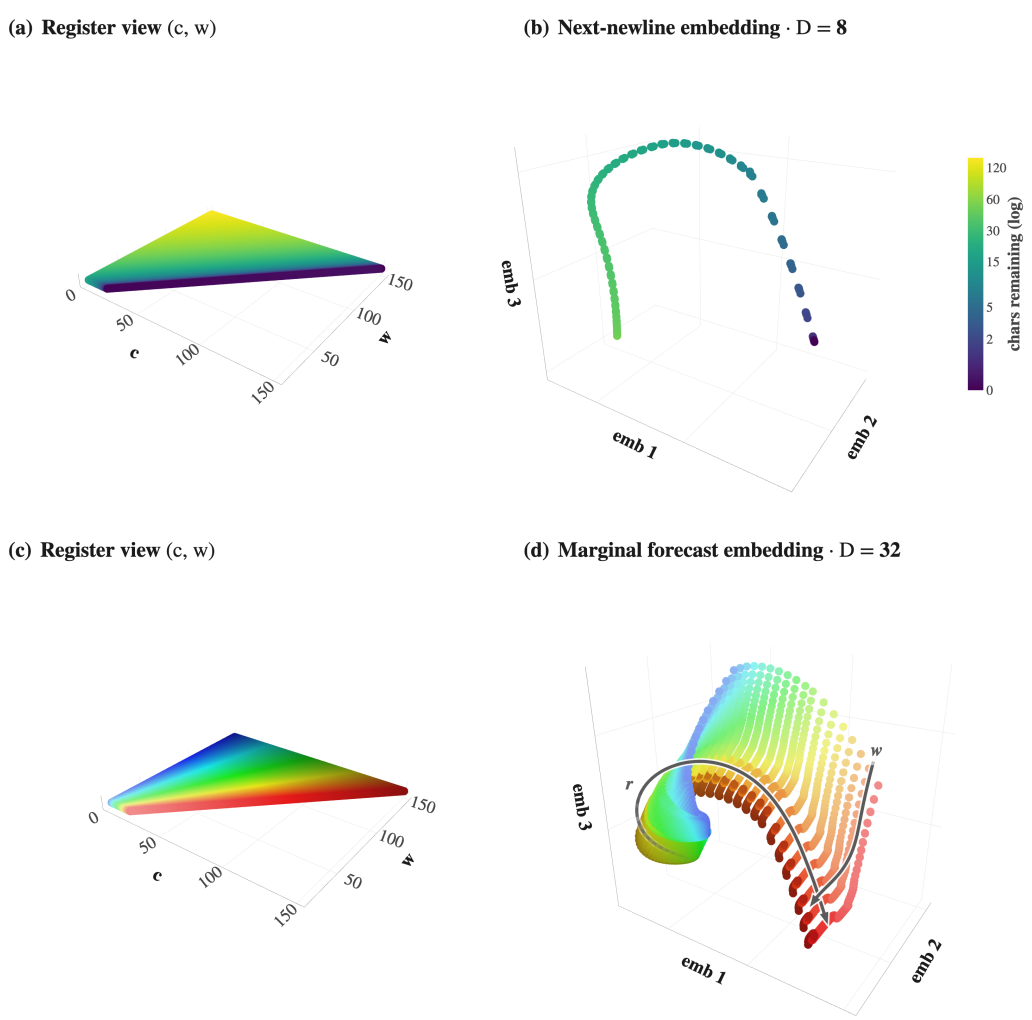

Figure 24: Register view (a, c) and forecast embedding (b, d) of the (c, w) cells for a mixture (token lengths 2–8, widths 20–150). Top: the next-newline embedding at horizon D = 8, colored by characters remaining. Cells far from the line end share nearly identical forecasts and collapse toward a point. Bottom: the top-3 PCA of the marginal newline forecast at horizon D = 32, colored by progress into the line (hue) and width (lightness). Embedding axes are PC scores.

Let's start by investigating the next-newline embedding. If we are c characters into a line of width w, whether any string of tokens $x_1, \ldots, x_m$ fits on the current line is decided by

$$c_{x_1} + \cdots + c_{x_m} \;\leq\; w - c$$

Thus, a state's forecast of everything up to the next newline can be expressed as a function of the room $r = w - c$ alone. We can see this visually: the register embedding folds up upon itself, and the organizing variable becomes characters remaining before the next newline $r = w - c$ (this behavior is also observed in Gemma-2-9B activations; we quantify it in Section E.4).

The mismatch between register and forecast embeddings also shows up as uneven spacing along a single line. In the register view, successive positions are evenly spaced: each character moves the state by the same amount. In the forecast embedding the spacing varies. While there are many characters remaining, each character barely changes the predictions, so successive positions sit nearly on top of one another. Closer to w, each character changes the forecast significantly, and the spacing grows. The marginal statistic shows this spacing at every horizon D (Figure 24, bottom, shows $D = 32$), sharpest at one step and softer as the horizon grows.

The total collapse by characters remaining is a statement about forecasts that stay within the current line. In the marginal view, the position of every later newline depends on the width because each subsequent line runs for about w characters. At short horizons, it agrees with the next-newline embedding. At long horizons, the embedding increasingly includes information about future lines. This means that cells with equal r but different w pull apart and a width structure emerges in the embedding. This regime seems to be the regime most closely matched by the geometry of the (c, w) manifolds in both our toy models and in Gemma. Figure 25 shows Gemma-2-9B activation centroids at L15, aligned to fit with a marginal forecast embedding at a horizon of 32; as well as the centroids of a small toy model– harvested after the penultimate attention block.

Here, we note that, because r is a linear function of w and c– we could just as well write all of our registers in terms of r as in terms of w. This changes nothing about the embedding geometry, but would have some effect on the distances between register values. We do not follow up on this thread here, but it is worth noting that $(c, L - c, U - c)$ and (c, r) make equally valid registers as (c, L, U) and (c, w). Another side effect of this, is that many of our interpretations of "width" embeddings can often be equally considered c or r embeddings. That the conclusion above: "cells with equal r but different widths pull apart and a width structure emerges in the embedding". Since this "width structure" happens at fixed r, they can just as easily be considered as character count structures. More careful analysis is required to try and separate these two things, we touch on this again in the next section.

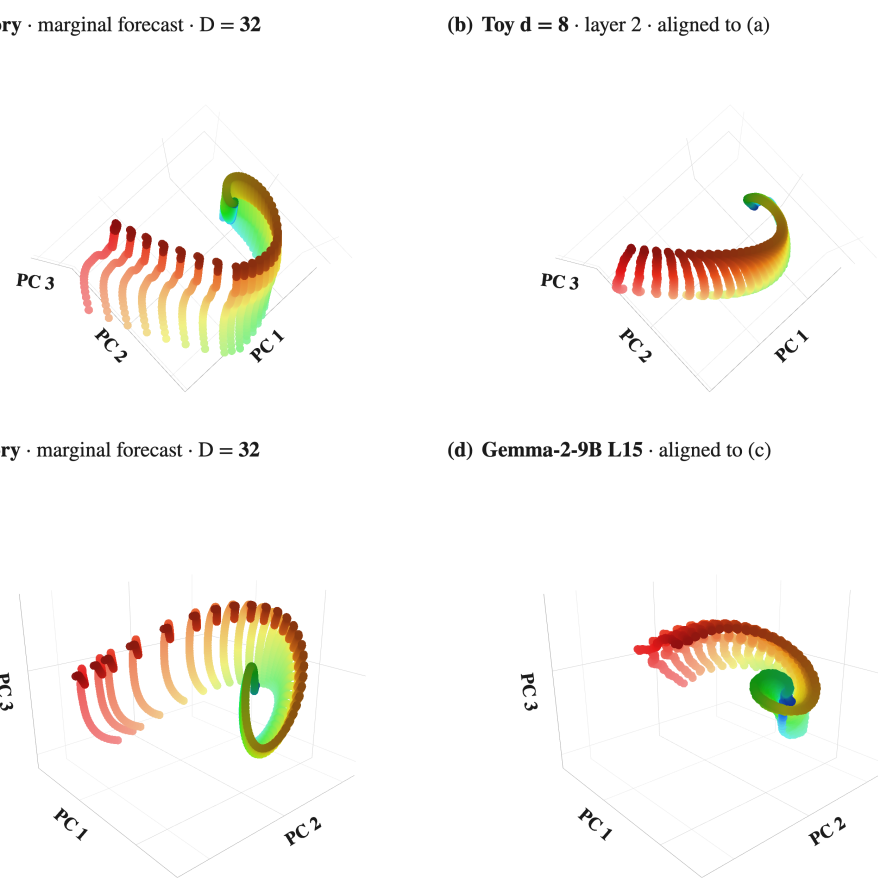

Figure 25: (a) The marginal forecast embedding of the theoretical (c, w) cells of the toy models task (token lengths 2–8 and widths 20–150) at $D = 32$, in its own top-3 PCs. (b) the toy models layer-3 (c, w) centroids, rigid-Procrustes-aligned onto the marginal forecast. (c) The marginal forecast embedding of the theoretical (c, w) cells of Gemma-2-9B's empirical task (token lengths 1–15 with the empirical length distribution, and widths 20–150) at $D = 32$, in its own top-3 PCs. (d) Gemma-2-9B layer-15 (c, w) centroids, rigid-Procrustes-aligned onto (c). The left and right panels share one scale.

The first few PC's of the trained model's activation centroids allocate distance as our marginal forecast embedding predicts. Neighboring positions nearly coincide far from the newline, and spread apart as room runs out. We can fit these manifolds using procrustes alignment, but ultimately this comparison should be treated as qualitative. Different values of D differ mainly in how the stretch is distributed; our evidence supports the pattern rather than identifying a particular horizon for either our toy models or Gemma-2-9B. As a closing note, we point out that the neural network centroids align to *each other* more readily than they do to their respective forecast embeddings— this suggests a different or additional embedding (though largely shared between the toys and Gemma-2-9B) is responsible for the exact arrangement we see, and our marginal forecast is merely an approximation that captures the most prominent features.

E.4 Tracing and steering c, r, and w

In the previous section we described the joint width–count sheet as a geometric object: a two-dimensional surface in the residual stream, parameterized by the current character count c and the line width w.

In this section we ask how this object relates to the model's *computation*. Where, in the network, are the variables of the linebreaking task represented, and where are they actually used? These two questions can have different answers: a variable can be linearly decodable at a layer without the model reading it there. Furthermore, prior steering work Gurnee et al. (2025) intervened using count centroids averaged over documents of all line widths, which left open whether the model's operational count state is a width-independent scalar or a genuinely joint (w, c) coordinate on the sheet. We address these questions in two steps. First, we train linear probes for each variable at every layer, tracking how well each variable can be linearly decoded as a function of the layers of the model. We also do variance decomposition of the activations in terms of these variables. Second, we perform targeted steering experiments: we overwrite the model's count state with a chosen point on the (c, w) sheet, using either the width-conditioned $\mu(c \mid w)$ or the width-agnostic $\bar{\mu}(c)$ mean activation, and read out the change in the model's newline prediction. We report results for Gemma-2-9B.

E.4.1 Probes

Where can each variable can be linearly read out? We wrap English prose at line widths $w = 20$ to 150 and collect residual-stream activations at every content token, skipping the first three lines of each document so the width has been established. Each token is labeled with its character count c and with the width w of its document. For each variable (c, w, $w - c$, and a control variable $w + c$) we fit a linear probe on the activations at every layer and evaluate it on held-out documents. In Figure 26 we report held-out R^2 as a function of layer.

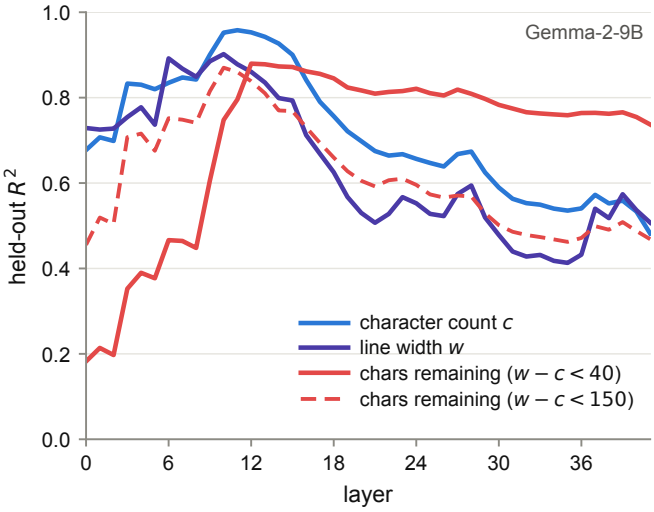

Figure 26: Held-out probe R^2 by layer in Gemma-2-9B. Linear probes for the character count c, the line width w, and the characters remaining $w - c$ (fit on positions with $r < 40$, and on all positions, $r < 150$), trained on residual-stream activations at content tokens and evaluated on held-out documents. The fact that we find a stronger linear encoding of r towards the end of the line is consistent with the theoretical predictions in Section E.3 that the model should separate states by r increasingly as the end of the line is approached.

Any linear function of c and w is decodable wherever both variables are present, so a successful chars-remaining probe cannot by itself be distinguished from a linear combination of a count and a width representation. We therefore perform variance decomposition. For each candidate variable we group the token activations by the variable's value and compute the fraction of total activation variance that lies between the group means. A high score means activations sharing a value of the variable sit close together, whether or not the variable is encoded linearly. Because the characters remaining is a function of c and w when $c \leq w$, a raw score can simply be correlated with the character count.

We calibrate each curve against a full permutation of the labels, which gives the score expected by chance at these group sizes. We plot the 99th percentile of this null, and credit a variable with structure of its own only where the observed curve exceeds it. The variable $w + c$, which the model has no computational reason to track, serves as a check on the method. Its bump around layer 10 indicates that the line width bump at the same location may be due to correlations with c and r rather than the line width in its own right. We leave this investigation for future work.

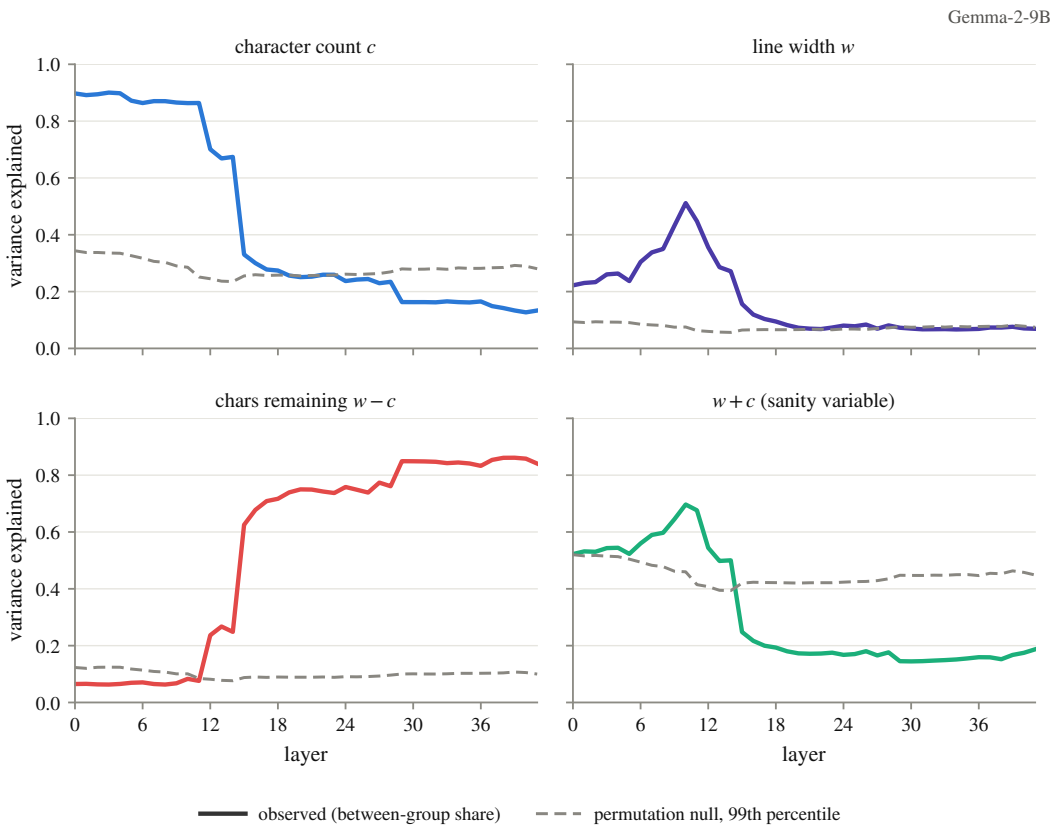

Figure 27: Fraction of residual-stream variance that lies between group means, by layer, when Gemma-2-9B's activations are grouped by the character count c, the line width w, the characters remaining $w - c$, and the sanity variable $w + c$. Dashed: 99th percentile of a full label permutation.

The decomposition in Figure 27 gives a clear picture of the computation across layers of the model. In the first third of the network the residual stream is organized by absolute count: grouping by c explains about 90% of activation variance from layer 0 through layer 11, far above the null. Here we see a quantitative signature of the geometric reorganization from "register" to "forecast": between layers 12 and 15 the organization changes to be organized primarily by characters remaining. The count score collapses toward its permutation null, while chars remaining $w - c$ jumps from near its null to about 0.7 and continues to about 0.85 by layer 30. After the transition, only r does better than its null baseline: the late-network representations are primarily organized by the distance to the boundary specifically, not by count and width separately. The early layers represent how far into the line a token is, the late layers by characters remaining, which is the quantity actually relevant for predicting newlines.

Together, the probe and variance analyses suggest that the residual stream becomes increasingly organized around characters remaining in the middle of the network. The same pattern observed in the toy models, and the forecast embedding prediction. These tests do not, however, establish that the model uses this geometry to predict a newline. We therefore next intervene directly on the corresponding representations.

E.4.2 Steering

We intervene on the width-conditioned character count sheet to test its causal effect on predicting newline tokens. We take English documents wrapped at width 90 and read the model's newline probability $P(\text{newline})$ at content tokens in the middle of lines, where the correct prediction is to continue the line.

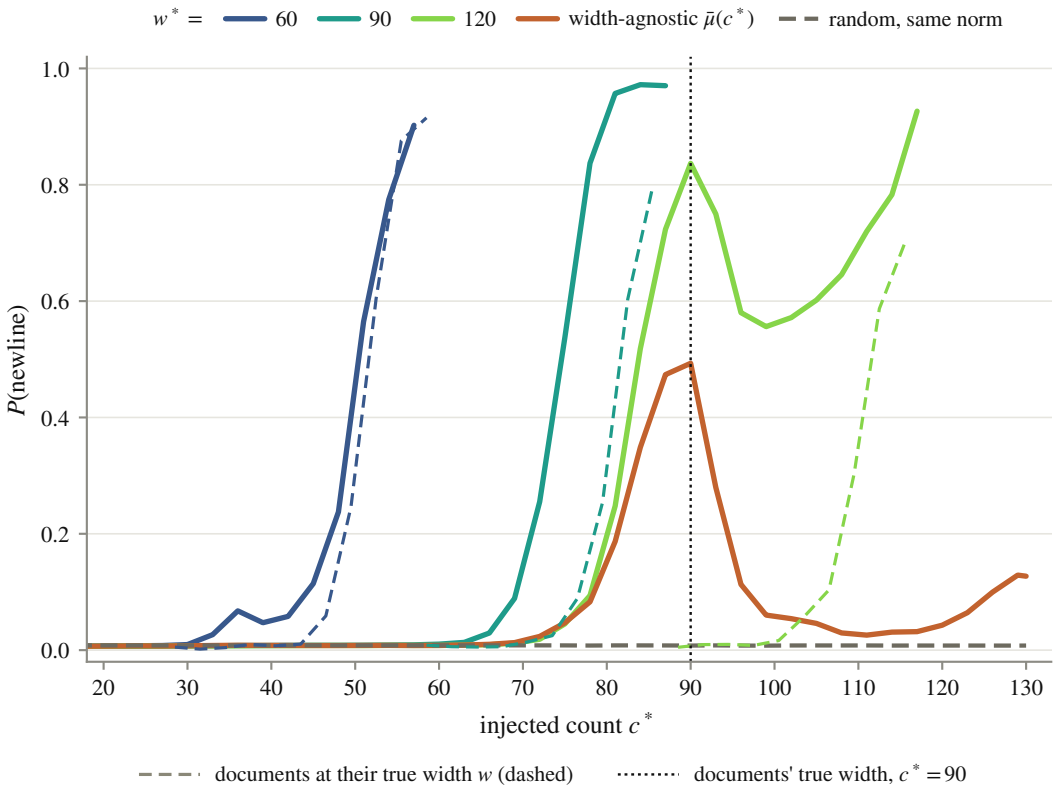

Figure 28: Width-agnostic count steering fires only at the document's true width. Here, we intervene on layers 10–14 with the additive intervention, and plot against the injected count c^*. Injecting the width-conditioned mean $\mu(c^* \mid w^*)$ drives a newline at each injected width's own boundary($w^* = 60, 90, 120$; solid). Injecting the width-agnostic centroid $\bar{\mu}(c^*)$, the mean activation over tokens with count c^* in documents of every width, fires only where the injected count reaches the documents' true width; past that, the probability returns to baseline. The data averages over 63 different contexts whose most likely next token is exactly 5 characters. The gray dashed horizontal line is from a random direction with the norm of the width-agnostic patch.

At these tokens, we shift the (c, w) component of the residual stream vector h from the model's own state to a chosen counterfactual point:

$$h' = h + UU^\top \big[\mu(c^* \mid w^*) - \mu(c \mid w)\big],$$

where U spans a rank-8 subspace fit on the full set of width-conditioned character count centroids, $\mu(c \mid w)$ is the centroid for the token's actual c and w, and $\mu(c^* \mid w^*)$ is the centroid for the injected count c^* on a line of injected width w^*. The same displacement is added to the read token and to the two content tokens before it, so the local count trajectory stays consistent.

The injected state is meant to tell the model it has seen c^* characters on a line of width w^*; everything else in the document, including the width evidence at the newline tokens, remains visible to the model. Importantly, only the content tokens are patched: the newline tokens are left untouched.

We first compare these injections to ones that use width-agnostic centroids $\bar{\mu}(c^*)$, the intervention from Gurnee et al. (2025). We sweep the injected count c^* for different counterfactual widths w^* and report their impact on the probability of newline prediction. Figure 28 shows that the width-agnostic fires only at the document's true width while injecting the width-conditioned mean $\mu(c^* \mid w^*)$ drives a newline at each injected width's own boundary. For example, injecting "five characters left on a width-60 line" makes the model behave similarly to a model that is five characters from the end of a width-60 line, even though every line it can actually see is 90 characters wide.

However, this is not the whole story. This may be because the (c^*, w^*) injections change what the model believes about its proximity to the end of the line, but leaves the width evidence at previous newlines untouched. We see evidence of these conflicting instructions in Figure 28: when $w^* = 120$, $P(\text{newline})$ also increases for injected counts near the document's *original boundary*. This suggests the model is still receiving previous newline information as evidence, and using that in service of prediction.

To test this, we try a simple intervention that stops the queried token from reading previous newline tokens directly. The left panel of Figure 29 shows the same width conditioned intervention as Figure 28, but scaled against $r^* = w^* - c^*$. The right hand panel, shows the same data and intervention but also masks the queried token's ability to attend to any of previous newline tokens. The peak at the original boundary drops significantly under this mask. However, some portion of the bump persists and the newline probabilities for all interventions fall. This suggests that the information is spread out over many token positions in a way that allows the model to partially circumvent our simple masking intervention. We leave further investigation of this for future work.

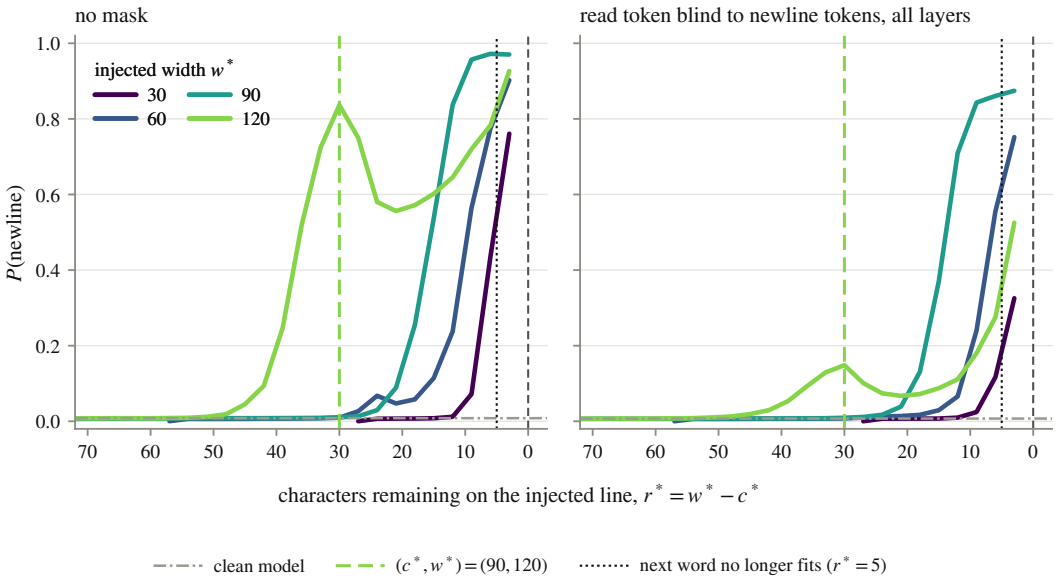

Figure 29: Left: $P(\text{newline})$ against the characters remaining on the injected line, $r^* = w^* - c^*$. Right: the same injections, with the read token's attention to every earlier newline token removed at all layers; all other tokens are processed normally.

We now explicitly test whether the activations at previous newline tokens carry the width evidence that the model uses when it predicts a newline. We therefore intervene on those tokens directly. Let $g(w)$ denote a width centroid: the mean residual stream vector at the newline tokens of documents wrapped at width w. These centroids span a low-dimensional manifold, and we take V to be the rank-4 subspace fit on them. In a document wrapped at width 90, we shift every newline token's width component to a counterfactual width w^*:

$$h' = h + VV^\top \big[g(w^*) - g(90)\big],$$

either at a single layer or at each of layers 10–14. The content tokens are left untouched. We then read $P(\text{newline})$ after every word and plot it against the character count c. A narrow injection ($w^* < 90$) should make it break early, at $c \approx w^*$ and a wide injection ($w^* > 90$) should suppress the break at the true boundary, $c \approx 90$. Figure 30 shows both behaviors, though only when the intervention is done across multiple layers.

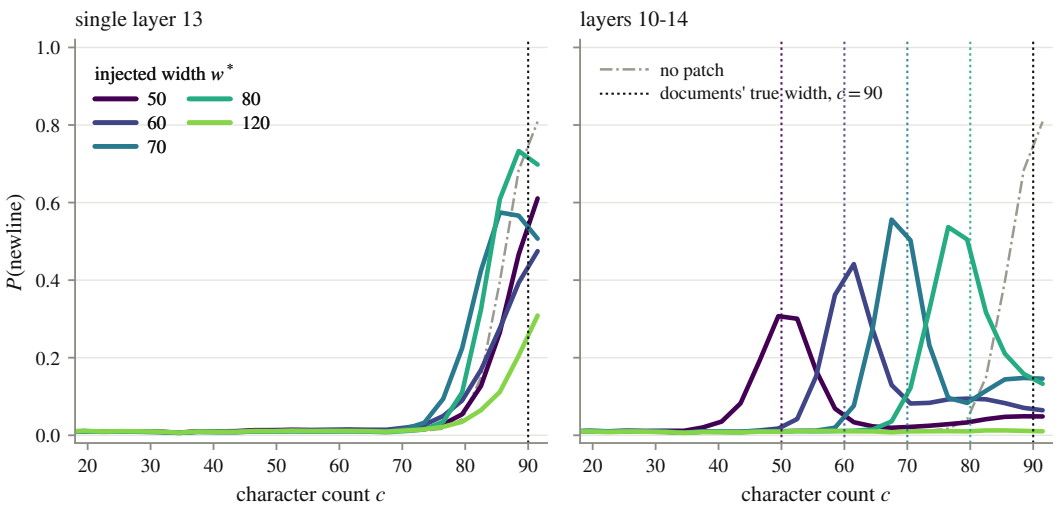

Figure 30: Newline-token width injections in 256 width-90 documents, $w^* = 50, 60, 70, 80, 120$. Left: patched at layer 13 only. Right: patched at layers 10–14 together. Dotted colored lines mark $c = w^*$ for each narrow injection.

F Technical details of the predictive-generator analysis and toy models of the count × width manifold

F.1 Predictive generators

Think of an exact finite-dimensional predictive generator of a process (here a process is simply a distribution over token sequences, $P(x_{0:t})$) as being composed of three parts:

1. A map from histories of tokens to *predictive vectors*. Given the history $h_t = x_{0:t}$, there exists a map Φ
$$\eta_t = \Phi(h_t) \in \mathbb{R}^d,$$
where d stays fixed as t increases.
2. This vector contains enough information to give the same next-token probabilities as the full history through a function E:
$$\Pr(X_{t+1} = x \mid X_{0:t} = h_t) = E(\eta_t, x).$$
3. The predictive vector can be updated iteratively through a function f, using only its current value and an observed token:
$$\eta_{t+1} = f(\eta_t, x_{t+1}) = \Phi(h_t\, x_{t+1}).$$

When the process is stationary, the same maps and rules must work at every time and for every history that can occur. Generation therefore consists of repeatedly sampling a token and updating the vector:

$$X_{t+1} \sim E(\eta_t, \cdot), \qquad \eta_{t+1} = f(\eta_t, X_{t+1}).$$

Starting from the vector for a given history, this generates continuations according to the process probabilities. These conditions make η_t a sufficient statistic of the history for the purposes of prediction. It preserves all information in the observed past that matters for predicting the future. Under next-token cross entropy loss, the optimal probabilistic prediction is the true conditional distribution of the next token given the current context. Since the function E reproduces this distribution, predicting from η_t achieves the optimal loss. In this way, our predictive generator can be thought of as a mechanism for achieving optimal prediction.

The same process can have different presentations: different predictive models that reproduce its sequence probabilities. We can define classes of presentations by restricting the state space and the allowed forms of E and f.

A familiar example is a finite-state HMM, which has η_t restricted to be a probability distribution. For an HMM, E and f are defined by the same set of nonnegative linear maps, which sum to a row-stochastic matrix. Using row vectors, let $T^{(x)}_{ij}$ be the probability of emitting x and moving to state j, given current state i. Then

$$E(\eta, x) = \eta T^{(x)} \mathbf{1}, \qquad f(\eta, x) = \frac{\eta T^{(x)}}{\eta T^{(x)} \mathbf{1}},$$

where $\mathbf{1}$ is the column vector of ones.

Not all processes possess a compact HMM presentation. In order to arrive at a compact set of predictive vectors, we must extend what operations E and f have access to. Another useful family uses a fixed number of numerical registers as its predictive state, rather than probability distributions. For our purposes, a register presentation has a finite collection of affine comparisons that divides the state space into regions. Within each region R, each token has a fixed emission probability and an affine update. Using row-vector notation,

$$E(\eta, x) = p_{x,R}, \qquad f(\eta, x) = \eta A_{x,R} + b_{x,R},$$

with

$$p_{x,R} \geq 0, \qquad \sum_x p_{x,R} = 1.$$

The update f is piecewise affine, and the readout E is piecewise constant. Unlike an HMM, they are specified separately. When each token uses the same affine update across all regions, the update is globally affine. This register-based viewpoint has its roots in the counter and register machines of Lambek (1961) and Minsky (1961). This framework extends the updates to include piecewise-affine maps; the resulting construction is an iterated function system with place-dependent probabilities (Barnsley et al., 1988).

F.2 Fully specifying the stochastic counter Machine

If the width w is known, the only thing about the past that matters for future predictions is the current line position c, the characters since the last newline. c is not hidden, it is a count over the observed stream. Assume token k has length c_k and is drawn with a position independent probability p_k; the whole next-token distribution is that draw put through a threshold, token k fits iff $c + c_k \leq w$:

$$P(\text{token } k \mid c) = p_k \, \mathbf{1}[\, c + c_k \leq w \,], \qquad P(\text{newline} \mid c) = \sum_k p_k \, \mathbf{1}[\, c + c_k > w \,]$$

Tokens that fit in the line keep their p_k selection probability and any tokens that would overflow have their probability mass collect on the newline. The overflowing token is emitted on the next line, so its characters are preserved.

Between the newline N and that token, the observer knows a newline happened at c but not which token caused it (any token with $c_k > w - c$ could be responsible). Thus, right after a newline the next-token question becomes "*which token overflowed?*". This means we are really just looking at the distribution that generated $P(\text{newline} \mid c)$, renormalized on the restricted domain of tokens that would have overflowed:

$$P(\text{token } k \mid c, \text{ N}) = \frac{p_k \, \mathbf{1}[\, c + c_k > w \,]}{P(\text{newline} \mid c)} \, .$$

Here, the conditioning on c, N indicates this is the probability distribution after having just observed a newline token from position c. This distribution only matters at this context position, until the next token (the deferred token that caused the newline token) opens the new line with c_k characters.

While we just described this generator in terms of a single count register c, the automaton that implements this process actually requires two character-counting registers. Let a count the characters on the current line, and let b temporarily hold the position the last newline was emitted from. The position that we describe above is actually $c = a + b$. These registers update from observing content tokens or newlines in the following way

- a content token k increments a: $a \leftarrow a + c_k$, and erases b: $b \leftarrow 0$
- a newline token swaps the registers: $(a, b) \leftarrow (b, a)$

These dynamics mean the (a, b) state is always $(c,\ 0)$ except for right after a newline, where it is $(0, c)$. The extra register slot is simply a way to keep track of the fact that we have a different probability distribution of tokens right after the newline. At a known width, then, the sufficient statistic is this register pair. The state updates are always affine, but the probability readout functions are nonlinear, in the inequality that compares $c + c_k$ against w and the one that compares a and b to change the next token prediction right after a newline.

Our task includes documents of various fixed widths, so we can think of the real data as drawing w from a prior τ over candidate widths W at the start of each sequence. The observer must infer this w from the token stream, since it is not directly observable the way that c is. A Bayesian observer should refine their estimate of w through context, starting from some prior on the task. Let's assume that the observer is Bayes optimal and knows that we are selecting from a fixed set of widths. At optimality, the observers prior would be τ itself. Our data draws w uniformly from a range of candidates, and the optimal posterior over widths is simply this prior restricted to an interval of plausible widths with lower bound L and upper bound U:

$$P(w) \ \propto \ \tau(w)\, \mathbf{1}[\, L \le w \le U\,]$$

Every token that moves us to position c proves $w \ge c$, which raises L; the beginning of each line shows us a token k that overflowed, proving $w < c + c_k$ and lowering U.

All said, the three formulas above need only a small change to accommodate inference over widths. Each of the indicator functions must be replaced with its average over the viable widths. If we define $M(l, u) = \sum_{w \in W,\ l \le w \le u} \tau(w)$ as the probability mass the prior assigns to the interval $[l, u]$, we simply implement the following transformations:

$$\mathbf{1}[\, c + c_k \le w\,] \ \longrightarrow \ \frac{M(\max(L,\ c + c_k),\ U)}{M(L, U)}\,,$$
$$\mathbf{1}[\, c + c_k > w\,] \ \longrightarrow \ \frac{M(L,\ \min(U,\ c + c_k - 1))}{M(L, U)}\,.$$

Substituting the above into our probability expressions yields

$$P(\text{token } k) = p_k\, \frac{M(\max(L,\ c + c_k),\ U)}{M(L, U)},$$
$$P(\text{newline}) = \sum_k p_k\, \frac{M(L,\ \min(U,\ c + c_k - 1))}{M(L, U)}\ , \text{ and}$$
$$P(\text{token } k \mid \text{N}) = \frac{p_k\, M(L,\ \min(U,\ c + c_k - 1))}{M(L, U)\ P(\text{newline})}\,.$$

The (a, b) register updates carry over completely unchanged, but we also need to include rules for updating the U, L registers. These updates are as follows:

- a normal content token can raise the width floor: $L \leftarrow \max(L,\ a + c_k)$.
- the first token after a newline can lower the width ceiling: $U \leftarrow \min(U,\ b + c_k - 1)$.

The entire probability distribution over the future can be represented with four registers $(a,\ b,\ L,\ U)$ and the prior over widths. While the position pair (a, b) still update with affine maps, L and U update through $\max$ and $\min$.

We have glossed over one point that is worth looking into quickly here. As long as the most recent token emitted is a content token, the posterior stays uniform over the viable widths. However, a newline token is itself evidence that *some* token overflowed at c, and that evidence is width-dependent:

$$P(w \mid h, \text{N}) \;\propto\; \tau(w)\,\mathbf{1}[L \le w \le U] \sum_k p_k\,\mathbf{1}[c + c_k > w] \tag{4}$$

The smaller w is, the more token lengths overflow so the post-newline posterior is not a flat interval. However, seeing which token was deferred brings the posterior back to a uniform interval again. We can effectively ignore this wrinkle, since the bulk of our geometry analysis neglects the activations above the newline regardless– but it is still worth mentioning.

Because we ignore the activations above the newline tokens, we can also ignore the b register– which is only nonzero right after seeing a newline. This means that, when looking at content tokens, we effectively only have three registers (a, L, U). When looking for content token geometry, we refer to this as (c, L, U) since the first register position is always equal to the character count on the current line.

F.3 Minimal linear dimension of the line-wrap process

The compact filter is four registers, but it is not linear. It is worth being precise about whether it *needs* nonlinearity. The position registers use affine transformations (addition and swapping, one operator per observed symbol, exactly as an HMM picks an operator per symbol). The interval endpoints L, U are the issue, which require a nonlinear comparison to update. What does it cost to represent the same process linearly — all latent state updates linear per symbol, with the readout linear as well.

The straightforward way to linearize a hard threshold is to simply enumerate it. If we add latent direction that effectively enumerate the register domain, we can create an HMM that generates the exact process for a single width w. With n distinct token lengths we need:

- $w + 1$ position states: $S_0 \dots S_w$. From S_i, a token that fits on the line ($i + c_k \le w$) moves $S_i \to S_{i+c_k}$ with certainty; if the token doesn't fit, a newline is emitted and we move to one of the buffer states B_i, defined below.
- n buffer states: $B_0 \dots B_{n-1}$ — one per distinct c_k, holding the deferred token so it can open the next line.

In total, this means an $w + 1 + n$ state HMM. As an example, let's write out the matrices for an HMM for data with $w = 3$ and $c_k \in 1, 2$. The HMM has $3+1+2 = 6$ states: $[S_0\ S_1\ S_2\ S_3\ B_0\ B_1]$, and a vocab of 3 tokens: $\{\text{tok0}\,(c=1),\ \text{tok1}\,(c=2),\ \text{newline}\}$. The token conditioned transition operators are:

$T_{[\text{tok0},\,c=1]} =$

from\to	S_0	S_1	S_2	S_3	B_0	B_1	
S_0	.	.5	.	.	.	.	
S_1	.	.	.5	.	.	.	
S_2	.	.	.	.5	.	.	
S_3	.	.	.	.	.	.	
B_0	.	1	.	.	.	.	$\leftarrow B_0$ emits tok0 $\to S_1$
B_1	.	.	.	.	.	.	

$T_{[\text{tok1},\,c=2]} =$

from\to	S_0	S_1	S_2	S_3	B_0	B_1	
S_0	.	.	.5	.	.	.	
S_1	.	.	.	.5	.	.	
S_2	.	.	.	.	.	.	
S_3	.	.	.	.	.	.	
B_0	.	.	.	.	.	.	
B_1	.	.	1	.	.	.	$\leftarrow B_1$ emits tok1 $\to S_2$

$T_{[\text{newline}]} =$

from\to	S_0	S_1	S_2	S_3	B_0	B_1	
S_0	.	.	.	.	.	.	
S_1	.	.	.	.	.	.	
S_2	.	.	.	.	.	.5	$\leftarrow$ tok1 overflows: newline $\rightarrow B_1$
S_3	.	.	.	.	.5	.5	$\leftarrow$ both overflow: newline $\rightarrow B_0$, B_1
B_0	.	.	.	.	.	.	
B_1	.	.	.	.	.	.	

This is a fairly compact model, but if we have tokens that reach up to 15 characters, and a line width of 120 characters, the HMM is 136 states. Maybe that is just a clumsy construction? Maybe a more clever linear model collapses the counter into a few latents? This question can be asked by constructing the spectrum of the Hankel matrix (Carlyle & Paz, 1971; Fliess, 1974). The matrix is defined by indexing rows by histories h and columns by futures f. Then

$$H[h, f] = P(h, f) \,,$$

the joint probability the process emits h followed by f. Any linear model factors every entry through its d-dimensional latent state: $P(hf) = (\eta_\varnothing T_h)(T_f \mathbf{1})$. This implies a factorization $H = AB$ with inner dimension d. Thus, $\operatorname{rank} H \leq d$ for *any* linear model of the process, no matter how clever the entries. (The bound is tight: a linear model of dimension $\operatorname{rank} H$ always exists, so the rank *is* the minimal linear dimension.) We can use any generator to efficiently calculate joint probabilities, so we use the explicit HMM construction above to calculate the rank of H for various single width machines. The table below gives the calculated rank:

process	HMM states $w + 1 + n$	Hankel rank
`linewrap(3, (1,2))`	6	5
`linewrap(5, (1,2,3))`	9	8
`linewrap(10, (1,2,3))`	14	13
`linewrap(20, (1,2,3))`	24	23
`linewrap(12, (1,2,3,5,8))`	18	17

Table 1: Calculating the rank of the Hankel matrix validates our naive construction as nearly optimal already

In every case we computed, the rank is $w+n$. All we can save is a single state. In retrospect, this can be accomplished easily by removing the start state S_0. Unfortunately, we already have the minimal linear model!

The nonergodic mixture of differing widths makes the dimensions climb even higher. The widths cannot share counters (each thresholds at a different place), so the natural exact linear model is the direct sum over widths, whose dimension is the sum of the individual machines' dimensions. Even for a modest mixture with token lengths $2, \ldots, 8$ over widths $20, 21, \ldots, 150$ the exact mixture HMM carries $\sim$12,000 latent states. If a network's residual stream is a linear state machine, it cannot afford an exact representation and must be running some *compressed, approximate* linear model of the counter. Such an approximation could very well achieve near optimal predictive loss, but looking for these approximations is beyond the scope of the current investigation.

F.4 Proof that the forecast embedding is independent of content token probabilities

Watching only the newlines, one step at a time, feels like it throws away most of the richness of the predictions. However, we are throwing away less (newline placement relevant) information than it appears.

Consider the probability some starting (c, w) state assigns to a particular continuation $\rightarrow$ consisting of a string of tokens with newlines in it. Let's break down the continuation into two distinct components. We have the sequence of content token draws $\vec{x}$ that the non-newline token generator made, and a list of newline placement positions $\hookleftarrow$. These draws are made with the same distribution from every (c, w) state. The likelihood each state assigns to $\rightarrow$ factors into a (c, w) independent factor

(the probability of the content tokens neglecting the newlines) times a factor that depends on the newline placement.

$$P(\rightarrow|\ c, w)\ =\ P(\vec{x}) \times P(\hookleftarrow |c, w)$$

Here, $P(\hookleftarrow |c, w)$ is just an indicator function that is either 0 or 1, depending on whether the newline placement is consistent with (c, w) or not. The same general argument applies when the width is uncertain, but the form of the function is more complicated.

Thus, we see the difference between two (c, w) states' forecasts lies entirely in the difference between newline placement. The token content itself does not separate different (c, w) states. Thus, we should expect that (c, w) centroids are organized according to future distributions over newlines only, rather than the full token distribution.

While we can safely ignore the content tokens, and only look at newline probabilities– it is not true that looking only at "next-newline" and the "marginal newline" probabilities is the same as looking at the full joint distribution of newlines.

F.5 Toy model data and training

F.5.1 Data

Every toy model in this paper is trained on the synthetic version of the line-breaking task, with data generated by the process described in Section F.2. A document has a fixed width w that is never emitted. Its vocabulary is n content tokens, with token k representing for a word of c_k characters, plus a newline token. Widths are drawn uniformly from $\{20, 21, \ldots, 150\}$.

Two token-length distributions were used. The *uniform 2–8* process has 7 tokens of lengths $2, \ldots, 8$ with $p_k = 1/7$. The *empirical 1–15* process has 15 tokens of lengths $1, \ldots, 15$ with p_k set to the token-length distribution of Gemma-2-9B's tokenizer on 5.4×10^8 tokens from the Gutenberg corpus, and then clipped to 15 characters.

$p_{1:15} = (.136,\ .061,\ .174,\ .215,\ .155,\ .090,\ .061,\ .046,\ .027,\ .018,\ .010,\ .0044,\ .0021,\ .0009,\ .0002)$.

This second process is the one whose ground truth is compared against Gemma in Figures 23 and 25.

Every document begins with a beginning-of-sequence token followed by 511 emitted symbols, filling the models' 512-token context; there is no end-of-sequence token.

Five widths, $w \in \{45, 70, 95, 120, 145\}$, were withheld from training so that the held-out evaluation loss measures interpolation to line widths the model has never seen. All geometry analyses in this paper (the register charts, the (c, L, U) and (c, w) centroids, and the forecast embeddings) are computed over all 131 widths, so about 4% of the analyzed documents have a width the toy models never trained on.

F.5.2 Training

All toys are 4-layer decoder-only transformers (TransformerLens `HookedTransformer`) with rotary position embeddings, RMSNorm, gated GELU MLPs of width $4\,d_{\text{model}}$, and a 512-token context. Table 2 lists the four models. They were trained with AdamW ($\beta = (0.9, 0.999)$, no weight decay) at a constant learning rate of 10^{-3} on next-token cross-entropy over every position, for 50,000 steps. Each step used 256 freshly sampled documents with no document ever repeated. Evaluation used fresh documents from a fixed seed, separately over the trained and the held-out widths. Runs were logged to MLflow with regular checkpoints, and the checkpoints and process definitions used in this paper are exported from those runs.

toy	process	d_{model}	heads (d_{head})	d_{mlp}	analyzed step
$d = 8$	uniform 2–8	8	2 (4)	32	45,000
$d = 32$	uniform 2–8	32	4 (8)	128	45,000
$d = 64$	uniform 2–8	64	8 (8)	256	45,000
$d = 32$	empirical 1–15	32	4 (8)	128	50,000

Table 2: The four toy models. All share the 4-layer architecture; only the token-length distribution and the size differ.

F.5.3 CENTROID METHODS

For the geometry analyses, each toy is run on 65,536 fresh documents (eight seeds of 8,192 documents, split into two independent halves to verify the sampling noise). We record the residual stream after transformer block ℓ(blocks.l.hook_resid_post); layers are zero-indexed, so "layer 1" is the stream after the second block, the same convention used for Gemma. Because the process is known exactly, every position can be associated with ground truth register values: the observable count c, the line index, the document's true width w, and the observer's bounds L and U from Section F.2.

Centroids are weighted means over positions sharing a cell, either the belief cell (c, L, U) or the pair (c, w). Unless stated otherwise we skip the first three lines of every document and positions with $c \leq 10$ are excluded. These means waiting until the the width bounds have tightened. Cells with fewer than 100 positions are dropped. The PCA basis of a layer is the weighted PCA of its (c, L, U) centroids, of which the top six components enter every fit. A rigid fit is a weighted similarity Procrustes map (rotation, reflection, uniform scale, translation) from those six coordinates onto the target; a linear fit is a count-weighted affine regression; both report the count-weighted R^2 over the target's coordinates. For (c, L, U) centroids the target is the register chart itself; for (c, w) centroids it is the chart of the cell's expected registers $(c, \mathbb{E}[L], \mathbb{E}[U])$ under the process occupancy. Raw-activation fits use every recorded position; the figures only draw a subsample of 15,000 points.

Forecast embeddings (Figures 24 and 25) involve no model: they are computed exactly from the process. Each (c, w) cell's forecast is the weighted average of its member states' newline forecasts to horizon D, either the per-step marginal newline probabilities or the first-newline probabilities, and the embedding is the weighted PCA of those vectors.