Priors in Time: A Generative View of Sparse Autoencoders for Sequential Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sparse Autoencoders (SAEs) are widely used to decompose neural network representations into interpretable concepts. Despite their success, SAEs often fail to capture all relevant concepts, raising the question of what assumptions underlie their limitations. We show that these challenges arise from a mismatch between the true data distribution and the implicit priors encoded in SAE architectures and sparsity regularizers. Taking language model representations as a case study, we demonstrate that these activations exhibit rich temporal structure—such as systematic growth in concept dimensionality, context-dependent correlations, and non-stationarity over time—that conflicts with SAE priors. Through experiments, we highlight how this mismatch leads to characteristic SAE pathologies, including degraded concept recovery and reconstruction quality over time. Our results point toward the need for new SAE designs that incorporate inductive biases aligned with temporal dynamics in sequential data.

1 Introduction

Scaling of neural networks [1–4] has been argued to enable better approximations of the data distribution [5–8], leading to learning of cognitive abilities generally associated with humans [9–12]. Since computations underlying these capabilities occur via intermediate representations, one can intuitively expect the representations encode concepts relevant to such abilities (i.e., latent factors of the generative process) within them [13–19]. With the dual goal of better understanding human cognition and controlling models', this argument has motivated recent interpretability work on identifying what concepts underlie a model's abilities. Specifically, such works build on hypothesized computational models of how concepts are encoded in neural network representations, motivating tools for unsupervised extraction of a dictionary of vectors that (ideally) correspond to meaningful concepts. The prime example of such computational models and corresponding tools includes the linear representation hypothesis (LRH) [20–23] and sparse autoencoders (SAEs) [24–27].

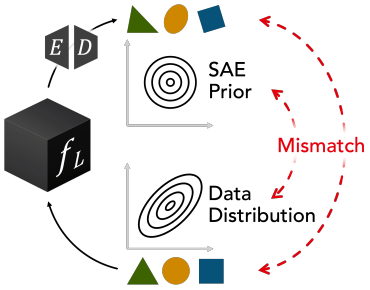

Figure 1: **SAEs assume a prior on concepts that generate data.** When this prior deviates from true data structure, SAEs end up learning different/distorted concepts.

This work. If neural network representations indeed encode concepts underlying the distribution, any time a computational model is proposed for how concepts organize in representations, one ends up making implicit assumptions about the data distribution. For example, motivated by literature on representation steering, SAEs assume model behavior can be manipulated along a given concept without interfering with other ones. This amounts to assuming statistical independence of concepts

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

involved in the generative process. For natural domains, e.g., language, such an assumption can be restrictively strict, since even simple linguistic concepts such as tense and prepositions will be correlated. The mismatch between such implicit assumptions versus the true generative process can thus constrain what concepts an SAE is able to identify from representations (Fig. 1). We aim to formalize this problem: specifically, we make explicit the assumptions about the data-generating process implicit in SAEs and demonstrate the consequences thereof in the specific setting of interpreting language models. In brief, our contributions can be summarized as follows.

- **Characterizing prior assumptions of existing SAEs.** By adopting a Bayesian perspective on SAE training objectives, we prove that SAEs implicitly assume a prior distribution over concepts.
- **Revealing temporal structure in LLM representations.** Contextualizing our theory with respect to large language models (LLMs), we demonstrate rich temporal dynamics in an LLM's representations—including systematic growth in the number of active concepts, context-sensitive correlations, and non-stationarity—that are inconsistent with implicit assumptions made by SAEs.
- **Demonstrating SAE limitations under prior–data mismatch.** The mismatch between prior assumptions of SAEs and actual structure of LLM representations suggests SAEs will exhibit poor reconstruction error with increasing sequence length. We empirically verify this claim with off-the-shelf, pretrained SAEs.

Overall, our results suggest interpretability should be driven by the behavior one is trying to explain—else, mismatch in structure of the distribution describing the behavior and tools used can lead to pathological results [28–32].

2 Priors of Sparse Autoencoders

Let bold, lowercase letters represent vectors (e.g., $\boldsymbol{z}$). Subscripts on vectors denote different samples (e.g., $\boldsymbol{z}_i$), while superscripts denote the index within the vector, leading to a scalar (e.g., z^k). We denote model activations by $\boldsymbol{x} \in \mathbb{R}^n$, SAE latents (sparse code) by $\boldsymbol{z} \in \mathbb{R}^M$, and the dictionary by $\boldsymbol{D} \in \mathbb{R}^{n \times M}$ (M is the dictionary size). SAEs solve the following optimization problem, which is equivalent to constrained dictionary learning ([29]): $\arg\min_{\boldsymbol{D},\boldsymbol{z}} \sum_{i=1}^{N} \frac{1}{N} \left(\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2 + \lambda \mathcal{R}(\boldsymbol{z}_i) \right)$, where $\forall k, \boldsymbol{z}_k = f_{\texttt{SAE}}(\boldsymbol{x}_k)$. This formulation affords a Bayesian lens on SAEs as a MAP (maximum-aposteriori) estimation problem for the sparse codes $\{\boldsymbol{z}_i\}$, given data $\{\boldsymbol{x}_i\}$, where the first term (MSE) captures the log likelihood, while the latter (sparsity penalty) is the log prior. This perspective leads us to the following theorem about priors of various SAEs.

Theorem 2.1 (SAE Priors on Sparse Code)**.** *Let $S_t = \mathrm{supp}(\boldsymbol{z}_t) = \{k : z_t^k > 0\}$ be the set of active latents in the sparse code $\boldsymbol{z}$ at time t, and $n_t = |S_t|$ be the cardinality of S_t (the number of active latents). Each SAE imposes a prior distribution on the sparse code $\boldsymbol{z}$, arising from its sparsity penalty $\mathcal{R}(\boldsymbol{z})$ or implicit conditions imposed on the sparse code. These conditions are highlighted in Table 1.*

Table 1: Priors over concept interactions and dynamics for various SAEs

$f_{\texttt{SAE}}, \mathcal{R}(\boldsymbol{z})$	Across-Concept Prior (interaction)	Across-time Prior (dynamics)
ReLU, L_1-norm	$z_t^1, \ldots, z_t^M \overset{\text{i.i.d.}}{\sim} \text{Laplace}(0, \cdot)$	$\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t \overset{\text{i.i.d.}}{\sim} P^{(1)}$
TopK	$z_t^{i_1}, \ldots, z_t^{i_K} \mid S_t \overset{\text{i.i.d.}}{\sim} U(0, \cdot)\ \forall i. \in S_t$, $S_t \sim U([M]^K)$	$(\boldsymbol{z}_1, S_1), \ldots, (\boldsymbol{z}_t, S_t) \overset{\text{i.i.d.}}{\sim} P^{(2)}$, $S_1, \ldots, S_t \overset{\text{i.i.d.}}{\sim} U([M]^K)$
JumpReLU, L_0-norm	$z_t^{i_1}, \ldots, z_t^{i_{n_t}} \mid S_t \overset{\text{i.i.d.}}{\sim} U(0, \cdot)\ \forall i. \in S_t$, $S_t \mid n_t \sim U([M]^{n_t})$	$(\boldsymbol{z}_1, S_1, n_1), \ldots, (\boldsymbol{z}_t, S_t, n_t) \overset{\text{i.i.d.}}{\sim} P^{(3)}$, $n_1, \ldots, n_t \overset{\text{i.i.d.}}{\sim} P^{(4)}$
BatchTopK	$z_t^{i_1}, \ldots, z_t^{i_{n_t}} \mid S_t \overset{\text{i.i.d.}}{\sim} U(0, \cdot)\ \forall i. \in S_t$, $S_t \mid n_t \sim U([M]^{n_t})$	$(\boldsymbol{z}_1, S_1, n_1), \ldots, (\boldsymbol{z}_t, S_t, n_t) \overset{\text{i.i.d.}}{\sim} P^{(5)}$, $n_1, \cdots, n_t \overset{\text{i.i.d.}}{\sim} P^{(6)}$, $\mathbb{E}[n_t] = K$

Implications for concepts: ReLU SAE assumes an independence prior over concepts at each time t, as well as independence of concepts over time. TopK, BatchTopK and JumpReLU SAEs assume conditional independence of concepts conditioned on the set of active concepts at each time, and independence of concepts along with their sparsity over time.

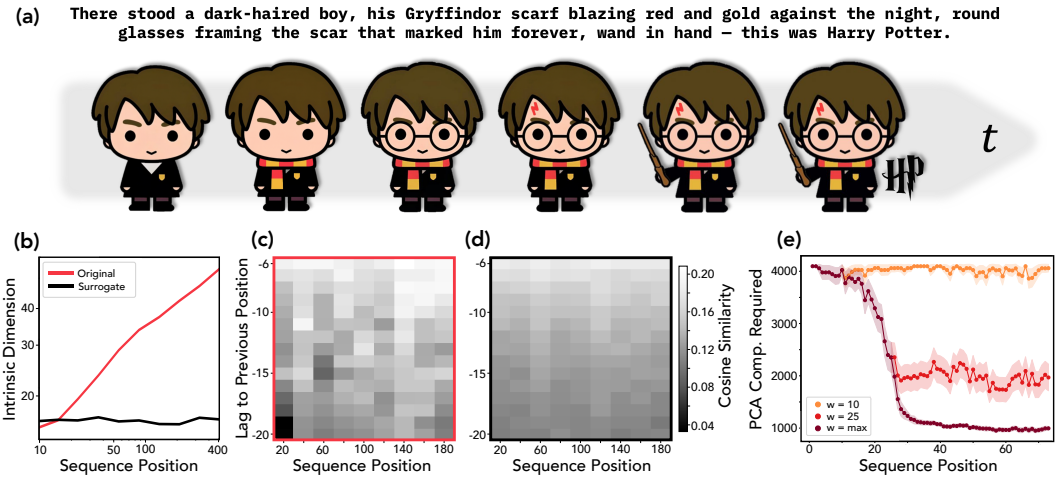

Figure 2: **Language model representations have rich temporal structure**: Illustration (in (a)) of and evidence (b–e) for temporal properties of LLM representations. (a) An illustrative, toy example of Harry Potter demonstrating the evolution of character attributes over a sentence. (b) Intrinsic dimension (measured using a U-statistic of cosine similarities between vectors) as a function of time. (c) Autocorrelation function of representations as a function of time (x-axis) and lag/ shift (y-axis); (d) baseline autocorrelation for a stationary process. (e) Number of PCA components (basis constructed from past context) needed to explain 90% variance in representations as a function of time.

3 Temporal Structure in LLM Representations

Natural language has a temporal structure, as illustrated in Fig. 2(a). The individual characters depict the reader's growing knowledge across the given sentence about Harry Potter. There are three notable observations: (1) the number of concepts (attributes of the character) increases with time, with changes being incremental at each timestep; (2) the sum total of concepts at a given time t is strongly correlated with the past context, while only a few additions are novel; and (3) the sentence is non-stationary. Since a different concept is added at each step, the correlations between concepts at a given position and past positions are time-dependent.

As LLMs model natural language, we expect LLM representations to reflect such temporal structure. To this end, we investigate the temporal structure in LLM representations when processing webtext, to enable a comparison with the priors of existing SAEs discussed in Section 2. Specifically, we investigate the LLama-3.1-8B model representations on the Pile dataset [33]. Fig. 2(b)–(e) provide evidence for the non-stationarity of LLM representations. Panel (b) shows a steady increase in the intrinsic dimension of representations over time. This trend indicates an increase in the number of concepts, assuming concept representations are quasi-orthogonal [34, 30]. The autocorrelation in Panels (c) and (d) show the similarity between a representation at position t with earlier representations. The autocorrelation pattern of LLM representations (c) is different across time t, indicating that representations are non-stationary. For comparison, Panel (d) shows that the autocorrelation of a stationary process is approximately uniform across the sequence. Appendix B provides further details on stationarity measures and the surrogate stationary process. Finally, in Panel (e), we project representations at a given time onto the PCA basis obtained using a context window with varying size. We note that 90% of the variance in representation can be explained using a context window of the past 50 tokens, strongly suggesting correlations between a representation and its context.

4 SAE Pathologies on Temporal Dynamics

The mismatch between SAE priors (Theorem 2.1) and the actual temporal structure of LLM representations (Section 3) leads to three systematic failures, illustrated in Fig. 3.

Sparsity dynamics (Column a): Neither ReLU nor TopK SAEs capture the systematic growth in concept count observed in LLM representations (Fig. 2(b)). ReLU SAEs maintain nearly-constant sparsity throughout most of the sequence, with L_0 norm increasing only at late positions. TopK

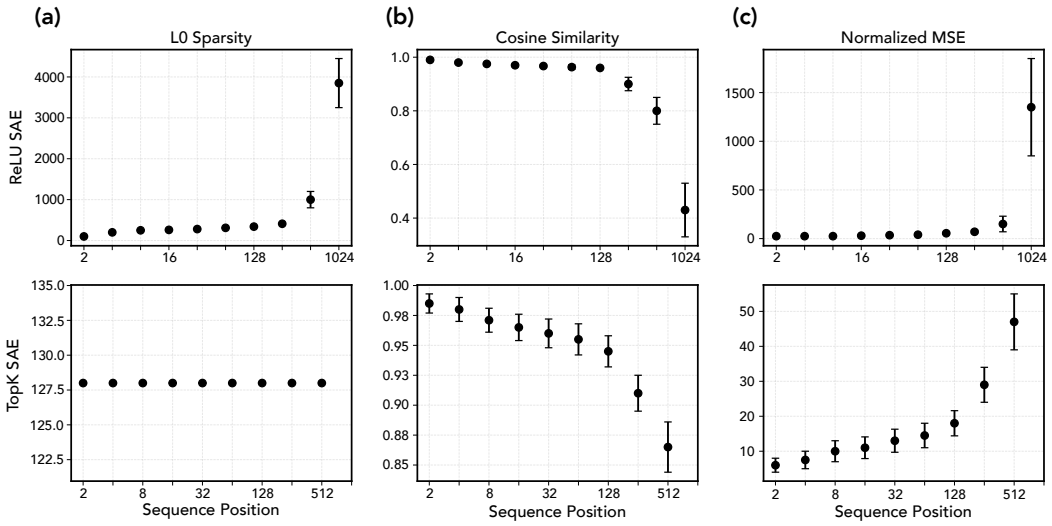

Figure 3: **SAEs fail to capture temporal structure in LLM representations**: Results from ReLU, TopK SAEs trained on GPT-2 model activations. (a) The sparsity of representations (L_0 (pseudo-)norm) as a function of time. (b) Cosine similarity between learnt features and true features decreases with time, making concept extraction progressively worse with time. (c) Normalized MSE increases with time, indicating worse reconstruction for later parts of a story.

SAEs, constrained by their architecture to exactly K active features, cannot adapt to varying sparsity levels at all.

Concept recovery (Column b): Cosine similarity between learned and true concept directions degrades progressively along the sequence for both SAE types, indicating deteriorating concept recovery—whereas an ideal SAE would maintain high similarity throughout.

Reconstruction quality (Column c): Normalized mean-squared error (NMSE) increases monotonically with sequence position, demonstrating that both SAE variants struggle increasingly to reconstruct representations with increasing sequence length.

5 Discussion

Our findings highlight fundamental limitations of existing SAEs when applied to sequential data. Most SAEs implicitly assume independence of concepts across time, yet our analysis shows that LLM representations exhibit rich temporal dynamics. These include a systematic growth in the number of active concepts, strong correlations with recent context, and non-stationarity in their temporal dependencies. This prior–data mismatch leads to systematic pathologies: in particular, we show that ReLU and TopK SAEs fail to capture the progressive increase in conceptual complexity and become increasingly ineffective at both concept recovery and representation reconstruction over time. We hypothesize that abstract, context-dependent phenomena—such as humor, sarcasm, or other higher-level semantic constructs—are especially unlikely to be recovered under current SAE formulations, as they emerge later in sequences and rely heavily on contextual correlations.

These results suggest the need for SAE variants with stronger inductive biases tailored to sequential representations. Future architectures may explicitly account for temporal correlations and incremental novelty in activations, for example by incorporating recurrence or self-attention over past concepts, and imposing sparsity penalty on incremental novel concepts rather than all concepts. Such modifications could enable SAEs to better align with the true dynamics of LLM representations, thereby improving their ability to extract meaningful and temporally coherent concepts.

References

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[3] Jonathan S Rosenfeld. Scaling laws for deep learning. *arXiv preprint arXiv:2108.07686*, 2021.

[4] Corinna Cortes, Lawrence D Jackel, Sara Solla, Vladimir Vapnik, and John Denker. Learning curves: Asymptotic values and rate of convergence. *Advances in neural information processing systems*, 6, 1993.

[5] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

[6] Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

[7] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*, 2024.

[8] Daniel Wurgaft, Ekdeep Singh Lubana, Core Francisco Park, Hidenori Tanaka, Gautam Reddy, and Noah D Goodman. In-context learning strategies emerge rationally. *arXiv preprint arXiv:2506.17859*, 2025.

[9] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[10] Google Deepmind. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-i 2025.

[11] James WA Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, et al. Testing theory of mind in large language models and humans. *Nature Human Behaviour*, 8(7):1285–1295, 2024.

[12] Laura Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. The goldilocks of pragmatic understanding: Fine-tuning strategy matters for implicature resolution by llms. *Advances in Neural Information Processing Systems*, 36:20827–20905, 2023.

[13] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[14] 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.

[15] 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, 2025.

[16] Bo Zhao, Maya Okawa, Eric J Bigelow, Rose Yu, Tomer Ullman, Ekdeep Singh Lubana, and Hidenori Tanaka. Emergence of hierarchical emotion organization in large language models. *arXiv preprint arXiv:2507.10599*, 2025.

[17] Michael Pearce, Elana Simon, Michael Byun, and Daniel Balsam. Finding the tree of life in evo 2. *Goodfire Research*, August 2025. Correspondence to michael@goodfire.ai.

[18] Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. *arXiv preprint arXiv:2406.01506*, 2024.

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

[20] 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$_m odel/index.html$.

[21] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

[22] Jack Merullo, Noah A Smith, Sarah Wiegreffe, and Yanai Elazar. On linear representations and pretraining data frequency in language models. *arXiv preprint arXiv:2504.12459*, 2025.

[23] 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. *arXiv preprint arXiv:2308.09124*, 2023.

[24] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

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

[26] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders, 2024.

[27] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024.

[28] David Chanin, James Wilken-Smith, Tomas Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2025.

[29] Sai Sumedh R Hindupur, Ekdeep Singh Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *arXiv preprint arXiv:2503.01822*, 2025.

[30] Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. *arXiv preprint arXiv:2506.03093*, 2025.

[31] Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

[32] 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.

[33] Monology. The pile: Uncopyrighted subset. https://huggingface.co/datasets/monology/pile-uncopyrighted, 2021. Based on the original Pile dataset by Gao et al.

[34] 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.

[35] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

A Further Theory Results

A.1 Priors on the Sparse Code for various SAEs

We first (re)state the loss function used in training SAEs:

$$\arg\min_{\boldsymbol{D},\boldsymbol{z}} \sum_{i=1}^{N} \frac{1}{N} \left(\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2 + \lambda \mathcal{R}(\boldsymbol{z}_i) \right)$$
$$\text{s.t. } \forall k, \boldsymbol{z}_k = f_{SAE}(\boldsymbol{x}_k)$$

As described in Section **??**, the above loss can be viewed as the negative log posterior in a MAP estimation problem for the sparse code $\{\boldsymbol{z}^{(i)}\}_i$ given data $\{\boldsymbol{x}^{(i)}\}_i$. In this perspective, the prior is given as:

$$\log P(\boldsymbol{z}) \propto -\sum_{i=1}^{N} \mathcal{R}(\boldsymbol{z}) \tag{1}$$

A.1.1 ReLU SAE

The vanilla ReLU SAE ([32], [25]) is trained with the L_1-norm penalty:

$$\mathcal{R}(\boldsymbol{z}) = \|\boldsymbol{z}\|_1. \tag{2}$$

The prior over $\boldsymbol{z}$ for the above case is:

$$\log P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto -\sum_{i=1}^{N} \sum_{k=1}^{M} |z_i^k|, \tag{3}$$

$$\implies P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto \prod_{i=1}^{N} \left(\prod_{k=1}^{M} \exp -\nu |z_i^k| \right). \tag{4}$$

This joint distribution implies that for each sample i, different indices k are sampled i.i.d. from the same distribution:

$$z_i^1, \ldots, z_i^M \overset{\text{i.i.d.}}{\sim} \text{Laplace}(0, 1/\nu), \tag{5}$$

and different samples are all independently sampled from the same product Laplace distribution:

$$\boldsymbol{z}_1, \ldots \boldsymbol{z}_N \overset{\text{i.i.d.}}{\sim} \text{Laplace}^M(0, 1/\nu). \tag{6}$$

This concludes the proof for priors of ReLU SAE trained with L_1 norm sparsity penalty. □

A.1.2 TopK SAE

The TopK SAE ([35], [24]) directly controls the sparsity of the representation $\boldsymbol{z}$ by fixing it at $\|\boldsymbol{z}\|_0 = K$, instead of imposing an explicit sparsity penalty $\mathcal{R}(\boldsymbol{z})$ in the loss function. The objective function for TopK SAE is:

$$\arg\min_{\boldsymbol{D},\boldsymbol{z}} \sum_{i=1}^{N} \frac{1}{N} \|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2, \tag{7}$$

$$\text{s.t. } \forall j, \boldsymbol{z}_j = f_{TopK}(\boldsymbol{x}_j), \ \|\boldsymbol{z}_j\|_0 = K. \tag{8}$$

Since the fixed sparsity is a hard constraint that depends on $\boldsymbol{z}$ alone (and not the data $\boldsymbol{x}$), we can use Lagrange multipliers to reformulate it as an effective prior:

$$\arg\min_{\boldsymbol{D},\boldsymbol{z},\{\lambda_i\}} \sum_{i=1}^{N} \frac{1}{N} \left(\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2 + \lambda_i \left(\left| \|\boldsymbol{z}_i\|_0 - K \right| \right) \right), \tag{9}$$

$$\text{s.t. } \forall j, \boldsymbol{z}_j = f_{TopK}(\boldsymbol{x}_j). \tag{10}$$

The prior over $\boldsymbol{z}$ for the above (effective) regularizer is:

$$\log P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto -\sum_{i=1}^{N} \lambda_i \left(\left| \|\boldsymbol{z}_i\|_0 - K \right| \right) \tag{11}$$

$$\implies P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto \prod_{i=1}^{N} \exp\left(-\lambda_i \left| \|\boldsymbol{z}_i\|_0 - K \right| \right) \tag{12}$$

Note that the above prior is finite for finite values of λ_i, but the overall objective optimizes over $\{\lambda_i\}$, resulting in a *hard* prior peaked at $\|\boldsymbol{z}_i\|_0 = K$ for each sample i.

The factorization over samples i implies mutual independence of $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$: $P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) = \prod_{i=1}^{N} P(\boldsymbol{z}_i)$.

As defined in Theorem 2.1 (and restated here for convenience), let $S_i = \text{supp}(\boldsymbol{z}_i) = \{k : z_i^k > 0\}, n_i = |S_i| = \|\boldsymbol{z}_i\|_0$ denote the active indices and their number (sparsity) respectively.

For individual samples $\boldsymbol{z}_i$, if we condition on the set of active indices S_i, the sparsity gets fixed since $\|\boldsymbol{z}_i\|_0 = |S_i| = n_i$, and the distribution becomes constant:

$$P(\boldsymbol{z}_i \mid S_i) = C \tag{13}$$

$$\implies z_i^{\mu} \mid S_i \sim \begin{cases} U(0, \kappa) & \mu \in S_i \\ \delta_0 & \mu \notin S_i \end{cases}, \text{and} \tag{14}$$

$$z_i^{\mu_1}, \ldots, z_i^{\mu_{|S_i|}} \mid S_i \overset{\text{i.i.d.}}{\sim} U(0, \kappa) \;\; \text{for } \mu_\cdot \in S_i \tag{15}$$

where C, κ are appropriate constants.

Since $\{\boldsymbol{z}_i\}_i$s are mutually independent, any measurable function of each is also independent. The indices of nonzero entries of $\boldsymbol{z}_j$, i.e., S_j is a measurable function since it is a map $S : \mathbb{R}_+^M \to 2^M$ which is discrete valued, and pre images of each value—a set of nonzero indices—are measurable since they equal the cartesian products of the measurable sets $\{z = 0\}, \{z > 0\}$ over all indices. Hence, $S_1, \ldots, S_n$ are also independent.

Since $S_i = g(\boldsymbol{z}_i)$ and the distribution of $\boldsymbol{z}_i$ depends only on $n_i = \|\boldsymbol{z}_i\|_0$ (Eq.11), the distribution of S_i will also depend only on n_i, becoming uniform when conditioned on n_i. In TopK SAE, $n_i = K$ is a constant. Therefore, each $S_i \sim U([M]^K)$, and together with independence argued above,

$$S_1, \ldots, S_N \overset{\text{i.i.d.}}{\sim} U([M]^K) \tag{16}$$

This completes the proof for the priors of TopK SAE. □

A.1.3 BatchTopK SAE

BatchTopK SAE ([26]) is a modification of the TopK SAE. Instead of fixing sparsity like TopK, BatchTopK allows variable sparsity per input while fixing the mean sparsity over a batch at K. The objective function for BatchTopK SAE can equivalently be written as:

$$\arg\min_{\boldsymbol{D}, \boldsymbol{z}} \sum_{i=1}^{N} \frac{1}{N} \|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2, \tag{17}$$

$$\text{s.t. } \forall j, \boldsymbol{z}_j = f_{TopK}(\boldsymbol{x}_j), \;\; \frac{1}{N} \sum_{j=1}^{N} \|\boldsymbol{z}_j\|_0 = K. \tag{18}$$

While BatchTopK imposes a mean sparsity per batch, for simplicity, we use the batch size to match the size of the entire dataset (WLOG). Smaller batch sizes can easily be incorporated by adding separate constraints, each over the entire batch (only leads to a change in constants—lagrange multipliers—in the analysis).

Following similar analysis as for TopK SAE (App. A.1.2), we can derive an equivalent prior over $\boldsymbol{z}$ for BatchTopK SAE:

$$P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto \prod_{i=1}^{N} \exp\left(-\lambda \left| \|\boldsymbol{z}_i\|_0 - K \right| \right) \tag{19}$$

The sparse codes for different samples $\{\boldsymbol{z}_i\}_i$ are thus sampled i.i.d. from a distribution that only depends on the sparsity penalty. While this prior looks very similar to the prior of TopK SAE, the difference is that in TopK, the fixed sparsity constraint is imposed per sample, leading to a different Lagrange multiplier λ_i per sample to optimize over, while in BatchTopK, we have a common multiplier λ over all examples in a batch (with multiple batches, we will have one multiplier per batch), which is then optimized over to ensure that average sparsity per batch constraint is met.

Similar to the analysis for the TopK SAE, we get the following prior over different latents per sample:

$$z_i^{\mu} \mid S_i \sim \begin{cases} U(0,\kappa) & \mu \in S_i \\ \delta_0 & \mu \notin S_i \end{cases}, \text{and} \tag{20}$$

$$z_i^{\mu_1}, \ldots, z_i^{\mu_{|S_i|}} \mid S_i \overset{\text{i.i.d.}}{\sim} U(0,\kappa) \;\; \text{for } \mu_\cdot \in S_i \tag{21}$$

The active indices S_i are sampled uniformly conditioned on the number of active indices n_i:

$$S_i \mid n_i \sim U([M]^{n_i}) \tag{22}$$

The number of active latents n_i are themselves sampled i.i.d. (since $n_i = \tilde{g}(\boldsymbol{z}_i)$ and $\{\boldsymbol{z}_i\}_i$ are i.i.d.) from a distribution whose mean is fixed:

$$n_1, \ldots, n_N \overset{\text{i.i.d.}}{\sim} P, \text{ s.t. } \mathbb{E}[n_\cdot] = K \tag{23}$$

This completes the derivation for the BatchTopK prior. □

A.1.4 JumpReLU SAE

JumpReLU SAE ([27]) is trained with the L_0 (pseudo-)norm regularizer. This leads to the following optimization problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{D},\boldsymbol{z}} \sum_{i=1}^{N} \frac{1}{N} \left(\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2^2 + \lambda \|\boldsymbol{z}_i\|_0 \right) \\ &\text{s.t. } \forall k, \boldsymbol{z}_k = f_{JumpReLU}(\boldsymbol{x}_k) \end{aligned}$$

This objective is equivalent to the following prior over $\boldsymbol{z}$:

$$P(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \propto \prod_{i=1}^{N} \exp\Big(-\eta \|\boldsymbol{z}_i\|_0 \Big) \tag{24}$$

Noting the similarity with the TopK/ BatchTopK cases, we use the same analysis to derive the following conditions:

$$z_i^{\mu} \mid S_i \sim \begin{cases} U(0,\kappa) & \mu \in S_i \\ \delta_0 & \mu \notin S_i \end{cases}, \text{and} \tag{25}$$

$$z_i^{\mu_1}, \ldots, z_i^{\mu_{|S_i|}} \mid S_i \overset{\text{i.i.d.}}{\sim} U(0,\kappa) \;\; \text{for } \mu_\cdot \in S_i \tag{26}$$

$$S_i \mid n_i \sim U([M]^{n_i}) \tag{27}$$

The number of active latents n_i are again i.i.d., but there is no constraint on the mean of the distribution (unlike BatchTopK which constrained the mean of n_i to equal K):

$$n_1, \ldots, n_N \overset{\text{i.i.d.}}{\sim} P, \tag{28}$$

which completes the analysis for JumpReLU SAE. □

B Stationarity measures

LLM activations are empirically non-stationary across the sequence. We quantify the non-stationary nature by measuring autocorrelations and the U-statistic.

B.1 Autocorrelation

We compute autocorrelation by selecting evenly spaced tokens across the sequence and measuring the cosine similarity between each token and tokens at various lags in the past. Specifically, for tokens at position t, we compute similarities to tokens at $t - w$ where lag w ranges from 5 to 20. This creates a heatmap where rows represent lag offsets and columns represent token positions.

For a stationary process, we expect the autocorrelation pattern to remain consistent across time—that is, the relationship between a token and its historical context should be similar regardless of position in the sequence. This would manifest as similar autocorrelation patterns repeating horizontally across token positions. In contrast, for a non-stationary process where representations evolve over time, we expect the autocorrelation patterns to vary systematically across positions, with columns showing different temporal dependency structures as the sequence progresses.

B.2 U-statistic

We measure the effective dimensionality of LLM representations using a U-statistic based on pairwise cosine similarities.

$$\text{U-stat}(t) = \frac{M^2 - M}{\|\mathbf{G}_t\|_F^2 - M} \tag{29}$$

where $\|\mathbf{G}_t\|_F^2$ is the squared Frobenius norm of the Gram matrix. This quantity estimates the effective rank $1/\text{tr}(\mathbf{C}_t^2)$, where $\mathbf{C}_t = \mathbb{E}[\boldsymbol{x}_t \boldsymbol{x}_t^T]$ is the second moment matrix and $\boldsymbol{x}_t$ is the activation vector at time t. Under stationarity, U-stat remains constant. When representations evolve over time, U-stat increases systematically as more orthogonal directions become active.

B.3 Surrogate

For both metrics, we compare LLM activations to a known stationary distribution, referred to as surrogate. The surrogate is obtained from observed LLM activation distribution by randomize the phase in Fourier space. It preserves the magnitude of each dimension while randomizing the phases, which creates a stationary process that maintains the same marginal distribution as the original activations.

C Further Results

C.1 Autocorrelation plots for other LLMs

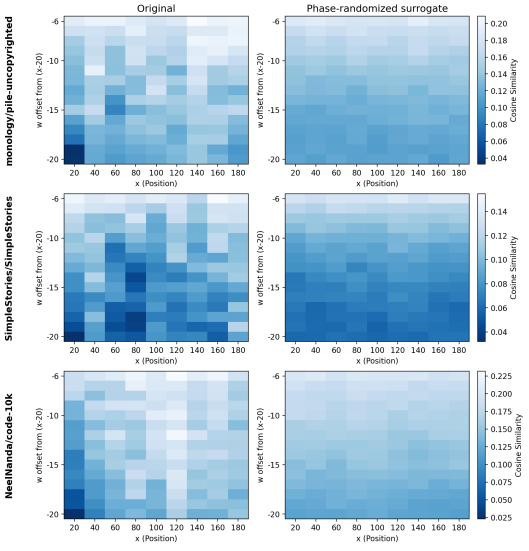

Figure 4: Autocorrelation for Llama 3.1 8B and a surrogate stationary process

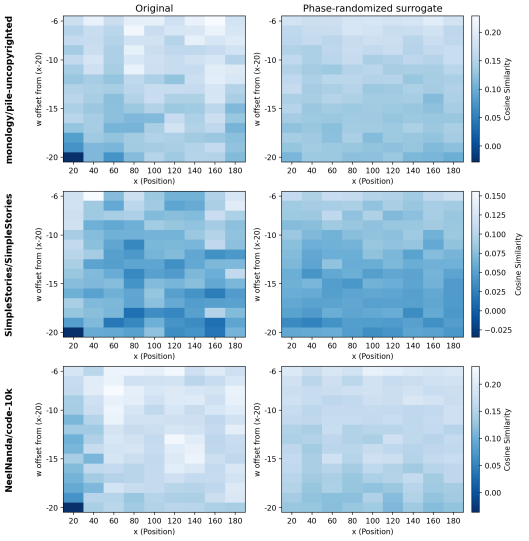

Figure 5: Autocorrelation for Gemma 2 2B and a surrogate stationary process

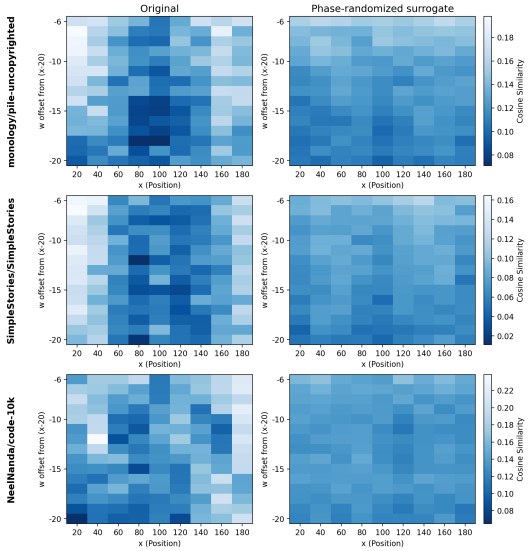

Figure 6: Autocorrelation for GPT 2 small and a surrogate stationary process