Position: Mechanistic Interpretability Should Prioritize Feature Consistency in SAEs

Xiangchen Song[*1] **Aashiq Muhamed**[*1] **Yujia Zheng**[1] **Lingjing Kong**[1]
Zeyu Tang[1,2] **Mona T. Diab**[1] **Virginia Smith**[1] **Kun Zhang**[1,3]
[1]Carnegie Mellon University [2]Stanford University [3]MBZUAI
{xiangchs,amuhamed}@andrew.cmu.edu

Abstract

Sparse Autoencoders (SAEs) are a prominent tool in mechanistic interpretability (MI) for decomposing neural network activations into interpretable features. However, the aspiration to identify a canonical set of features is challenged by the observed inconsistency of learned SAE features across different training runs, undermining the reliability and efficiency of MI research. **This position paper argues that mechanistic interpretability should prioritize feature consistency in SAEs**—the reliable convergence to equivalent feature sets across independent runs. We propose using the Pairwise Dictionary Mean Correlation Coefficient (PW-MCC) as a practical metric to operationalize consistency and demonstrate that high levels are achievable (0.80 for TopK SAEs on LLM activations) with appropriate architectural choices. Our contributions include detailing the benefits of prioritizing consistency; providing theoretical grounding and synthetic validation using a *model organism*, which verifies PW-MCC as a reliable proxy for ground-truth recovery; and extending these findings to real-world LLM data, where high feature consistency strongly correlates with the semantic similarity of learned feature explanations. We call for a community-wide shift towards systematically measuring feature consistency to foster robust cumulative progress in MI.[2]

1 Introduction

Mechanistic Interpretability (MI) seeks to reverse-engineer neural networks into human-understandable algorithms [40, 14], with Sparse Autoencoders (SAEs) emerging as a prominent tool for decomposing model activations into more interpretable, monosemantic features [6, 11, 19, 43]. The aspiration within the MI community is often to identify a canonical set of features—unique, complete, and atomic units of analysis that faithfully represent the model's internal computations [28]. However, a significant challenge highlighted by recent work [44, 32, 15, 28, 36] is the observed inconsistency of features learned by SAEs across different training runs, even when using identical data and model architectures. This instability, potentially arising from phenomena like feature splitting [28, 10] or the amortization gap [42], undermines the reliability of derived interpretations, reduces research efficiency, and impacts the trust in findings derived from MI.

Mechanistic Interpretability Should Prioritize Feature Consistency in SAEs. We argue that the reliable convergence to equivalent feature sets across independent SAE training runs should be elevated from a secondary concern to an essential evaluation criterion and an active research priority. We present the Pairwise Dictionary Mean Correlation Coefficient (PW-MCC) as a concrete, working example of how consistency can be operationalized. Furthermore, we demonstrate that high levels

[*]Equal contribution.
[2]Code is available at https://github.com/xiangchensong/sae-feature-consistency.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Workshop on Mechanistic Interpretability.

of consistency are achievable with appropriate architectural and training choices (e.g., with TopK SAEs), and highlight the benefits of such prioritization for scientific rigor and practical utility in MI.

Our main contributions are:

- We advocate for prioritizing feature consistency in MI for SAEs, detailing its benefits for scientific reproducibility, research efficiency, and trustworthiness of interpretations. We propose using PW-MCC as a practical metric for operationalizing run-to-run feature consistency (Section 3).
- We provide theoretical grounding for achieving strong feature consistency by connecting SAEs to established identifiability results in overcomplete sparse dictionary learning. We validate this using a synthetic *model organism*, demonstrating that PW-MCC reliably tracks ground-truth feature recovery (GT-MCC) and that a specific SAE architecture (TopK SAE) achieves high consistency (≈ 0.97) under idealized, matched-capacity conditions (Section 4).
- We demonstrate empirically on large language model activations that high feature consistency (PW-MCC $\approx$ 0.80 for TopK SAEs) is attainable with appropriate architectural choices and training. Our real-world data experiments reflect findings from the synthetic setting (e.g., frequency-dependent consistency) and critically show that high PW-MCC scores correlate strongly with the semantic similarity of feature explanations (Section 5).

Ultimately, this work calls for a community-wide shift towards valuing and systematically measuring feature consistency for cumulative progress in understanding the inner workings of complex neural models, and we outline several open questions and future research directions to achieve this.

2 Background and Related Work

Sparse Autoencoders for MI. MI aims to reverse-engineer neural networks into human-understandable algorithms by identifying and explicating their internal components and computational processes [40, 43, 11]. A central challenge in MI is polysemanticity, where individual neurons respond to multiple, unrelated concepts, which obscures straightforward interpretation of model internals [13]. SAEs have emerged as a tool to address this challenge by decomposing high-dimensional neural network activations into a sparser, higher-dimensional representation that aims to isolate *monosemantic* features [6, 11]. An SAE comprises an encoder and decoder network. The encoder transforms an input activation vector $\mathbf{x} \in \mathbb{R}^m$ into a sparse latent representation $\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}}$ (where $d_{\text{sae}} > m$ establishes an overcomplete dictionary): $\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{W}_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}})$. Here, $\mathbf{W}_{\text{enc}}$ and $\mathbf{b}_{\text{enc}}$ are the encoder weights and biases, while σ is a non-linear activation function that is sparsity-inducing. The decoder reconstructs the input from $\mathbf{f}(\mathbf{x})$ as $\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{f}(\mathbf{x}) + \mathbf{b}_{\text{dec}}$, where $\mathbf{W}_{\text{dec}}$ and $\mathbf{b}_{\text{dec}}$ are the decoder weights and biases. SAEs are trained by minimizing a loss function that balances reconstruction fidelity with sparsity: $L(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda S(\mathbf{f}(\mathbf{x}))$, where $S(\cdot)$ represents a sparsity-inducing penalty (e.g., L1 norm) and λ controls the sparsity trade-off. Once trained, SAEs provides a decomposition of the input activation as $\mathbf{x} \approx \sum_{i=1}^{d_{\text{sae}}} f_i(\mathbf{x})\mathbf{a}_i$, where $f_i(\mathbf{x})$ are the sparse feature activations, and $\mathbf{a}_i$ are dictionary elements corresponding to columns of feature dictionary $\mathbf{A}$ (i.e., $\mathbf{W}_{\text{dec}}$). Several SAE variants have been proposed to improve feature quality and sparsity control. These include Standard ReLU [6], TopK [18], BatchTopK [7], Gated [46], and JumpReLU [47]. The ultimate aspiration for many researchers employing SAEs is to identify a *canonical* set of features: unique, complete, and atomic units of analysis that faithfully represent the model's internal computations [28].

The Challenge of Feature Consistency. Despite their promise, SAEs trained on identical data and architectures but different random initializations often converge to substantially different feature sets [32, 26], with overlap sometimes as low as 30% for Standard SAEs [44]. This inconsistency manifests through several documented phenomena, including *feature splitting*, where concept representations vary across runs [28], and *feature absorption*, where general features are usurped by more specific ones [10]. These empirical instabilities stem from fundamental limitations in overcomplete dictionary learning [28]. Despite theoretical advances [48, 12], the gap between idealized assumptions and practical implementations undermines guarantees for unique feature recovery. Existing approaches to address these limitations include Mutual Feature Regularization [32], which forces alignment between concurrently trained SAEs but addresses the effect rather than underlying causes, and Archetypal SAEs [15], which impose geometric constraints that may sacrifice representational power for stability. These challenges have fostered skepticism, with some [44] suggesting that SAE features should be viewed as pragmatically useful decompositions rather than exhaustive, universal

sets. Contrary to this prevailing skepticism, our work shows that high feature consistency is attainable through careful architectural and training choices without explicit alignment mechanisms.

3 Mechanistic Interpretability should prioritize Feature Consistency in SAEs

A prerequisite for the scientific validity and practical utility of features extracted in SAEs, is their *consistency*—the reliable convergence to equivalent feature sets across independent training runs given the same data and model architecture. Feature consistency should be elevated from a secondary concern to an essential evaluation criterion and an active research priority for the following reasons.

Firstly, achieving consistency yields substantial benefits for current MI practices that use SAEs.

- *Improved Scientific Reproducibility:* Reproducibility is a cornerstone of science. If SAEs produce different feature dictionaries run-to-run [44, 15], feature explanations and discovered circuits become difficult to replicate. Consistency ensures findings are robust, not initialization artifacts, and helps foster cumulative progress.
- *Improved Research Efficiency and Resource Allocation:* Significant effort is invested in interpreting SAE features, whether manually or through automated methods [6]. If features are not consistent across training runs, this entire interpretation process: identifying, labeling, and understanding features must be repeated for each new SAE training instance. Consistent features, however, can be reliably matched across instances, allowing interpretations to be reused and incrementally refined, thereby saving substantial researcher time and computational resources.
- *Increased Trust in Explanations from SAEs on Private Data:* When SAEs are trained on private data and their dictionaries and feature explanations are shared, feature consistency increases credibility of the explanations. Although the training process (e.g., random initialization) can create features that are artifacts of that specific run, a feature consistently emerging across multiple runs is more likely a stable abstraction learned from the data distribution than an ephemeral artifact. This measurable robustness to training variability lends greater confidence that the interpretations reflect genuine learned patterns, which is even more important when direct data validation is impossible.

Secondly, many current SAE techniques already implicitly assume feature consistency, even if this assumption is not explicitly verified. The long-term MI ambition of identifying *canonical units of analysis*—features that are complete, atomic, and unique [28]—requires run-to-run consistency as a prerequisite before addressing complexities like atomicity or completeness. Another example of implicit reliance is *feature stitching* [28], which compares or combines features across different models or SAEs; this process relies on identifiable and stable underlying features. Similarly, downstream applications like model steering [9], unlearning [37], bias removal [34]), and feature ablation assume that the targeted features are well-defined, consistent entities. If these base features are unstable or non-identifiable, the reliability of such applications is severely compromised.

Our Proposal: Defining and Measuring Feature Consistency. While prior work has highlighted the challenge of feature inconsistency in SAEs [44, 32, 42], often leading to pessimistic conclusions about achieving stable feature sets, our findings demonstrate that high levels of feature consistency *are* attainable, with appropriate architectural choices. This motivates a renewed focus on consistency as a key dimension for evaluation. Our central position is that the MI community **should prioritize feature consistency in SAEs**. This requires not only acknowledging its importance but also adopting rigorous methods for its quantification. Conceptually, feature consistency implies that two feature dictionaries, $\mathbf{A}$ and $\mathbf{A'}$ (both $\in \mathbb{R}^{m \times d_{\text{sae}}}$ with d_{sae} features), learned from independent training runs using same dataset, should capture the same underlying concepts. We formalize this ideal with an empirically tractable notion, **Strong Feature Consistency**, where the dictionaries are considered equivalent if their feature vectors align up to a permutation and individual non-zero scaling factors. That is, for each feature vector $\mathbf{a}_i$ in $\mathbf{A}$, there should ideally exist a corresponding feature vector $\mathbf{a}'_{\sigma(i)}$ in $\mathbf{A'}$ (where σ is a permutation) such that $\mathbf{a}_i = \lambda_i \mathbf{a}'_{\sigma(i)}$ for some scaling factor $\lambda_i \neq 0$. More general notions of consistency are detailed in Appendix C.

To make this actionable, throughout this paper we adopt a commonly used evaluation metric from the independent component analysis literature [24]: the Mean Correlation Coefficient (MCC). This metric directly evaluates permutation and scaling equivalence, making it a robust measure of Strong Feature Consistency for dictionary-based features in SAEs. Cosine similarity addresses arbitrary positive scaling, while the use of the absolute value in MCC accounts for feature sign. We present

this as a concrete, working example of how consistency can be operationalized, although alternative metrics may be more appropriate for other feature types or notions of equivalence.

We define a general **Mean Correlation Coefficient (MCC)** between any two feature dictionaries $\mathbf{A} \in \mathbb{R}^{m \times d_A}$ and $\mathbf{B} \in \mathbb{R}^{m \times d_B}$ with columns $\mathbf{a}_i$ and $\mathbf{b}_j$ respectively. Let $n = \min(d_A, d_B)$ and $\mathcal{M}_n(\mathbf{A}, \mathbf{B})$ be the set of all possible one-to-one matchings of size n between the features of $\mathbf{A}$ and $\mathbf{B}$. The MCC is defined as:

$$\text{MCC}(\mathbf{A}, \mathbf{B}) = \frac{1}{n} \max_{M \in \mathcal{M}_n(\mathbf{A},\mathbf{B})} \sum_{(i,j) \in M} \frac{|\langle \mathbf{a}_i, \mathbf{b}_j \rangle|}{\|\mathbf{a}_i\|_2 \|\mathbf{b}_j\|_2}.$$

The optimal matching M^* that achieves this maximum is typically found using the Hungarian algorithm. From this general definition, we derive two specific metrics for our evaluations:

1. **Pairwise Dictionary Mean Correlation Coefficient (PW-MCC)**: When comparing two dictionaries $\mathbf{A}$ and $\mathbf{A}'$ of learned features, both of size d_{sae}, we use PW-MCC$(\mathbf{A}, \mathbf{A}') = \text{MCC}(\mathbf{A}, \mathbf{A}')$ where $n = d_{\text{sae}}$. A PW-MCC approaching unity signifies robust convergence to highly similar feature dictionaries across independent training runs.

2. **Ground-Truth MCC (GT-MCC)**: In controlled synthetic environments, where a ground-truth dictionary $\mathbf{A}_{\text{gt}} \in \mathbb{R}^{m \times d_{\text{gt}}}$ is known, we use GT-MCC$(\mathbf{A}, \mathbf{A}_{\text{gt}}) = \text{MCC}(\mathbf{A}, \mathbf{A}_{\text{gt}})$ to evaluate the recovery quality of a learned dictionary $\mathbf{A} \in \mathbb{R}^{m \times d_{\text{sae}}}$, where $n = \min(d_{\text{sae}}, d_{\text{gt}})$. GT-MCC can be used for validating PW-MCC as a proxy for consistency.

Prioritizing consistency, and employing well-defined metrics such as PW-MCC to quantify it, offers several advantages: (i) it provides an objective measure of run-to-run stability for this key notion of feature equivalence; (ii) it facilitates equitable comparisons across methods and settings; and (iii) it incentivizes the development of techniques that yield more reliable features. In the following sections, we provide evidence from theory, synthetic experiments, and real-world applications to support our position and illustrate both the attainability and the challenges of achieving high feature consistency.

4 Evidence from Theoretical Analysis and Synthetic Experiments

4.1 Theoretical Foundations for Feature Consistency

SAEs learn to represent input data $\mathbf{X} \in \mathbb{R}^{m \times n}$ through a dictionary $\mathbf{A} \in \mathbb{R}^{m \times d_{\text{sae}}}$ and corresponding sparse activations $\mathbf{F} \in \mathbb{R}^{d_{\text{sae}} \times n}$, such that $\mathbf{X} \approx \mathbf{AF}$. Previous work often dismisses non-invertible dictionaries as non-consistent [42, 25], particularly in the overcomplete regime of dictionary learning where $d_{\text{sae}} > m$. However, this overlooks the natural sparsity present in real signals. Drawing inspiration from sparse dictionary learning literature, we show how sparsity enables feature consistency guarantees even in overcomplete settings. We build our analysis on the **spark condition** [22, 12], which precisely characterizes when unique sparse representations exist:

Definition 1 (Spark condition)**.** *A dictionary $\mathbf{A} \in \mathbb{R}^{m \times d_{sae}}$ satisfies the spark condition at sparsity level k if for any two k-sparse vectors $\mathbf{f}, \mathbf{f}' \in \mathbb{R}^{d_{sae}}$, the equality $\mathbf{Af} = \mathbf{Af}'$ implies that $\mathbf{f} = \mathbf{f}'$.*

This condition ensures that distinct k-sparse vectors produce distinct outputs when transformed by the dictionary $\mathbf{A}$. Equivalently, it provides *injectivity of the linear map $\mathbf{A}$ on the set of k-sparse vectors* $\Sigma_k := \{\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}} : \|\mathbf{f}\|_0 \leq k\}$. This is precisely the algebraic property needed for uniqueness of sparse representations. We leverage the following result from [22]:

Theorem 1 (Adapted from [22])**.** *Fix sparsity level k. There exists a witness set of $n = k\binom{d_{sae}}{k}^2$ k-sparse vectors $\mathbf{f}_1, \ldots, \mathbf{f}_n \in \Sigma_k$ such that for* any *pair of dictionaries $\mathbf{A}, \mathbf{A}' \in \mathbb{R}^{m \times d_{sae}}$ satisfying the spark condition, the factorizations $\mathbf{X} = [\mathbf{Af}_1, \ldots, \mathbf{Af}_n]$ and $\mathbf{X} = [\mathbf{A}'\mathbf{f}'_1, \ldots, \mathbf{A}'\mathbf{f}'_n]$ with k-sparse codes must coincide up to a permutation and scaling of columns: $\mathbf{A}' = \mathbf{APD}$ and $\mathbf{F}' = \mathbf{D}^{-1}\mathbf{P}^\top\mathbf{F}$ for some permutation matrix $\mathbf{P}$ and diagonal invertible $\mathbf{D}$.*

Implications for TopK SAE Feature Consistency. TopK SAEs achieve feature consistency by satisfying the conditions required for unique sparse factorization. Consider a TopK SAE with encoder E and decoder $\mathbf{A}$ that enforces exactly k-sparse activations via $\mathbf{f} \mapsto \text{TopK}_k(\mathbf{f})$. The training objective simultaneously encourages three key properties: **(1) Exact k-sparsity** by construction of the TopK constraint, which zeros all but the k largest coordinates; **(2) Zero reconstruction error**

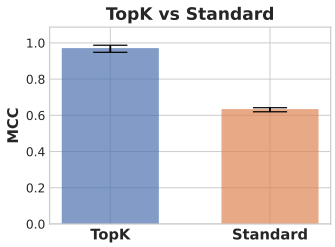

Figure 1: TopK SAE is significantly better than Standard SAE (0.97 vs 0.63) in terms of GT-MCC.

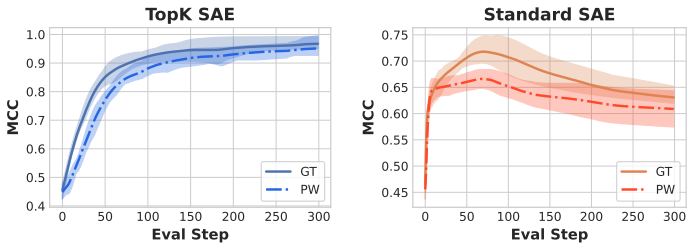

Figure 2: GT-MCC and PW-MCC for TopK and Standard SAE. PW-MCC follows the same trend as GT-MCC, both converging to comparable values. Shaded region represents max-min range across seeds.

by minimizing $\|\mathbf{X} - \mathbf{AF}\|_F$ on data containing the witness set from Theorem 1; and **(3) The spark condition** through what we term the *round-trip property* $E(\mathbf{Af}) = \mathbf{f}$. As we prove in Appendix D, this round-trip property directly implies the spark condition. When these three conditions hold with data coverage meeting the requirements of Theorem 1, any two TopK SAEs trained on the same data must learn dictionaries that are identical up to permutation and scaling, establishing **strong feature consistency** we introduced in Section 3 even in overcomplete regimes. This explains why TopK SAEs can achieve consistent features: their training objective directly optimizes for the mathematical prerequisites required by the identifiability theorem.

Takeaway: SAEs with k-sparsity and minimal reconstruction error satisfy strong feature consistency when the learned dictionary meets the spark condition.

4.2 Synthetic Verification

To empirically validate our theoretical analysis, we conduct synthetic experiments comparing two representative SAE variants: TopK SAE and Standard SAE. We show that models designed according to our theoretical criteria achieve consistent feature representations.

Following conventions in dictionary learning literature, we generate synthetic data by first sampling a ground-truth feature dictionary $\mathbf{A}_{\text{gt}} \in \mathbb{R}^{m \times d_{\text{gt}}}$ from a standard normal distribution. We can represent all data points as $\mathbf{X} = \mathbf{A}_{\text{gt}}\mathbf{F}_{\text{gt}}$, where $\mathbf{F}_{\text{gt}} \in \mathbb{R}^{d_{\text{gt}} \times n}$ contains the activations for all n data points. For each individual data sample $\mathbf{x}$, we enforce the k-sparse condition by randomly selecting k features and setting their values to independent Gaussian samples: $\mathbf{x} = \mathbf{A}_{\text{gt}}\,\mathbf{f}_{\text{gt}}(\mathbf{x})$, where $\mathbf{f}_{\text{gt}}(\mathbf{x}) \in \mathbb{R}^{d_{\text{gt}}}$ represents a single column of $\mathbf{F}_{\text{gt}}$ corresponding to data point $\mathbf{x}$ and contains at most k non-zero entries. In this synthetic setting ($m = 8, d_{\text{gt}} = 16, k = 3, n = 5e4$) we can evaluate the estimated feature dictionary $\mathbf{A}$ against the ground-truth $\mathbf{A}_{\text{gt}}$ using GT-MCC. We also conduct additional experiments by training multiple SAEs (5 seeds) with identical data and model architecture but different weight initializations, comparing the PW-MCC curves with the GT-MCC curves. Figure 1 presents the final MCC evaluation results, showing that TopK SAE achieves significantly better consistency than Standard SAE, which confirms our analysis in Theorem 1. More importantly, Figure 2 demonstrates that the empirical PW-MCC values follow the same trend as GT-MCC, achieving comparable final values, suggesting that PW-MCC serves as an effective alternative to GT-MCC when ground truth dictionaries are unavailable. We refer to this setting as the **matched regime**, where the empirical dictionary size d_{SAE} matches d_{gt}. In all experiments for TopK SAE, the empirical sparsity value k used in during training matches the ground truth sparsity. See Appendix F for the extended analysis when k is misspecified.

Takeaway: We observe that Pairwise MCC converges to GT-MCC and strong feature consistency is achieved with TopK SAE in synthetic matched settings.

4.3 A Synthetic Model Organism for Analyzing Feature Consistency

While TopK SAEs can achieve high feature consistency under idealized matched-capacity scenarios, real-world data introduces substantial complexities that degrade this ideal. To show how these complexities affect feature consistency, we develop and analyze a synthetic model organism,

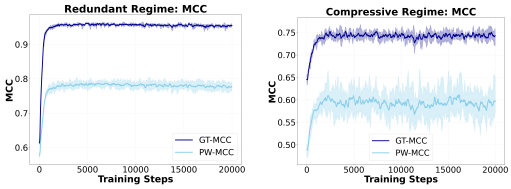

Figure 3: Left: Redundant regime with high GT-MCC but lower PW-MCC due to selection ambiguity. Right: Compressive regime with lower GT-MCC and PW-MCC. Max-min range across 5 seeds is shaded.

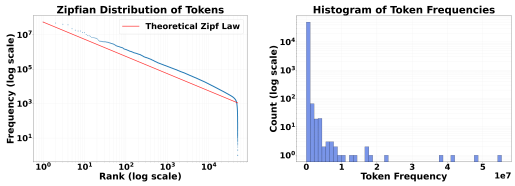

Figure 4: Token frequency in 1M tokens from Pile, showing the Zipfian distribution in real data, with a long and sparse tail.

progressively introducing realistic data characteristics. This allows us to observe how metrics like PW-MCC respond to these challenges providing insights into their continued diagnostic utility.

Consistency and Global Capacity Regimes. Fundamental challenges to feature consistency arise even before considering heterogeneous ground-truth distributions, stemming from the relationship between the SAE's dictionary width d_{sae} and ground-truth dictionary width d_{gt}. Using the linear generative model ($\mathbf{X} = \mathbf{A}_{\text{gt}}\mathbf{F}_{\text{gt}}$) earlier, where ground-truth $\mathbf{f}(\mathbf{x})$ are k-sparse ($k = 8$ in our experiments) and when all ground-truth features are uniformly sampled, we observe distinct behaviors.

In a **globally redundant regime** ($d_{\text{sae}} > d_{\text{gt}}$), where the SAE has more dictionary features than the ground truth (e.g., $d_{\text{sae}} = 160, d_{\text{gt}} = 80, k = 8, n = 5e4$), it can achieve high alignment with the ground-truth dictionary (GT-MCC 0.95). This suggests learned features accurately represent underlying concepts (Figure 3). However, run-to-run consistency is often significantly lower (PW-MCC 0.77). This discrepancy arises from **selection ambiguity**: with excess capacity, multiple learned dictionary vectors can be comparably good matches for a single ground-truth feature, leading to different, yet individually valid, feature sets being learned across runs. The lower PW-MCC here appropriately reflects this reduced stability in feature selection.

Conversely, in a **globally compressive regime** ($d_{\text{sae}} < d_{\text{gt}}$), where the SAE has insufficient capacity (e.g., $d_{\text{sae}} = 80, d_{\text{gt}} = 800, k = 8, n = 5e4$), both GT-MCC ($0.75$) and PW-MCC ($0.60$) are diminished due to the inability to represent all true features (Figure 3). The parallel decline of both metrics indicates their shared sensitivity to fundamental capacity limitations. These global capacity mismatches show that consistency in practice might be harder to achieve, and PW-MCC provides a direct measure of this practical stability.

Zipfian Feature Frequencies and Non-Uniform Capacity Allocation. A primary characteristic of natural language data is the Zipfian (power-law) distribution of underlying feature frequencies (Figure 4)—a few features are common, many are rare. We study the effect of this heterogeneity in the globally compressive regime where SAEs operate in, given the vast number of true concepts versus typical dictionary sizes [5]. To model this, we partition our synthetic ground truth features into C clusters uniformly, each containing d features, and impose an arbitrary ranking on these clusters. Data points are generated by first sampling a cluster i with probability p_i (following a Zipfian distribution with exponent α), and then sampling k true features from that cluster.

Post-training analysis of TopK SAEs trained on this data show that SAEs do not allocate their dictionary capacity uniformly across these clusters. Instead, the effective capacity D_i (number of learned SAE features, matched via Hungarian algorithm, corresponding to ground-truth cluster i) is well-approximated by a power law: $D_i = d_{\text{sae}} \cdot p_i^{\beta} / \sum_j {p_j}^{\beta}$. Our experiments empirically find $\beta \approx 1.4$. Thus, in a globally compressive setting (e.g., $C = 10$, $d = 80$ per cluster, total $d_{\text{gt}} = 800$; $d_{\text{sae}} = 80, k = 8$), more frequent clusters (higher p_i) receive a proportionally larger share of the SAE's limited dictionary representation and, as a result, exhibit higher GT-MCC scores, indicating better feature recovery for more common concepts (Figure 6). This differential ground-truth recovery suggests that run-to-run consistency would similarly depend on the frequency of features, a pattern that feature-level PW-MCC analysis would capture. For additional details see Appendix E.

Emergence of Local Identifiability Regimes and Frequency-Dependent Consistency. This non-uniform capacity allocation driven by Zipfian frequencies means that different ground-truth clusters experience varied effective representational capacity within the same SAE. We define a **local redundancy factor** for each ground-truth cluster i (containing d true features) as $R_i := D_i/d$. This factor characterizes three distinct **local identifiability regimes**: *Locally Redundant* ($R_i > 1$)

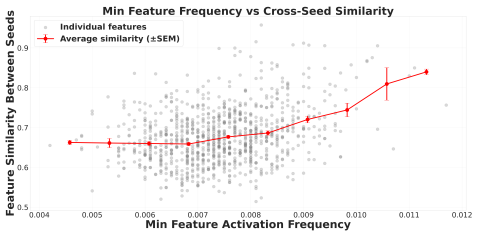

Figure 5: Min activation frequency between matched feature pairs vs. pairwise similarity. Data from two-phase Zipfian model ($d_{\text{gt}} = 5000$, $d_{\text{sae}} = 1000$). Feature-level similarity captures the influence of local consistency regimes across the frequency spectrum.

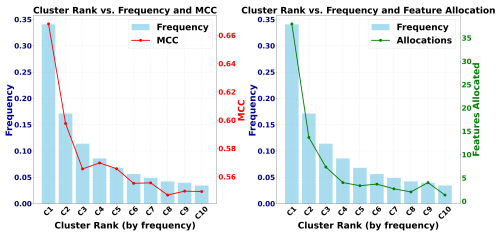

Figure 6: Globally Compressive Zipfian Model ($\alpha = 1$, 10 clusters, $d_{gt} = 800, d_{SAE} = 80$) shows cluster frequency-dependent GT-MCC (red line, left y-axis) and SAE feature allocation (green line, right y-axis).

where the SAE has allocated more features to this cluster than true underlying features, risking selection ambiguity for this cluster's concepts; *Locally Matched* ($R_i \approx 1$) where the allocated capacity approximately matches the cluster's complexity; and *Locally Compressive* ($R_i < 1$) where insufficient capacity allocated for the cluster prevents full recovery. These local regimes coexist within a single, even globally compressive SAE. Frequent clusters may become locally redundant or matched, while rare clusters inevitably remain locally compressive. This coexistence translates to frequency-dependent consistency (PW-MCC) of individual learned features.

Probing the Full Spectrum of Consistency: A Two-Phase Zipfian Model. To more robustly investigate how these local regimes affect the consistency of individual features across a wide dynamic range, especially for very rare concepts, we further enhance our model organism. As real-world feature distributions exhibit an extremely long and sparse tail, we employed a two-phase feature cluster frequency distribution for 50000 clusters with $d_{\text{gt}} = 400000$: a Mandelbrot-Zipf function ($g(r; s_1, q) = (r + q)^{-s_1}$, with $s_1 = 1.05, q = 5.0$) for common concepts (rank $r < 40,000$), transitioning to a steeper power law ($g(r; s_2) \propto r^{-s_2}$, with $s_2 = 30.0$) for the long tail (see Appendix E.4). Training a TopK SAE with a relatively ample dictionary width ($d_{\text{sae}} = 1000$) on this data reveals a clear positive correlation between the minimum activation frequency of matched feature pairs and their inter-run representational similarity as shown in Figure 5.

This correlation emerges naturally from the interplay of local regimes: Frequent features are more likely to be in locally redundant or matched regimes. They receive sufficient allocated capacity, leading to stable learning and high GT-MCC. As true feature frequency decreases, the corresponding learned features are more likely to transition into locally compressive regimes. Here, insufficient allocated capacity relative to the conceptual complexity leads to lower inter-run similarity scores. Features in the extreme tail become so deeply locally compressive that they may be learned inconsistently across runs or not at all, resulting in minimal inter-run similarity. This spectrum of varying stability is effectively quantified by analyzing PW-MCC at the individual feature level.

Takeaway: Our synthetic model organism validates PW-MCC as a robust diagnostic, capturing how global capacity, Zipfian skew, and local identifiability regimes affect feature consistency.

5 Evidence from Applications and Large-Scale Validation

5.1 Evaluating Consistency in Real-World Applications

Prevailing practices in SAE evaluation prioritize metrics like reconstruction error (L2 loss), Fraction of Variance Explained, and sparsity (L0/L1) [6, 11]. While reconstruction fidelity is one aspect of SAE quality, incorporating feature consistency, quantified by PW-MCC, into the evaluation process offers benefits for both practical model development and the interpretation of learned features.

PW-MCC enables more decisive SAE model comparisons and hyperparameter selection where conventional metrics like reconstruction loss prove ambiguous. This advantage is particularly pronounced when controlling feature sparsity. When tuning the L1 coefficient, lower penalties improve reconstruction loss while potentially degrading feature quality. Similarly, in TopK SAEs where reconstruction loss improves monotonically with k, reconstruction loss alone fails to identify

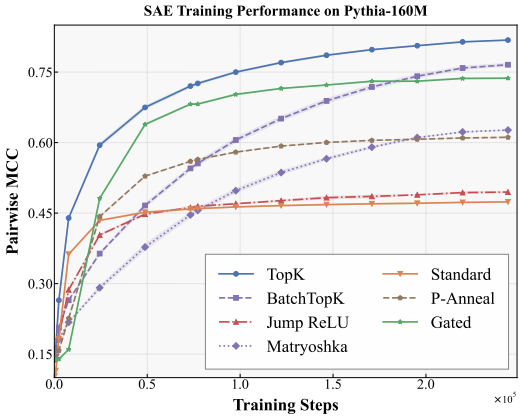

Figure 7: PW-MCC vs. train steps for BatchTopK, Gated, P-Anneal, JumpReLU, Standard, TopK, and Matryoshka BatchTopK SAEs on Pythia-160M activations. Higher PW-MCC indicates greater feature consistency.

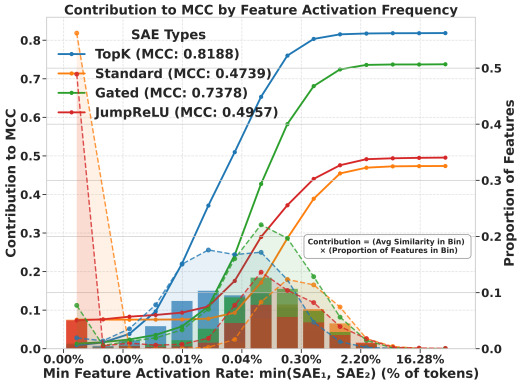

Figure 8: PW-MCC contribution by feature activation frequency for TopK, Standard, Gated, and JumpReLU SAEs. Bars (left axis) show each bin's contribution; solid lines show cumulative contribution. Dashed lines (right axis) show feature distribution across bins.

optimal sparsity levels for feature quality. PW-MCC addresses this limitation by identifying the underlying trade-off. Insufficient sparsity from excessively large k or low L1 penalties causes feature selection ambiguity, while excessive sparsity from small k or high L1 penalties leads to inconsistent concept representation. Both extremes degrade GT-MCC and PW-MCC, enabling clear identification of optimal k. We demonstrate how PW-MCC guides this selection in practice (Appendix F).

PW-MCC acts as a justifiable proxy for ground-truth alignment in unsupervised settings. Its utility stems from observations in our synthetic experiments where PW-MCC strongly correlated with GT-MCC, tracked its progression during training, and served as a practical lower bound. Consequently, low PW-MCC across independent training runs suggests that an SAE is unlikely to converge to a well-defined, *true* feature set. This ability to signal robust feature learning, even without ground truth, underpins its use as a key evaluation metric in the real-world experiments presented next.

5.2 Training SAEs on LLM Activations: Empirical Consistency Results

Experimental Setup. We train SAEs on Pythia-160M model [4], with width 2^{14} on 500 million tokens from `monology/pile-uncopyrighted` [17], using residual stream activations from layer 8. For each SAE, we performed a hyperparameter sweep, selecting the configuration that yielded the highest final PW-MCC across three independent training runs. Further details on the training setup and hyperparameters as well as results for Gemma-2-2B are provided in Appendix G.

Overall Consistency and Training Dynamics. Figure 7 shows the evolution of PW-MCC during training for several SAE architectures. We observe the steady increase in PW-MCC over training steps, indicating that as SAEs learn, their feature dictionaries converge and PW-MCC captures the emergence of consistent representations. Among the evaluated architectures, TopK and BatchTopK

Table 1: TopK SAE (Pythia-160M): Higher activation frequency features show stronger mean cosine similarity (and lower variance) between matched pairs from independently trained SAEs, indicating greater consistency for more prevalent features.

Act freq/1M tokens	Features	Similarity
0.1–2.4	127	0.514 ± 0.280
2.4–54.1	2,542	0.742 ± 0.295
54.1–1.2k	10,013	0.837 ± 0.209
1.2k–26.7k	3,548	0.888 ± 0.157
26.7k–592.2k	33	0.964 ± 0.087

Table 2: TopK SAE (Pythia-160M): Semantic similarity scores (GPT Score, 1-10 scale) for automatically generated explanations of matched feature pairs correlate strongly with the features' dictionary vector cosine similarity. GPT-score is averaged over 20 pairs.

Similarity Range	Feat pairs	GPT Score
0.0654–0.1128	34	1.71
0.1128–0.1947	311	2.19
0.1947–0.3359	975	3.27
0.3359–0.5795	1,423	4.12
0.5795–0.9999	13,640	8.28

SAEs achieved the highest PW-MCC scores. The PW-MCC for some architectures, like BatchTopK, has not fully saturated by 2.5×10^5 training steps, indicating that longer training might yield even higher consistency. The curves also reveal interesting dynamics; for instance, some methods (e.g., BatchTopK) may start with lower consistency but exhibit faster improvement, eventually surpassing others (e.g., Gated SAE), suggesting that different SAEs impose varied structural assumptions [23].

Consistency Across the Feature Frequency Spectrum. The insights from our synthetic model organism, particularly the relationship between feature frequency and consistency, are reflected in these real-data experiments. Table 1 quantifies this: features with higher activation frequencies exhibit markedly stronger inter-run similarity. For example, the rarest features show an average similarity of 0.514, while the most frequent features achieve a much higher 0.964. This trend is broadly observed across architectures (see Appendix G.2). This confirms that frequently occurring concepts are generally learned more stably, and PW-MCC reveals this spectrum of consistency.
Figure 8 further dissects this relationship by showing the contribution of different feature frequency bins to the overall PW-MCC. For the TopK SAE, we observe a relatively symmetric contribution from features across a wide range of activation frequencies, with few dead features. In contrast, the Standard SAE exhibits a larger proportion of features in the lowest frequency bins which contribute minimally to the cumulative PW-MCC, effectively pulling down its overall consistency. Gated SAEs perform well, approaching TopK SAEs but with a slightly larger tail of less active, less consistent features. PW-MCC thus enables a nuanced understanding of how different SAEs utilize their dictionary and achieve consistency across the frequency spectrum.

Correlation with Semantic Similarity. We find that feature consistency is related to learning semantically related concepts. Table 2 shows that when we matched features between two independent SAE runs, binned these pairs by their cosine similarity, used an automated interpretation pipeline (details in Appendix G.4) to generate explanations for each feature in a pair, and used an LLM to rate the semantic similarity of these two explanations (GPT Score), the results show a strong positive correlation. Feature pairs with high dictionary vector similarity receive high semantic similarity scores from the LLM, while pairs with low vector similarity are judged as semantically divergent. This validates that PW-MCC and feature-level similarity are indicative of genuine stability in the learned semantic representations, making them valuable for ensuring the reliability of interpretability findings.

Takeaway: High PW-MCC is achievable on LLM activations with TopK SAEs. PW-MCC tracks feature consistency, reveals frequency-dependent consistency, and correlates with feature semantic similarity.

6 Conclusion and Call to Action

Prioritizing feature consistency in SAEs is important for advancing MI towards a more robust engineering science. This requires a shift in how we develop and evaluate feature extraction methods. **We call upon the community to routinely report quantitative consistency scores** (e.g., PW-MCC), ideally contextualized by feature frequency, alongside standard metrics, enabling meaningful comparisons. **Furthermore, we propose developing standardized benchmarks for consistency**, such as challenging synthetic model organisms with known ground-truth features and data heterogeneity. **Finally, focused research is needed to deeply understand the determinants of consistency**, including the interplay between SAE architecture, optimization, data characteristics, and evaluation metrics.

Our work highlights several fertile avenues for future research: (a) designing SAEs for robust consistency across diverse LLM activation statistics (e.g., early vs. late layers) and developing adaptive sparsity mechanisms responsive to local data properties; (b) improving consistency for less frequent yet potentially critical features, and exploring techniques to target specific parts of the feature spectrum; (c) defining broader notions of feature equivalence beyond strong feature consistency (e.g., functional, subspace alignment) and corresponding consistency metrics; (d) understanding how the SAE encoder's amortization gap [42] influences dictionary stability and whether encoder improvements or explicit consistency regularizers can enhance it; (e) establishing stronger theoretical guarantees for the consistency of features from modern SAEs under realistic data assumptions. Addressing these challenges and embracing a research culture that values and quantifies feature consistency will be pivotal in building a more reliable and cumulative science of MI.

References

[1] Samir Abdaljalil, Filippo Pallucchini, Andrea Seveso, Hasan Kurban, Fabio Mercorio, and Erchin Serpedin. Safe: A sparse autoencoder-based framework for robust query enrichment and hallucination mitigation in llms. *arXiv preprint arXiv:2503.03032*, 2025.

[2] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025.

[3] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Conference on Learning Theory*, pages 779–806. PMLR, 2014.

[4] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

[5] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

[6] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

[7] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

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

[9] Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*, 2024.

[10] David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

[11] 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.

[12] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[13] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[14] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[15] Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *arXiv preprint arXiv:2502.12892*, 2025.

[16] Timo Freiesleben, Gunnar König, Christoph Molnar, and Alvaro Tejero-Cantero. Scientific inference with interpretable machine learning: Analyzing models to learn about real-world phenomena. *Minds and Machines*, 34(3):32, 2024.

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

[18] 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.

[19] Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, et al. Causal abstraction: A theoretical foundation for mechanistic interpretability. *arXiv preprint arXiv:2301.04709*, 2023.

[21] Ruben Härle, Felix Friedrich, Manuel Brack, Björn Deiseroth, Patrick Schramowski, and Kristian Kersting. Scar: Sparse conditioned autoencoders for concept detection and steering in llms. *arXiv preprint arXiv:2411.07122*, 2024.

[22] Christopher J Hillar and Friedrich T Sommer. When can dictionary learning uniquely recover sparse data from subsamples? *IEEE Transactions on Information Theory*, 61(11):6290–6297, 2015.

[23] 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.

[24] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[25] Shruti Joshi, Andrea Dittadi, Sébastien Lachapelle, and Dhanya Sridhar. Identifiable steering via sparse autoencoding of multi-concept shifts. *arXiv preprint arXiv:2502.12179*, 2025.

[26] Adam Karvonen, Can Rager, Jessica Lin, Curt Tigges, Jacob Bloom, Daniel Chanin, Yue-Ting Lau, Euan Farrell, Arthur Conmy, Callum McDougall, Kolawole Ayonrinde, Martin Wearden, Logan Marks, and Neel Nanda. SAEBench: A Comprehensive Benchmark for Sparse Autoencoders. `https://www.neuronpedia.org/sae-bench/info`, 2024.

[27] Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. *Advances in Neural Information Processing Systems*, 37:83091–83118, 2024.

[28] Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. *arXiv preprint arXiv:2502.04878*, 2025.

[29] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

[30] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[31] Aleksandar Makelov, Georg Lange, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. *arXiv preprint arXiv:2311.17030*, 2023.

[32] Luke Marks, Alasdair Paren, David Krueger, and Fazl Barez. Enhancing neural network interpretability with feature-aligned sparse autoencoders. *arXiv preprint arXiv:2411.01220*, 2024.

[33] Samuel Marks, Adam Karvonen, and Aaron Mueller. Dictionary learning. `https://github.com/saprmarks/dictionary_learning`, 2024.

[34] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

[35] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[36] Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? *arXiv preprint arXiv:2502.20914*, 2025.

[37] Aashiq Muhamed, Jacopo Bonato, Mona Diab, and Virginia Smith. Saes can improve unlearning: Dynamic sparse autoencoder guardrails for precision unlearning in llms. *arXiv preprint arXiv:2504.08192*, 2025.

[38] Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangde. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*, 2024.

[39] Chris Olah. Mechanistic interpretability, variables, and the importance of interpretable bases. *Transformer Circuits Thread*, 2(4), 2022.

[40] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[41] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[42] Charles O'Neill, Alim Gumran, and David Klindt. Compute optimal inference and provable amortisation gap in sparse autoencoders. *arXiv preprint arXiv:2411.13117*, 2024.

[43] Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *arXiv preprint arXiv:2504.02821*, 2025.

[44] Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. *arXiv preprint arXiv:2501.16615*, 2025.

[45] Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

[46] Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

[47] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

[48] Yuchen Sun and Kejun Huang. Global identifiability of overcomplete dictionary learning via l1 and volume minimization. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[49] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[50] Kexin Wang and Anna Seigal. Identifiability of overcomplete independent component analysis. *arXiv preprint arXiv:2401.14709*, 2024.

A Alternative Views

While we advocate for prioritizing feature consistency in SAEs, we acknowledge and address potential counterarguments from the community.

Some argue that SAE feature consistency is fundamentally unachievable, viewing features merely as a useful, pragmatic decomposition[44]. This view is bolstered by findings that multiple, incompatible mechanistic explanations can coexist for the same model behavior [36, 31], questioning the existence of a single, canonical feature set. While perfect universality for every feature on real data is indeed challenging, our work demonstrates that *high levels of consistency are attainable* with appropriate methods and evaluation (e.g., TopK SAEs achieving PW-MCC ≈ 0.80; Sections 5.2). A pragmatic decomposition gains significant scientific utility when its components are demonstrably stable. The focus, therefore, should be on understanding, maximizing, and characterizing this stability.

Another perspective is that *sufficiently good* interpretability can be achieved without demanding perfect feature consistency, and an excessive focus on stability might stifle exploratory research[30, 16]. Initial exploration using single-run features certainly has value. However, for claims aspiring to scientific robustness—such as those underpinning causality [35, 20], safety verification [1, 21], or canonical understanding [38, 39]—sufficiently good stability must be quantitatively defined and verified. The current degree of instability in many applications is often underestimated [44]. We advocate for establishing measurable baselines for consistency to add rigor for cumulative progress.

It is also suggested that the pursuit of low-level feature stability might divert from the arguably more important goal of understanding higher-level conceptual abstractions or circuits [40]. We contend, however, that reliable higher-level understanding requires robust lower-level foundations. If the fundamental feature vocabulary is ill-defined or shifts between runs, any circuits or mechanisms built upon them become inherently suspect and difficult to validate [41]. Stable features are important, dependable anchors for trustworthy compositional analyses and the mapping of learned circuits.

Finally, there's a view that the field will organically converge on more consistent SAE methods without explicit mandates for consistency benchmarking. While scientific fields do self-correct over time, feature instability remains a significant and often under-reported issue [15], even in widely-used methods. An active, concerted push for prioritizing consistency, supported by standardized metrics (like PW-MCC) and benchmarks, can substantially accelerate progress, guiding the community towards more scientifically sound practices.

B Additional Related Work

B.1 Sparse Autoencoders for Mechanistic Interpretability

This section provides further context on the specific SAE architectures evaluated in our work, complementing the broader discussion of SAEs in Section 2.

SAEs aim to learn overcomplete dictionaries that can decompose high-dimensional neural network activations into sparser, potentially more interpretable feature representations. The core principle involves training an autoencoder to reconstruct an input activation $\mathbf{x}$ while simultaneously encouraging the latent representation $f(\mathbf{x})$ to be sparse. Various SAE architectures implement different mechanisms to achieve this sparsity objective. The key architectures employed in our experiments are described below.

Standard SAE. The architecture we refer to as *Standard SAE* is an L1-penalized ReLU SAE that incorporates several contemporary training practices aimed at improving stability and reducing the incidence of inactive (dead) features [6, 33]. A distinguishing characteristic of this variant is the application of the L1 penalty to feature activations $f(\mathbf{x})$ after they have been scaled by the L2 norm of their corresponding decoder dictionary vectors: $\lambda_1 \sum_j |f_j(\mathbf{x})| \cdot \|\mathbf{a}_j\|_2$, where $\mathbf{a}_j$ is the j-th column of the decoder matrix. Unlike some earlier L1 SAEs that explicitly constrain decoder column norms to unity during optimization, this approach omits such a constraint, integrating the decoder norm directly into the sparsity term. We use Adam optimization and gradient clipping.

TopK SAE. TopK SAEs [18] enforce sparsity structurally, rather than through a continuous penalty term. For each input, only the k features with the highest pre-activation values (typically after a

ReLU non-linearity) are selected to be active, while all other feature activations are set to zero. The integer k directly determines the L0 norm of the feature activation vector. This design obviates the need for tuning an L1 coefficient but introduces k as a crucial hyperparameter. We do not incorporate additional auxiliary loss terms designed to prevent feature death in this work.

BatchTopK SAE. The BatchTopK SAE architecture [7] adapts the TopK principle by enforcing the k-sparsity constraint on average across a batch of inputs, rather than strictly on a per-sample basis. This is achieved by learning a global activation threshold that is dynamically adjusted during training (using an Exponential Moving Average of feature pre-activations) to ensure that, averaged over a batch, approximately k features are active per input sample. This allows for greater variability in per-sample sparsity while maintaining a target average L0 norm.

Gated SAE. Gated SAEs [46] are designed to decouple the decision of whether a feature activates from the magnitude of that activation. They employ two distinct pathways for processing the input: a *gate* pathway, which produces values (near 0 or 1 via an L1 or similar sparsity penalty on the gate outputs) that determine if each feature should be active, and a *magnitude* pathway, which computes the strength of each feature if it is gated on. The final feature activation is then the element-wise product of the outputs from these two pathways. The rationale behind this design is to allow features to activate with strong magnitudes when relevant, without these magnitudes being directly suppressed by the primary sparsity-inducing penalty, as that penalty is instead applied to the gating mechanism.

P-Anneal SAE. This SAE variant [27] modifies L1-penalized SAEs by employing a dynamic sparsity penalty based on an L_p^p-norm. In this approach, the exponent p in the sparsity term $\lambda_s \|f(\mathbf{x})\|_{p_s}^{p_s}$ is annealed during the training process. Training typically commences with $p_s = 1$ (equivalent to L1 minimization, which offers a convex optimization landscape) and p_s is progressively decreased towards a target value $p_{end} < 1$ (e.g., $p_{end} = 0.2$ in the original work). This annealing schedule aims to first guide the optimization towards a good region using the L1 penalty, and then gradually shift towards a non-convex objective that more closely approximates L0 sparsity, potentially yielding sparser solutions. To ensure the effective strength of the sparsity penalty remains relatively consistent as p_s changes, the scaling coefficient λ_s is also adaptively adjusted during training based on statistics derived from recent batches of feature activations.

JumpReLU SAE. JumpReLU SAEs [47] employ a JumpReLU activation function which uses per-feature learnable thresholds, θ_j. For an input language model activation $x \in \mathbb{R}^n$, the encoder computes pre-activations $\pi_j(x) = (W_{\text{enc}}x + b_{\text{enc}})_j$ for each feature j. The feature activation $f_j(x)$ is then given by $f_j(x) = \text{JumpReLU}_{\theta_j}(\pi_j(x)) = \pi_j(x)H(\pi_j(x) - \theta_j)$, where H is the Heaviside step function and $\theta_j > 0$ is the learned threshold for feature j. Sparsity in the feature representation $f(x)$ is encouraged by an L_0 penalty on the feature activations: for instance, using a loss term like $\lambda(\|f(x)\|_0/L_0^{\text{target}} - 1)^2$ to drive the average number of active features towards a predefined target L_0^{target}. The non-differentiable nature of both the JumpReLU (with respect to θ_j) and the L0 penalty is handled during training using Straight-Through Estimators. This architecture allows the model to learn distinct activation sensitivities for different features, as each θ_j can be optimized independently.

Matryoshka BatchTopK SAE. Matryoshka SAEs [8] introduce a hierarchical structure to the learned dictionary. In this paradigm, multiple, nested dictionaries of progressively increasing sizes are trained simultaneously within a single model. Features are ordered or grouped, and the training objective is designed to encourage more general or broadly important features to be learned by the smaller, inner dictionaries (analogous to the inner dolls in a Matryoshka set). More numerous or specialized features are then captured by the larger, outer dictionaries. This hierarchical approach aims to learn features at multiple levels of granularity and can offer computational efficiencies at inference time if a smaller, inner dictionary provides sufficient representational power for a given task. The Matryoshka BatchTopK variant evaluated in our study combines this hierarchical dictionary organization with the BatchTopK mechanism for selecting active features.

Each architecture comes with its own set of hyperparameters, computational considerations, and characteristic effects on the learned feature space.

B.2 Extended Discussion on Dictionary Learning Identifiability

The feature consistency challenges observed in SAEs can be understood through the theoretical lens of dictionary learning identifiability. Dictionary learning identifiability addresses a fundamental question: under what conditions can we guarantee that a learning algorithm will recover the true underlying dictionary (or an equivalent version up to permutation and scaling) from observed data? This question directly parallels our inquiry into when SAEs can consistently learn the same features across different initializations. Dictionary learning can be formalized as the problem of finding a dictionary matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ and a sparse coefficient matrix $\mathbf{F} \in \mathbb{R}^{d \times n}$ such that $\mathbf{X} \approx \mathbf{AF}$, where $\mathbf{X} \in \mathbb{R}^{m \times n}$ represents observed data. In the overcomplete setting ($d > m$), which is most relevant to SAEs, the problem becomes particularly challenging because infinitely many solutions can potentially fit the data equally well.

Several lines of theoretical work establish conditions under which overcomplete dictionary recovery is possible. For example, as discussed in our paper, the Spark condition introduced by [12] states that when spark($\mathbf{A}$) is sufficiently large relative to the sparsity level, unique recovery becomes possible. Specifically, a unique sparse representation is guaranteed when spark($\mathbf{A}$) $> 2k$, where k is the sparsity level. Building on this foundation, [48] recently established more comprehensive identifiability results for overcomplete dictionary learning. Their work introduces conditions under which global identifiability holds, showing that the identifiability of dictionaries depends on both the structure of the dictionary itself and the generative mechanism for coefficient vectors. A key insight from [48] is that traditional dictionary identifiability frameworks rely on verifying two conditions: (1) coefficients are sufficiently diverse to span the full space of possibilities, and (2) dictionaries satisfy appropriate structural conditions such as the Spark condition. When both conditions hold, the dictionary can be uniquely determined up to permutation and scaling—exactly the type of consistency we seek in SAE features. The Restricted Isometry Property (RIP) provides another important set of conditions. A dictionary matrix $\mathbf{A}$ satisfies RIP of order k with constant δ_k if $(1 - \delta_k)|\mathbf{x}|_2^2 \leq |\mathbf{Ax}|_2^2 \leq (1 + \delta_k)|\mathbf{x}|_2^2$ for all k-sparse vectors $\mathbf{x}$. When RIP holds with sufficiently small δ_k, consistent recovery becomes possible even in overcomplete regimes [3]. In our discussion, we opted to use the Spark condition due to its clearer connections to the training objectives of SAEs and the simplicity for implementing the condition in the learning algorithm.

We also draw inspiration from the independent component analysis (ICA) literature for defining our evaluation metric MCC, as dictionary learning has deep connections to ICA, particularly in its overcomplete form. ICA assumes that observed signals are linear mixtures of statistically independent source signals and aims to recover both the mixing matrix and the source signals [24], an objective shared with dictionary learning for finding the dictionary and sparse coefficient matrix [50].

C Formal Definitions of Feature Consistency

This appendix provides more formal definitions for the concepts of $\mathcal{T}$-Feature Consistency and Strong Feature Consistency, briefly introduced in Section 3. These definitions help to precisely articulate what it means for two sets of learned features to be considered *equivalent.*

Definition 2 ($\mathcal{T}$-Feature Consistency)**.** *Let $\mathbf{A}$ and $\mathbf{A}'$ be two dictionaries (matrices whose columns are feature vectors), each containing d features, learned from the same dataset $\mathcal{D}$ using the same algorithm and hyperparameters but with different random initializations. These dictionaries are said to be $\mathcal{T}$-**consistent** ($\mathbf{A} \sim_{\mathcal{T}} \mathbf{A}'$) if there exists a permutation $\sigma \in S_d$ (the set of all permutations of $\{1, \ldots, d\}$) and a specified transformation $\mathcal{T}$ such that for all feature indices $i \in \{1, \ldots, d\}$:*

$$\mathbf{a}_i = \mathcal{T}(\mathbf{a}'_{\sigma(i)}),$$

where $\mathbf{a}_j^{(k)}$ denotes the j-th feature vector (column) of dictionary $\mathbf{A}^{(k)}$.

The transformation $\mathcal{T}$ can take various forms. For instance, in some contexts, $\mathcal{T}$ might represent a more complex function if features are not considered atomic or have internal structure. However, for dictionary learning in sparse autoencoders, where features are typically represented by individual vectors (dictionary atoms), a more specific and common notion of equivalence is based on permutation and scaling.

Definition 3 (Strong Feature Consistency)**.** *The dictionaries $\mathbf{A}$ and $\mathbf{A}'$ from Definition 2 are said to exhibit **Strong Feature Consistency** if the transformation $\mathcal{T}$ corresponds to an individual, per-*

feature scaling. That is, there exists a permutation $\sigma \in S_d$ and a set of non-zero scaling factors $\lambda_1, \ldots, \lambda_d \in \mathbb{R} \setminus \{0\}$ such that for all $i \in \{1, \ldots, d\}$:

$$\mathbf{a}_i = \lambda_i \mathbf{a}'_{\sigma(i)}.$$

This definition implies that each feature learned in one run has a one-to-one correspondent in the other run that points in the same (or exactly opposite, if $\lambda_i < 0$) direction, differing only in magnitude. If feature vectors are constrained to have unit ℓ_2-norm (either by explicit normalization during training or as a post-processing step before comparison), the scaling factors λ_i would effectively become ± 1. The Mean Correlation Coefficient (MCC) metrics used in this paper—namely PW-MCC and GT-MCC—are designed to measure this Strong Feature Consistency. The use of cosine similarity $|\langle \mathbf{u}, \mathbf{v} \rangle| / (\|\mathbf{u}\|_2 \|\mathbf{v}\|_2)$ inherently accounts for differences in magnitude (norm), and the absolute value handles the sign ambiguity (features pointing in opposite directions are still considered perfectly correlated in direction).

We prioritize Strong Feature Consistency—alignment up to permutation and scaling—as a foundational and empirically tractable starting point. This notion directly connects to identifiability results in dictionary learning and allows for straightforward quantification using metrics like MCC. While other, broader notions of consistency, such as functional equivalence (where features achieve similar outcomes despite different dictionary vectors) or subspace alignment, are undoubtedly important and represent valuable avenues for future research, establishing robust vector-level consistency is a critical first step.

D How the Round-Trip Condition Guarantees Spark in SAEs

This section proves that the round-trip property directly implies the spark condition for TopK SAEs. The argument is purely algebraic.

D.1 Setting and Notation

Throughout this section, we fix a sparsity level $k \geq 1$ and denote by $\Sigma_k := \{\mathbf{f} \in \mathbb{R}^d : \|\mathbf{f}\|_0 \leq k\}$ the set of k-sparse vectors.

SAE Components. Let $\mathbf{A} \in \mathbb{R}^{m \times d}$ be the decoder (dictionary) learned by a TopK SAE, and let $\mathbf{E} : \mathbb{R}^m \to \Sigma_k$ be its deterministic TopK encoder. The encoder $\mathbf{E}$ selects the k largest magnitude inner products $|\langle \mathbf{a}_j, \mathbf{x} \rangle|$ and returns their signed values, with ties broken lexicographically to ensure $\mathbf{E}$ is a deterministic function.

Round-Trip Property. We assume that the encoder-decoder pair $(\mathbf{E}, \mathbf{A})$ satisfies the round-trip property if:

$$\forall \mathbf{f} \in \Sigma_k, \quad \mathbf{E}(\mathbf{A}\mathbf{f}) = \mathbf{f}. \tag{1}$$

k-Injectivity and Spark. A dictionary $\mathbf{A}$ is *k-injective* if $\forall \mathbf{f}, \mathbf{f}' \in \Sigma_k$, $\mathbf{A}\mathbf{f} = \mathbf{A}\mathbf{f}'$ implies $\mathbf{f} = \mathbf{f}'$. This is equivalent to the spark condition $\mathrm{spark}(\mathbf{A}) > 2k$, where $\mathrm{spark}(\mathbf{A})$ is the size of the smallest linearly dependent column set of $\mathbf{A}$ [12]:

$$\mathbf{A} \text{ is } k\text{-injective} \quad \iff \quad \mathrm{spark}(\mathbf{A}) > 2k.$$

D.2 Key Decomposition Lemma

The following lemma provides the crucial technical tool for our main result:

Lemma 1 (Two-Vector Decomposition)**.** *Let $\mathbf{h} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ with $\|\mathbf{h}\|_0 \leq 2k$. There exist distinct vectors $\mathbf{f}, \mathbf{f}' \in \Sigma_k$ with disjoint supports such that $\mathbf{h} = \mathbf{f} - \mathbf{f}'$. Consequently, if $\mathbf{A}\mathbf{h} = \mathbf{0}$, then $\mathbf{A}\mathbf{f} = \mathbf{A}\mathbf{f}'$.*

Proof. Let $S = \mathrm{supp}(\mathbf{h})$, so $|S| \leq 2k$. We can partition S into two disjoint sets S_1, S_2 such that $|S_1|, |S_2| \leq k$. This is always possible since $|S| \leq 2k$.

Define vectors $\mathbf{f}, \mathbf{f}' \in \mathbb{R}^d$ by:

$$\mathbf{f}_j := \begin{cases} \mathbf{h}_j & \text{if } j \in S_1 \\ 0 & \text{if } j \notin S_1 \end{cases}, \quad \mathbf{f}'_j := \begin{cases} -\mathbf{h}_j & \text{if } j \in S_2 \\ 0 & \text{if } j \notin S_2 \end{cases}.$$

By construction:

1. $\mathbf{f}, \mathbf{f}' \in \Sigma_k$ since $\text{supp}(\mathbf{f}) \subseteq S_1$ and $\text{supp}(\mathbf{f}') \subseteq S_2$ with $|S_1|, |S_2| \leq k$
2. $\text{supp}(\mathbf{f}) \cap \text{supp}(\mathbf{f}') = S_1 \cap S_2 = \emptyset$ (disjoint supports)
3. $\mathbf{f} \neq \mathbf{f}'$ since $\mathbf{h} \neq \mathbf{0}$ implies at least one of S_1, S_2 is non-empty
4. $\mathbf{h} = \mathbf{f} - \mathbf{f}'$ by direct verification on each coordinate

If $\mathbf{Ah} = \mathbf{0}$, then $\mathbf{A}(\mathbf{f} - \mathbf{f}') = \mathbf{0}$, which immediately gives $\mathbf{Af} = \mathbf{Af}'$. □

D.3 Main Result

Theorem 2 (Round-Trip Implies k-Injectivity)**.** *If the round-trip property* (1) *holds, then* $\text{spark}(\mathbf{A}) > 2k$*. Equivalently,* $\mathbf{A}$ *is* k*-injective.*

Proof. We proceed by contradiction. Assume $\text{spark}(\mathbf{A}) \leq 2k$. Then there exists a non-zero vector $\mathbf{h} \in \mathbb{R}^d$ with $\|\mathbf{h}\|_0 \leq 2k$ such that $\mathbf{Ah} = \mathbf{0}$.

By Lemma 1, we can decompose $\mathbf{h} = \mathbf{f} - \mathbf{f}'$ where $\mathbf{f}, \mathbf{f}' \in \Sigma_k$ are distinct vectors with disjoint supports, and $\mathbf{Af} = \mathbf{Af}'$ (since $\mathbf{Ah} = \mathbf{0}$).

Let $\mathbf{x} := \mathbf{Af} = \mathbf{Af}'$ denote the common image. Since $\mathbf{E}$ is a deterministic function, $\mathbf{E}(\mathbf{x})$ is uniquely determined. However, applying the round-trip property (1) to both representations:

$$\mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{Af}) = \mathbf{f} \quad \text{and} \quad \mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{Af}') = \mathbf{f}'.$$

This implies $\mathbf{f} = \mathbf{f}'$, contradicting the fact that $\mathbf{f}$ and $\mathbf{f}'$ are distinct.

Therefore, our assumption $\text{spark}(\mathbf{A}) \leq 2k$ must be false, which means $\text{spark}(\mathbf{A}) > 2k$. □

D.4 Application to TopK SAE

Corollary 1 (Spark Condition for TopK SAEs)**.** *Let* $X \subset \mathbb{R}^m$ *be a training dataset and suppose a TopK SAE with learned dictionary* $\mathbf{A} \in \mathbb{R}^{m \times d}$ *and deterministic Top-*k *encoder* $\mathbf{E} : \mathbb{R}^m \to \Sigma_k$ *achieves:*

1. ***Zero reconstruction error****:* $\mathbf{AE}(\mathbf{x}) = \mathbf{x}$ *for all* $\mathbf{x} \in X$
2. ***Reachability****: For every* k*-sparse vector* $\mathbf{f} \in \Sigma_k$*, there exists* $\mathbf{x} \in X$ *such that* $\mathbf{E}(\mathbf{x}) = \mathbf{f}$

Then the learned dictionary $\mathbf{A}$ *is* k*-injective:* $\text{spark}(\mathbf{A}) > 2k$*.*

Proof. Let $\mathbf{f} \in \Sigma_k$ be arbitrary. By reachability, there exists $\mathbf{x} \in X$ such that $\mathbf{E}(\mathbf{x}) = \mathbf{f}$.

Zero reconstruction error gives:

$$\mathbf{Af} = \mathbf{AE}(\mathbf{x}) = \mathbf{x}. \tag{2}$$

Applying the encoder to both sides:

$$\mathbf{E}(\mathbf{Af}) = \mathbf{E}(\mathbf{x}) = \mathbf{f}. \tag{3}$$

Since this holds for arbitrary $\mathbf{f} \in \Sigma_k$, the round-trip property is satisfied. Theorem 2 then immediately implies $\text{spark}(\mathbf{A}) > 2k$. □

D.5 Implications for Feature Consistency

Theoretical Guarantee. Corollary 1 establishes that when TopK SAEs achieve zero reconstruction error and reachability on their training data, the learned dictionary satisfies the spark condition. This provides a theoretical foundation for feature consistency in TopK SAEs based purely on operational training outcomes.

Practical Interpretation. The two conditions serve complementary but distinct roles in ensuring the spark guarantee. Zero reconstruction error prevents encoder collapse by forcing the encoder to distinguish among all training inputs—if multiple inputs collapsed to the same sparse code, some could not be perfectly reconstructed by the decoder. Reachability ensures comprehensive coverage by guaranteeing that every possible k-sparse code appears in the training dataset X. This coverage requirement enables the theoretical result to apply universally across all codes in Σ_k. In practice, exact reachability cannot be verified on finite datasets, so this condition is approximated through diverse training data that provides broad coverage across the feature space.

Connection to Identifiability. Our result shows that learning a decoding dictionary satisfying the spark condition does not require access to the ground truth, which significantly broadens the scope of traditional identifiability discussions. When the ground truth $\mathbf{A}_{\text{gt}}$ is assumed to satisfy the spark condition, our result naturally recovers standard identifiability guarantees. More importantly, even in the absence of any ground truth information, our result ensures strong consistency across all learned dictionaries. This is especially valuable for practitioners who cannot reliably make assumptions about the data generation process.

E Supplementary Analysis of SAEs trained on Synthetic Data

E.1 Detailed Analysis of Learning Regimes

This section provides additional analysis of the redundant and compressive learning regimes introduced in the main text. In all experiments, we maintain a constant TopK sparsity parameter $k = 8, n = 5e4$.

E.1.1 Redundant Regime Analysis

In the redundant regime, we set the ground truth dimension $d_{gt} = 80$ and the SAE dictionary size $d_{sae} = 160$, creating a setting where the SAE has twice the capacity needed to represent all ground truth features ($d_{sae} > d_{gt}$).

Figures 9 and 10 illustrate a key characteristic of the redundant regime: the SAE learns multiple good representations for each ground truth feature. Figure 9 shows substantial overlap between features with high similarity to ground truth and features with high similarity across runs. Figure 10 demonstrates that cosine similarity to ground truth decays very slowly, remaining high well beyond the ground truth dimension.

This redundancy creates a fundamental selection ambiguity problem when comparing features across different SAE initializations. The large pool of near-equally good candidates for the top-d_{gt} matches makes the optimal feature matching determined by the Hungarian algorithm highly sensitive to small variations between runs. This dictionary instability persists despite high average MCC scores with the ground truth. In essence, while the SAE learns good representations of the underlying features (as evidenced by high GT-MCC), it lacks a consistent way to select among multiple valid alternatives, leading to different features being selected across runs and consequently comparatively lower pairwise consistency.

As shown in Figure 11, the Mean GT-MCC (Maximum Correlation Coefficient) reaches high values, indicating strong recovery of ground truth features. However, Figure 12 shows that the PW-MCC across different SAE initializations is lower, reflecting the challenge of consistent feature selection despite good ground truth recovery.

Intersection Ratio The Intersection Ratio measures the consistency of feature selection across different training runs by quantifying how often the same learned features that match well to ground

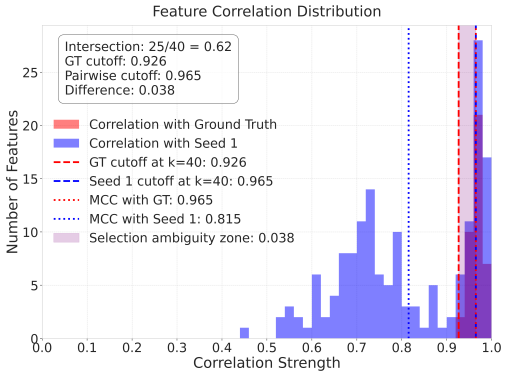

Figure 9: Feature Correlation Distribution ($d_{gt} = 40, d_{sae} = 160, k = 8$). Compares similarities of Run 0 features to ground truth (red) and Run 1 features (blue). The substantial overlap in high-similarity regions (purple) demonstrates ambiguity where multiple SAE features are good matches to both ground truth and features learned in other runs, creating selection ambiguity despite high feature quality.

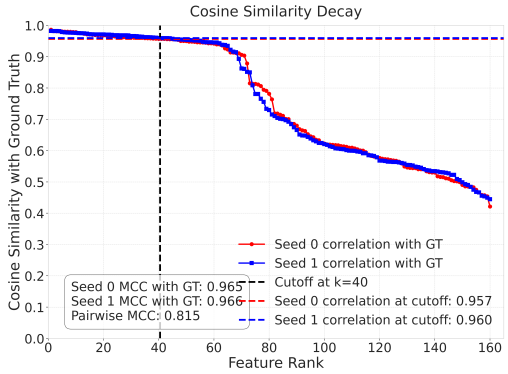

Figure 10: Cosine Similarity Decay with Ground Truth ($d_{gt} = 40, d_{sae} = 160, k = 8$). Features are ranked by ground truth similarity for Run 0 (red) and Run 1 (blue). Similarity decays very slowly, remaining high well past rank $d_{gt} = 40$, indicating that the SAE learns multiple good representations for each ground truth feature, creating a selection challenge when comparing across runs.

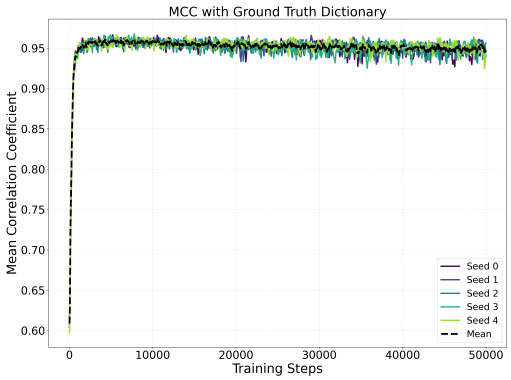

Figure 11: Redundant Regime: TopK SAE Mean GT-MCC (across 5 seeds) vs. Training Steps ($d_{gt} = 80, d_{sae} = 160, k = 8$). The GT-MCC reaches high values, indicating strong recovery of ground truth features.

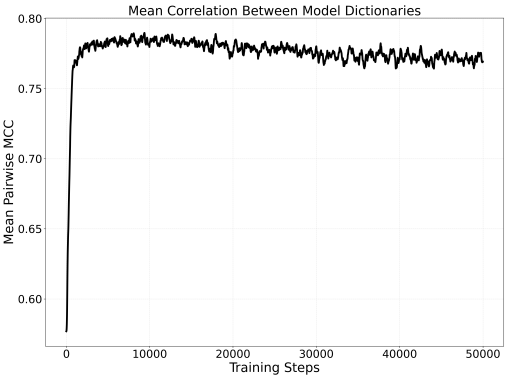

Figure 12: Redundant Regime: TopK SAE Mean PW-MCC (across 5 seeds) vs. Training Steps ($d_{gt} = 80, d_{sae} = 160, k = 8$). The PW-MCC reaches lower values than GT-MCC, reflecting the challenge of feature consistency across different SAE initializations due to selection ambiguity.

truth also match well between different runs. For a pair of runs (run_1, run_2), we first find $M_{1\to GT}$, the optimal matching between dictionaries A_1 and A_{gt}, and define $I_{1\to GT} = \{i \mid \exists j, (i,j) \in M_{1\to GT}\}$ as the set of feature indices from Run 1 that successfully match to ground truth features. Next, we find $M_{1\to 2}$, the optimal matching between dictionaries A_1 and A_2, and let $I'_{1\to 2}$ be the set of the top d_{gt} feature indices from Run 1 that participate in the highest-scoring similarity pairs $(i,k) \in M_{1\to 2}$ (if $d_{sae} < d_{gt}$, we use all d_{sae} indices). The intersection ratio is then computed as $R_{1,2} = \frac{|I_{1\to GT} \cap I'_{1\to 2}|}{\min(d_{gt}, d_{sae})}$. We report $\mathbb{E}_{i\neq j}[R_{i,j}]$, the expected intersection ratio estimated by averaging over multiple distinct pairs of runs, where higher values indicate reduced selection ambiguity and more consistent feature discovery across training runs.

We find that the Intersection Ratio increases over training steps, indicating that SAEs converge toward more consistent feature selection, though perfect consistency remains challenging due to the fundamental ambiguity introduced by excess capacity.

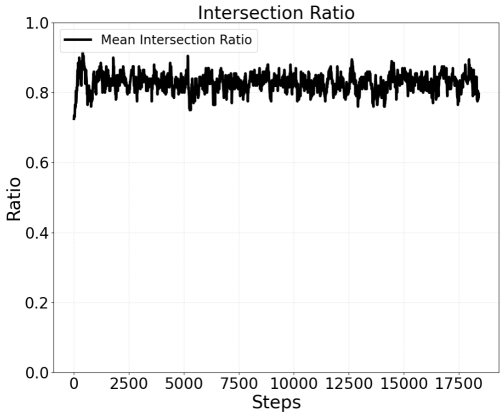

Figure 13: Redundant Regime: TopK SAE Mean Intersection Ratio (across 5 seeds) vs. Training Steps ($d_{gt} = 80, d_{sae} = 160, k = 8$). The Intersection Ratio measures the consistency of feature selection indices across different SAE initializations, with higher values indicating more stable feature recovery.

E.1.2 Compressive Regime Analysis

In the compressive regime, we set the ground truth dimension $d_{gt} = 800$ and the SAE dictionary size $d_{sae} = 80$, creating a setting where the SAE has only one-tenth of the capacity needed to represent all ground truth features ($d_{sae} < d_{gt}$). This capacity limitation forces the SAE to prioritize which features to learn.

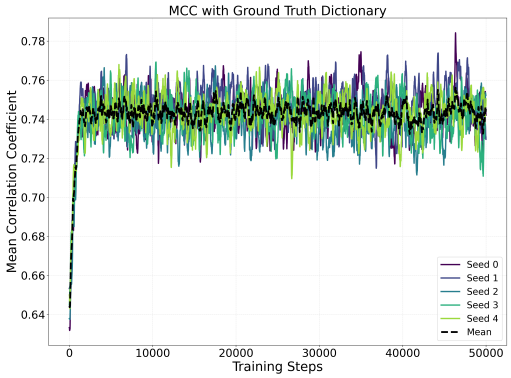

Figure 14: Compressive Regime: TopK SAE Mean GT-MCC (across 5 seeds) vs. Training Steps ($d_{gt} = 800, d_{sae} = 80, k = 8$). The GT-MCC reaches lower values compared to the redundant regime, reflecting the fundamental capacity limitation that prevents complete recovery of all ground truth features.

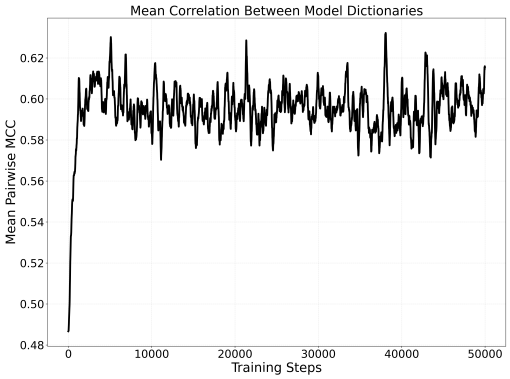

Figure 15: Compressive Regime: TopK SAE Mean PW-MCC (across 5 seeds) vs. Training Steps ($d_{gt} = 800, d_{sae} = 80, k = 8$). The PW-MCC values are also lower compared to the redundant regime, indicating lower overall recovery quality.

Figures 14 and 15 illustrate the key characteristics of the compressive regime. Unlike the redundant regime, where feature selection ambiguity was the primary challenge, the compressive regime faces a fundamental capacity limitation.

Figure 14 shows that both the PW-MCC and the Mean GT-MCC reach significantly lower values compared to the redundant regime (Figure 11), reflecting the inability to recover all ground truth features with limited capacity.

E.2 Uniform Partitioning Experiments

We analyze how partitioning ground truth features into uniform clusters affects SAE learning dynamics. In these experiments, we maintain a constant total number of ground truth features ($d_{gt} = 800$) while varying the number of clusters they are organized into. The dimension of each cluster is

d_{gt}/num_clusters, resulting in fewer features per cluster as the number of clusters increases. The complete hyperparameter settings for these experiments are presented in Table 3.

Table 3: Hyperparameters for Uniform Clustering Experiments

Parameter	Value
TopK sparsity parameter (k)	8
Activation dimension	20
Dictionary size (d_{sae})	80
Training examples	100,000
Training steps	20,000
Learning rate	0.04
Learning rate decay factor	0.1
Learning rate decay steps	[20,000]
Warmup steps	1,000
Minimum learning rate	1e-05
L1 coefficient	0.1
Batch size	4,096
Number of seeds	3
Cluster distribution	uniform
Ground truth dimension (d_{gt})	800
Number of clusters	varies (1, 10, 50, 100)
Cluster dimensions	d_{gt}/num_clusters

Table 4: Effect of uniform partitioning of ground truth vectors. As the number of clusters increases while keeping the total number of ground truth vectors constant, mean PW-MCC shows a weak but consistent increase (0.621 to 0.665), while ground truth MCC remains stable around 0.74.

Clusters	Mean GT-MCC	Std GT-MCC	Mean PW-MCC
1	0.742	0.006	0.621
10	0.747	0.002	0.634
50	0.740	0.002	0.651
100	0.746	0.007	0.665

Table 4 presents the results of our uniform partitioning experiments. The key finding is that as we impose additional structure by increasing the number of clusters from 1 to 100 while keeping the total number of ground truth vectors constant, the mean PW-MCC between SAEs increases consistently from 0.621 to 0.665. This suggests that clustered organization of features promotes more consistent feature learning across different SAE initializations. Interestingly, the mean ground truth MCC remains stable around 0.74 across all cluster configurations, indicating that the overall recovery quality of ground truth features is not significantly affected by the clustering structure.

E.3 Feature Recovery Across Zipf Distributions

This section provides additional experimental results showing how feature recoverability in SAEs varies across different Zipf distributions. We analyzed distributions with exponents $\alpha \in \{1.0, 1.1, 1.5, 2.0\}$ to understand how the skewness of ground truth feature cluster probability affects SAE learning dynamics and feature reproducibility. For all experiments in this section, we set the ground truth dimension $d_{gt} = 800$, SAE dictionary size $d_{sae} = 80$, and TopK sparsity parameter $k = 8$, placing us in the compressive regime where the SAE must learn a compressed representation of the underlying features. The ground truth features are organized into 10 clusters, with 80 ground truth features per cluster, where the probability of each cluster appearing in the data follows a Zipf distribution with varying exponents α. Table 5 provides the complete hyperparameter settings used in these experiments.

E.3.1 Zipf Distribution with $\alpha = 1.0$

For $\alpha = 1.0$, we observe a moderate skew in cluster probabilities with a corresponding power-law allocation of dictionary features. As shown in Figure 16 (left), the SAE capacity allocation via

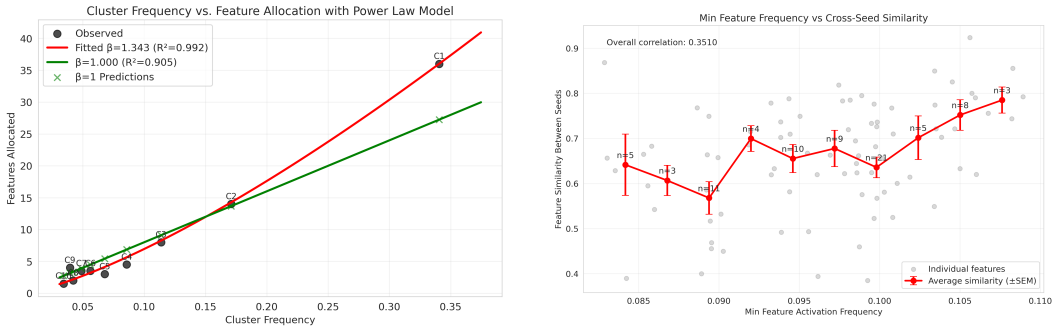

Figure 16: Left: Capacity allocation model for Zipf distribution with $\alpha = 1.0$, showing how SAE features are allocated to clusters based on cluster probability. The red curve shows the fitted power law model, following $D_i \propto p_i^{\beta}$ where $\beta \approx 1.343$. Right: Feature similarity between independently trained SAEs as a function of minimum feature activation frequency, with bucketed averages (red) showing a positive trend between activation frequency and feature reproducibility.

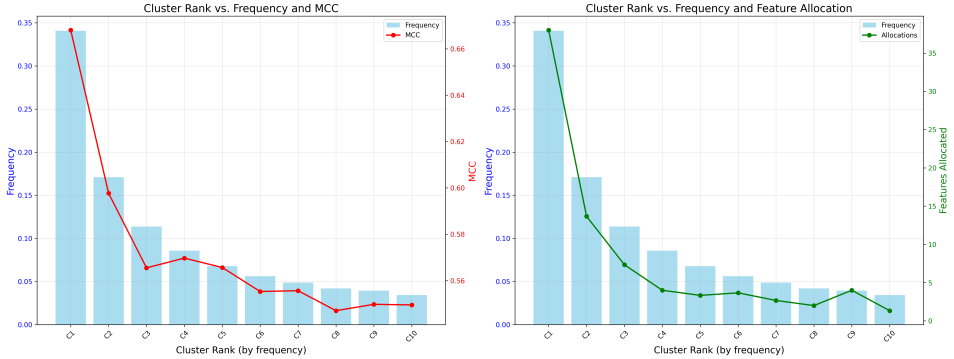

Figure 17: Cluster metrics for Zipf distribution with $\alpha = 1.0$. Left: Cluster rank vs. probability (blue bars) and MCC scores (red line), showing how feature recovery quality varies with cluster probability. The MCC scores demonstrate a positive correlation with cluster probability, with lower-ranked (more probable) clusters achieving better feature recovery. Right: Cluster rank vs. probability (blue bars) and feature allocation (green line), demonstrating how the model allocates dictionary features based on cluster probability, with more frequent clusters receiving proportionally more features.

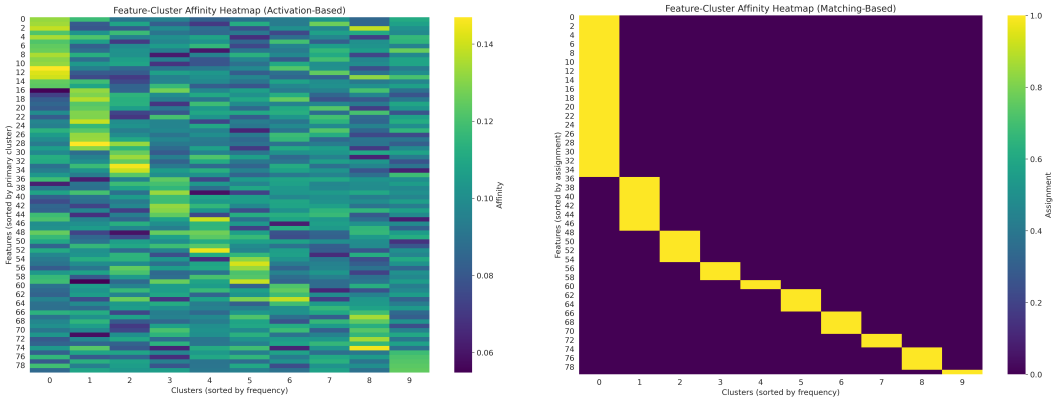

Figure 18: Feature-cluster relationships for Zipf distribution with $\alpha = 1.0$. Left: Activation-based affinity heatmap showing how features (y-axis, sorted by primary cluster) are activated by different clusters (x-axis, sorted by probability). Brighter colors indicate stronger activation, showing that more frequent cluster features activate more features. Right: Matching-based affinity heatmap showing global assignment of features to clusters using Hungarian matching, with features on y-axis and clusters on x-axis

Table 5: Hyperparameters for Zipf Distribution Experiments

Parameter	Value
TopK sparsity parameter (k)	8
Activation dimension	20
Dictionary size (d_{sae})	80
Training examples	100,000
Training steps	20,000
Learning rate	0.04
Learning rate decay factor	0.1
Learning rate decay steps	[20,000]
Warmup steps	1,000
Minimum learning rate	1e-05
L1 coefficient	0.1
Batch size	4,096
Number of models trained (seeds)	3
Number of clusters	10
Cluster dimensions	80 per cluster
Distribution	zipf
Zipf skew (α)	varies (1.0, 1.1, 1.5, 2.0)

Hungarian matching follows $D_i \propto p_i^\beta$ with $\beta \approx 1.343$, where D_i is the number of SAE features allocated to cluster i and p_i is the cluster's probability in the data distribution. This superlinear relationship indicates that more probable clusters receive disproportionately more dictionary features. Figure 16 (right) demonstrates that features with higher activation frequencies also show greater reproducibility across different SAE initializations, indicating that frequently activated features are more robustly learned.

The cluster metrics in Figure 17 further support this relationship, showing that more probable clusters achieve better feature recovery quality as measured by MCC scores. The feature-cluster activation-based affinity map in Figure 18 reveal that while features tend to specialize for specific clusters, there is some activation overlap, particularly among the most probable clusters. The activation-based affinity (left) displays more diffuse relationships between features and clusters and exhibits less frequency skew compared to the discrete one-to-one assignments established through Hungarian matching (right), which more strongly favors high-probability clusters.

E.3.2 Zipf Distribution with $\alpha = 1.1$

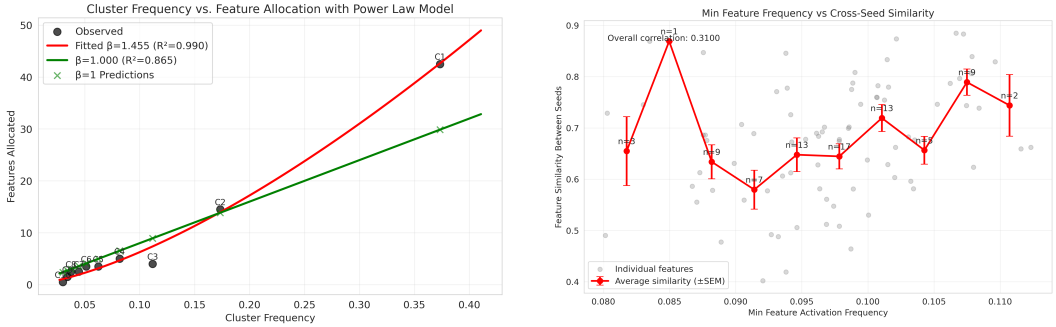

Figure 19: Left: Capacity allocation model for Zipf distribution with $\alpha = 1.1$, showing how SAE features are allocated to clusters based on cluster probability. Red curve shows fitted power law model with $D_i \propto p_i^\beta$ where $\beta \approx 1.455$. Right: Feature similarity between independently trained SAEs as a function of minimum feature activation frequency, with bucketed averages (red) showing a positive trend between activation frequency and feature reproducibility.

Increasing the exponent to $\alpha = 1.1$ creates a slightly more skewed distribution. Comparing Figure 19 with Figure 16, we observe a steeper power law curve in the capacity allocation model ($D_i \propto p_i^\beta$ with $\beta \approx 1.455$), indicating that high-probability clusters now receive an even larger share of the

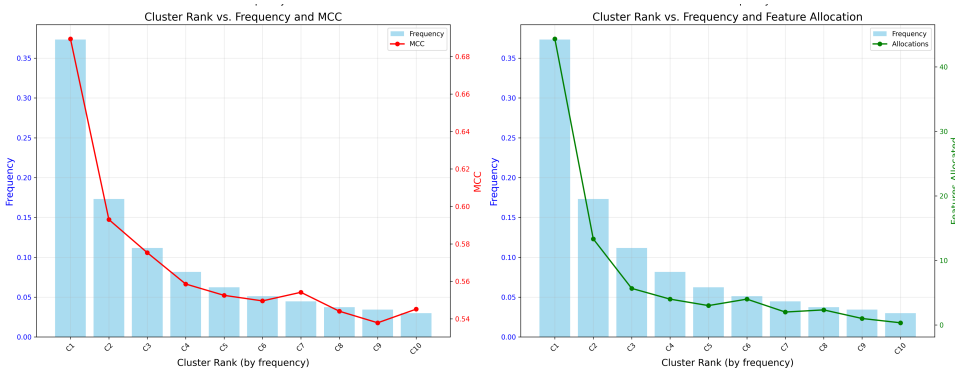

Figure 20: Cluster metrics for Zipf distribution with $\alpha = 1.1$. Left: Cluster rank vs. probability (blue bars) and MCC scores (red line), showing a steeper decline in feature recovery quality for less probable clusters compared to $\alpha = 1.0$. Right: Cluster rank vs. probability (blue bars) and feature allocation (green line), demonstrating more skewed allocation of dictionary features toward high-probability clusters.

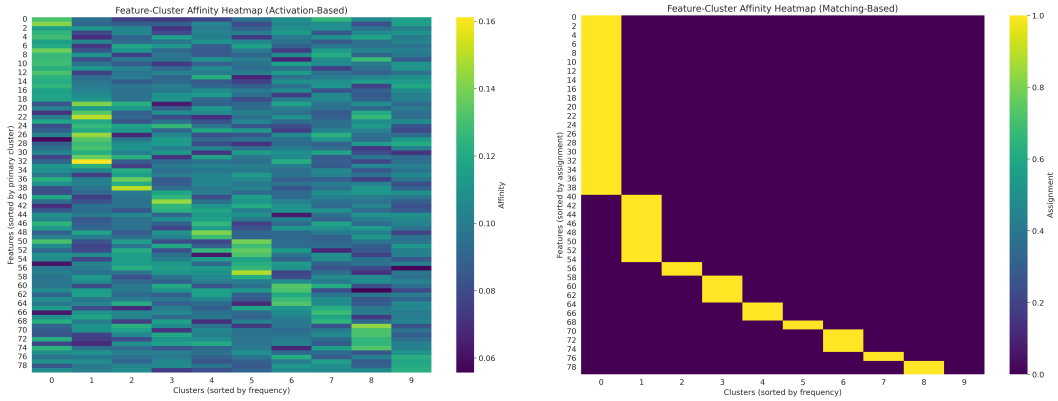

Figure 21: Feature-cluster relationships for Zipf distribution with $\alpha = 1.1$. Left: Activation-based affinity heatmap showing stronger feature-to-cluster specialization. compared to $\alpha = 1.0$. Right: Matching-based affinity heatmap showing increased skew in feature assignments, with high-probability clusters receiving disproportionately more feature allocations compared to $\alpha = 1.0$.

dictionary capacity. We see a similar positive trend between feature activation frequency and feature reproducibility, with no noticeable increase in the trend.

Figure 20 reveals a more dramatic drop-off in feature recoverability for lower-probability clusters, and Figure 21 shows increased skew in feature allocation, with high-probability clusters receiving proportionally more features than in the $\alpha = 1.0$ case. This skew is less pronounced when measured through activation-based affinity (left) compared to the more extreme allocation in the Hungarian matching assignments (right).

E.3.3 Zipf Distribution with $\alpha = 1.5$

With $\alpha = 1.5$, we observe a highly skewed distribution where a small number of probable clusters dominate. Figure 22 shows that the capacity allocation follows a power law with $D_i \propto p_i^\beta$ where $\beta \approx 1.35$, with the majority of dictionary features allocated to the highest-probability clusters. The feature similarity plot shows a positive relationship between the feature activation frequency and feature similarity between independently trained SAEs, but the effect is not much stronger than the $\alpha = 1$ or $\alpha = 1.1$ case.

The cluster metrics in Figure 23 show MCC scores dropping precipitously beyond the most probable clusters. The feature-cluster affinity maps in Figure 24 also show highly specialized features based on frequency but the skew when measured through activation-based affinity is less pronounced as compared to Hungarian matching assignments.

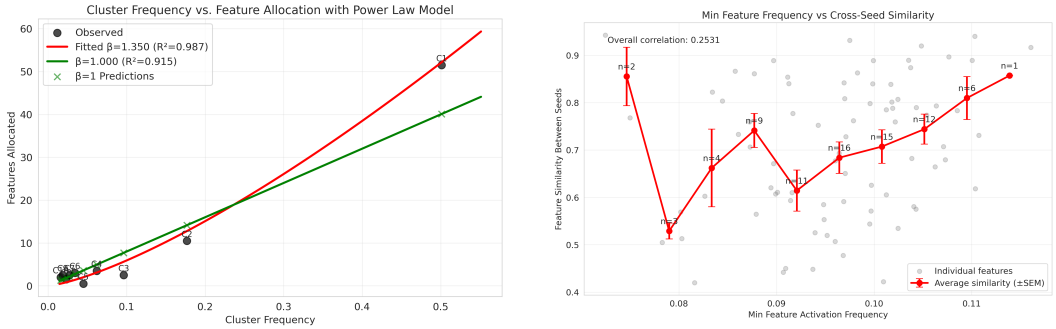

Figure 22: Left: Capacity allocation model for Zipf distribution with $\alpha = 1.5$, showing significantly more skewed allocation of SAE features to clusters. Red curve shows fitted power law model with $D_i \propto p_i^{\beta}$ where $\beta \approx 1.35$. Right: Feature similarity between independently trained SAEs as a function of minimum feature activation frequency showing a weak positive trend.

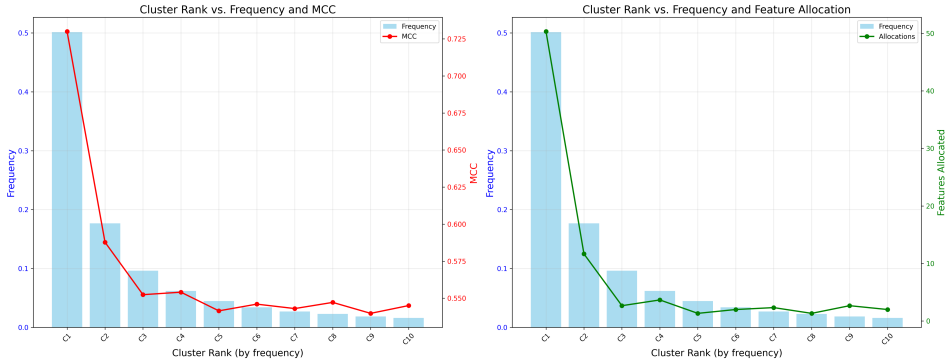

Figure 23: Cluster metrics for Zipf distribution with $\alpha = 1.5$. Left: Cluster rank vs. probability (blue bars) and MCC scores (red line), showing a sharp threshold effect where feature recovery quality drops dramatically beyond the highest-probability clusters. Right: Cluster rank vs. probability (blue bars) and feature allocation (green line), demonstrating highly concentrated allocation of dictionary features to the most probable clusters.

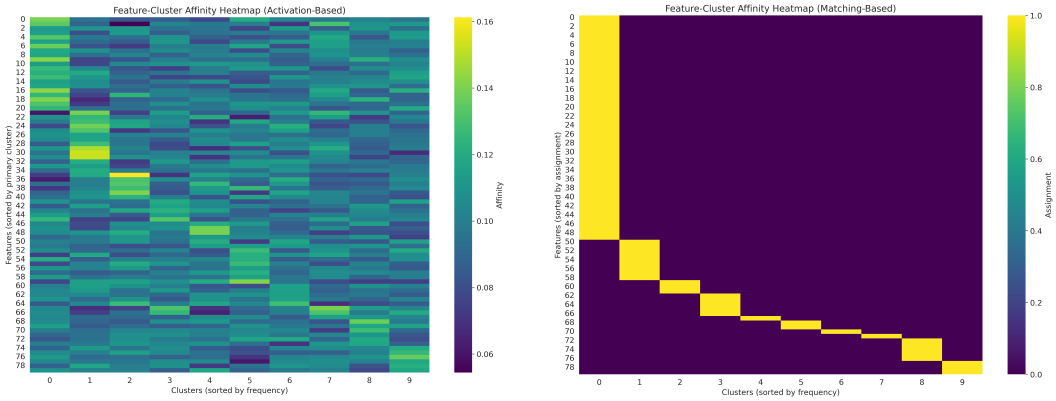

Figure 24: Feature-cluster relationships for Zipf distribution with $\alpha = 1.5$. Left: Activation-based affinity heatmap showing high feature specialization with minimal cross-activation. Right: Matching-based affinity heatmap showing strong one-to-one mapping for high-probability clusters but poor assignment for low-probability clusters.

E.3.4 Zipf Distribution with $\alpha = 2.0$

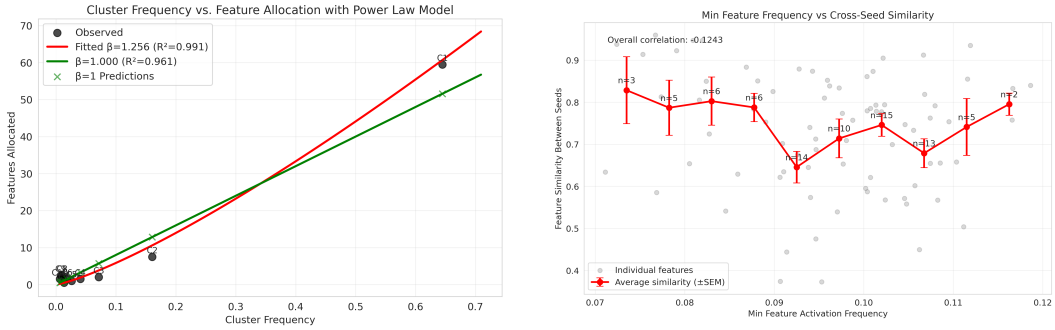

Figure 25: Left: Capacity allocation model for Zipf distribution with $\alpha = 2.0$, showing extreme concentration of SAE features to the highest-probability clusters. Red curve shows fitted power law model with $D_i \propto p_i^\beta$ where $\beta \approx 1.256$. Right: Feature similarity between independently trained SAEs as a function of minimum feature activation frequency showing a flat to weak positive trend.

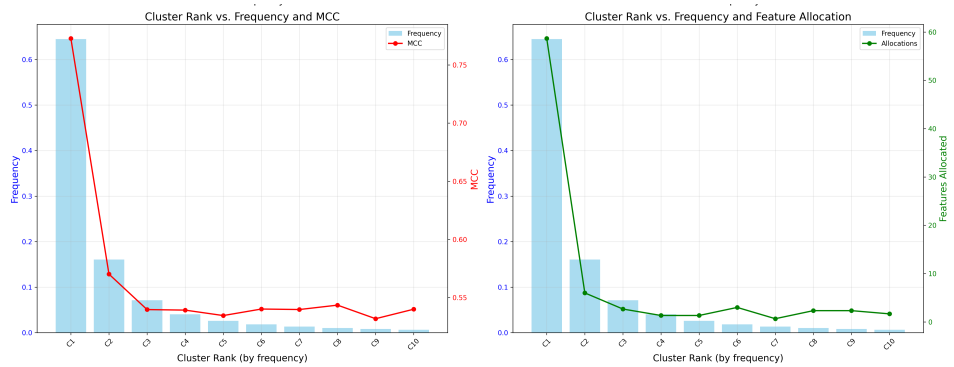

Figure 26: Cluster metrics for Zipf distribution with $\alpha = 2.0$. Left: Cluster rank vs. probability (blue bars) and MCC scores (red line), showing that the very highest-probability clusters achieve good feature recovery. Right: Cluster rank vs. probability (blue bars) and feature allocation (green line), demonstrating that dictionary features are almost exclusively allocated to the top clusters, with negligible capacity for the long tail.

At $\alpha = 2.0$, we observe an extremely skewed distribution where a handful of clusters dominate the probability distribution. Figure 25 shows that dictionary capacity is allocated according to a power law with $D_i \propto p_i^\beta$ where $\beta \approx 1.256$, with capacity concentrated in the highest-probability clusters. The feature similarity plot shows a flat to weak positive relationship between the feature activation frequency and feature similarity between independently trained SAEs.

The cluster metrics in Figure 26 confirm that only the very top clusters achieve meaningful feature recovery, with the vast majority of clusters poorly represented.

E.4 Analysis of Dictionary Size Effects in Two-Phase Distributions

To better approximate real language data distributions, we developed a two-phase model that combines different power laws. This section examines how dictionary size affects feature learning in this more realistic distribution. We enhanced our model organism with a two-phase feature cluster frequency distribution combining a Mandelbrot-Zipf function ($g(r; s_1, q) = (r+q)^{-s_1}$, $s_1 = 1.05$, $q = 5.0$) for common concepts ($r < 40,000$) and a steeper power law ($g(r; s_2) \propto r^{-s_2}$, $s_2 = 30.0$) for the long tail. For all experiments in this section, we set the ground truth dimension $d_{gt} = 400000$, with 50000 clusters (each cluster represented by 8 ground truth features on average), and maintain a constant TopK sparsity parameter $k = 8$, while varying the SAE dictionary size $d_{sae} \in \{80, 160, 1000, 10000\}$ to analyze the impact of SAE capacity on feature learning. Table 6 summarizes the hyperparameters used in these experiments.

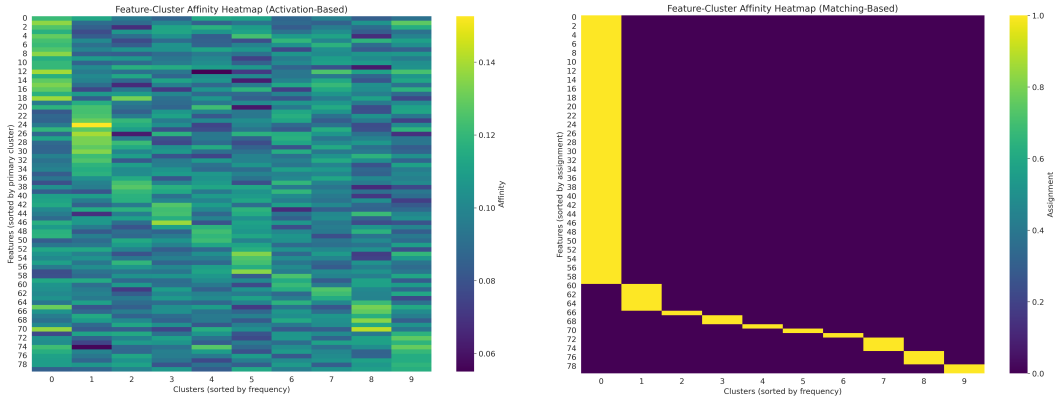

Figure 27: Feature-cluster relationships for Zipf distribution with $\alpha = 2.0$. Left: Activation-based affinity heatmap showing specialization to high-probability clusters. Right: Matching-based affinity heatmap showing strong assignment for only the highest-probability clusters, with the majority of clusters receiving minimal or no feature representation.

Table 6: Hyperparameters for Two-Phase Distribution Experiments

Parameter	**Value**
TopK sparsity parameter (k)	8
Activation dimension	20
Dictionary size (d_{sae})	varies (80, 160, 1000, 10000)
Training examples	100,000
Training steps	20,000
Learning rate	0.04
Learning rate decay factor	0.1
Learning rate decay steps	[20,000]
Warmup steps	1,000
Minimum learning rate	1e-05
L1 coefficient	0.1
Batch size	4,096
Number of models trained	3
Number of clusters	50,000
Cluster dimensions	8 per cluster
First power law exponent (s_1)	1.05
Second power law exponent (s_2)	30.0
Transition rank	40,000
q parameter	5.0

Our two-phase distribution depicted in Figure 28 combines a shallow power law for frequent tokens ($s_1 = 1.05$ until rank 40,000) with a much steeper power law ($s_2 = 30.0$) for the long tail. This creates a realistic approximation of natural language distributions, which exhibit similar two-phase characteristics (Figure 4).

Figures 29 through 32 demonstrate how dictionary size affects feature reproducibility across the activation frequency spectrum. Several key trends emerge:

1. With small dictionary sizes (80-160 features), we observe only a weak relationship between activation frequency and feature reproducibility. Only the most frequent clusters show consistent reproducibility, indicating severe capacity limitations where the dictionary should prioritize only the dominant clusters.

2. As dictionary size increases to 1000 features, the relationship between activation frequency and reproducibility becomes more pronounced. A wider range of moderately frequent features begins to show improved reproducibility, as the increased capacity allows the model to represent more clusters with sufficient local redundancy.

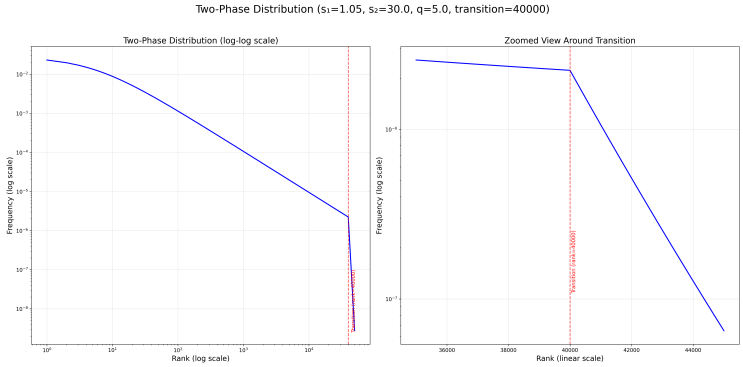

Figure 28: Two-phase cluster probability distribution used to approximate real language data. The distribution follows a Mandelbrot-Zipf pattern ($s_1 = 1.05$) until rank 40,000, then transitions to a steeper power law ($s_2 = 30.0$) capturing the long tail characteristics of natural language.

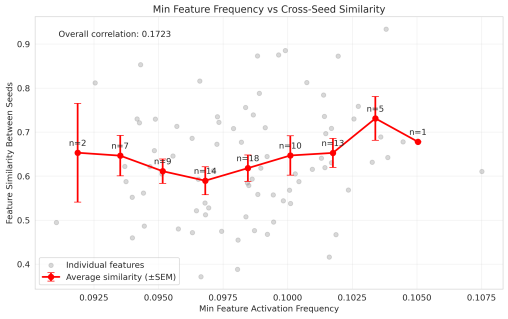

Figure 29: Two-phase model with dictionary size 80. Feature reproducibility shows a weak positive relationship with activation frequency.

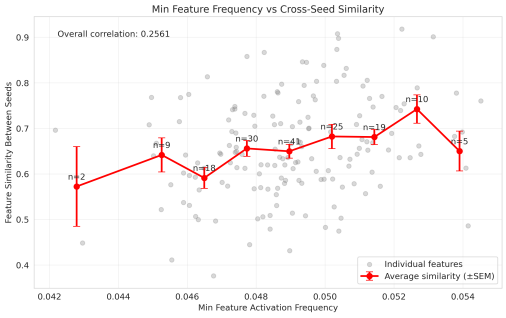

Figure 30: Two-phase model with dictionary size 160. The relationship between activation frequency and feature reproducibility remains weak but becomes slightly more pronounced compared to dictionary size 80.

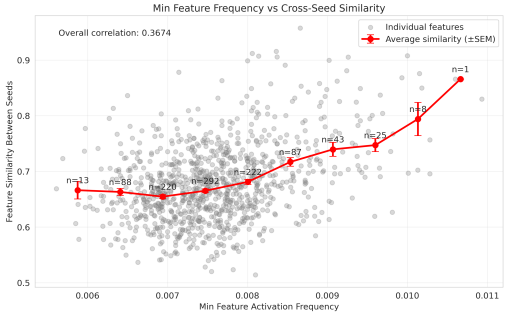

Figure 31: Two-phase model with dictionary size 1000. Feature reproducibility shows a moderately strong positive correlation with activation frequency especially at higher activation frequencies. Increased model capacity creates sufficient local redundancy for high probability clusters.

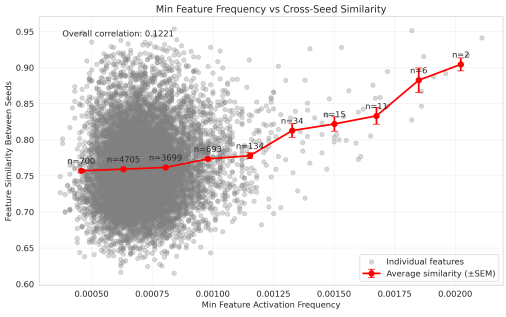

Figure 32: Two-phase model with dictionary size 10000. With substantially increased capacity, feature reproducibility exhibits a strong positive correlation with activation frequency across a wide frequency range. Increased model capacity creates sufficient local redundancy for high probability clusters.

3. At dictionary size 10000, we observe a positive relationship between activation frequency and reproducibility across a wide frequency range. The substantial increase in capacity creates local redundancy for many more clusters, enabling consistent representation of features.

These results demonstrate that dictionary size relative to the distribution characteristics plays a crucial role in determining which features can be consistently learned. In particular, the local redundancy—defined as the ratio of dictionary size to the effective number of clusters above a certain frequency threshold—determines the model's ability to learn reproducible features across different frequency bands. Rather than a threshold effect, we observe a continuum where the strength of correlation between feature frequency and reproducibility increases as more clusters benefit from sufficient local redundancy.

These results demonstrate that dictionary size relative to the distribution characteristics plays an important role in determining which features can be consistently learned. In particular, local redundancy—defined as the ratio of dictionary capacity allocated to a cluster relative to the cluster dimension—determines the model's ability to learn reproducible features across different frequency bands. With larger dictionaries, more clusters achieve sufficient local redundancy, leading to a continuum where the strength of correlation between feature frequency and reproducibility increases as dictionary capacity expands.

F Effect of Misspecifying Encoder Sparsity Parameter

In practical applications of TopK SAEs, practitioners must choose the sparsity parameter k without knowledge of the true underlying sparsity s of the data-generating process. This section investigates how misspecification of k relative to the optimal value s affects dictionary recovery quality and training stability.

Consider a conceptual cluster i within the data with ground-truth dictionary $A_i^\star \in \mathbb{R}^{m \times d_{\text{gt}}}$ having unit-norm columns. Let $s < d_{\text{gt}}$ be the true sparsity of coefficient vectors $s^\star$ for signals $x = A_i^\star s^\star$ from this cluster. The SAE uses a TopK encoder with parameter k. We define the **sparsity ratio** as $\rho := k/s$ and focus on understanding the asymmetric effects of under-sparsity ($\rho < 1$) versus over-sparsity ($\rho > 1$) on feature learning.

We conduct experiments in the matched regime where $d_{\text{sae}} = d_{\text{gt}}$, generating synthetic data by first sampling the ground-truth dictionary $\mathbf{A}_{\text{gt}} \in \mathbb{R}^{m \times d_{\text{gt}}}$ from a standard normal distribution with unit-norm columns. For each data point, we randomly select exactly s features and set their activations to independent Gaussian samples, yielding coefficient vector $\mathbf{f}_{\text{gt}}(\mathbf{x}) \in \mathbb{R}^{d_{\text{gt}}}$ with s non-zero entries, then construct data points as $\mathbf{x} = \mathbf{A}_{\text{gt}}\mathbf{f}_{\text{gt}}(\mathbf{x})$. Our experimental configuration uses input dimension $m = 8$, dictionary sizes $d_{\text{gt}} = d_{\text{sae}} = 40$, true sparsity $s = 8$, training samples $N = 50,000$, TopK parameter range $k \in \{2, 4, 6, 8, 10, 12, 14, 16\}$, and 5 independent runs per configuration. We evaluate dictionary recovery using GT-MCC, which measures correlation between learned dictionary $\mathbf{A}$ and ground-truth $\mathbf{A}_{\text{gt}}$ using optimal permutation matching, computed over the final 100 training steps and averaged to reduce noise.

Figure 33 demonstrates that GT-MCC peaks precisely at $k = s = 8$, confirming that matching encoder sparsity to true data sparsity yields optimal dictionary recovery. The performance curve exhibits notable asymmetry around the optimal point: under-sparsity ($k < s$) causes sharp degradation in GT-MCC as k decreases below s, while over-sparsity ($k > s$) leads to more gradual decline as k increases beyond s. This asymmetry reflects fundamental differences in how sparsity misspecification affects the optimization process.

When $k < s$, the SAE lacks sufficient representational capacity to capture all active features in the ground-truth generating process. This constraint forces the model to either merge multiple ground-truth features into single learned features or systematically ignore some ground-truth features, both resulting in poor dictionary recovery. The sharp performance degradation occurs because the model cannot represent the true complexity of the data-generating process, leading to fundamental representational limitations that cannot be overcome through better optimization.

Conversely, when $k > s$, the SAE has excess representational capacity that allows recovery of all true features but may also lead to learning near-duplicate features representing similar directions, increased sensitivity to initialization, and selection ambiguity among competing feature representations. While

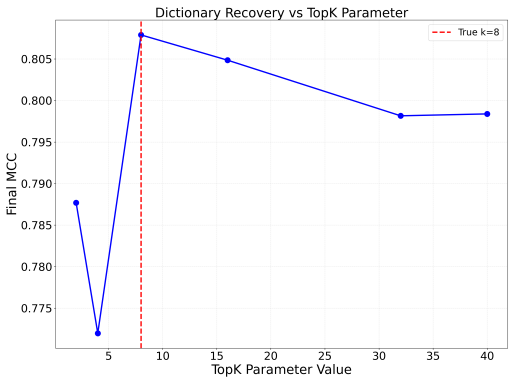

Figure 33: Effect of Activation Sparsity k in TopK SAE in the Matched Regime ($d_{\text{gt}} = d_{\text{sae}} = 40$, true $s = 8$). We plot final GT-MCC (averaged over the last 100 steps) vs. k. Performance peaks at $k = s = 8$, with underestimating k being more harmful than overestimating it.

this reduces run-to-run consistency, the gradual performance decline suggests that over-sparsity is less immediately harmful than under-sparsity, as the model can still capture the essential structure of the data even if it learns redundant or unstable features.

These findings have important practical implications. Because practitioners never observe $A_i^\star$ in real applications, measuring PW-MCC across multiple training runs becomes an important diagnostic tool. As demonstrated earlier in this paper, PW-MCC correlates strongly with GT-MCC, especially in the matched regime, providing a practical proxy for dictionary quality when ground truth is unavailable. Our observations reveal that when k is too low, the SAE lacks sufficient representational capacity, leading to poor feature recovery. When k is too high, features may become unstable across runs due to selection ambiguity among near-duplicates, even when reconstruction loss appears acceptable. Therefore, sweeping over k values while monitoring PW-MCC provides a principled approach to approximate effective sparsity for a given dataset and SAE architecture.

For practitioners selecting appropriate sparsity parameters, these results suggest that when uncertain about true sparsity, a conservative approach favoring slight over-sparsity ($\rho \approx 1.2$–1.5) is preferable to under-sparsity, as performance degradation is more gradual and reversible. Practitioners should systematically vary k while monitoring PW-MCC to identify regions of high consistency. Warning signs include low PW-MCC coupled with acceptable reconstruction loss, which may indicate over-sparsity leading to feature instability, and poor reconstruction performance combined with high PW-MCC, which may indicate under-sparsity with consistent but incomplete feature recovery.

These observations raise important theoretical questions for future research. Can we precisely characterize how dictionary learning error scales with the degree of sparsity misspecification under different conditions, potentially leading to theoretical bounds on performance degradation? What is the exact relationship between mutual coherence, sparsity ratio, and feature stability? How do these effects manifest in real LLM representations, where underlying sparsity may vary across different data regimes and semantic contexts? Can we develop methods that adaptively determine k or accommodate varying sparsities within datasets to better capture heterogeneous features?

A theoretical analysis of these questions, combined with further empirical investigation, could substantially advance our understanding of SAE training dynamics and improve feature extraction for mechanistic interpretability.

G Supplementary Analysis of SAEs Trained on Language Model Activations

This appendix provides additional experimental details, visualizations, and quantitative results for SAEs trained on activations extracted from LLMs. The primary objective of these experiments is to empirically evaluate the feature consistency and learned characteristics of different SAE architectures when applied to complex, real-world data. In the absence of ground-truth features for LLM activations, feature consistency is primarily assessed by training multiple instances of each SAE configuration

(differing only by random initialization seeds) and subsequently quantifying the similarity between their learned dictionaries using PW-MCC.

G.1 Experimental Setup for Real Data Analysis

G.1.1 General Methodology and Data

SAEs of various architectures were trained using activations derived from LLMs processing text from the `monology/pile-uncopyrighted` dataset. For each specific SAE architecture and hyperparameter set, three independent training runs were conducted using distinct random seeds (`random_seeds = [42, 43, 44]`) to evaluate consistency. The learned dictionaries from pairs of these runs were compared by first finding an optimal one-to-one matching between their features using the Hungarian algorithm, and then calculating the cosine similarity for each matched pair. The average of these similarities constitutes the PW-MCC for the dictionary. Separately, we plot the individual feature similarity and its correlation with feature activation statistics.

All SAEs were trained on 500 million tokens from the source dataset. Common training parameters, consistent across these experiments unless otherwise specified, are detailed in Table 7. Feature activation frequency for a given SAE feature is defined as the proportion of input tokens (within a representative sample of the training data) on which that feature exhibits a non-zero activation. When comparing a matched pair of features from two independently trained SAEs (run 1 and run 2), their joint activation behavior is characterized by `min(freq_run1, freq_run2)`. This metric provides a conservative estimate of their shared activity level, as a feature pair representing a truly consistent underlying concept should ideally be active on a substantial and largely overlapping set of inputs.

Table 7: Common Training Parameters for SAEs on LLM Activations.

Parameter	Value
Dataset for Activations	`monology/pile-uncopyrighted`
Total Training Tokens	5×10^8
SAE Batch Size	2048
Warmup Steps	1000
Sparsity Warmup Steps	5000 (for L1-based and JumpReLU)
Learning Rate Decay Start	0.8 (of total training steps)
Number of Random Seeds per Configuration	3 (42, 43, 44)
Activation Normalization	Applied (before SAE input)
Autocast Data Type	`torch.bfloat16`

G.1.2 Configuration for Pythia-160M Experiments

The detailed visualizations and quantitative comparisons presented in Section G.2 pertain to SAEs trained on activations extracted from the `EleutherAI/pythia-160m-deduped` model. These SAEs were trained on the residual stream activations output by layer 8 of the LLM (`resid_post_layer_8`). Specific parameters for this experimental setup are provided in Table 8 and inspired by the sweep used in [26].

For the specific configurations analyzed from this Pythia-160M setup, the SAE architectures demonstrated varying levels of pairwise dictionary consistency as measured by PW-MCC. TopK SAEs achieved the highest overall performance with a PW-MCC of 0.8181 using a target k of 20. Batch TopK SAEs followed with a PW-MCC of 0.7656, also at target k of 20. Gated SAEs produced a PW-MCC of 0.7370 with a sparsity penalty of 0.06. Among the remaining architectures, Matryoshka Batch TopK SAEs achieved 0.6267 at target k of 20, P-Anneal SAEs reached 0.6113 with an initial sparsity penalty of 0.025, JumpReLU SAEs attained 0.4947 at target L_0 of 40, and Standard SAEs achieved 0.4739 with a sparsity penalty of 0.06. All optimal results were observed at training step 244,140. In terms of feature utilization, TopK and Gated SAEs consistently produced fewer *dead* features (features with very low or zero activation rates across the evaluation data) compared to the Standard and JumpReLU variants, with the L_0-constrained architectures (TopK, Batch TopK, JumpReLU, and Matryoshka Batch TopK) demonstrating more controlled sparsity patterns than their L_1-penalized counterparts.

Table 8: SAE Training Parameters for EleutherAI/pythia-160m-deduped (Layer 8).

Parameter	Value
LLM Model	`EleutherAI/pythia-160m-deduped`
Targeted Layer	8 (residual stream output)
LLM Activation Dimension (m)	768
LLM Batch Size (for activation generation)	32
LLM Context Length	1024
LLM Data Type	`torch.float32`
SAE Dictionary Width	2^{14} (16,384)
SAE Learning Rate	3×10^{-4}
Architectures Evaluated	Standard, TopK, BatchTopK, Gated, P-Anneal, JumpReLU, Matryoshka BatchTopK
Sparsity Penalties Sweep (for L1-based architectures):	
Standard	[0.012, 0.015, 0.02, 0.03, 0.04, 0.06]
P-Anneal (initial penalty)	[0.006, 0.008, 0.01, 0.015, 0.02, 0.025]
Gated	[0.012, 0.018, 0.024, 0.04, 0.06, 0.08]
Target L0s / k (for L0-based architectures):	[20, 40, 80, 160, 320, 640]

G.1.3 Configuration for Gemma-2-2B Experiments

Additional experiments were conducted using activations from the `google/gemma-2-2B` model [49], targeting the residual stream output of layer 12 (`resid_post_layer_12`). Table 9 documents the pertinent hyperparameters for these larger-scale runs.

For the Gemma-2-2B experimental configuration, the SAE architectures exhibited similar performance characteristics in terms of PW-MCC. TopK SAEs achieved the highest pairwise dictionary consistency with a PW-MCC of 0.7898 using a target k of 80. JumpReLU SAEs demonstrated competitive performance with a PW-MCC of 0.7405 at target k of 40, closely followed by Batch TopK SAEs with a PW-MCC of 0.7403 at target k of 80. Gated SAEs produced a PW-MCC of 0.7033 with a sparsity penalty of 0.04. The remaining architectures showed more modest performance levels, with Matryoshka Batch TopK SAEs achieving 0.5842 at target k of 80, P-Anneal SAEs reaching 0.5731 with an initial sparsity penalty of 0.025, and Standard SAEs attaining 0.5717 with a sparsity penalty of 0.03. The performance patterns observed in the larger Gemma-2-2B model generally maintained the relative ordering established in the Pythia-160M experiments, with L_0-constrained architectures consistently outperforming their L_1-penalized counterparts in terms of dictionary consistency metrics.

Table 9: SAE Training Parameters for google/gemma-2-2B (Layer 12).

Parameter	Value
LLM Model	`google/gemma-2-2B`
Targeted Layer	12 (residual stream output)
LLM Activation Dimension (m)	2304
LLM Batch Size (for activation generation)	4
LLM Context Length	1024
LLM Data Type	`torch.bfloat16`
SAE Dictionary Width	2^{16} (65,536)
SAE Learning Rate	3×10^{-4}
Random Seeds Used	42, 43
Architectures Evaluated	Standard, TopK, BatchTopK, Gated, P-Anneal, JumpReLU, Matryoshka BatchTopK Standard
Sparsity Penalties Sweep (for L1-based architectures):	
Standard	[0.012, 0.015, 0.02, 0.03, 0.04, 0.06]
P-Anneal (initial penalty)	[0.006, 0.008, 0.01, 0.015, 0.02, 0.025]
Gated	[0.012, 0.018, 0.024, 0.04, 0.06, 0.08]
Target L0s / k (for L0-based architectures):	[20, 40, 80, 160, 320, 640]

G.2 Detailed Analysis of SAEs Trained on Pythia-160M Layer 8 Activations

This section presents a comparative analysis of feature consistency and activation patterns for four different SAE architectures (TopK, Gated, Standard, JumpReLU) trained on Pythia-160M layer 8 activations.

G.2.1 Cross-Architectural Comparison of Feature Similarity and Activation Patterns

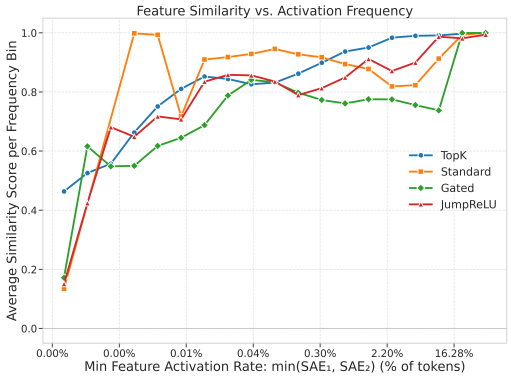

Figure 34: Average feature similarity (PW-MCC of matched individual features) versus activation rate for four SAE architectures. The activation rate is defined as `min(freq_run1, freq_run2)`, representing the minimum percentage of tokens activating the feature across two independent runs. Data is from SAEs trained on Pythia-160M layer 8 activations. A strong positive correlation is evident, indicating that features with higher shared activation rates are learned more consistently.

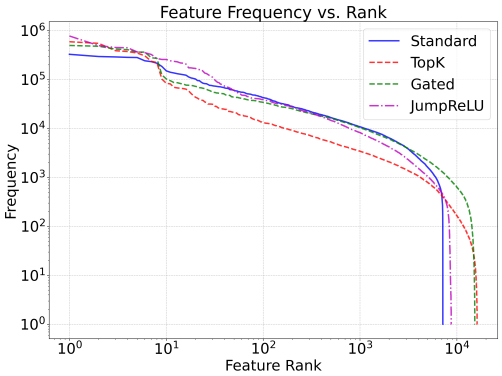

Figure 35: Log-log plot of feature activation frequency versus feature rank for four SAE architectures trained on Pythia-160M layer 8 activations. All architectures exhibit power-law-like distributions of feature usage, but with distinct slopes and differing prevalence of very low-frequency (potentially dead) features. Notably, TopK and Gated architectures tend to show fewer dead features.

A general trend observed across SAE architectures is the positive correlation between a feature's activation frequency and its consistency across independent training runs. Figure 34 illustrates this for Standard, TopK, Gated, and JumpReLU SAEs. The x-axis, `min(freq_run1, freq_run2)`, represents a condition for shared activity; features satisfying this with higher values (i.e., both are frequently active) exhibit higher pairwise similarity. This suggests that features corresponding to more prevalent patterns in the LLM's activations are more robustly learned. While this correlation is seen in all SAEs, the overall level of consistency varies, with TopK SAEs achieving the highest aggregate PW-MCC.

Concurrently, Figure 35 reveals that different SAE architectures induce distinct feature utilization profiles. While all exhibit power-law-like distributions for feature activation frequencies (when features are ranked by frequency), the slopes of these distributions and the number of extremely low-frequency or dead features vary. Architectures such as TopK and Gated SAEs tend to result in fewer dead features compared to Standard and JumpReLU SAEs, indicating potentially more efficient use of their learned dictionaries.

G.2.2 Standard SAE

The Standard SAE architecture, characterized by an L1 penalty on activations, achieved a relatively low overall PW-MCC of approximately 0.4739 in these experiments. A detailed breakdown of its feature similarity characteristics is provided in Figure 36. The density map (top-left) and the bucketed average similarity (bottom-left) illustrate that while higher minimum activation frequencies correlate with improved similarity, a significant portion of features exhibit low similarity. The histogram of similarity scores (top-right) is also peaked at zero, indicating several dead features. The scatter plot of paired frequencies (bottom-right) shows that features matched by the Hungarian algorithm (and thus contributing to the similarity score) often, but not always, possess similar activation frequencies across the two runs. Features lying closer to the diagonal (similar frequencies in both runs) and having higher joint frequencies (top-right of this sub-panel) tend to exhibit higher similarity. Figure 37

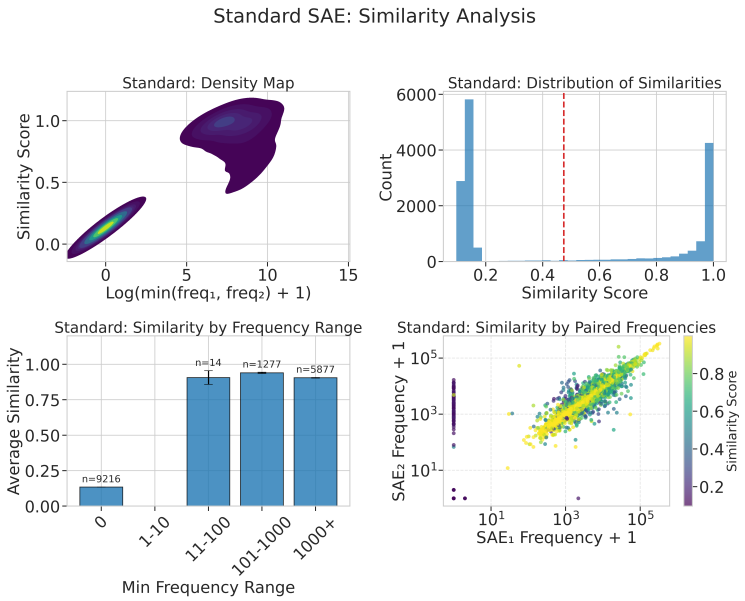

Figure 36: Feature similarity analysis for a Standard SAE (L1-penalized) trained on Pythia-160M layer 8 activations. The overall PW-MCC for this configuration was approximately 0.4739. Top left: Density map of pairwise feature similarity vs. log minimum activation frequency (`min(freq_run1, freq_run2)`). Top right: Histogram of pairwise feature similarity scores. Bottom left: Average similarity bucketed by log minimum frequency range. Bottom right: Scatter plot of feature activation frequencies from two runs (`freq_run1` vs. `freq_run2`), colored by their pairwise similarity score. Points along the diagonal represent features with similar activation frequencies in both runs.

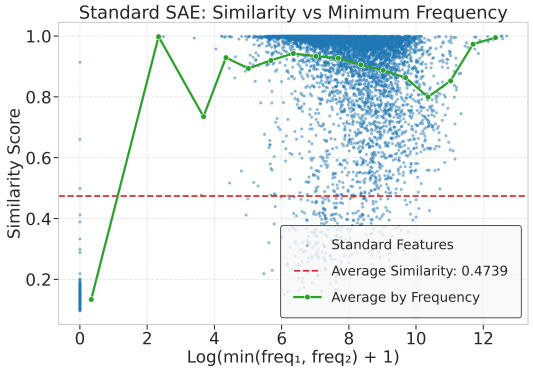

Figure 37: Average feature similarity versus `min(freq_run1, freq_run2)` for the Standard SAE. This plot highlights the positive trend: features with higher shared activation levels tend to exhibit greater pairwise similarity, though the overall consistency for this architecture is modest.

isolates and clearly depicts the positive relationship between shared activation frequency and pairwise similarity for this architecture.

G.2.3 TopK SAE

In contrast, the TopK SAE architecture demonstrated superior performance, achieving the highest overall PW-MCC of approximately 0.8188. The analyses in Figure 38 and Figure 39 illustrate this. The density map in Figure 38 (top-left) shows a strong concentration of features in the high-similarity, high-shared-frequency region. The histogram of similarity scores (top-right) is markedly skewed towards higher values compared to the Standard SAE, indicating more consistently learned features across the dictionary. The scatter plot of paired frequencies (bottom-right) again suggests that matched features tend to have similar activation rates especially at higher frequencies. Figure 39

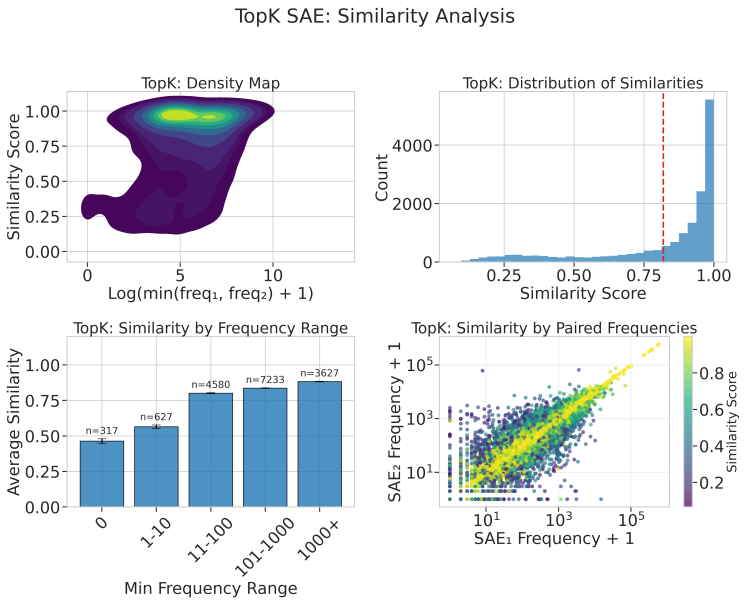

Figure 38: Feature similarity analysis for a TopK SAE trained on Pythia-160M layer 8 activations. This architecture achieved a high overall PW-MCC of approximately 0.8188. Panels are analogous to Figure 36.

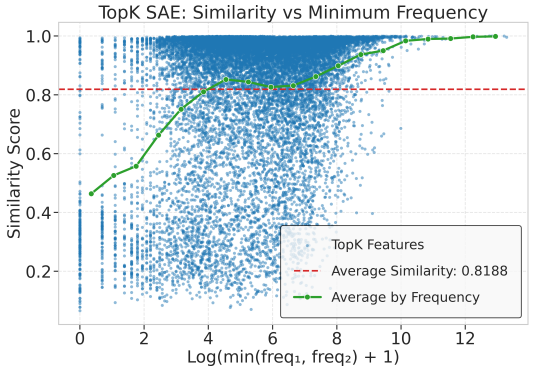

Figure 39: Average feature similarity versus `min(freq_run1, freq_run2)` for the TopK SAE. This architecture demonstrates both high overall similarity levels and a strong positive correlation between shared activation frequency and feature reproducibility.

clearly shows a robust positive correlation between shared activation frequency and high pairwise similarity, underscoring the stability of frequently used features learned by TopK SAEs.

G.2.4 Gated SAE

The Gated SAE architecture yielded an overall PW-MCC of approximately 0.7378, exhibiting good consistency, second only to TopK SAEs in this comparison. Figure 40 and Figure 41 detail its characteristics. The four-panel plot in Figure 40 displays trends consistent with other architectures regarding the relationship between shared frequency and similarity. The distribution of similarity scores (top-right) is more favorable than that of Standard SAEs, leaning towards higher consistency. The paired frequency scatter plot (bottom-right) also indicates that matched features tend to share similar activation levels. Figure 41 confirms the strong positive correlation between `min(freq_run1, freq_run2)` and feature similarity.

G.2.5 JumpReLU SAE

The JumpReLU SAE achieved an overall PW-MCC of approximately 0.4957, placing its aggregate consistency on par with Standard SAEs in these experiments. The results are presented in Figure 42

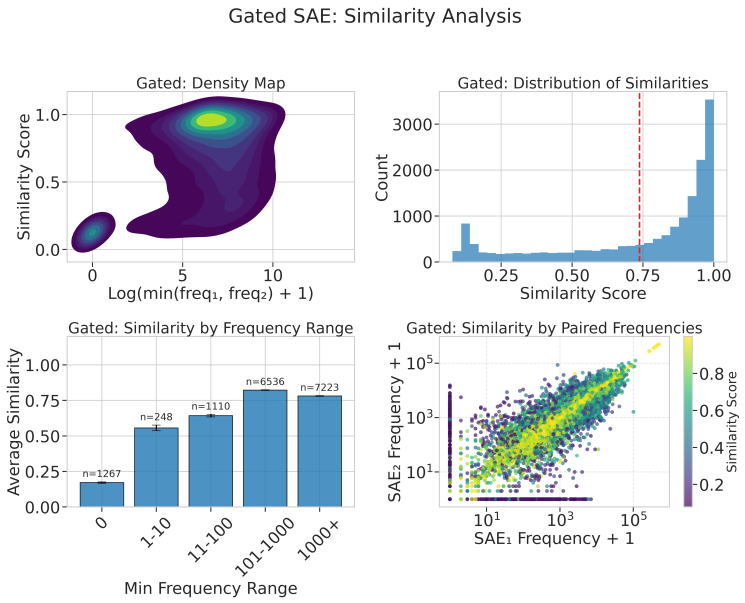

Figure 40: Feature similarity analysis for a Gated SAE trained on Pythia-160M layer 8 activations, with an overall PW-MCC of approximately 0.7378. Panels are analogous to Figure 36.

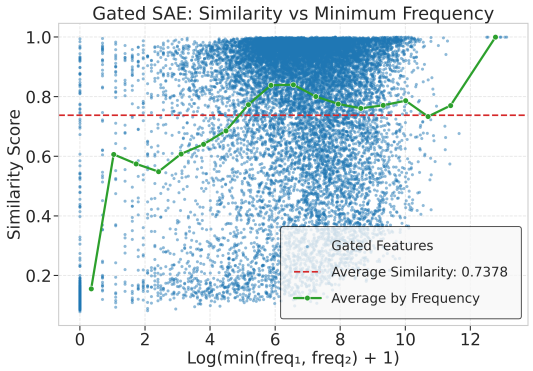

Figure 41: Average feature similarity versus `min(freq_run1, freq_run2)` for the Gated SAE. The overall dictionary PW-MCC for this configuration is 0.7378. A strong positive correlation is evident between shared activation frequency and individual feature similarity.

and Figure 43. The general trend of higher similarity for more frequently and jointly active features persists. The scatter plot of paired frequencies in Figure 42 (bottom-right) also suggests that features matched by the Hungarian algorithm tend to possess similar activation frequencies across runs. Figure 43 shows that JumpReLU also produces several dead features that affect the net consistency.

G.2.6 Summary of Real Data Findings

The empirical investigations on Pythia-160M activations consistently reveal a significant positive correlation between the activation frequency of learned features—particularly their shared activation level across independent runs, `min(freq_run1, freq_run2)`—and their inter-run similarity or consistency. This observation holds across diverse SAE architectures. The use of `min(freq_run1, freq_run2)` as a metric for joint activity is justified by the intuition that a feature representing a stable, underlying concept should be consistently activated by a similar set of inputs across different model initializations, thus implying that both individual frequencies should be high for a robust match. Furthermore, the scatter plots of paired feature frequencies (e.g., bottom-right panels in the four-panel figures) visually corroborate that features deemed "the same" by the Hungarian matching process

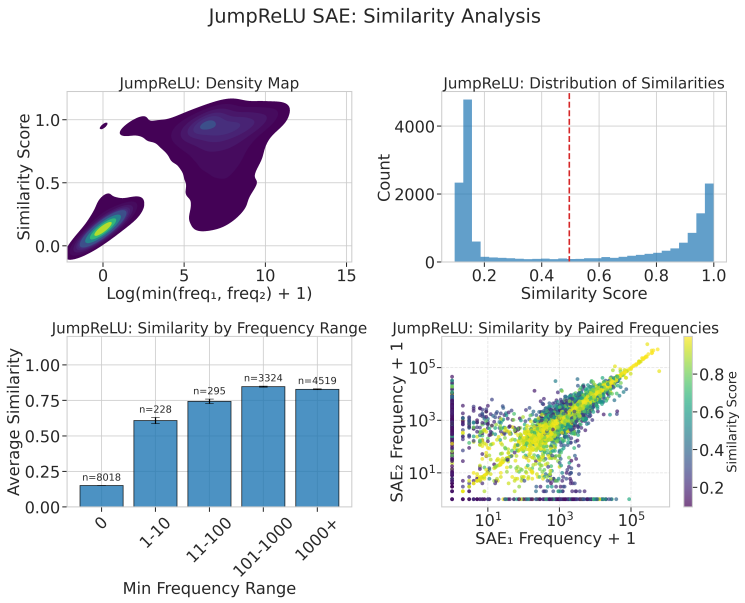

Figure 42: Feature similarity analysis for a JumpReLU SAE trained on Pythia-160M layer 8 activations. The overall PW-MCC was approximately 0.4957. Panels are analogous to Figure 36.

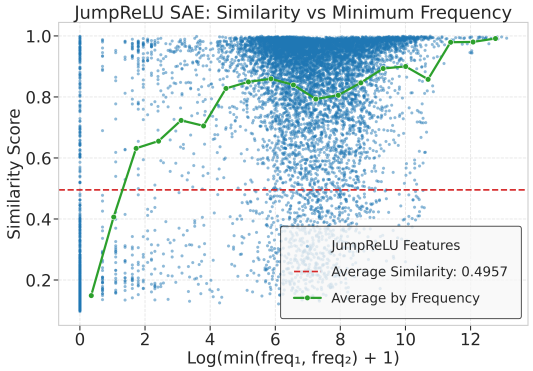

Figure 43: Average feature similarity versus `min(freq_run1, freq_run2)` for the JumpReLU SAE. This plot shows increasing similarity with higher shared activation frequency. The presence of two distinct clusters suggests potential subpopulations of features with differing learning or consistency characteristics within this architecture.

(and thus contributing to similarity scores) indeed tend to exhibit comparable activation frequencies in the respective runs.

Among the architectures quantitatively compared, TopK SAEs demonstrated the highest overall dictionary consistency as measured by PW-MCC, followed in order by Gated, JumpReLU, and Standard SAEs. This ranking correlates with qualitative observations regarding feature utilization; TopK and Gated SAEs also tended to produce fewer "dead" or very sparsely used features, suggesting a more effective and stable learning dynamic that leverages a greater portion of their dictionary capacity.

While the frequency-consistency trend is a common thread, different SAE architectures clearly lead to varying aggregate levels of feature consistency and distinct feature activation profiles. The detailed multi-panel visualizations offer a more granular understanding of consistency than a single global PW-MCC score, by illustrating how stability is distributed across features with different activation characteristics within each architectural type. These findings highlight the interplay of architectural choice and feature activation statistics in determining the reliability and interpretability of features learned by SAEs from real-world LLM activations.

G.3 Analysis of SAEs Trained on Gemma-2-2B Layer 12 Activations

This section presents results from training SAEs on activations from layer 12 of the Gemma-2-2B model [49] and analyzes the mean pairwise MCC averaged across two training seeds. Due to computational constraints, we report only the final aggregated pairwise MCC values. Computing the MCC requires solving an assignment problem with complexity $\mathcal{O}(N^3)$, where N represents the dictionary size. This becomes particularly challenging for Gemma-2-2B, which employs a dictionary four times larger than Pythia-160M ($65\,536$ versus $16\,384$).

Figure 44 shows the final training PW-MCC of SAEs trained on 500M tokens from `monology/pile-uncopyrighted`. TopK SAE achieves the highest pairwise MCC on this larger model, corroborating our findings on Pythia-160M and supporting our theoretical analysis. Most other SAE variants exhibit performance patterns similar to those observed in Figure 7. One notable exception is JumpReLU, which previously showed relatively poor consistency on Pythia-160M but achieves substantially improved feature consistency on Gemma-2-2B. Despite this improvement, JumpReLU still does not match the consistency levels attained by TopK SAE. JumpReLU is commonly used in extremely large-scale SAEs within the community [2, 29].

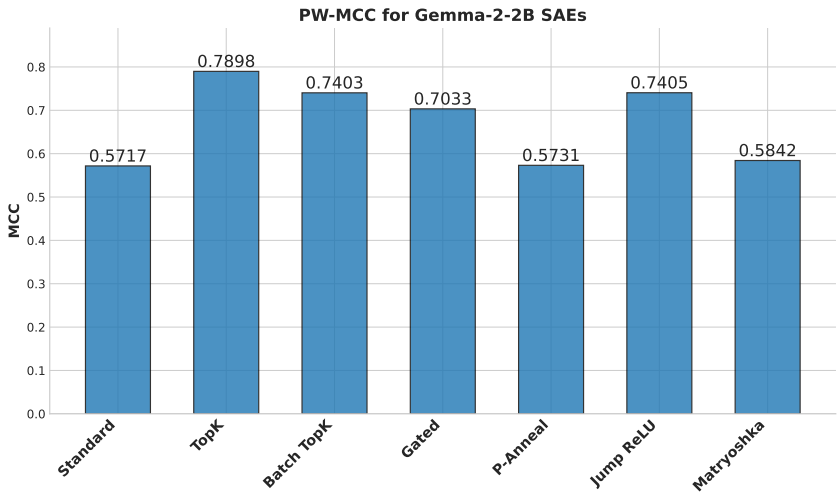

Figure 44: Final PW-MCC for BatchTopK, Gated, P-Anneal, JumpReLU, Standard, TopK, and Matryoshka BatchTopK SAEs on Gemma-2-2B activations. Higher PW-MCC indicates greater run-to-run feature consistency.

G.4 Qualitative Analysis: Example Feature Pairs across Similarity Buckets

To complement the quantitative assessments of feature consistency, this section provides a qualitative examination of example feature pairs extracted from 2 SAEs trained on Pythia-160M. We leverage an automated interpretation pipeline [45], to generate natural language explanations for individual SAE features and to measure the functional similarity between pairs of features from independently trained models.

The process for generating an explanation for a single SAE feature is as follows: first, activations for the feature are recorded across a substantial corpus (100,000 diverse text examples from `monology/pile-uncopyrighted` in our setup). The 10 text segments eliciting the strongest (highest magnitude) activations for that feature are then selected. These top-activating examples are formatted, the top activating tokens are emphasized (surrounded by « »), the activation strength of the tokens is shown after each example and presented to an explainer LLM, specifically `gpt-4.1-2025-04-14`. The LLM is prompted to identify common patterns or semantic concepts within these examples and produce a concise natural language interpretation of the feature's apparent function.

For the analysis presented in Table 10, we first matched features between two independently trained SAEs (SAE1 and SAE2). This matching was achieved by computing the pairwise cosine similarity of all their dictionary vectors and subsequently applying the Hungarian algorithm to find an optimal one-to-one correspondence. These matched feature pairs were then categorized into five buckets

based on their dictionary vector cosine similarity scores (Bucket 1: low vector similarity; Bucket 5: high vector similarity).

To measure the functional similarity of each matched pair, their independently generated explanations were provided to the same `gpt-4.1-2025-04-14` model. This model was then prompted to evaluate the semantic resemblance between the two explanations and assign a functional similarity score on a 1-10 scale. This score is presented as the **GPT-Score** score in Table 10, alongside the feature indices and their generated explanations.

Table 10 shows a clear concordance between the quantitative cosine similarity of the feature dictionary vectors and the functional similarity of their roles, as determined by the automated interpretability pipeline. Feature pairs with low vector similarity (e.g., Buckets 1 and 2) typically receive divergent functional explanations and low similarity scores from GPT-4.1. For instance, one feature might be interpreted as activating on LaTeX math symbols, while its low-vector-similarity counterpart from another SAE is interpreted as activating on Go/Rust code structures. Conversely, feature pairs exhibiting high vector similarity (e.g., Buckets 4 and 5) frequently obtain near-identical functional explanations and high similarity scores, such as both features being interpreted as responding to phrases indicating future events or specific Wikipedia category tags like births.

This analysis reinforces the utility of dictionary vector cosine similarity as a meaningful measure of feature consistency. The strong correlation observed suggests that high vector similarity between features learned across different runs reliably indicates the stable learning of semantically coherent and functionally equivalent interpretable units.

Table 10: Representative feature pairs from two independently trained SAEs (SAE1 and SAE2) organized by similarity buckets, with corresponding GPT-evaluated functional similarity scores. Buckets represent increasing levels of feature similarity, demonstrating a strong correlation between computed similarity measures and functional equivalence. Lower-similarity pairs (Buckets 1-2) exhibit largely unrelated behavioral patterns, while higher-similarity pairs (Buckets 4-5) demonstrate nearly identical semantic functions across both SAEs.

Bucket	SAE1, SAE2	GPT-Score	SAE1 Feature Explanation	SAE2 Feature Explanation
1	10742, 13528	3/10	Activates on punctuation marks and transition words marking syntactic or discourse boundaries.	Activates on section separator "Background {Sec1}" in scientific writing.
	37, 6034	2/10	Activates on LaTeX/math environments and symbols within mathematical markup.	Activates at the start of code block bodies after opening braces in Go and Rust.
	11993, 9627	3/10	Activates on bibliographic reference tokens and citation markers in academic writing.	Activates on closing parenthesis and angle bracket sequence at the end of figure captions.
	16044, 13563	2/10	Activates on common 2-4 letter substrings within larger tokens or variable names.	Activates on log message prefixes like "W/" and "E/" in Android logcat output.
	5177, 15030	3/10	Activates on LaTeX math tokens beginning or ending with backslash or angle brackets.	Activates on file paths or URLs containing delimiter sequences between directory or resource names.
2	15430, 4143	4/10	Activates on ordinal terms and comparative adjectives marking ordered elements in a sequence.	Activates on words following punctuation or in enumerated lists, especially suffixes or grammatical constructs.
	13156, 12246	6/10	Activates on markup-like sequences of repeated or paired symbols used as section separators in code.	Activates on closing angle brackets that terminate LaTeX-style or math expression delimiters.
	2292, 15215	4/10	Activates on forms of the verb "to be" used as auxiliary verbs or main verbs.	Activates on the auxiliary verb "have" used for forming present perfect constructs.
	4789, 3718	4/10	Activates on phrases like "sounds like", "feels like" expressing resemblance or subjective impressions.	Activates on verbs expressing mental or emotional impact in reactions or realizations.
	15326, 4995	2/10	Activates on the token "int" in various contexts, both as a suffix and as a programming token.	Activates on "Image" and variants in variable names, software names, and UI elements.
3	10739, 10630	5/10	Activates on paired angle brackets used as delimiters or markers in code and technical writing.	Activates on code keywords, method names, and identifiers in programming contexts.
	1203, 2543	3/10	Activates on tokens containing the sequence "red", "whit", or "Reddit" within longer words.	Activates on scientific nouns denoting materials used in laboratory or industrial contexts.
	9668, 13698	6/10	Activates on past-tense passive forms of verbs indicating completion or accomplishment.	Activates on tokens signifying the achievement or fulfillment of requirements or conditions.
	14177, 3256	4/10	Activates on delimiters for copyright and license statements in code and documentation.	Activates on punctuation elements and conjunctions serving as delimiters in complex sentences.
	7237, 7841	4/10	Activates on wordpieces containing "wor" and similar substrings within longer words.	Activates on "War" in proper nouns and as a standalone word in relevant contexts.
4	8557, 8334	9/10	Activates on mentions of the color "yellow" when describing objects or attributes.	Activates on the token "yellow" as a standalone word or within color-related phrases.
	3796, 6569	4/10	Activates on ".>" token in numeric, scientific notation or code fragments.	Activates on the period character when used as a decimal point in numerical values.
	3161, 11659	6/10	Activates on multi-word phrases with verbs plus prepositions introducing perspectives.	Activates on "in" within common prepositional phrases introducing abstract relationships.
	2045, 14698	5/10	Activates on mentions of diseases, particularly cancer and related medical conditions.	Activates on scientific terms denoting important issues, processes, or domains in research.
	10453, 9653	7/10	Activates on tokens related to biometric identification, especially fingerprints and forensics.	Activates on technical nouns denoting identifiers, labels, codes, or tags in specialized domains.
5	13974, 1289	10/10	Activates on phrases indicating future events or developments with specific timeframes.	Activates on future-related phrases indicating when something is expected or planned.
	9040, 9704	10/10	Activates on "births" in Wikipedia-style category tags denoting birth years.	Activates on "births" within Wikipedia category tags indicating year of birth.
	13514, 4930	9/10	Activates on multi-token prepositions and conjunctions, especially with "of", "by", "to".	Activates on phrases containing prepositions that indicate causes, relationships, or compositions.
	3430, 6144	9/10	Activates on function definitions in code following parameter lists before function bodies.	Activates on opening curly braces that begin function or method bodies in code.
	3215, 7110	9/10	Activates on "exclude", "excluded", or "exclusion" in procedural or scientific contexts.	Activates on "exclude" and related forms in the context of setting boundaries or omission criteria.