# Predicting Emergent Capabilities Using Sparse Features

**Aneesh Kumar, Oskar Herlitz**

aneesh.kumar6214@gmail.com, oskar.herlitz@gmail.com

## Abstract

Reports of "emergent" capabilities in transformer-based LLMs (abrupt, non-linear improvements in task performance) remain controversial due to **post-hoc** measurement and ambiguous definitions. We investigate whether such transitions can be predicted **pre-hoc** from internal representations. For each training checkpoint, we train sparse autoencoders (SAEs) on model activations, construct a co-activation graph over SAE features, and track graph statistics (e.g. density, clustering, etc.). We analyze a 2-layer grokking-style transformer with aggregate graph metrics over eight SAE initializations, and test lead-lag relationships between changes in graph metrics and subsequent changes in accuracy under a formalized emergence criterion. Across both settings, we find no statistically significant evidence that global co-activation topology forecasts emergent jumps in performance. If pre-hoc indicators exist, they may lie outside the global graph measures analyzed here (e.g., in task-specific circuits or localized subgraphs).

## Introduction

**Motivation and Definition** Emergent capabilities in large language models refer to discontinuous improvement in tasks that appear abruptly after a certain scale or threshold (Wei et al. 2022). We define emergence as a sharp, sustained jump in test accuracy (formalized further in the Methodology section). While existing scaling laws capture the trends of emergence (Kaplan et al. 2020), they do not identify *when* sharp changes occur. Predictive indicators of emergence could help anticipate model behavior during training, but existing analyses are largely retrospective, identifying abrupt improvements after training.

**Hypothesis** We hypothesize that emergent capabilities can be predicted *pre-hoc* by analyzing internal representations, rather than post-hoc detection. Specifically, we propose that emergence may be preceded by a reorganization of sparse features that underlie model activations. Sparse Autoencoders (SAEs) have shown that LLM activations can be decomposed into interpretable, disentangled features corresponding to semantically meaningful directions in activation space (Cunningham et al. 2024). If emergent behavior suggests a structural phase transition in how such features interact, then the topology of their co-activation graphs might exhibit measurable shifts before emergent jumps in accuracy (Nanda et al. 2023).

**Methods and Our Contribution** We operationalize this hypothesis by checkpointing every 1k steps during model training, where at each checkpoint we:

(1) Train an SAE (seed consistent across steps);

(2) Construct a co-activation graph whose nodes represent individual sparse features and weighted edges represent the magnitude of normalized feature co-activation; and

(3) Compute graph-level statistics such as density, clustering, and component size.

We repeat this process across eight independent SAE initializations and aggregate the resulting metrics to capture consistent representational dynamics. Finally, we perform a *lead-lag* correlation analysis between changes in graph metrics and subsequent changes in model accuracy, testing whether any graph-level reorganization systematically precedes behavioral emergence.

Our framework bridges interpretability and scaling work by observing emergence to probing its underlying representational precursors and assessing the limits of predictability.

## Related Work

**Emergence, Scaling, and Training Dynamics** Scaling laws outline smooth performance gains with model and data size (Kaplan et al. 2020; Hoffmann et al. 2022), while qualitative behaviors (e.g. reasoning, arithmetic) appear abruptly beyond critical scales (Wei et al. 2022). These results frame emergence as an outcome of scaling, not as a predictable training-time event. Closer to training dynamics, work on grokking and phase transition-like behavior analyzes abrupt generalization after prolonged overfitting (Power et al. 2022), typically via loss-landscape diagnostics. The predominant methodology is retrospective detection from validation metrics rather than pre-hoc signals.

**Representation Comparisons Across Checkpoints** Several studies quantify how neural representations evolve during training using similarity or geometric measures such as SVCCA (Raghu et al. 2017), CKA (Kornblith et al. 2019), and linear mode connectivity (Frankle et al. 2020). These approaches analyze representations over time for studying representational change; however, they remain descriptive rather than predictive. Our work complements this by applying a predictive framework.

**Sparse, Interpretable Features** Sparse autoencoders (SAEs) decompose activations into disentangled, often interpretable features that capture semantically meaningful directions (Michael et al. 2023; Cunningham et al. 2024). These results motivate the hypothesis that qualitative behavioral changes may coincide with reorganizations in sparse feature usage or interactions. We leverage SAEs as a measurement tool to probe whether feature-level structure provides pre-hoc signals of behavioral change.

**Topology of Representations** Beyond single-feature analysis, representation geometry and topology have been used to characterize structure in activations (Barrett et al. 2021). Co-activation graphs summarize representational properties over time, but these tools are largely descriptive. We examine whether **temporal** changes in graph topology precede behavioral jumps.

**Positioning** Prior studies either characterize emergence retrospectively or describe representational structure without linking it to future behavior. Our work extends this by testing whether temporal changes in representational topology—derived from sparse features—bear predictive relationships to behavioral transitions during training.

## Methodology

We test whether internal representational metrics from Sparse Autoencoders (SAEs) and Co-activation Graphs (CAGs) predict or explain emergent behavior in transformer-based LLMs. The pipeline has three stages: Sparse Feature Extraction, Co-activation Graph Construction, and Metric Analysis with a statistical predictive test.

### Defining Emergence (Detector)

We define emergence to be a sharp and sustained increase in test accuracy during training. A training step $i$ exhibits emergence if:

(1) The accuracy jump exceeds a threshold $a_i - a_{i-1} \geq \Delta$

(2) Accuracy remains stable within a tolerance ($|a_k - a_i| \leq \tau$) over the next $H$ evaluations.

Default hyper-parameters ($\Delta = 0.20, \tau = 0.02, H = 3$) can detect a large performance jump followed by a short plateau. The method is deterministic, operating retrospectively on metrics logged during the training step, reporting the earliest step satisfying conditions (1) and (2).

### Grokking Experiment

We follow the small-algorithmic dataset paradigm and model scale in (Power et al. 2022) by implementing a two-layer Transformer encoder with learned positional embeddings on the task of modular addition ( $\bmod p = 97$), where the goal is to predict $(a + b) \mod p$ from the pair $(a, b)$.

**Dataset generation** We generate a fully specified synthetic dataset for modular addition: for a fixed prime $p = 97$, we enumerate all ordered pairs $(a, b) \in \{0, \ldots, p - 1\}^2$ (total $p^2 = 9,409$ examples) and label each with $c = (a + b) \bmod p$. We create a deterministic split by shuffling indices with NumPy seeded at 42 and taking the first $\lfloor 0.2 \cdot p^2 \rfloor = 1{,}881$ examples as train and the remaining 7,528 as test. Inputs are integer tokens $(a, b)$; the target is the class index $c$. No additional preprocessing is applied.

**Model and Training Setup** The architecture is a 2-layer Transformer encoder ($d_{model} = 128, n_{heads} = 4, d_{ff} = 256$) with GELU activations, no dropout, and learned positional embeddings for three token positions. The prediction is taken from the final position. We trained the model using AdamW ($lr = 10^{-3}, weight\_decay = 0.1$) on all $p^2 = 9409$ input pairs, with a deterministic 80/20 train/test split. We trained up to 120k steps using full-batch gradient updates (1881 examples), evaluating every 1k steps.

**Sparse Feature Extraction** At each checkpoint, we record the hidden-layer activations of the Transformer encoder before output projection using a forward hook. These activations have shape $(N, 3, d_{model})$ and are later reduced by position (averaged). Each activation vector is flattened and used to train a Sparse Autoencoder (SAE) with ReLU activations, MSE reconstruction loss, and an L1 penalty ($\lambda = 0.01$) to enforce sparsity. The SAE learns an overcomplete basis (8192 features) representing sparse directions in model activation space.

**Co-activation Graph Construction** We binarize SAE activations using a threshold of 0.01 and compute a normalized co-activation matrix:

$$C_{ij} = \frac{A_i^T A_j}{N sqrt(r_i r_j)}$$

where $A_i$ and $A_j$ are the binary activation vector of features $i$ and $j$ (respectively) across samples, while $r_i$ $r_j$ are their respective activation rates. Undirected edges are added between features with $C_(ij) > 0.3$, and isolated nodes are removed. The resulting graph captures feature-level co-activation structure of the model.

**Graph Metrics** We compute standard graph statistics: number of nodes and edges, density, average and maximum degree, number of connected components, largest component size, and average clustering coefficient. Modularity is computed when clusters are present. This characterizes the evolution of feature topology across training.

**Multi-seed setup.** To increase robustness and account for variability in SAE initialization, we run this pipeline across SAE random seeds ($n = 8$).

Each seed produces independent sparse features and co-activation graphs, generating a full timeline of graph metrics aligned by training step.

**Aggregating across seeds.** For each training step and graph metric $M$, we compute the mean, standard deviation, and 95% confidence interval across seeds:

$$\bar{M}_t = \frac{1}{n}\sum_{s=1}^{n} M_{t,s}, \qquad \mathrm{CI}_{95} = 1.96\frac{\sigma_t}{\sqrt{n}}.$$

**Detecting predictive signals.** We formalize a *predictive signal* as a statistically significant lead-lag correlation between changes in a graph metric and subsequent changes in accuracy. For each metric $M$, lag $k$, and test accuracy $A$ (across 5 different lags), we compute per seed:

$$\begin{aligned} r_k &= \mathrm{corr}(\Delta M_{t-k},\, \Delta A_t)\,, \\ \Delta M_t &= M_t - M_{t-1}, \\ \Delta A_t &= A_t - A_{t-1}. \end{aligned}$$

Correlations are averaged across seeds using the Fisher $z$-transform:

$$z = \tanh^{-1}(r), \quad \bar{z} = \frac{1}{n}\sum_{s} z_s, \quad \bar{r} = \tanh(\bar{z}),$$

with 95% confidence intervals computed in $z$-space and transformed back. We consider a metric practically predictive if it exhibits an $\bar{r} > 0.5$, following the conventional interpretation of effect sizes (Cohen 1988).

**Visualization.** Figure 1 shows the cross-seed aggregation of each metric at each training step, emphasizing variability/consistency across seeds. Figure 2 presents a heatmap of mean lead-lag correlations ($\bar{r}$) for all metrics and lags, centered at zero.

## Experiments & Results

We conduct a controlled grokking experiment on a two-layer transformer trained on modular addition, enabling fine-grained temporal analysis of representational dynamics during a hypothesized phase transition (emergence). We test the framework's ability to correlate sparse feature-based co-activation graph metrics with test accuracy.

### Grokking Experiment (2-Layer Transformer)

This experiment was implemented in PyTorch and run locally on an NVIDIA RTX 4070 GPU. We analyze whether the graph-based metrics derived from sparse feature co-activation can anticipate emergence during training. On the two-layer transformer trained on modular addition, we checkpoint and evaluate every 1k steps. At each checkpoint, we extract hidden activations, train a Sparse Autoencoder with 8192 features, binarize activations at a threshold of 0.01, and construct a co-activation graph using normalized pairwise co-activations $C_{ij}$ (edges for $C_{ij} > 0.3$).

**Findings** We compute graph statistics (density, average clustering, edge count, largest-component size, and modularity) across training and align them with test accuracy across 8 SAE seeds. We observe emergence detected at step 22,000 using the method defined previously.

Graph density and clustering fluctuate within narrow ranges, modularity shows transient local variation, and SAE sparsity decreases steadily until convergence (Figure 1). The lack of structure across all seeds suggests that feature-level

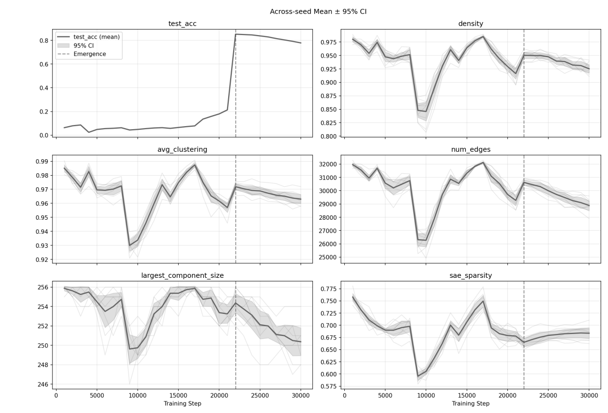

Figure 1: Evolution of test accuracy and graph metrics over the first 30k steps. The dashed line marks the detected emergence point (at step 22,000). No metric exhibited clear predictive structure preceding the accuracy jump.

co-activation topology remains stable, implying structural graph statistics alone are insufficient predictors.

To formally examine whether any metric changes were correlated with abrupt improvements in test accuracy, we conducted a lead-lag correlation analysis. For each metric $M$ and lag $k$, we compute per-seed correlations $r_k = \mathrm{corr}(\Delta M_{t-k}, \Delta A_t)$ and aggregate them across seeds using Fisher $z$-transformation.

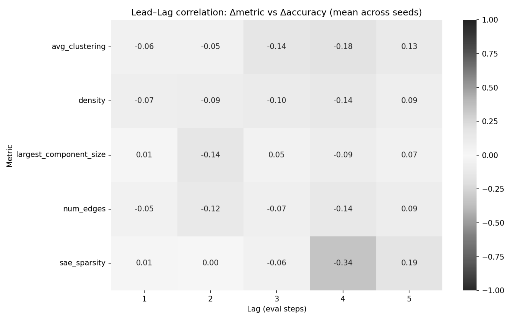

Figure 2: lead-lag correlations ($\bar{r}$) between changes in graph metrics and changes in test accuracy, averaged across eight SAE seeds. Rows denote metrics; columns denote lag offsets in 1k-step intervals. All correlations remain weak ($|\bar{r}| < 0.35$), implying the absence of strong predictive structure.

Figure 2 summarizes the resulting mean correlations $\bar{r}$ as a heatmap. All correlations are of insignificant magnitude ($|\bar{r}| < 0.5$), and their 95% confidence intervals include zero, indicating no statistically reliable predictive signals. The largest effect was a modest negative correlation for SAE sparsity at lag 4 ($\bar{r} = -0.34$), suggesting that transient increases in sparsity slightly preceded decreases in accuracy, but this trend was not significant.

## Conclusion

We tested whether co-activation topology of sparse features is correlated with behavioral emergence during training. In a grokking setting with a clear accuracy jump, none of the graph-level statistics we analyzed (density, clustering, edge count, largest-component size, modularity)–aggregated across eight SAE seeds–exhibited reliable lead-lag correlations with subsequent accuracy changes. All mean effects were small $|\bar{r}| < 0.35$. Taken together, our results provide negative evidence for global co-activation topology as a pre-hoc predictor of emergence under our methods and configuration.

**Limitations** Our analysis is limited to one algorithmic task. Future work may examine other tasks as well as: temporal derivatives of graph structure, richer similarity measures (e.g. spectral embeddings), and scaling trends across models to identify whether more subtle structural signatures anticipate emergent behavior.

## References

Barrett, D. G. T.; Dherin, B.; Chien, S.; and Botev, A. 2021. Analyzing internal representations of neural networks using persistent homology. *arXiv preprint arXiv:2106.05304*.

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Hillsdale, NJ: Lawrence Erlbaum Associates, 2nd edition.

Cunningham, H.; et al. 2024. Sparse Autoencoders Find Highly Interpretable Features in Language Model Activations. In *Proceedings of ICLR 2024*.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020. Linear Mode Connectivity and the Lottery Ticket Hypothesis. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D. d. L.; Hendricks, L. A.; Welbl, J.; Clark, A.; et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of Neural Network Representations Revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 3519–3529.

Michael, J.; Zou, A.; Raffel, C.; Bansal, M.; et al. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2310.13072*.

Nanda, N.; Chan, L.; Lieberum, T.; Smith, J.; and Steinhardt, J. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

Power, A.; Burda, Y.; Edwards, H.; Babuschkin, I.; and Misra, V. 2022. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. *arXiv preprint arXiv:2201.02177*.

Raghu, M.; Gilmer, J.; Yosinski, J.; and Sohl-Dickstein, J. 2017. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

## Appendices

### Appendix A: Pythia as a Non-Emergent Baseline

**Role and scope.** We include EleutherAI Pythia-410M as a *non-emergent baseline* to characterize SAE-based co-activation topology in a regime where validation behavior is smooth, contrasting our grokking-style transformer. We implement a separate feature extraction and co-activation pipeline here. Because no behavioral jump was observed, we report representational topology only and make no prediction claims.

**Setup** We extracted hidden activations from layer $\ell$=20 at publicly released checkpoints and trained a *separate* linear Sparse Autoencoder (SAE) per checkpoint:

$$\hat{h} = D \cdot \mathrm{ReLU}(E \cdot h), \tag{1}$$

where $E \in \mathbb{R}^{m \times d}$ and $D \in \mathbb{R}^{d \times m}$ with $m = 8192$.

SAEs were optimized with MSE reconstruction plus an $\ell_1$ sparsity penalty ($\lambda$=0.01), using AdamW (lr=$10^{-3}$, wd=$10^{-5}$), batch size 1024, and 800 epochs. Latents $z$=$\mathrm{ReLU}(Eh)$ were binarized by a fixed threshold $\tau$=0.01 to obtain activation indicators $a_f$=$\mathbf{1}[z_f{>}\tau]$.

**Co-activation graph.** For sparse features $f, g$, we computed Jaccard similarity

$$\mathrm{Jaccard}_{fg} = \frac{|A_f \cap A_g|}{|A_f \cup A_g|}, \quad A_f = \{t \in S : a_f(t) = 1\} \tag{2}$$

Edges were retained when $\mathrm{Jaccard}_{fg} \geq t$ with $t = 0.01$. For computational efficiency, we analyzed 1024 features per checkpoint (subset of 8192). We reported node/edge counts, modularity $Q$ density $\delta$=$\frac{2|E|}{|V|(|V|-1)}$, node/edge counts, degree statistics, and community detection via spectral clustering.

**Computational constraints** Given the model size and our computational limits, we analyzed a single checkpoint (the final one) and a *subset of SAE features per checkpoint* (up to 1024). We cached co-activation matrices and skipped zero-variance features. These choices were pragmatic.

**Results** Validation accuracy evolved smoothly with no discrete jump; hence, no emergence event was detected. Co-activation graphs were large and structured: at the last training checkpoint we observed $1,024$ nodes and $69,433$ edges, with $10$ communities via spectral clustering. Across the inspected checkpoints, density and average clustering varied modestly, and the largest component size was stable. In line with the negative-control role, we did not perform forecasting (lead–lag) analysis for Pythia.

**Interpretation.** In a non-emergent, large-model regime, SAE-derived co-activation topology appears coherent at the. This complements the grokking setting (where an emergence event exists and forecasting was tested): global co-activation topology neither forecasts behavior when an event *does* occur (grokking) nor fabricates forecasting claims when no event exists (Pythia).

**Limitations specific to Pythia** (1) Separate implementation: results are not a code-level replication of the grokking pipeline. (2) Subsampling of features and checkpoints due to compute constraints; quantitative values may shift with larger budgets. (3) Single-checkpoint descriptive analysis (no multi-seed aggregation), does not show temporal aspect of representational change. (4) No predictive-analysis or layer sweeps are reported; we used $\ell = 20$, $\tau = 0.01$, $t = 0.01$.

## Appendix B: AAAI Reproducibility Checklist

- Includes a conceptual outline and/or pseudocode description of AI methods introduced: Yes
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results: Yes
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper: Yes
- Does this paper make theoretical contributions?: No
- Does this paper rely on one or more datasets? Yes
- A motivation is given for why the experiments are conducted on the selected datasets: Yes
- All novel datasets introduced in this paper are included in a data appendix: N/A
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes: N/A
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations: N/A
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available: N/A
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing: Yes
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting: Yes
- Any code required for pre-processing data is included in the appendix: Yes
- All source code required for conducting and analyzing the experiments is included in a code appendix: Yes
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes: Yes
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from: Partial
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results: Yes
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks: Partial
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics: Yes
- This paper states the number of algorithm runs used to compute each reported result: No
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information: Yes
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank): Yes.
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments: Yes

## Appendix C: Code Appendix

https://github.com/aneesh6214/nyx-soso-predicting-emergence/tree/aneesh/grokking-experiment