Chasing the Counting Manifold in Open LLMs

Llama3.1-8B · layer 5 · main + SAE span

Gemma-2-9B · layer 11 · main + SAE span

Qwen3-8B · layer 5 · main only

Pythia-410m · layer 6 · main only

How open LLMs track line breaks and where that signal lives.

AUTHORS AFFILIATION PUBLISHED

In Gurnee et al. (2025), Anthropic showed that language models can represent a counting variable tied to line formatting. In their setup, the model effectively tracks distance since the previous line break, which helps maintain consistent wrapped text generation.

This project asks whether the same behavior appears in open-weight models and whether we can recover the underlying representation with fully open tooling. We focus on model families with different training data and scales to test how robust this phenomenon is outside the original closed-model setting. Code for this project is available at github.com/corl-team/counting_manifolds.

What We Reproduced

We reproduced the wrapped-text evaluation pipeline by converting long documents into width-limited lines and measuring newline prediction directly. We then ran layer-wise linear probes on hidden states to identify where character-offset information is most recoverable, and complemented that analysis with PCA-based geometry and manifold visualizations of the position representation. Finally, we examined SAEs to isolate individual features that track position.

What Is New in This Reproduction

Our main extension is breadth and comparability across model families including GPT-2, Pythia, Llama, Gemma, and Qwen, rather than focusing on a single lineage. This allows us to separate architecture and data effects from simple scale effects, and shows that model size alone does not determine newline competence. We also provide an SAE-focused analysis workflow that captures non-monotonic position features and explicitly compares those features against AutoInterp descriptions to reveal interpretation failure modes.

TL;DR

- Several modern open-weight families predict wrapped newlines reliably, while GPT-2 remains weak even at larger size (including GPT-2-XL), and Pythia improves sharply by the Pythia-410m scale.
- Position information is clearly present in hidden states but usually peaks in mid layers, not consistently in the earliest layers.

- SAE features recover fine-grained position structure, including hill-shaped tuning curves that encode position without simple linear trends.
- AutoInterp labels for SAE features often miss the actual mechanistic role of these features in counting.

We now follow this pipeline step by step: first the data setup and labeling scheme, then a tokenizer-agnostic newline definition, then behavioral comparisons across models, and finally hidden-state and SAE-level analysis.

Setup

We adopted the line-wrapping setup from the original paper. Specifically, we filtered documents from the `HuggingFaceFW/fineweb` dataset to include only those that (i) contained no newline characters in the raw text (i.e., each document was a single long line) and (ii) yielded at least 10 lines after wrapping. We sampled 1,000 such documents. We then wrapped each document into 150-character lines using `textwrap` and assigned each token a position index equal to the number of characters since the most recently inserted newline (`\n`).

Text Sample With Tokens Labeled by Characters Since the Previous \n (Hover to See the Count)

Our Angel, Sarah Abigail Neal, age 16, was called Home to be with the Lord and her beloved and waiting heavenly family on February 9, 2013. Abby was
born October 2, 1996, in Atlanta, Georgia. She lived her early life for eleven years in Acworth, Georgia, after which she moved to Big Canoe, Georgia
where she resided until the day God called her Home. Abby was a borrowed gift from God that touched everyone in a way that only an angel could do.
Although afflicted her entire life with severe mitochondrial disorder, she carried out His mission here on earth by reaching out to and touching the
hearts and souls of her family, friends and complete strangers with unconditional love and compassion. Abby is survived by her father Robert K. Neal
(and stepmother Tonya) of Jasper, Georgia, mother Debbie F. Neal of Cumming, Georgia, brother Seph Neal of Brunswick, Georgia, twin sisters Ava and
Ella Neal of Cumming, Georgia, stepbrothers Haiden and Zakaryah Bain of Jasper, Georgia, grandparents Bob and Linda Neal of Big Canoe, Georgia, and
Jerry and Judy Williams of White Oak, Georgia and her homecare nurse, Mrs. Gina Crowe of Ball Ground, Georgia. She is preceded in death by her
grandmother Gail H. Williams and great grandparents Mr. and Mrs. James W. Williams, Mr. and Mrs. George B. Harvey, Jr., Mr. and Mrs. Fred H. Martin,
Robert M. Neal, Jr. and Helen F. Neal. Arrangements are being handled by Cagle Funeral Home in Jasper, Georgia with visit ation from 6 until 8 pm
Thursday, February 14. Memorial and Celebration Services will be held at Trinity Church - 2685 Steve Tate Highway, Marble Hill, Georgia - at 11 am on
Friday, February 15. Pall Bearers include Bob Neal, Keith Neal, Kevin Neal, Seph Neal, Jerry Williams, Wade Williams and Walker Neal (Honorary).
Contributions in memory of Abby may be made to the Children's at Scottish Rite Hospital: Children's Foundation, Abby Neal Fund, 1687 Tullie Circle NE,
Atlanta, GA 30329.

Having established the dataset and labels, the next step is to define what counts as a newline prediction in a tokenizer-agnostic way.

How We Determine \n for Different Tokenizers

Comparing model families requires a tokenizer-agnostic newline definition, because tokenizers may encode line breaks as standalone tokens or merge them with spaces and other characters.

We use one rule everywhere: a token counts as a newline token if its decoded text contains at least one `\n`.

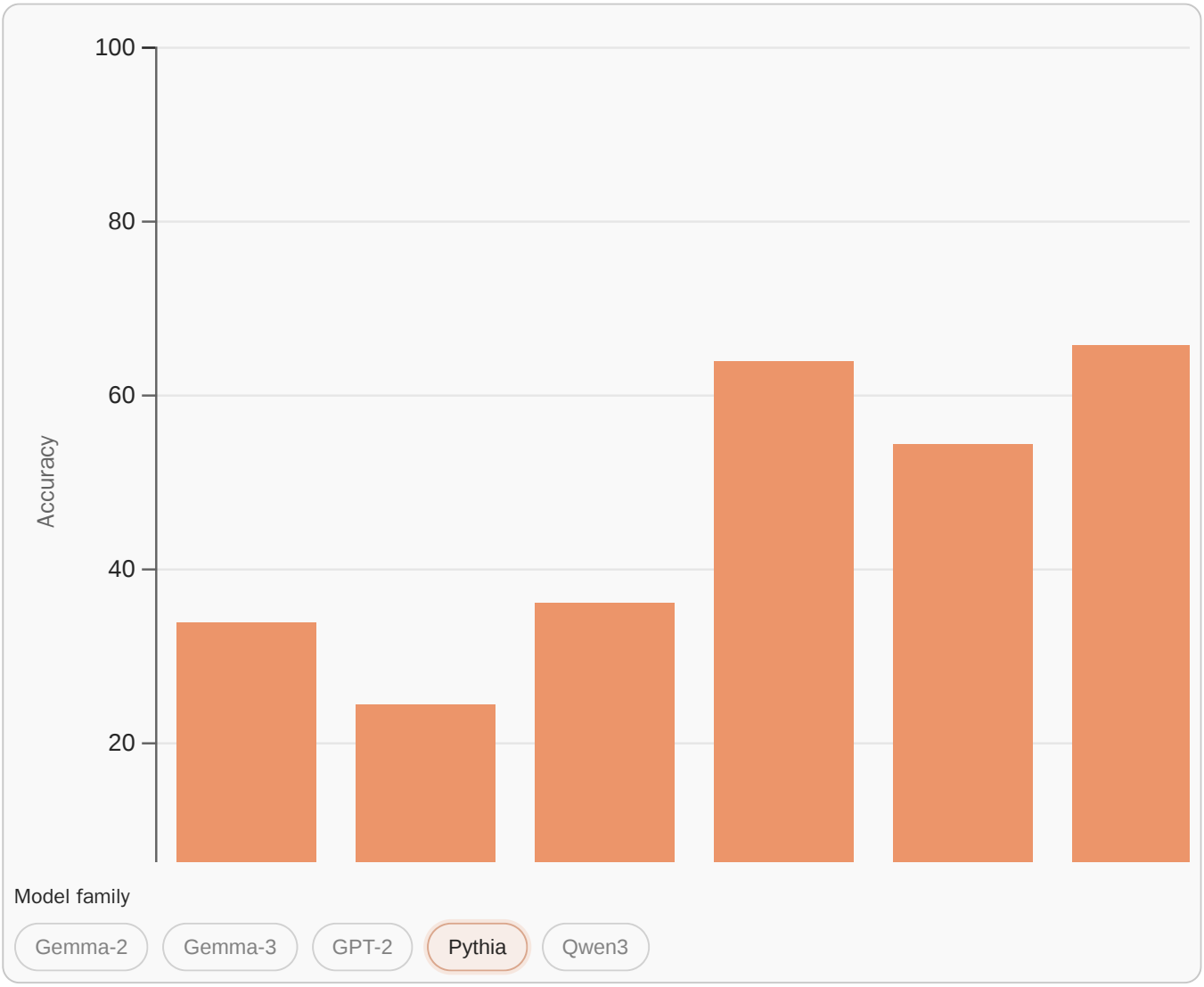
We apply this rule consistently in both metrics. For probability plots, we sum the next-token probability over all vocabulary items containing `\n`. For accuracy, a prediction is counted as correct when the predicted token contains `\n` at a wrapped position where a line break is expected.

This does not separate pure newline tokens from mixed tokens (newline plus extra text), but it directly matches the question we care about: does the model place a line break at the right location, independent of tokenizer boundaries.

Which Models Can Predict \n Correctly?

In our first experiment, we tested whether open-source language models could correctly predict newlines in wrapped text. We evaluated a range of model sizes from the GPT-2 and Pythia families, as well as the more recent Gemma-3 family. We report newline prediction accuracy using the tokenizer-agnostic newline rule described in the previous section.

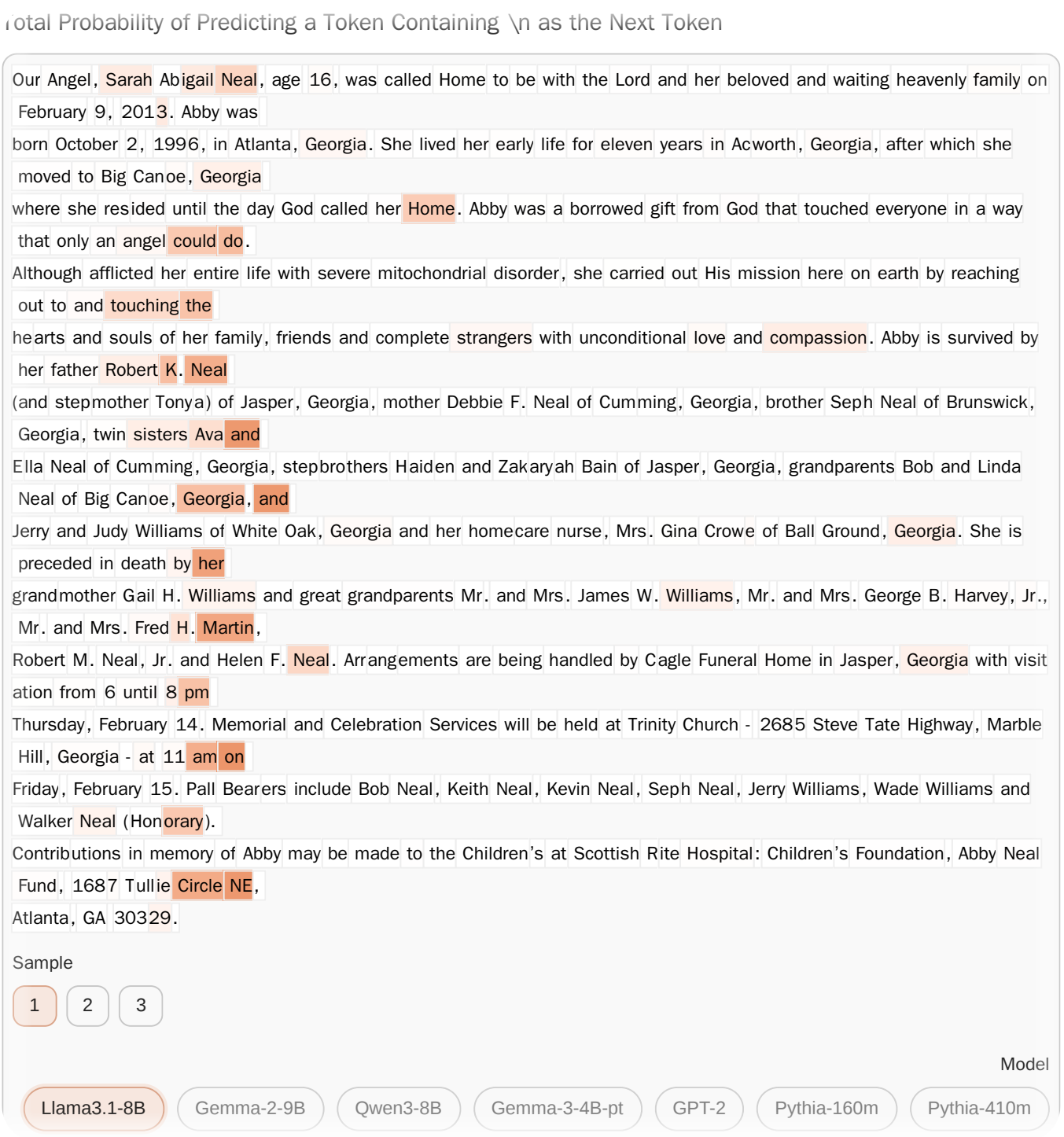
Newline Token Prediction Accuracy by Model Family



The family-level comparison shows a clear quantitative split. GPT-2 stays low on newline accuracy, ranging from 12.3% to 19.7% across Small through XL. Pythia shows a sharp transition between Pythia-160m and Pythia-410m (36.1% to 63.8%) and reaches 66.4% at 1.8B. Gemma is consistently strong (Gemma-3: 63.9%-75.1%, Gemma-2: 71.1%-78.1%), and Qwen3 is strong from 1.7B upward (60.7%-67.8%; 0.6B is 39.0%).

This pattern suggests that scale alone is not enough for learning line counts. A direct comparison is GPT-2-XL (19.7%) versus Pythia-410m (63.8%), a +44.2 point gap despite the smaller model. The effect appears to depend more on training data composition and formatting exposure.

Accuracy shows whether the top prediction is correct at wrap points. To see how strongly models lean toward newline even when it is not top-1, we next inspect total newline probability mass.



The probability-mass view supports the same conclusion at a finer level. Models that learned the wrapped-text pattern assign substantial next-token probability to newline-containing tokens at expected wrap points, while weaker models fail to align that mass as consistently.

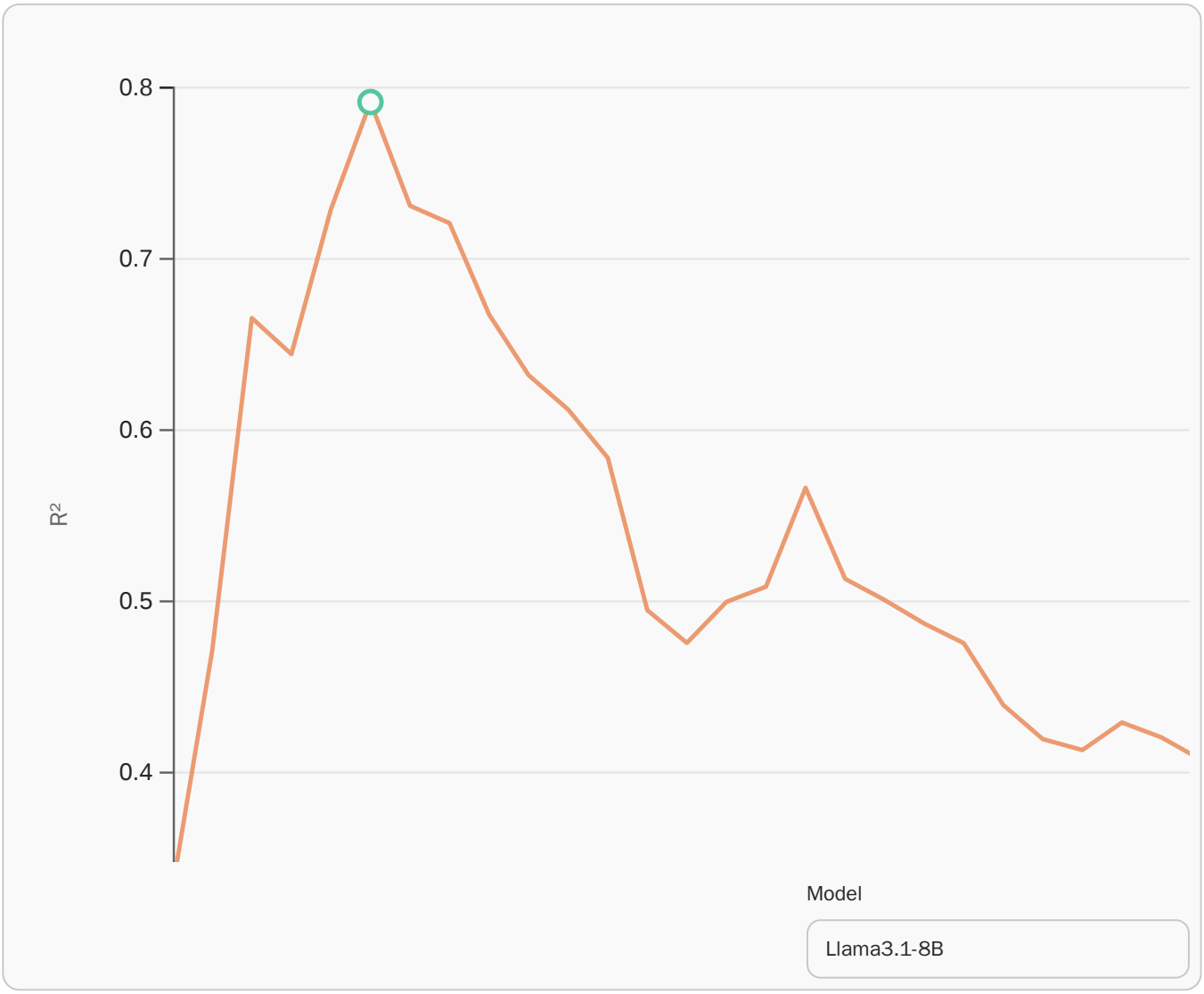
This is also reflected in aggregate probabilities at expected wrap positions: GPT-2 remains low (0.053-0.099), while stronger models are much higher (for example, Pythia-410m at 0.391, Gemma-2-9B at 0.575, and Qwen3-8B at 0.550). Taken together, the accuracy and probability evidence indicates a genuine representational gap in how strongly different model families internalize line-break structure.

Given this behavioral gap, the next question is representational: where in the hidden states is line-position information stored, and how compact is that signal?

Predicting Token Position

We next ask where line-position information is most accessible in the network. At each layer, we fit a linear probe that predicts character offset since the last \n from hidden states, and report R^2. This is correlational rather than causal, but it localizes the layers where the position signal is strongest.

R^2 of a Linear Probe Predicting Token Position From Hidden States



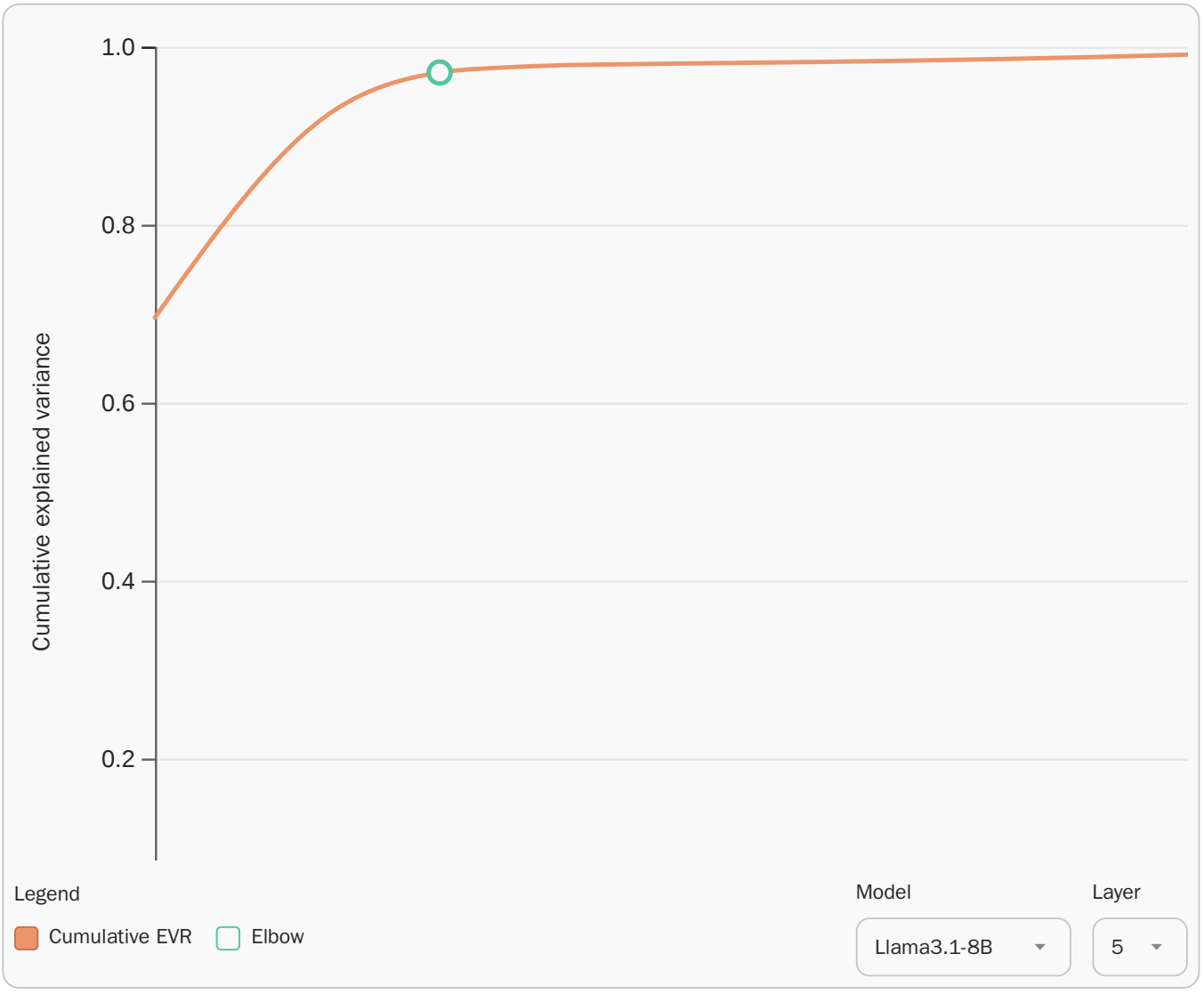
Starting with the default view, Llama3.1-8B reaches a peak probe R^2 of 0.791 at layer 5, indicating strong recoverability of token position in mid layers. More broadly, peak R^2 is 0.868 for Gemma-2-9B (layer 11), 0.852 for Qwen3-8B (layer 5), 0.744 for Pythia-160m (layer 3), and

0.724 for Pythia-410m (layer 6). GPT-2 is substantially weaker, peaking at 0.363 (layer 6), while Gemma-3-4B-pt reaches 0.491 (layer 18).

In contrast to Anthropic's original result, where peaks appeared in very early layers, the best layer is never in the first two layers for any tested model here. Across models, best layers are 3, 5, 5, 6, 6, 11, and 18 (median 6), so the position signal is usually early-to-mid (layers 3-6), with later-layer exceptions in Gemma-2-9B (11) and Gemma-3-4B-pt (18).

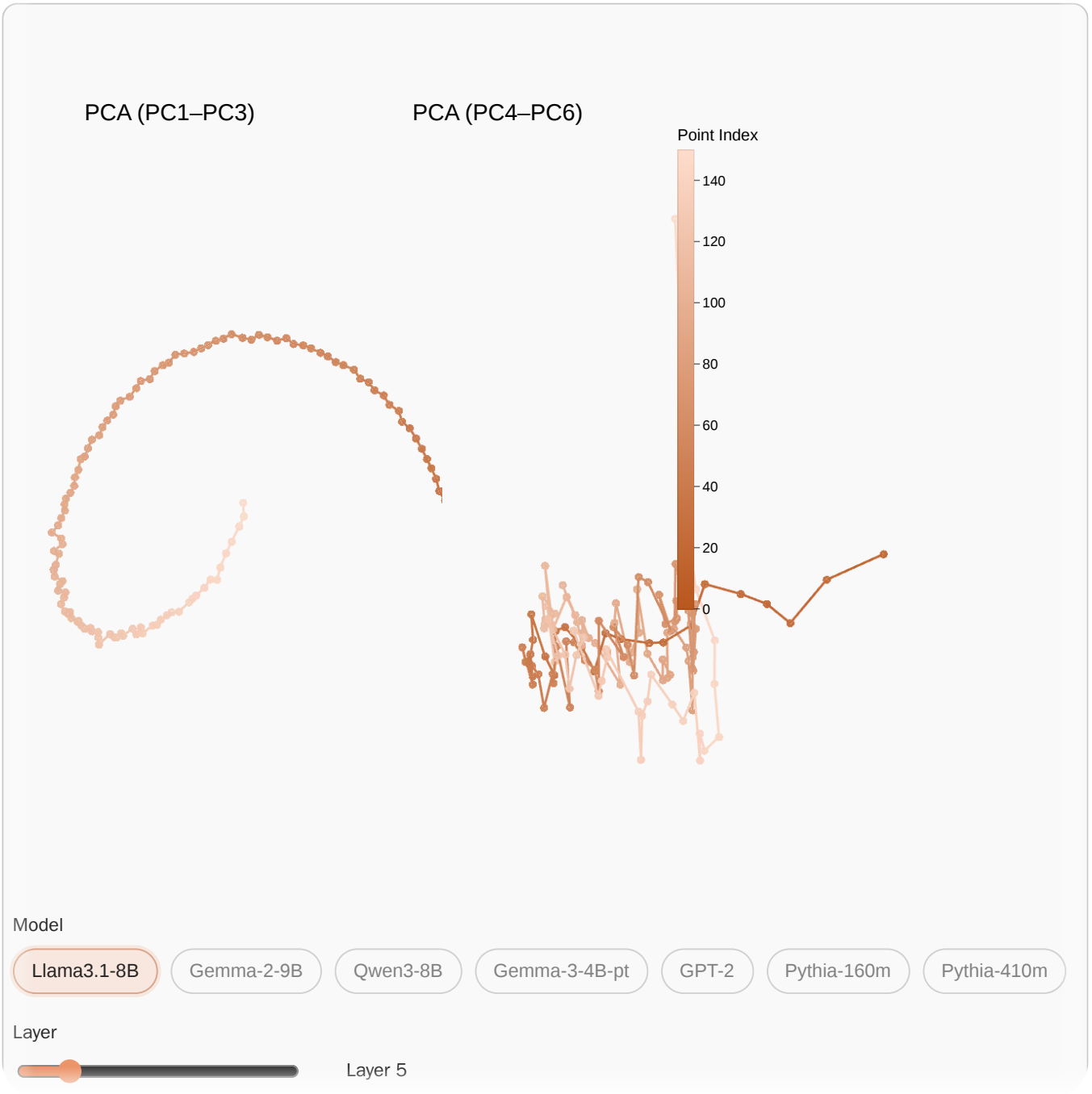
Next, for the best-performing layer, we checked whether this "counting signal" lives in a compact subspace. We averaged the hidden state for each position, ran PCA, and reported cumulative explained variance for the first n principal components. The variance curves suggest that top-3 components already capture most of the signal in most models, so we use a top-3 elbow by default. We keep two explicit exceptions: Gemma-2-9B uses top-4 because PC4 adds a large jump (93.9% to 97.6%), while GPT-2 is left without a fixed elbow because its spectrum is weaker and less cleanly separated.

Cumulative Explained Variance (PCA)



For the default Llama3.1-8B view (layer 5), PC1-3 already explains 97.1% of variance, and PC1-6 explains 98.0%, so most structure is concentrated in the first three components. Across models, PC1-3 explains 93.9%-98.2% for Gemma-2, Gemma-3, Qwen3, and both Pythias (78.8% for GPT-2), while PC1-6 explains 96.6%-98.7% for those stronger models (81.7% for GPT-2). Following the original Anthropic setup, we inspect both 3D component groups (PC1-3 and PC4-6) in the projection plots below.

PCA Projections of Counts' Mean Hidden States



These manifold views are consistent with the probe and PCA results: models with high linear recoverability show smooth, position-ordered trajectories, while weaker models show less coherent geometry.

Why Do We See This Shape?

Relative to the original Anthropic result, our reproduction more often shows the cleanest helix in PC1-3, while PC4-6 is usually less consistent, matching the variance concentration in the first three components.

Next, we move to SAEs to ask which individual features carry the position signal.

Finding Position Features in SAEs

Next, we look for individual SAE features that encode character offset in open-source SAEs. We use GemmaScope and LlamaScope (He et al., 2024; Lieberum et al., 2024), together with GPT-2 SAEs from (Gao et al., 2025), for this analysis. For each model, we take the SAE from the layer with the highest R^2, compute the mean feature activation at each position, and then select features whose activation patterns vary strongly with position.

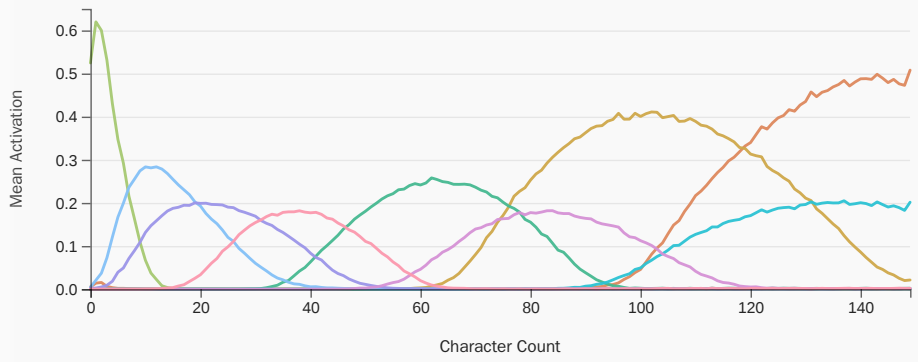
Some features activate only within a narrow range of positions, producing “hill-shaped” tuning curves (for example, a bump around positions 5–10). Such features can encode position very well even when their activation is non-monotonic across the position axis. Variance across positions captures these patterns more reliably.

For the default Llama3.1-8B view (layer 5), the top variance-ranked features are 2424, 72720, 73327, 126269, and 4041. The default synced feature 126269 is rank 4 and is clearly non-monotonic, peaking near position 62. For Gemma-2-9B (layer 11), the top features are 124789, 114007, 862, 41464, and 44341, and the default feature 44341 is also non-monotonic, peaking near 69. GPT-2 (layer 6) still has position-sensitive features (104860, 72264, 6666, 117724, 22611), but their tuning is less clean and more edge-biased.

Across the top-20 features, peak positions span almost the full range (0-149) in all three models, which suggests a distributed bank of position-tuned features rather than a single scalar counter.

AutoInterp descriptions are often only partially aligned with this position-tracking behavior, so the activation curves are still necessary for interpretation. Excluding undefined early positions, feature ranking in the view below is computed from position-wise activation variance at the selected layer.

Mean SAE Feature Activation + Neuronpedia Scope (Synced)



Legend

2424 72720 73327 126269 4041 49358 61073 109458 33835

Model Llama3.1-8B

Layer 5

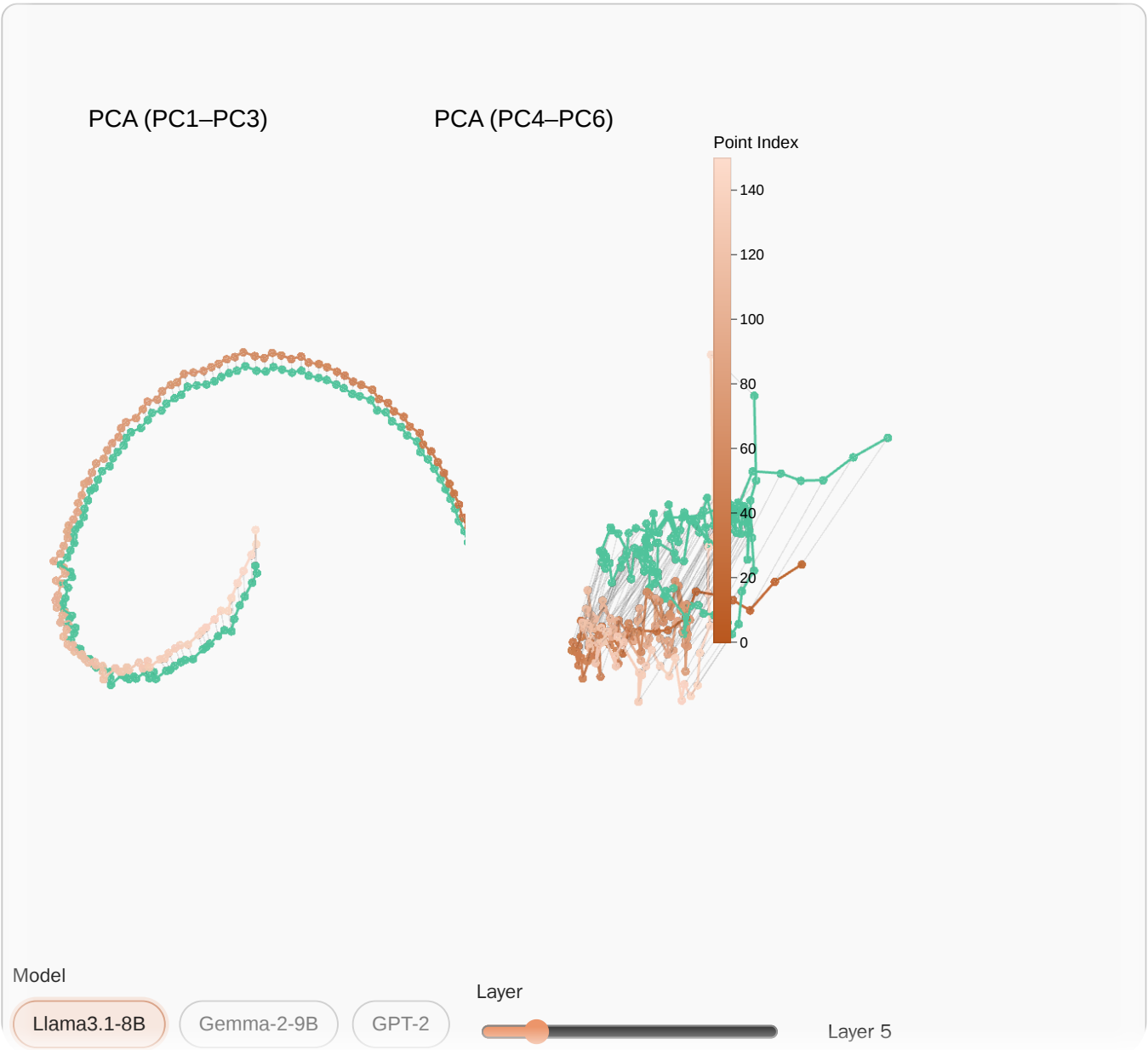
Max features 9

Neuronpedia feature · Llama3.1-8B · #126269 · layer 5

llama3.1-8b · 5-llamascope-res-131k

Finally, we test whether the identified SAE features actually preserve the same geometry as the full residual stream. For each position, we take the mean hidden state at the selected layer, then compare two trajectories in PCA space: (1) the original full-state trajectory, and (2) the trajectory obtained after projecting to the span of the selected SAE features. We use the same PCA coordinates for both views, so direct overlap means the selected features capture the position manifold well, while systematic gaps indicate position information that is still outside this SAE subspace.

PCA Projections (Main vs SAE-Subspace Projection)



Excluding undefined early positions, reconstruction quality is strong for Llama3.1-8B and Gemma-2-9B at their best layers: the main and SAE-subspace trajectories nearly overlap in PC1-5 (with weaker agreement in PC6 for Llama). GPT-2 is visibly less stable in this comparison, with larger gaps in the middle components, matching its weaker probe results in the previous section.

This closes the chain from behavior to hidden-state geometry to feature-level mechanisms, which we summarize in the final section.

Results Summary

Across open models, token position is linearly recoverable from hidden states, but both strength and layer location vary by architecture. The table below summarizes the main quantitative results used throughout this post.

Model	Best layer	Max probe R^2	PC1-3 variance at best layer	SAE-span manifold c
Llama3.1-8B	5	0.791	97.1%	Strong alignment (main vs
Gemma-2-9B	11	0.868	93.9%	Strong alignment (main vs
Qwen3-8B	5	0.852	95.3%	Not available in this SAE c
Gemma-3-4B-pt	18	0.491	98.2%	Not available in this SAE c
GPT-2	6	0.363	78.8%	Weaker alignment than Ll
Pythia-160m	3	0.744	95.4%	Not available in this SAE c
Pythia-410m	6	0.724	97.4%	Not available in this SAE c

For newline prediction quality, results also vary strongly by model family and training setup: Gemma-2-9B reaches 77.9% exact-match on newline tokens, Qwen3-8B reaches 62.5%, and Pythia-410m reaches 63.8%, while GPT-2 family models remain much lower (12.3%-19.7%).

Final Thoughts

Open-weight models do learn internal signals that track position, but where this signal is most recoverable is not consistent across architectures. In our runs, it usually peaked around mid layers rather than the earliest ones. SAE analysis then lets us move from a coarse layer-level statement to specific feature-level mechanisms, and many of the strongest position features show non-monotonic, hill-shaped tuning curves instead of a simple linear trend with position. At the same time, automatic feature labeling remains fragile: AutoInterp descriptions are often plausible-sounding but still miss the actual role of these features in position tracking.

Overall, this behavior appears across most modern LLM families we tested, and it can be studied end-to-end with publicly available models, datasets, and interpretability tooling.

Citation

For attribution in academic contexts, please cite this work as

```
Viacheslav Sinii, Nikita Balagansky (2026). "Chasing the Counting Manifold in Open LLMs".
```

BibTeX citation

```
@misc{sinii2026_chasing_the_counting_manifold_in_open_llms,
  title={Chasing the Counting Manifold in Open LLMs},
  author={Viacheslav Sinii and Nikita Balagansky},
  year={2026},

}
```

Reuse

References

1. Gurnee, W., Ameisen, E., Kauvar, I., Tarng , J., Pearce, A., Olah, C., & Batson, J. (2025). When Models Manipulate Manifolds: The Geometry of a Counting Task. *Transformer Circuits Thread*. https://transformer-circuits.pub/2025/linebreaks/index.html↑

2. Kantamneni, S., & Tegmark, M. (2025). Language Models Use Trigonometry to Do Addition. *arXiv Preprint arXiv:2502.00873*. 10.48550/arXiv.2502.00873↑

3. Nanda, N., Rajamanoharan, S., Kramár, J., & Shah, R. (2023). Fact Finding: Attempting to Reverse-Engineer Factual Recall on the Neuron Level. In *Alignment Forum*. https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall↑

4. Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., & Wu, J. (2025). Scaling and evaluating sparse autoencoders. *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=tcsZt9ZNKD↑

5. He, Z., Shu, W., Ge, X., Chen, L., Wang, J., Zhou, Y., Liu, F., Guo, Q., Huang, X., Wu, Z., Jiang, Y.-G., & Qiu, X. (2024). Llama Scope: Extracting Millions of Features from Llama-3.1-8B with Sparse Autoencoders. *arXiv Preprint arXiv: 2410.20526*. 10.48550/arXiv.2410.20526↑

6. Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kram’ar, J., Dragan, A., Shah, R., & Nanda, N. (2024). Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2. *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. 10.48550/arXiv.2408.05147↑