Mechanistic Interpretability Should Prioritize Feature Consistency in Sparse Autoencoders

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

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 reproducibility and complicating model comparison. We study *run-to-run feature consistency* in SAEs and argue that it should be reported as a standard evaluation axis alongside reconstruction and sparsity. We propose the Pairwise Dictionary Mean Correlation Coefficient (PW-MCC) as an assignment-based metric to quantify consistency and demonstrate that high levels are achievable (PW-MCC $\approx$ 0.80 for TopK SAEs on LLM activations) with appropriate architectural choices. Our contributions include: (i) theoretical grounding for strong consistency in the idealized setting of TopK SAEs; (ii) synthetic validation using a *model organism*, which verifies PW-MCC as a reliable proxy for ground-truth recovery; and (iii) empirical analysis on LLM activations, where PW-MCC correlates with the similarity of automatically generated natural-language feature explanations. We hope these results encourage routine reporting of feature consistency to support more robust cumulative progress in MI.[1]

1 Introduction

Mechanistic Interpretability (MI) seeks to reverse-engineer neural networks into human-understandable algorithms (Olah et al., 2020; Elhage et al., 2021), with Sparse Autoencoders (SAEs) emerging as a prominent tool for decomposing model activations into more interpretable, monosemantic features (Bricken et al., 2023a; Cunningham et al., 2023; Gao et al., 2025; Pach et al., 2025). A common aspiration in MI is to identify a canonical set of features—unique, complete, and atomic units of analysis that faithfully represent a model's internal computations (Leask et al., 2025). However, recent work (Paulo and Belrose, 2025; Marks et al., 2024a; Fel et al., 2025; Leask et al., 2025; Méloux et al., 2025) highlights substantial *run-to-run inconsistency*: SAEs trained with identical data and hyperparameters but different random seeds can learn meaningfully different feature dictionaries. This instability, potentially arising from phenomena like feature splitting (Leask et al., 2025; Chanin et al., 2024) or the amortization gap (O'Neill et al., 2024), undermines reproducibility, reduces research efficiency, and weakens the evidential value of derived interpretations.

Our position is that run-to-run *feature consistency* should be treated as a first-class evaluation axis for SAEs used in mechanistic interpretability and reported alongside reconstruction and sparsity. This is particularly important in architecture design, benchmarking, and reusable *foundation* SAEs. We present the Pairwise Dictionary Mean Correlation Coefficient (PW-MCC) as a concrete metric that operationalizes dictionary-level alignment across independent runs, and we show that high consistency is achievable with appropriate architectural and training choices (e.g., TopK nonlinearities). Our main contributions are:

- We formalize run-to-run feature consistency for SAE dictionaries and propose PW-MCC as a practical metric to quantify it, clarifying when and how it is most useful in practice (Section 3).
- We provide theoretical grounding for strong feature consistency in the idealized setting of TopK SAEs by connecting them to identifiability results in overcomplete sparse dictionary learning. We validate the theory using a

[*]Equal contribution. Correspondence to xiangchs@cs.cmu.edu and amuhamed@cs.cmu.edu.

[1]Code: https://github.com/xiangchensong/sae-feature-consistency.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2172–2210
July 2-7, 2026

synthetic *model organism*, demonstrating that PW-MCC reliably tracks ground-truth feature recovery (GT-MCC) and that TopK SAEs achieve high consistency (GT-MCC $\approx$ 0.97) under matched-capacity conditions (Section 4).

- We empirically study feature consistency on LLM activations across several SAE variants, showing that high consistency is attainable (PW-MCC ≈ 0.80 for TopK SAEs in our setup). We further show that higher PW-MCC is associated with higher similarity of automatically generated natural-language feature explanations across runs (Section 5).

2 Background and Related Work

Sparse Autoencoders. Mechanistic interpretability aims to reverse-engineer neural networks into human-understandable algorithms (Olah et al., 2020; Pach et al., 2025; Cunningham et al., 2023), a goal challenged by *polysemanticity*, where individual neurons respond to multiple, unrelated concepts (Elhage et al., 2022). SAEs address this by decomposing high-dimensional activations into a sparse, overcomplete representation of *monosemantic* features (Bricken et al., 2023a; Cunningham et al., 2023). An SAE's encoder, parameterized by weights $\mathbf{W}_{\text{enc}}$ and biases $\mathbf{b}_{\text{enc}}$, transforms an input $\mathbf{x} \in \mathbb{R}^m$ into a sparse latent representation $\mathbf{f} \in \mathbb{R}^{d_{\text{sae}}}$ ($d_{\text{sae}} > m$) via $\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{W}_{\text{enc}}\mathbf{x}+\mathbf{b}_{\text{enc}})$. The decoder $\mathbf{W}_{\text{dec}}$ and $\mathbf{b}_{\text{dec}}$ then reconstruct the input as $\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{f}(\mathbf{x}) + \mathbf{b}_{\text{dec}}$. SAEs are trained by minimizing a loss that balances reconstruction fidelity and sparsity: $L(\mathbf{x}) = \|\mathbf{x}-\hat{\mathbf{x}}\|_2^2 + \lambda S(\mathbf{f}(\mathbf{x}))$, where $S(\cdot)$ is a sparsity penalty. Once trained, an SAE provides the decomposition $\mathbf{x} \approx \sum_{i=1}^{d_{\text{sae}}} f_i(\mathbf{x})\mathbf{a}_i$, where dictionary elements $\mathbf{a}_i$ are the columns of feature dictionary $\mathbf{A}$ (i.e., $\mathbf{W}_{\text{dec}}$). Several SAE variants like Standard ReLU (Bricken et al., 2023a), TopK (Gao et al., 2024), BatchTopK (Bussmann et al., 2024), Gated (Rajamanoharan et al., 2024a), and JumpReLU (Rajamanoharan et al., 2024b) exist, with the ultimate aspiration of identifying a *canonical* set of features (Leask et al., 2025). This is a stronger ambition than correlational analyses such as probing classifiers, which have known methodological limitations (Belinkov, 2022).

Activation Functions, Geometry, and Trade-offs. TopK SAEs are a convenient case study for our theory because they enforce exact k-sparsity, but TopK-style SAEs can still exhibit fragmentation and other representational issues, and can be sensitive to the choice of k and other hyperparameters (Karvonen et al., 2024a; Zhu et al., 2025; Gao et al., 2024). More broadly, different sparsity mechanisms induce different concept geometries and inductive biases over the learned features (Hindupur et al., 2025). In this work, we focus specifically on run-to-run dictionary consistency, treating fragmentation, coverage, and feature "quality" as complementary evaluation axes.

The Challenge of Feature Consistency. Despite their promise, SAEs trained with different random initializations often converge to substantially different feature sets (Marks et al., 2024a; Karvonen et al., 2024a), with overlap sometimes as low as 30% (Paulo and Belrose, 2025). This inconsistency manifests as phenomena like *feature splitting* (Leask et al., 2025) and *feature absorption* (Chanin et al., 2024), and stems from fundamental limitations in overcomplete dictionary learning where unique feature recovery is not guaranteed in practice (Leask et al., 2025). Despite theoretical advances (Sun and Huang, 2024; Donoho and Elad, 2003), the gap between idealized assumptions and practical implementations undermines guarantees for unique feature recovery. Proposed solutions to address this include Mutual Feature Regularization, which forces alignment (Marks et al., 2024a), and Archetypal SAEs, which use geometric constraints that may trade representational power for stability (Fel et al., 2025). These challenges have fostered skepticism that SAE features are merely a pragmatic decomposition rather than a universal set (Paulo and Belrose, 2025). Contrary to this, our work shows that high feature consistency is attainable through careful architectural and training choices.

3 Feature Consistency as an Evaluation Axis for SAEs

We study *run-to-run feature consistency*: holding the dataset, SAE architecture, and hyperparameters fixed (varying only training randomness), do independent runs converge to equivalent feature dictionaries? For mechanistic interpretability, we argue consistency should be reported alongside reconstruction and sparsity to enable reproducible feature-based analyses and principled comparison of SAE designs.

Why measure consistency?

- *Benchmarking and architecture design:* Reconstruction metrics alone can be insufficient for

comparing SAE variants or tuning hyperparameters; a consistency metric can reveal whether a seemingly strong configuration is stable across random seeds.

- *Reusable foundation SAEs:* When SAEs (and associated feature explanations) are shared and reused across projects, run-to-run consistency supports reproducibility by reducing dependence on idiosyncrasies of a single training run.
- *Stability for downstream MI methods:* Many common workflows (e.g., feature stitching, ablations, and steering) implicitly rely on the learned features being sufficiently well-defined across runs; severe inconsistency can compromise the reliability of these analyses.

Consistency is an implicit assumption in many SAE techniques. The long-term ambition in mechanistic interpretability to identify *canonical units of analysis* (Leask et al., 2025) presupposes some degree of run-to-run stability before addressing complexities like atomicity or completeness. Techniques like *feature stitching* (Leask et al., 2025) similarly require matchable features across runs or models. Downstream intervention workflows—including steering (Chalnev et al., 2024), unlearning (Muhamed et al., 2025), bias removal (Marks et al., 2024c), and feature ablation—also rely on targeted features being sufficiently well-defined across runs; severe instability can turn these analyses into seed-dependent artifacts.

Making consistency usable therefore requires two things: (i) a formal notion of when two learned dictionaries should be considered equivalent, and (ii) a metric that measures the degree of agreement on finite samples. We develop both in turn.

Let $\mathbf{A}$ and $\mathbf{A}'$ (both $\in \mathbb{R}^{m \times d_{\text{sae}}}$) denote decoder dictionaries learned by independent runs of the same SAE configuration. Because dictionary columns can be permuted and rescaled without changing the induced decomposition (up to a corresponding reparameterization of the sparse codes), we adopt the empirically tractable notion of **Strong Feature Consistency**: $\mathbf{A}$ and $\mathbf{A}'$ are equivalent if $\mathbf{A}' = \mathbf{APD}$ for a permutation matrix $\mathbf{P}$ and an invertible diagonal matrix $\mathbf{D}$ (allowing sign flips). Equivalently, for each feature vector $\mathbf{a}_i$ in $\mathbf{A}$ there exists a corresponding 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 are detailed in Appendix B.

To quantify this notion on finite samples, we use the **Mean Correlation Coefficient (MCC)** from independent component analysis and dictionary learning (Hyvärinen and Oja, 2000). MCC operationalizes Strong Feature Consistency: the best one-to-one matching captures the permutation, scale-invariance of cosine similarity absorbs the diagonal $\mathbf{D}$, and the absolute value handles sign flips. Given 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 $\mathbf{b}_j$, let $n = \min(d_A, d_B)$ and let $\mathcal{M}_n(\mathbf{A}, \mathbf{B})$ be the set of all one-to-one matchings of size n. We define:

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

The optimal matching M^* that achieves this maximum is typically found using the Hungarian algorithm (Kuhn, 1955), which solves the linear assignment problem in $O(n^3)$ time. In our largest experiment ($d_{\text{sae}} = 65{,}536$ on Gemma-2-2B), a single PW-MCC computation completed in a few hours on a multi-core CPU and was performed offline. 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 **Ground-Truth MCC (GT-MCC)** defined as $\text{GT-MCC}(\mathbf{A}, \mathbf{A}_{\text{gt}}) := \text{MCC}(\mathbf{A}, \mathbf{A}_{\text{gt}})$, where $n = \min(d_{\text{sae}}, d_{\text{gt}})$. We use GT-MCC to evaluate recovery quality and to validate PW-MCC as a proxy when ground truth is unavailable.

When the true underlying dictionary is unknown, we instead compare independently learned dictionaries and introduce **Pairwise Dictionary Mean Correlation Coefficient (PW-MCC)**. When comparing two learned dictionaries $\mathbf{A}$ and $\mathbf{A}'$, both of size d_{sae}, we use $\text{PW-MCC}(\mathbf{A}, \mathbf{A}') := \text{MCC}(\mathbf{A}, \mathbf{A}')$, where $n = d_{\text{sae}}$. When multiple runs are available, we report mean PW-MCC over all run pairs. A PW-MCC approaching unity signifies robust convergence to highly similar feature dictionaries across independent training runs.

When is PW-MCC worth computing? Computing PW-MCC requires training at least two independent runs of the same SAE configuration. In *SAE methodology* settings (architecture development, benchmarking, or training shared SAEs), we recommend reporting PW-MCC (e.g., mean over run pairs) as a standard stability metric. In *one-off applied interpretability* settings on a fixed architecture, it can be more sensible to allocate budget toward longer training or broader hyperparameter sweeps; in this regime, PW-MCC remains a useful

diagnostic when feasible, and if it is not computed, this should be treated as a limitation.

Consistency is *not* a substitute for faithfulness: highly consistent dictionaries can still yield explanations that are plausible but unfaithful, so PW-MCC should be treated as complementary to faithfulness-oriented evaluations and sanity checks (Jacovi and Goldberg, 2020; Oikarinen et al., 2025).

The following sections study when and why PW-MCC is high in theory and practice.

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 (O'Neill et al., 2024; Joshi et al., 2025), 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** (Hillar and Sommer, 2015; Donoho and Elad, 2003), which 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 Hillar and Sommer (2015):

Theorem 1 (Adapted from Hillar and Sommer (2015))**.** *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* $\mathbf{P}$ *and diagonal invertible* $\mathbf{D}$*.*

Implications for TopK SAE Feature Consistency. TopK SAEs can 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 $\mathbf{f} \mapsto \text{TopK}_k(\mathbf{f})$. A sufficient set of conditions for Theorem 1 to apply is: **(1) Exact k-sparsity** (true by construction); **(2) (Near-)zero reconstruction error** on data containing the witness set from Theorem 1; and **(3) (Approximate) round-trip** $E(\mathbf{Af}) \approx \mathbf{f}$ on k-sparse codes. As we prove in Appendix C, the round-trip property implies the spark condition. When these conditions hold with adequate data coverage, any two TopK SAEs trained on the same data must learn dictionaries that are identical up to permutation and scaling, establishing **strong feature consistency** (Section 3) even in overcomplete regimes.

In practice, condition (1) holds by construction for TopK SAEs, while conditions (2) and (3) can be checked approximately by measuring reconstruction error and round-trip error. In contrast, the witness-set assumption in Theorem 1 concerns the latent data-generating process and is not directly verifiable from finite observations. Accordingly, in our experiments on real LLM activations, high PW-MCC should be interpreted as empirical evidence of approximate consistency rather than a formal identifiability guarantee.

Finally, our identifiability-style analysis focuses on TopK SAEs because per-example hard k-sparsity and a deterministic encoder enable a clean connection between approximate left-invertibility (round-trip) and the spark condition. Extending comparable theoretical guarantees to other SAE variants (e.g., L1-penalized or gated SAEs) would require new techniques to handle variable sparsity and amortized inference errors. Importantly, PW-MCC itself is architecture-agnostic and can be used to compare any learned dictionaries, as we do empirically in Section 5.

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 rep-

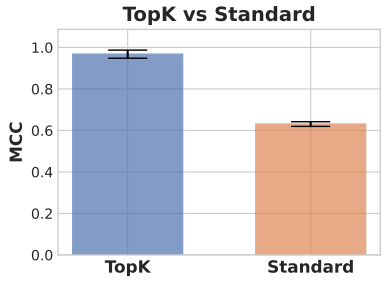

Figure 1: TopK vs. Standard SAE on a synthetic matched-capacity setting. TopK SAEs achieve substantially higher GT-MCC than Standard SAEs (0.97 vs. 0.63), validating the identifiability analysis of Section 4.1.

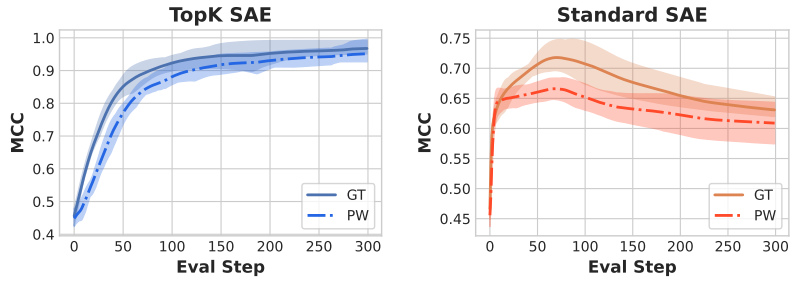

Figure 2: PW-MCC tracks GT-MCC during training (TopK left, Standard right). Both metrics follow the same trend and converge to comparable values, supporting PW-MCC as a ground-truth-free proxy for feature recovery. Shaded regions show max–min range across 5 seeds. The consistency (MCC) of TopK SAE increases steadily over the training curve until saturation (1.0), whereas Standard SAE reaches a peak MCC (0.72) but fails to maintain it, ultimately converging to 0.63.

resentative SAE variants: TopK SAE and Standard SAE. We show that models designed according to the criteria in our theoretical analysis converge to consistent features.

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 $(m = 8, d_{\text{gt}} = 16, k = 3, n = 5 \times 10^4)$, 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 (with 5 random seeds) on identical data and model architecture but different weight initializations, and compare the PW-MCC curves with the GT-MCC curves. Figure 1 presents the final GT-MCC evaluation results. TopK SAE achieves significantly higher GT-MCC (better ground-truth recovery) than Standard SAE, consistent with our identifiability analysis in Theorem 1. More importantly, Figure 2 demonstrates that the empirical PW-MCC values follow the same trend as GT-MCC and converge to comparable final values, suggesting that PW-MCC serves as an effective alternative 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 during training matches the ground-truth sparsity. See Appendix E for extended analysis when k is misspecified.

Takeaway: We observe that PW-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

Beyond the matched synthetic setting above, real SAE deployments face departures from the idealized assumptions that underlie strong consistency: SAEs are often globally compressive ($d_{\text{sae}} \ll d_{\text{gt}}$), feature frequencies are heavy-tailed, and the long tail contains extremely rare concepts. We therefore analyze a controlled synthetic *model organism* that progressively introduces these factors and uses the availability of ground truth to interpret PW-MCC as a diagnostic of run-to-run stability. Unless otherwise stated, synthetic results are averaged over multiple random seeds, and additional parameter sweeps are provided in Appendix D.

Consistency and Global Capacity Regimes. Even before introducing heterogeneous ground-truth frequencies, fundamental challenges to consistency arise from the global relationship between SAE width d_{sae} and ground-truth width d_{gt}. Under uniform sampling of ground-truth features in the linear model $\mathbf{X} = \mathbf{A}_{\text{gt}}\mathbf{F}_{\text{gt}}$ with k-sparse codes $\mathbf{f}_{\text{gt}}(\mathbf{x})$ $(k = 8)$, we observe two distinct regimes (Figure 3). In a **globally redundant regime** $(d_{\text{sae}} > d_{\text{gt}}$; e.g., $d_{\text{sae}} = 160, d_{\text{gt}} = 80, k = 8, n = 5 \times 10^4)$, the SAE achieves high alignment with the ground truth (GT-MCC 0.95), suggesting the learned features accurately recover the underlying concepts. However, run-to-run consis-

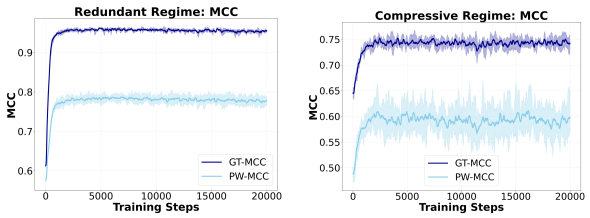

Figure 3: Two global capacity regimes driven by the ratio $d_{\text{sae}}/d_{\text{gt}}$. *Left:* Redundant regime ($d_{\text{sae}} > d_{\text{gt}}$) with high GT-MCC but lower PW-MCC due to selection ambiguity. *Right:* Compressive regime ($d_{\text{sae}} < d_{\text{gt}}$) with lower GT-MCC and PW-MCC due to insufficient capacity. The panels qualitatively illustrate the two regimes, not a head-to-head comparison at fixed d_{gt}. Shaded bands show max–min range across 5 seeds.

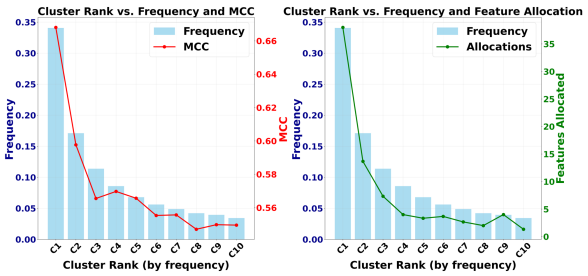

Figure 4: Non-uniform capacity allocation in the globally compressive Zipfian model ($\alpha = 1$, 10 clusters, $d_{\text{gt}} = 800$, $d_{\text{sae}} = 80$). More probable clusters receive proportionally more SAE capacity (green, right axis) and achieve higher GT-MCC (red, left axis).

tency is often substantially lower (PW-MCC 0.77). This discrepancy arises from **selection ambiguity**: with excess capacity, multiple learned features can be comparably good matches for a single ground-truth feature, leading different runs to converge to different, yet individually valid, dictionaries. The lower PW-MCC here reflects this reduced stability in feature selection. Conversely, in a **globally compressive regime** ($d_{\text{sae}} < d_{\text{gt}}$; e.g., $d_{\text{sae}} = 80, d_{\text{gt}} = 800, k = 8, n = 5 \times 10^4$), insufficient capacity prevents full recovery and diminishes both GT-MCC (0.75) and PW-MCC (0.60). The parallel decline of both metrics indicates their shared sensitivity to fundamental capacity limitations. Taken together, these global capacity mismatches illustrate why PW-MCC is valuable in practice: high GT-MCC does not guarantee stability, and PW-MCC provides a direct measure of practical run-to-run consistency.

Zipfian Feature Frequencies and Non-Uniform Capacity Allocation. Natural language exhibits a Zipfian (power-law) frequency spectrum (Appendix Figure 15)—a small head of common concepts and a large tail of rare ones. In the globally compressive regime relevant to SAEs in LLMs (Bricken et al., 2023b), this heterogeneity means limited dictionary capacity cannot be allocated uniformly across true features. To model this, we partition ground-truth features into C equal-sized clusters (each with d features) and impose an arbitrary ranking on these clusters. Data points are generated by first sampling a cluster $i \in \{1, \ldots, C\}$ with Zipf probability p_i, where $p_i := i^{-\alpha} / \sum_{j=1}^{C} j^{-\alpha}$ for rank i, and then sampling k true features from that cluster. Post-training analysis shows that SAEs do not allocate their dictionary capacity uniformly across clusters. We match learned features to clusters and define the effective capacity D_i of cluster i as the number of SAE features assigned to it (via Hungarian matching). Empirically, D_i is well-approximated by a superlinear power-law allocation in p_i:

$$D_i \approx d_{\text{sae}} \cdot \frac{p_i^{\beta}}{\sum_{j=1}^{C} p_j^{\beta}}.$$

We find $\beta > 1$, with $\beta \approx 1.34$ for $\alpha = 1$ and a modest range across Zipf exponents (roughly 1.26–1.46; Appendix D.3). Thus, in a globally compressive setting (e.g., $C = 10$, $d = 80$ per cluster, total $d_{\text{gt}} = 800$; $d_{\text{sae}} = 80, k = 8$), frequent clusters receive a disproportionate share of the dictionary and achieve higher GT-MCC, while rare clusters are systematically under-recovered (Figure 4). This differential recovery suggests that run-to-run consistency will also depend on feature frequency, a pattern that feature-level PW-MCC analysis is designed to capture. See Appendix D.3 for additional sweeps and diagnostics.

Emergence of Local Identifiability Regimes and Frequency-Dependent Consistency. Non-uniform allocation implies that, within a single SAE, different clusters effectively operate under different capacity ratios. For ground-truth cluster i with d true features, we define the **local redundancy factor** $R_i := D_i/d$. This yields three **local identifiability regimes**: *locally redundant* ($R_i > 1$), where the SAE allocates more learned features to the cluster than there are true features, risking selection ambiguity; *locally matched* ($R_i \approx 1$), where allocated capacity approximately matches the cluster's complexity; and *locally compressive* ($R_i < 1$), where insufficient capacity prevents full recovery. Because p_i is Zipfian, these regimes coexist within a single, globally compressive SAE: frequent clusters can be locally

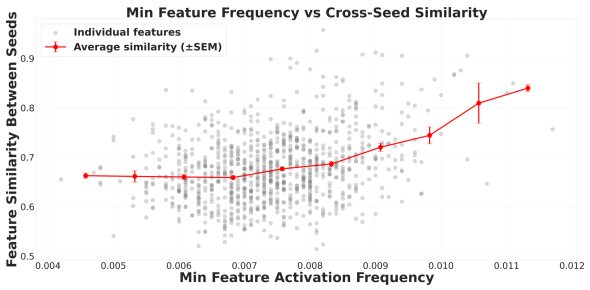

Figure 5: Frequency-dependent consistency at the individual-feature level. Minimum activation frequency of matched feature pairs vs. their pairwise similarity, from the two-phase Zipfian model organism ($d_{\text{gt}} = 400{,}000$, $d_{\text{sae}} = 1{,}000$). Feature-level similarity captures the influence of local identifiability regimes across the frequency spectrum.

redundant or matched, whereas rare clusters remain locally compressive. This coexistence translates directly to frequency-dependent run-to-run consistency (PW-MCC) at the individual-feature level. We therefore expect a wide distribution of matched-pair similarities across runs, with stability concentrated in the head and instability in the tail.

Probing the Full Spectrum of Consistency: A Two-Phase Zipfian Model. To probe this prediction across a wide dynamic range, especially for very rare concepts, we scale the model organism to 50,000 clusters ($d_{\text{gt}} = 400{,}000$) and adopt a two-phase frequency law: a Mandelbrot–Zipf head $g(r; s_1, q) = (r+q)^{-s_1}$ ($s_1 = 1.05$, $q = 5.0$) for ranks $r < 40{,}000$, followed by a steeper power-law tail $g(r; s_2) \propto r^{-s_2}$ ($s_2 = 30.0$) (Appendix D.4). Training a TopK SAE with $d_{\text{sae}} = 1{,}000$ reveals a clear positive relationship between the minimum activation frequency of matched feature pairs and their inter-run similarity (Figure 5).

This trend matches the local-regime picture. Frequent concepts are more likely to be locally redundant or matched: they receive sufficient allocated capacity, leading to stable learning and higher GT-MCC, though selection ambiguity can still prevent perfect consistency. As frequency decreases, clusters transition into locally compressive regimes; here insufficient allocated capacity relative to conceptual complexity leads to lower inter-run similarity scores. In the extreme tail, clusters can become so deeply locally compressive that corresponding features are learned inconsistently across runs or not at all, producing minimal inter-run similarity. This spectrum of varying stability is made visible by analyzing matched-pair similarities and summarizing them via PW-MCC.

Takeaway: Our synthetic model organism experiments validate PW-MCC as a robust diagnostic, capturing how global capacity, Zipfian skew, and local identifiability regimes jointly shape feature consistency across the full frequency spectrum.

This head–tail picture suggests that aggregate PW-MCC can hide substantial long-tail instability. In our real-world experiments (subsection 5.2), we therefore report both overall PW-MCC and a decomposition by activation frequency (Figure 7) to show whether consistency is broadly distributed or dominated by the most frequent features.

5 Evidence from Applications and Large-Scale Validation

5.1 Evaluating Consistency in the Real-World

Prevailing practices in SAE evaluation prioritize metrics like reconstruction error (L_2 loss), Fraction of Variance Explained, and sparsity (L_0/L_1) (Bricken et al., 2023a; Cunningham et al., 2023). 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 clear when controlling feature sparsity. For example, since reconstruction loss improves monotonically with lower L_1 penalties or a larger k in TopK SAEs, it alone fails to identify an optimal sparsity level. PW-MCC addresses this limitation by identifying the underlying trade-off. Insufficient sparsity from excessively large k or low L_1 penalties causes feature selection ambiguity, while excessive sparsity from small k or high L_1 penalties leads to inconsistent concept representation. Both extremes degrade GT-MCC and PW-MCC, thus enabling clear identification of an optimal k as we demonstrate in practice (Appendix E).

Empirically, across the regimes we study, we do not observe a strong trade-off between reconstruction quality and consistency: variants with similar reconstruction can differ substantially in PW-MCC. We do not claim such trade-offs cannot arise in other training regimes or at different scales; systematically characterizing reconstruction–consistency–diversity trade-offs remains an important direction for future work.

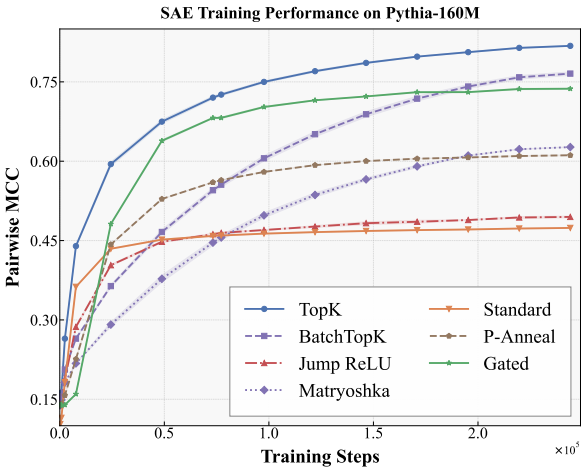

Figure 6: PW-MCC over training for seven SAE variants on Pythia-160M layer-8 activations. TopK and BatchTopK reach the highest run-to-run consistency; BatchTopK has not saturated by 2.5×10^5 steps. Curves are averages over 3 seeds with shaded regions showing the standard deviation.

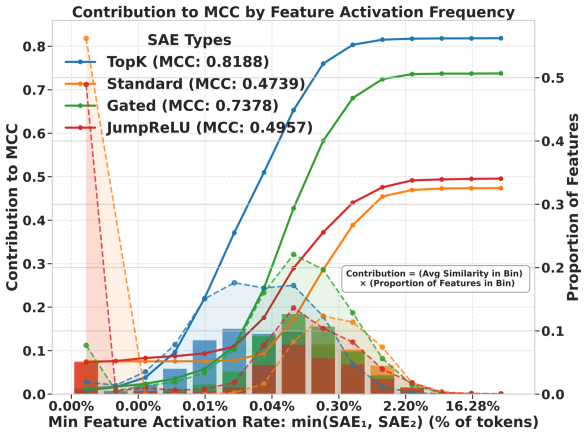

Figure 7: Contribution of each activation-frequency bin to overall PW-MCC across four SAE variants. Bars (left axis): each bin's contribution. Solid lines: cumulative contribution. Dashed lines (right axis): feature distribution across bins. TopK SAEs contribute more evenly across frequencies, while Standard and JumpReLU have a heavy tail of low-frequency, low-consistency features.

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. 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 (Biderman et al., 2023), with width $16,384$ on 500 million tokens from `monology/pile-uncopyrighted` (Gao et al., 2020), 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 training configurations. Further details on the training setup and hyperparameters as well as results for larger language model Gemma-2-2B (Team et al., 2024b) are provided in Appendix F. PW-MCC is computed post hoc via Hungarian matching, and in our experience 2–3 independent runs are sufficient to obtain a stable estimate of mean PW-MCC for a fixed configuration.

Overall Consistency and Training Dynamics. Figure 6 shows the evolution of PW-MCC during training for 7 different SAE architectures. For all SAE architectures we observe a steady increase in PW-MCC over training steps, indicating that as SAEs learn, their feature dictionaries become more aligned across independent runs. Among the evaluated architectures, TopK and BatchTopK SAEs achieved the highest PW-MCC scores. We emphasize that we do *not* claim TopK is universally best across all desiderata; TopK-style SAEs can be sensitive to hyperparameters and can exhibit fragmentation and other representational issues (Karvonen et al., 2024a; Zhu et al., 2025; Hindupur et al., 2025). Our claim here is specific to MCC-based run-to-run consistency. 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 initial consistency but exhibit faster improvement during the training process, eventually surpassing others (e.g., Gated SAE), suggesting that different SAEs impose varied structural assumptions (Hindupur et al., 2025). Overall, TopK- and BatchTopK-style SAEs achieve the highest run-to-run consistency under PW-MCC in our experiments, and we offer PW-MCC as a concrete diagnostic that we hope the community will extend with complementary measures.

Activation freq. (per 1M tokens)	# Features	Cosine similarity (mean ± std.)
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 1: Consistency increases with activation frequency (TopK SAE, Pythia-160M). For matched pairs of features from two independently trained SAEs, we bin by activation frequency and report the mean (± std.) cosine similarity within each bin. More frequently activated features exhibit higher run-to-run similarity.

Cosine similarity range	# Feature pairs	Explanation similarity (1–10)
0.065–0.113	34	1.71
0.113–0.195	311	2.19
0.195–0.336	975	3.27
0.336–0.580	1,423	4.12
0.580–1.000	13,640	8.28

Table 2: Dictionary alignment tracks natural-language explanation similarity (TopK SAE, Pythia-160M). We bin matched feature pairs by cosine similarity, generate natural-language explanations for each feature via an automated interpretation pipeline (Appendix F.4), and have an LLM rate the similarity of the two explanations on a 1–10 scale. Scores are averaged over 20 sampled pairs per bin.

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 F.2). This confirms that frequently occurring concepts are generally learned more stably, and PW-MCC reveals this spectrum of consistency. Figure 7 further shows 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 Natural-Language Explanation Similarity. We test whether dictionary alignment corresponds to stable *natural-language descriptions* of features across runs. In Table 2, we match features between two independent SAE runs, bin the matched pairs by cosine similarity, use an automated interpretation pipeline (Appendix F.4) to generate explanations for each feature, and ask an LLM to rate the similarity between the two explanations (GPT Score). We find a strong positive correlation: feature pairs with high dictionary-vector similarity receive high explanation-similarity scores, while low-similarity pairs tend to receive divergent explanations. This provides evidence that PW-MCC and feature-level cosine similarity track stability in the concepts surfaced by an automated explanation pipeline, though explanation similarity remains a partial proxy for downstream functional usefulness (e.g., in steering or circuit tracing).

Takeaway: High PW-MCC is achievable on LLM activations with TopK SAEs. PW-MCC tracks feature consistency, reveals frequency-dependent consistency, and strongly correlates with natural-language explanation similarity across runs.

6 Conclusion and Future Work

While we acknowledge alternative perspectives (Appendix G), our findings show that high, measurable feature consistency is an achievable and necessary standard for advancing MI toward a robust engineering science. We contend this requires a shift in community practices, including routinely reporting quantitative consistency scores (e.g., PW-MCC), developing standardized benchmarks, and researching the determinants of consistency. This motivates several key research directions: designing SAEs that are robust to diverse LLM activation statistics and can preserve consistency for rare features; defining broader notions of feature equivalence beyond strong consistency (e.g., functional or subspace alignment); understanding the impact of the SAE encoder's amortization gap (O'Neill et al., 2024) on dictionary stability; and establishing stronger theoretical guarantees for modern SAEs. Embracing a research culture that values and quantifies consistency will be pivotal in building a more reliable and cumulative science of MI.

Acknowledgements

The authors would like to acknowledge support from NSF Award No. 2229881, AI Institute for Societal Decision Making (AI-SDM), the National Institutes of Health (NIH) under Contract R01HL159805, and grants from Quris AI, Florin Court Capital, MBZUAI-WIS Joint Program, and the Al Deira Causal Education project. Aashiq Muhamed gratefully acknowledges support from the Amazon AI Ph.D. Fellowship, the Cooperative AI PhD Fellowship, and the ML Alignment and Theory Scholars Program.

Limitations

This paper should be viewed as a starting point rather than a final prescription. While we emphasize the importance of *feature consistency*, our formalization and evaluation through the Pairwise Dictionary Mean Correlation Coefficient (PW-MCC) is only one possible approach. Alternative notions of equivalence (e.g., functional similarity, subspace alignment) and corresponding metrics may prove more suitable in different contexts or for other interpretability goals.

Moreover, our discussion is framed around sparse autoencoders (SAEs). Although SAEs are currently central in mechanistic interpretability, the principle of consistency is relevant to a broader range of representation learning methods, and future work should investigate how stability manifests beyond this setting. Importantly, high consistency does not guarantee semantic interpretability or completeness, nor does the lack of consistency preclude useful exploratory analysis.

Finally, our empirical notion of run-to-run consistency holds all factors fixed except training randomness (e.g., random seed). Other sources of variation—such as changes in training data, shifts across domains, or changes in the underlying base model—may also affect learned features and their stability. We do not systematically evaluate these axes of consistency in this work.

Our aim is therefore not to provide a definitive framework, but to encourage community-wide engagement. We hope this work serves as a catalyst for developing richer definitions, metrics, and benchmarks of feature consistency, and invite the community to contribute towards shaping this research agenda.

Ethical Considerations

Mechanistic interpretability research, including the focus on feature consistency, carries both opportunities and risks. By providing tools that yield more stable and reproducible explanations, consistency-oriented methods may improve the reproducibility of model analyses and facilitate downstream applications such as safety auditing, bias detection, or model steering. At the same time, consistency alone is not a guarantee of faithfulness; it should be paired with dedicated faithfulness evaluations and sanity checks. We highlight several ethical considerations below.

First, there is a risk of *overconfidence*: highly consistent features may still fail to capture semantically meaningful or complete concepts. If consistency metrics are misinterpreted as guarantees of truth, they could lend unwarranted authority to flawed explanations. This is particularly concerning when interpretability outputs are used in high-stakes domains such as healthcare, law, or security.

Second, consistency metrics may inadvertently *privilege frequent or dominant patterns* at the expense of rare but socially or ethically significant features. For example, features representing minority groups, low-resource languages, or infrequent behaviors may be less consistent across runs, and thus overlooked or discounted in interpretability analyses. This could reinforce existing biases and further marginalize underrepresented populations.

Third, improved feature stability could *enable dual-use risks*. Consistent feature dictionaries may make it easier for malicious actors to reverse-engineer, manipulate, or exploit model internals (e.g., for creating more robust adversarial attacks or extracting private data). Researchers and practitioners should remain vigilant about how interpretability advances are communicated and deployed.

Finally, interpretability itself does not eliminate the need for broader accountability in the design, training, and use of machine learning systems. Feature consistency should be seen as one tool among many for supporting transparency and reliability, rather than a replacement for careful governance and ethical oversight.

References

Samir Abdaljalil, Filippo Pallucchini, Andrea Seveso, Hasan Kurban, Fabio Mercorio, and Erchin Serpedin. 2025. Safe: A sparse autoencoder-based framework

for robust query enrichment and hallucination mitigation in llms. *arXiv preprint arXiv:2503.03032*.

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, and 8 others. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*.

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

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

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, and 5 others. 2023a. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, and 1 others. 2023b. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

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

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

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

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

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

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

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and 1 others. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

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

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

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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

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

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and 1 others. 2023. Causal abstraction: A theoretical foundation for mechanistic interpretability. *arXiv preprint arXiv:2301.04709*.

Ruben Härle, Felix Friedrich, Manuel Brack, Björn Deiseroth, Patrick Schramowski, and Kristian Kersting. 2024. Scar: Sparse conditioned autoencoders for

concept detection and steering in llms. *arXiv preprint arXiv:2411.07122*.

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

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

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

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4198–4205.

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

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. 2024a. SAEBench: A Comprehensive Benchmark for Sparse Autoencoders. https://www.neuronpedia.org/sae-bench/info.

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

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

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

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, and 8 others. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

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

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

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

Samuel Marks, Adam Karvonen, and Aaron Mueller. 2024b. Dictionary learning. https://github.com/saprmarks/dictionary_learning.

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

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

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

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

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

Tuomas Oikarinen, Ge Yan, and Tsui-Wei Weng. 2025. Evaluating neuron explanations: A unified framework with sanity checks. *arXiv preprint arXiv:2506.05774*.

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

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

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

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

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

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

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

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

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

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

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024a. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Xudong Zhu, Mohammad Mahdi Khalili, and Zhihui Zhu. 2025. Abstopk: Rethinking sparse autoencoders for bidirectional features. *arXiv preprint arXiv:2510.00404*.

A Additional Related Work

A.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 (Bricken et al., 2023a; Marks et al., 2024b). 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 (Gao et al., 2024) 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 (Bussmann et al., 2024) 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 (Rajamanoharan et al., 2024a) 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 (Karvonen et al., 2024b) 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 (Rajamanoharan et al., 2024b) 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 (Bussmann et al., 2025) 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.

A.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 (Donoho and Elad, 2003) 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, (Sun and Huang, 2024) 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 (Sun and Huang, 2024) 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 (Arora et al., 2014). 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 (Hyvärinen and Oja, 2000), an objective shared with dictionary learning for finding the dictionary and sparse coefficient matrix (Wang and Seigal, 2024).

B 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 $\mathbf{a}_i$ *and* $\mathbf{a}'_i$ *denote the* i*-th feature vectors (columns) of dictionaries* $\mathbf{A}$ *and* $\mathbf{A}'$*, respectively.*

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.

C 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.

C.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:

$$\forall \mathbf{f} \in \Sigma_k, \quad \mathbf{E}(\mathbf{A}\mathbf{f}) = \mathbf{f}. \tag{2}$$

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 $\mathbf{A}$ (Donoho and Elad, 2003):

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

C.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:

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

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

By construction:

1. $\mathbf{f}, \mathbf{f}' \in \Sigma_k$ since $\mathrm{supp}(\mathbf{f}) \subseteq S_1$ and $\mathrm{supp}(\mathbf{f}') \subseteq S_2$ with $|S_1|, |S_2| \leq k$
2. $\mathrm{supp}(\mathbf{f}) \cap \mathrm{supp}(\mathbf{f}') = S_1 \cap S_2 = \emptyset$ (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 $\mathbf{A}\mathbf{h} = \mathbf{0}$, then $\mathbf{A}(\mathbf{f} - \mathbf{f}') = \mathbf{0}$, which immediately gives $\mathbf{A}\mathbf{f} = \mathbf{A}\mathbf{f}'$. □

C.3 Main Result

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

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

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 $\mathbf{A}\mathbf{f} = \mathbf{A}\mathbf{f}'$ (since $\mathbf{A}\mathbf{h} = \mathbf{0}$).

Let $\mathbf{x} := \mathbf{A}\mathbf{f} = \mathbf{A}\mathbf{f}'$ denote the common image. Since $\mathbf{E}$ is a deterministic function, $\mathbf{E}(\mathbf{x})$ is uniquely determined. However, applying the round-trip property (2) to both representations:

$$\mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{A}\mathbf{f}) = \mathbf{f},$$
$$\mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{A}\mathbf{f}') = \mathbf{f}'.$$

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

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

C.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**: $\mathbf{A}\mathbf{E}(\mathbf{x}) = \mathbf{x}$ 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: $\mathrm{spark}(\mathbf{A}) > 2k$.

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:

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

Applying the encoder to both sides:

$$\mathbf{E}(\mathbf{A}\mathbf{f}) = \mathbf{E}(\mathbf{x}) = \mathbf{f}. \tag{4}$$

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

C.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 approach differs from traditional dictionary learning identifiability results, which typically assume knowledge of an underlying ground truth dictionary and establish conditions for its recovery. While our result establishes spark condition without ground truth assumptions, guaranteeing full dictionary identifiability (recovery of $\mathbf{A}_{\text{gt}}$ up to permutation and scaling) requires explicit additional ground truth assumptions.

D Supplementary Analysis of SAEs trained on Synthetic Data

D.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 $k = 8, n = 5 \times 10^4$.

D.1.1 Redundant Regime Analysis

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

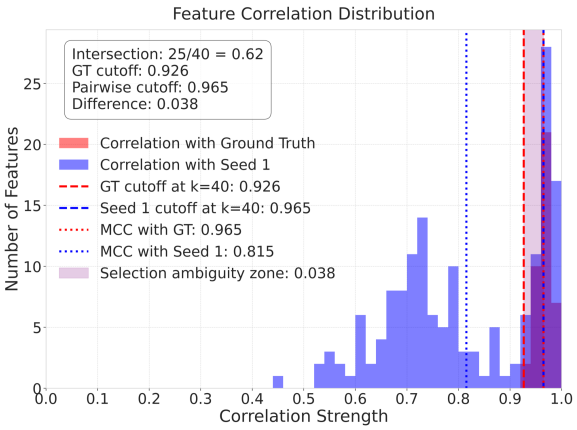

Figure 8: Feature-similarity distributions in the redundant regime ($d_{\text{gt}} = 80$, $d_{\text{sae}} = 160$, $k = 8$). Similarities of Run 1 features to ground truth (red) and to Run 2 features (blue). Overlap in the high-similarity region (purple) shows that multiple SAE features are good matches to both ground truth and to features from another run, producing selection ambiguity despite high feature quality.

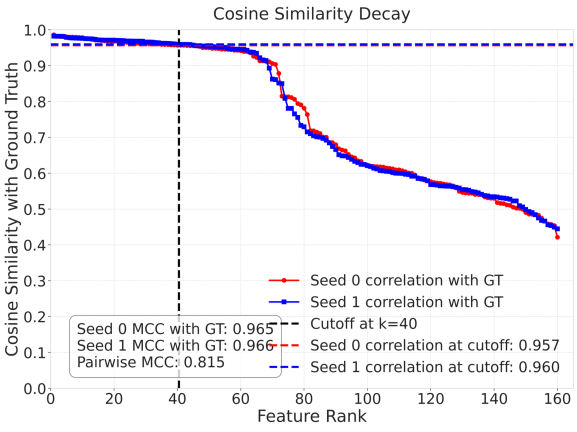

Figure 9: Slow decay of ground-truth similarity with feature rank ($d_{\text{gt}} = 80$, $d_{\text{sae}} = 160$, $k = 8$). Features of Run 1 (red) and Run 2 (blue) are ranked by cosine similarity to their matched ground-truth feature. Similarity stays high well past rank $d_{\text{gt}} = 80$, indicating that the SAE learns many near-duplicate representations of each ground-truth feature.

Figures 8 and 9 illustrate a key characteristic of the redundant regime: the SAE learns multiple good representations for each ground truth feature. Figure 8 shows substantial overlap between features with high similarity to ground truth and features with high similarity across runs. Figure 9 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 10, the Mean GT-MCC (Maximum Correlation Coefficient) reaches high values, indicating strong recovery of ground truth features. However, Figure 11 shows that the PW-MCC across different SAE initializations is lower, reflecting the challenge of consistent feature selection despite good ground truth recovery.

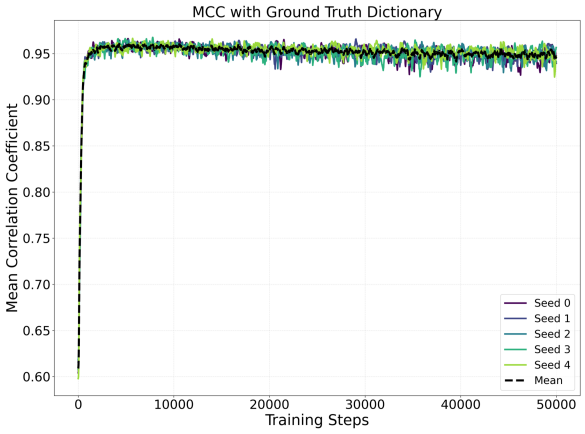

Figure 10: GT-MCC vs. training steps, redundant regime (TopK SAE, $d_{\text{gt}} = 80$, $d_{\text{sae}} = 160$, $k = 8$; mean over 5 seeds). GT-MCC reaches high values, indicating strong recovery of ground-truth features.

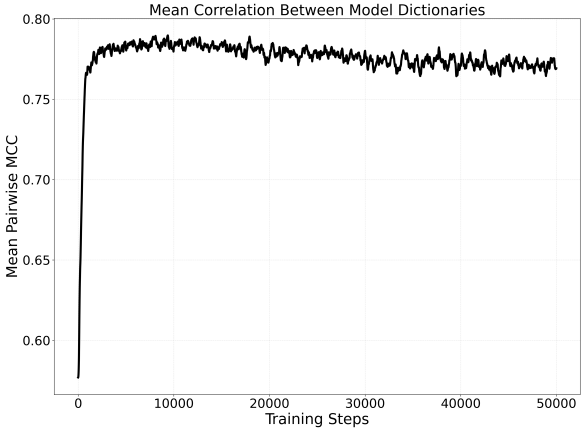

Figure 11: PW-MCC vs. training steps, redundant regime (TopK SAE, $d_{\text{gt}} = 80$, $d_{\text{sae}} = 160$, $k = 8$; mean over 5 seeds). PW-MCC stays below GT-MCC, reflecting selection ambiguity across independent SAE runs.

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 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 $d_{\text{sae}} < d_{\text{gt}}$, we use all d_{sae} indices). The intersection ratio is then computed as $R_{1,2} = \frac{|I_{1\to GT} \cap I'_{1\to 2}|}{\min(d_{\text{gt}}, d_{\text{sae}})}$. 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.

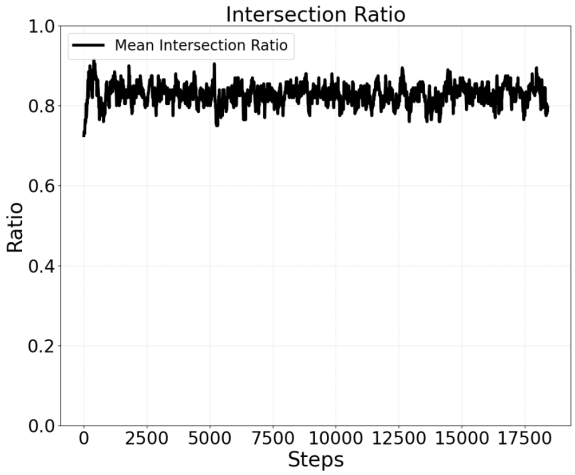

Figure 12: Intersection Ratio vs. training steps, redundant regime (TopK SAE, $d_{\text{gt}} = 80$, $d_{\text{sae}} = 160$, $k = 8$; mean over 5 seeds). Higher values indicate that the features matched to ground truth in one run also appear among the top-matched features in another run, reflecting more stable feature selection across initializations.

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.

D.1.2 Compressive Regime Analysis

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

ity limitation forces the SAE to prioritize which features to learn.

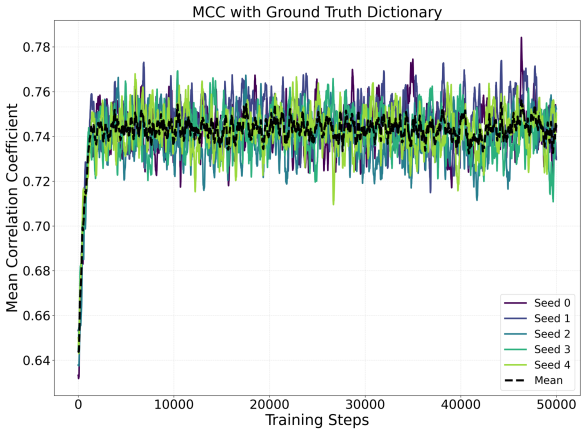

Figure 13: GT-MCC vs. training steps, compressive regime (TopK SAE, $d_{\text{gt}} = 800$, $d_{\text{sae}} = 80$, $k = 8$; mean over 5 seeds). GT-MCC is lower than in the redundant regime, reflecting the capacity limit that prevents recovering all ground-truth features.

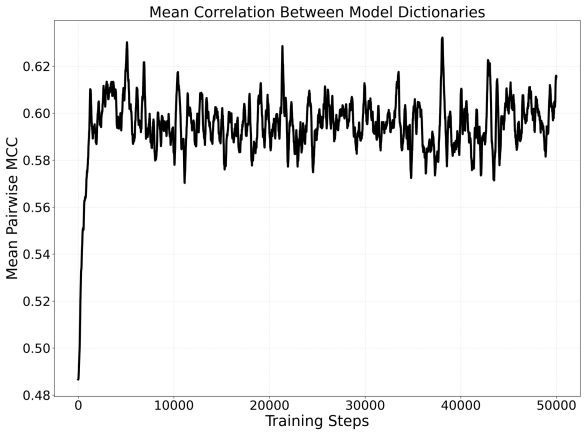

Figure 14: PW-MCC vs. training steps, compressive regime (TopK SAE, $d_{\text{gt}} = 800$, $d_{\text{sae}} = 80$, $k = 8$; mean over 5 seeds). PW-MCC also drops relative to the redundant regime, mirroring the reduced recovery quality.

Figures 13 and 14 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 13 shows that both the PW-MCC and the Mean GT-MCC reach significantly lower values compared to the redundant regime (Figure 10), reflecting the inability to recover all ground truth features with limited capacity.

D.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 ($d_{\text{gt}} = 800$) while varying the number of clusters they are organized into. The dimension of each cluster is $d_{\text{gt}}/\text{num\_clusters}$, resulting in fewer features per cluster as the number of clusters increases. The complete hyperparameter settings for these experiments are presented in Table 3.

Parameter	**Value**
TopK sparsity parameter (k)	8
Activation dimension (m)	20
SAE dictionary size (d_{sae})	80
Ground-truth dimension (d_{gt})	800
Number of clusters	$\{1, 10, 50, 100\}$
Cluster dimension	$d_{\text{gt}}/\text{num. clusters}$
Cluster distribution	uniform
Training examples	100,000
Training steps	20,000
Batch size	4,096
Learning rate	0.04
Minimum learning rate	1e−5
LR decay factor	0.1
LR decay steps	[20,000]
Warmup steps	1,000
L_1 coefficient	0.1
Random seeds	3

Table 3: Hyperparameters for the uniform-clustering synthetic experiments (Section D.2). The ground-truth dictionary of $d_{\text{gt}} = 800$ features is partitioned into a varying number of uniformly sampled clusters.

# Clusters	**GT-MCC**	**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: Effect of uniform partitioning of ground-truth features. Holding the total number of ground-truth features fixed at $d_{\text{gt}} = 800$ and varying the number of (uniformly sampled) clusters, mean PW-MCC increases modestly from 0.621 to 0.665, while GT-MCC remains near 0.74. Imposing cluster structure promotes more consistent feature selection without changing overall recovery quality.

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.

D.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 $d_{\text{gt}} = 800$, SAE dictionary size $d_{\text{sae}} = 80$, 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.

Parameter	Value
TopK sparsity parameter (k)	8
Activation dimension (m)	20
SAE dictionary size (d_{sae})	80
Number of clusters	10
Cluster dimension	80
Cluster distribution	Zipf
Zipf exponent (α)	$\{1.0, 1.1, 1.5, 2.0\}$
Training examples	100,000
Training steps	20,000
Batch size	4,096
Learning rate	0.04
Minimum learning rate	1e$-$5
LR decay factor	0.1
LR decay steps	[20,000]
Warmup steps	1,000
L_1 coefficient	0.1
Random seeds	3

Table 5: Hyperparameters for the Zipf-distribution synthetic experiments (Appendix D.3). The ground-truth dictionary is partitioned into 10 clusters of 80 features each, with cluster sampling probabilities following a Zipf distribution with exponent α.

D.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 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.

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

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 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).

D.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

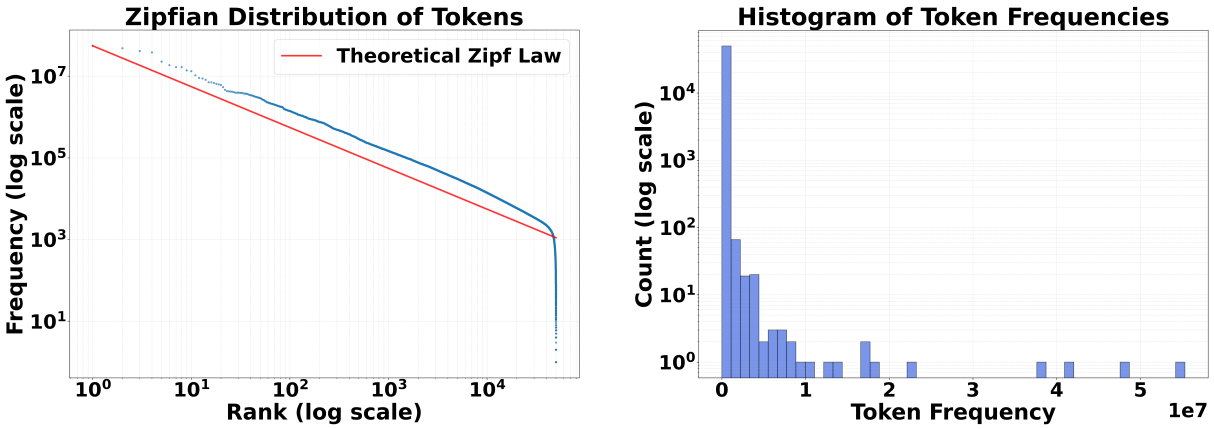

Figure 15: Token frequency on 1M tokens from the Pile. Log–log (left) and histogram (right) views, illustrating the Zipfian head and long sparse tail that motivate the two-phase model organism.

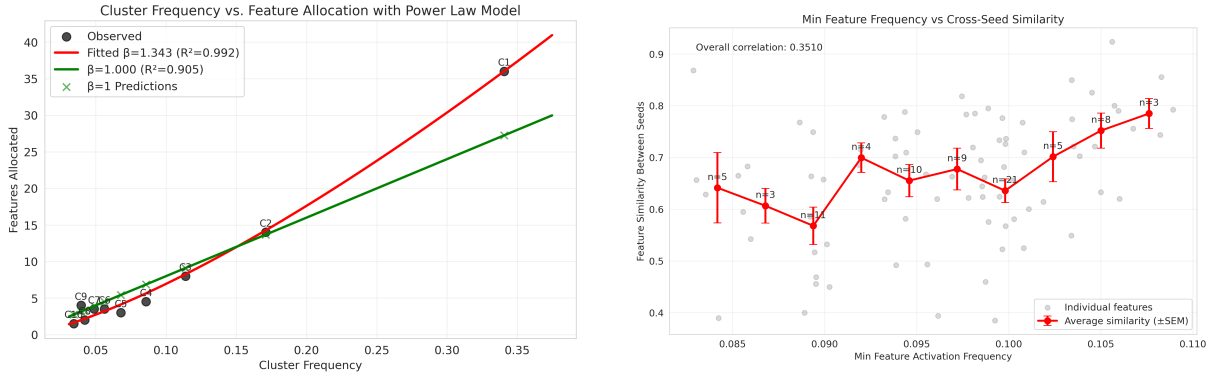

Figure 16: Capacity allocation and feature reproducibility under Zipf $\alpha = 1.0$. *Left:* SAE capacity allocated to each cluster vs. cluster probability, fit by a power law $D_i \propto p_i^{\beta}$ with $\beta \approx 1.343$ (red). *Right:* Matched-pair similarity vs. minimum activation frequency, with bucketed averages (red) showing a positive relationship between activation frequency and reproducibility.

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.

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

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.

D.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 $d_{\text{gt}} = 400{,}000$, with $50{,}000$ clusters (each cluster represented by 8 ground truth features on average), and maintain a constant TopK sparsity parame-

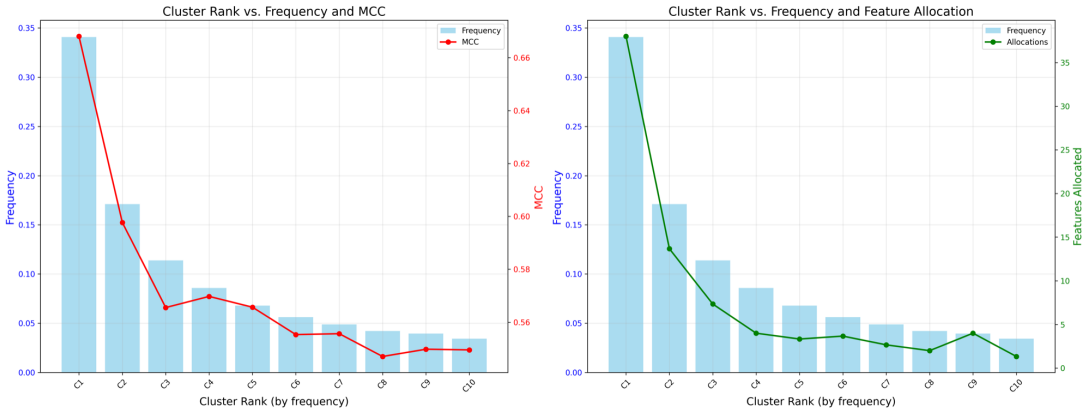

Figure 17: Per-cluster metrics under Zipf $\alpha = 1.0$. *Left:* Cluster probability (blue bars) and MCC (red line); more probable clusters achieve higher feature recovery. *Right:* Cluster probability (blue bars) and allocated SAE capacity (green line); more frequent clusters receive proportionally more features.

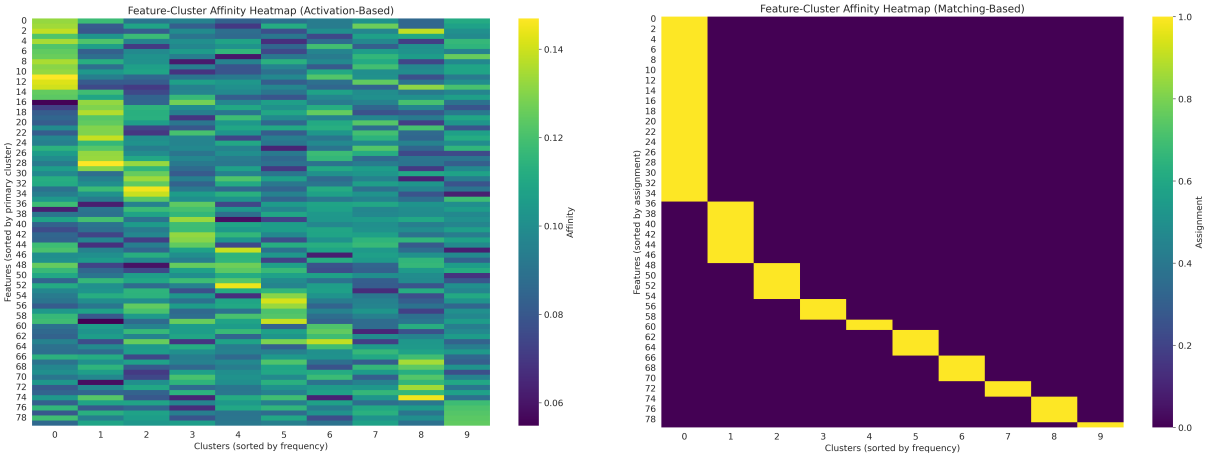

Figure 18: Feature–cluster affinities under Zipf $\alpha = 1.0$. *Left:* Activation-based affinity heatmap, where rows (features, sorted by primary cluster) vs. columns (clusters, sorted by probability); brighter entries indicate stronger activation. *Right:* Matching-based affinity via Hungarian assignment of features to clusters.

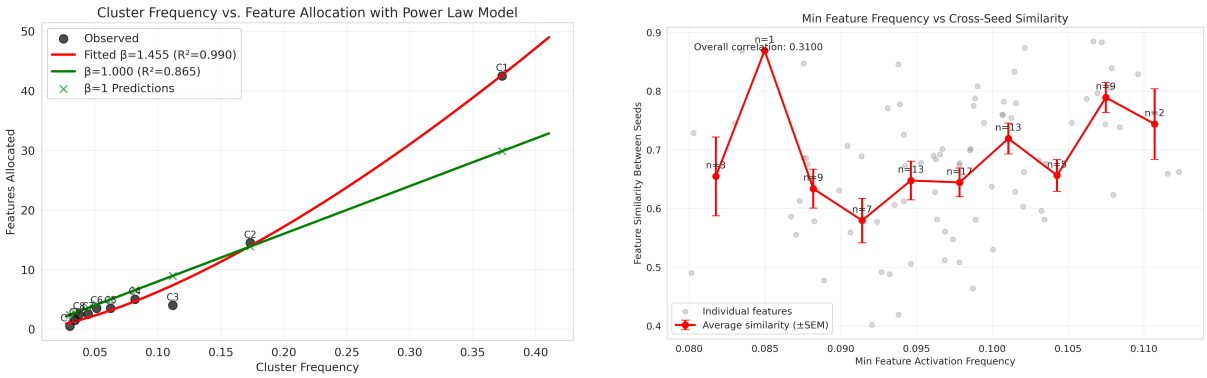

Figure 19: Capacity allocation and feature reproducibility under Zipf $\alpha = 1.1$. *Left:* Power-law fit $D_i \propto p_i^{\beta}$ with $\beta \approx 1.455$ (red). *Right:* Matched-pair similarity vs. minimum activation frequency, bucketed (red).

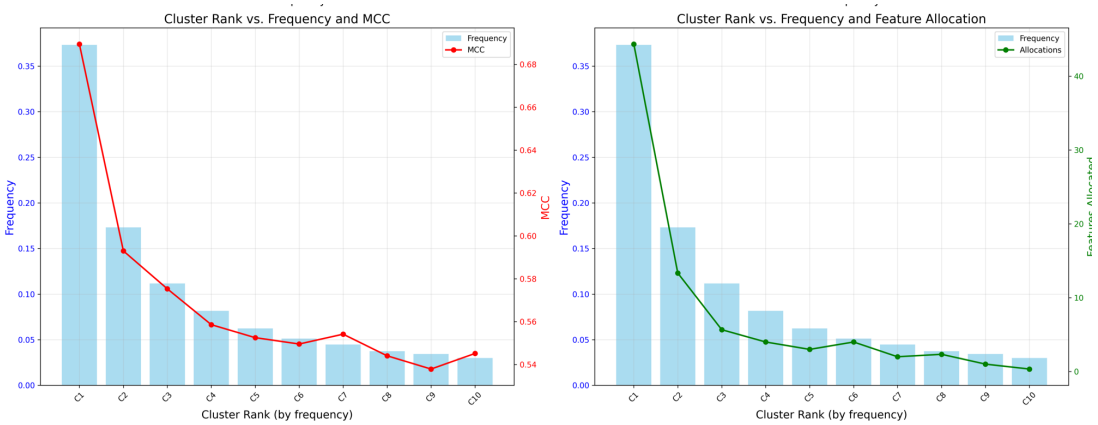

Figure 20: Per-cluster metrics under Zipf $\alpha = 1.1$. *Left:* Cluster probability (blue bars) and MCC (red line); recovery drops more sharply for low-probability clusters than at $\alpha = 1.0$. *Right:* Capacity allocation (green line) is even more skewed toward high-probability clusters.

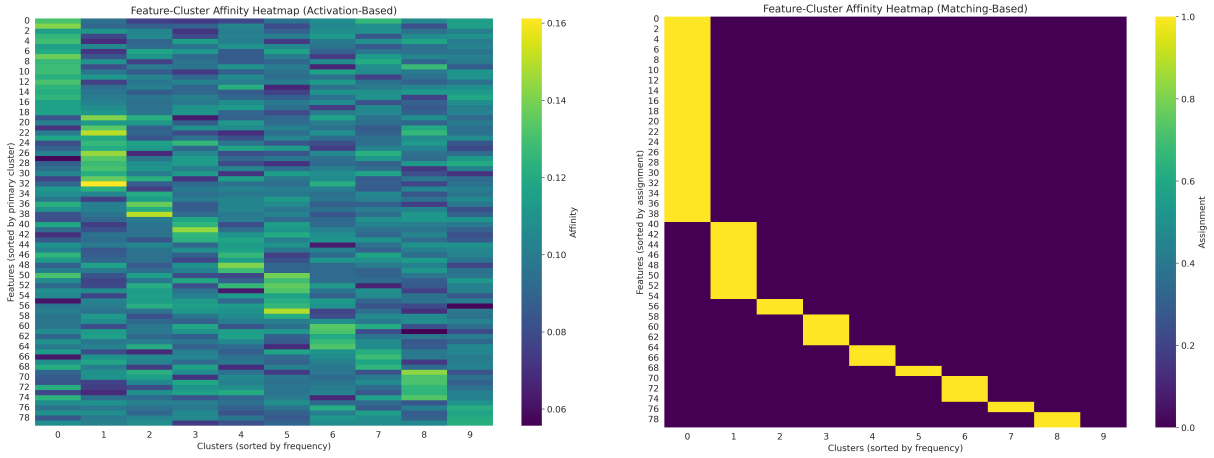

Figure 21: Feature–cluster affinities under Zipf $\alpha = 1.1$. *Left:* Activation-based affinity shows sharper feature-to-cluster specialization than at $\alpha = 1.0$. *Right:* Matching-based affinity shows more assignments concentrated on high-probability clusters.

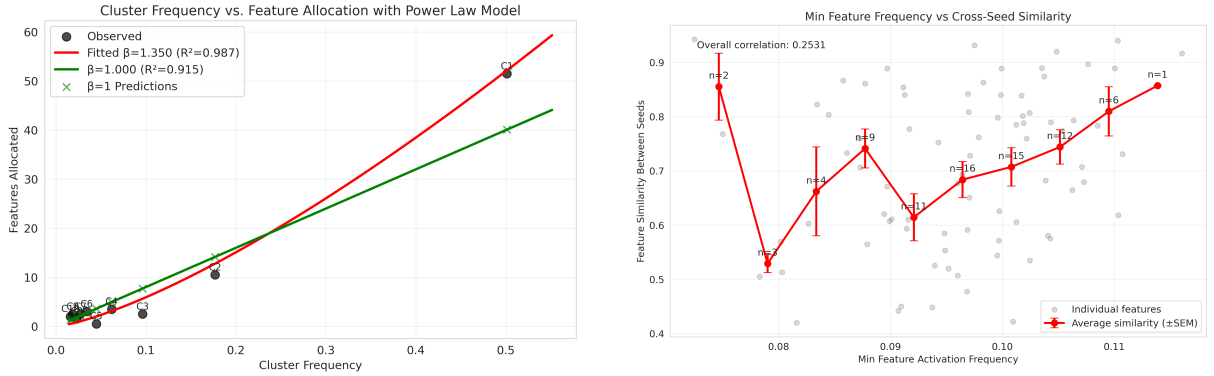

Figure 22: Capacity allocation and feature reproducibility under Zipf $\alpha = 1.5$. *Left:* Power-law fit $D_i \propto p_i^{\beta}$ with $\beta \approx 1.35$ (red). *Right:* Matched-pair similarity vs. minimum activation frequency shows a weak positive trend.

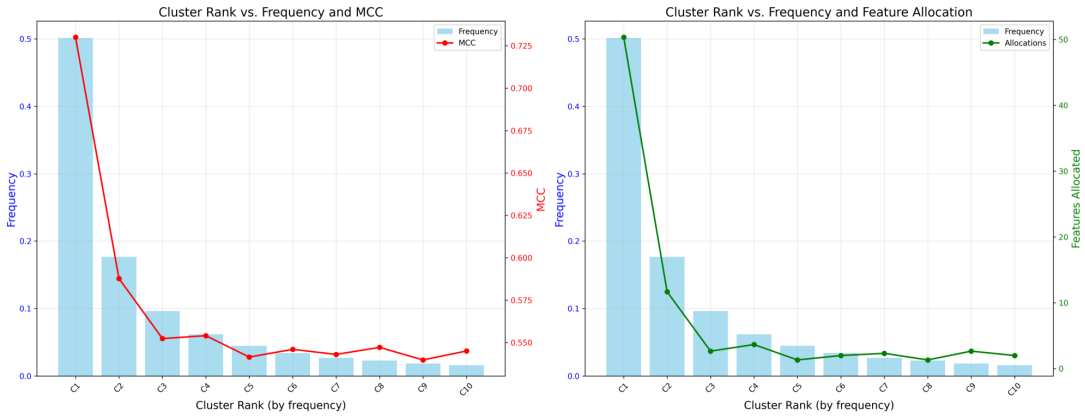

Figure 23: Per-cluster metrics under Zipf $\alpha = 1.5$. *Left:* A sharp threshold effect: MCC (red) drops steeply past the highest-probability clusters. *Right:* Capacity (green) is highly concentrated on the top clusters.

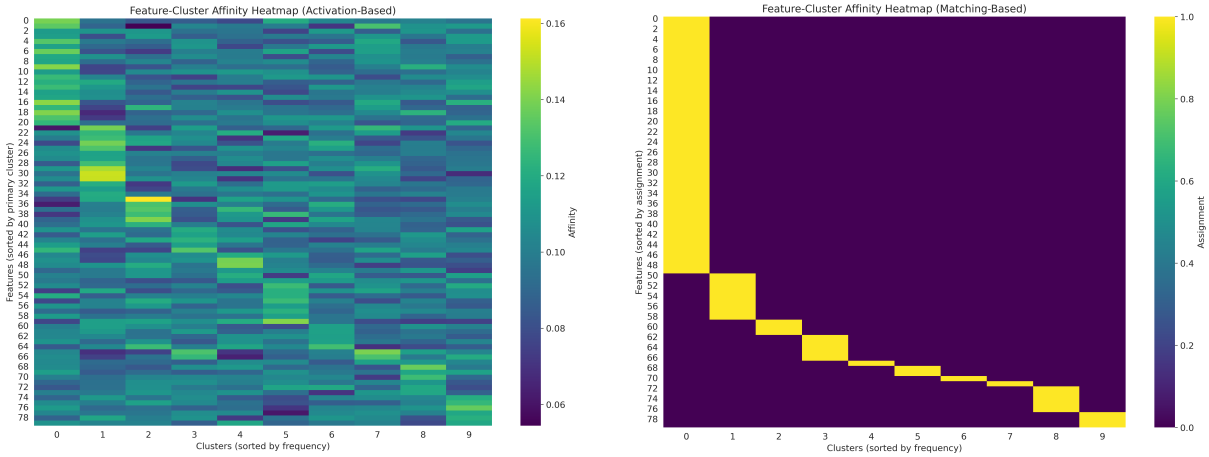

Figure 24: Feature–cluster affinities under Zipf $\alpha = 1.5$. *Left:* Activation-based affinity shows high specialization with little cross-activation. *Right:* Matching-based affinity shows strong one-to-one mapping for high-probability clusters and poor assignment for low-probability ones.

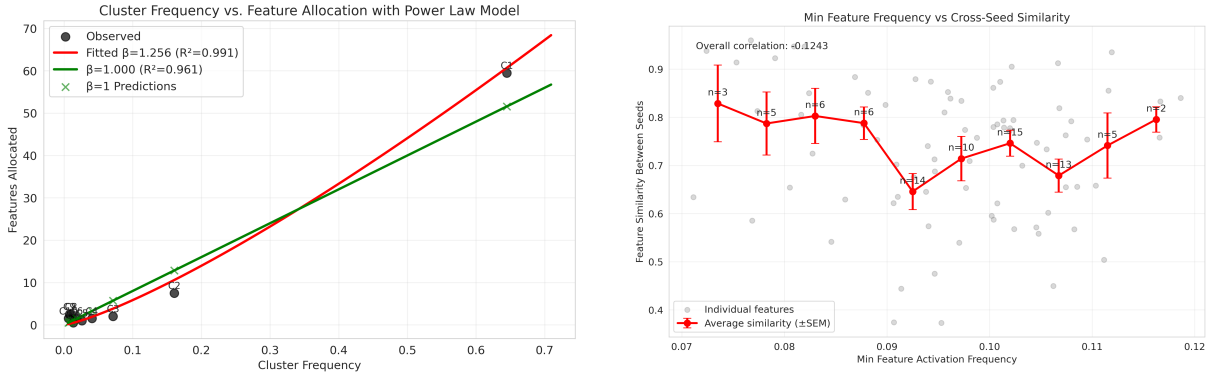

Figure 25: Capacity allocation and feature reproducibility under Zipf $\alpha = 2.0$. *Left:* Power-law fit $D_i \propto p_i^{\beta}$ with $\beta \approx 1.256$ (red); capacity is extremely concentrated on top clusters. *Right:* Matched-pair similarity vs. minimum activation frequency shows a flat-to-weakly-positive trend.

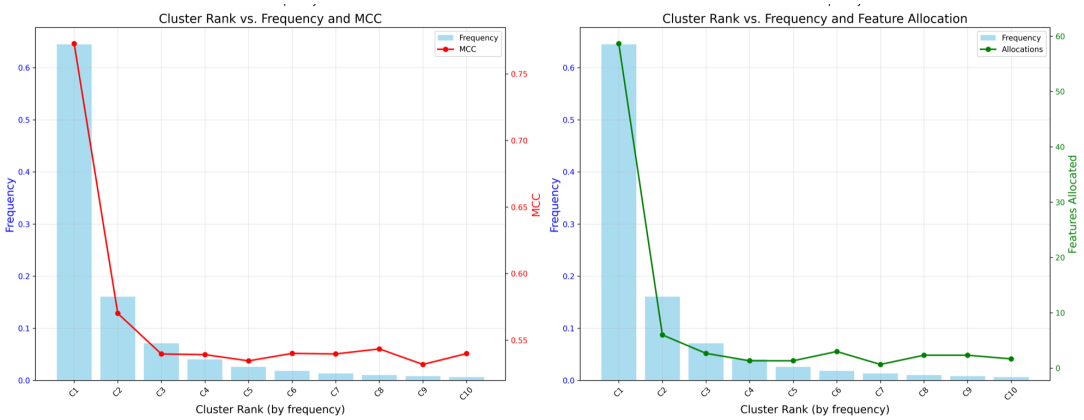

Figure 26: Per-cluster metrics under Zipf $\alpha = 2.0$**.** *Left:* Only the highest-probability clusters achieve good feature recovery (red). *Right:* Capacity (green) is concentrated almost entirely on the top clusters, leaving negligible capacity for the long tail.

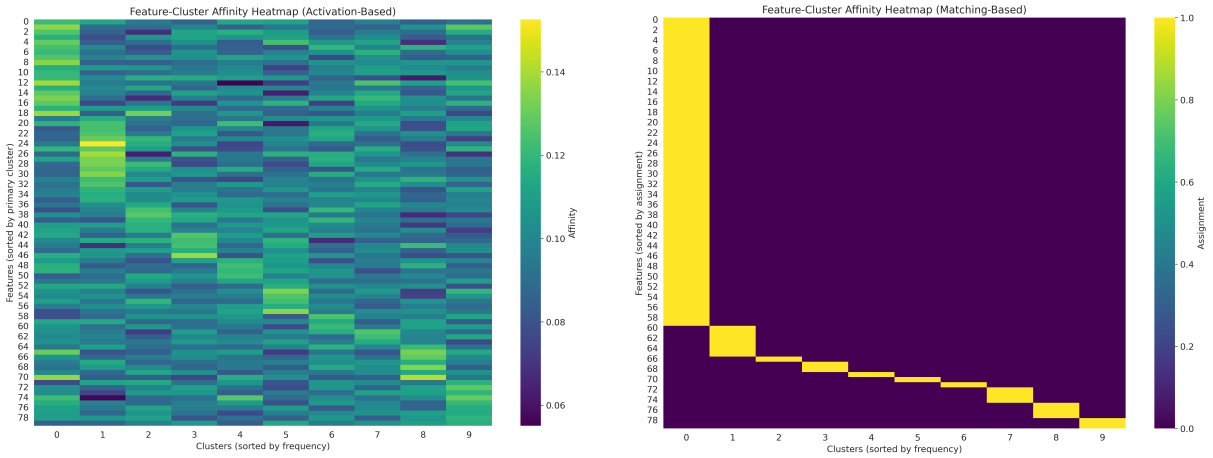

Figure 27: Feature–cluster affinities under Zipf $\alpha = 2.0$**.** *Left:* Activation-based affinity shows specialization to high-probability clusters. *Right:* Matching-based affinity shows strong assignment only for the very top clusters; most clusters receive little to no representation.

ter $k = 8$, while varying the SAE dictionary size $d_{\text{sae}} \in \{80, 160, 1{,}000, 10{,}000\}$ to analyze the impact of SAE capacity on feature learning. Table 6 summarizes the hyperparameters used in these experiments.

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 15).

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.

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

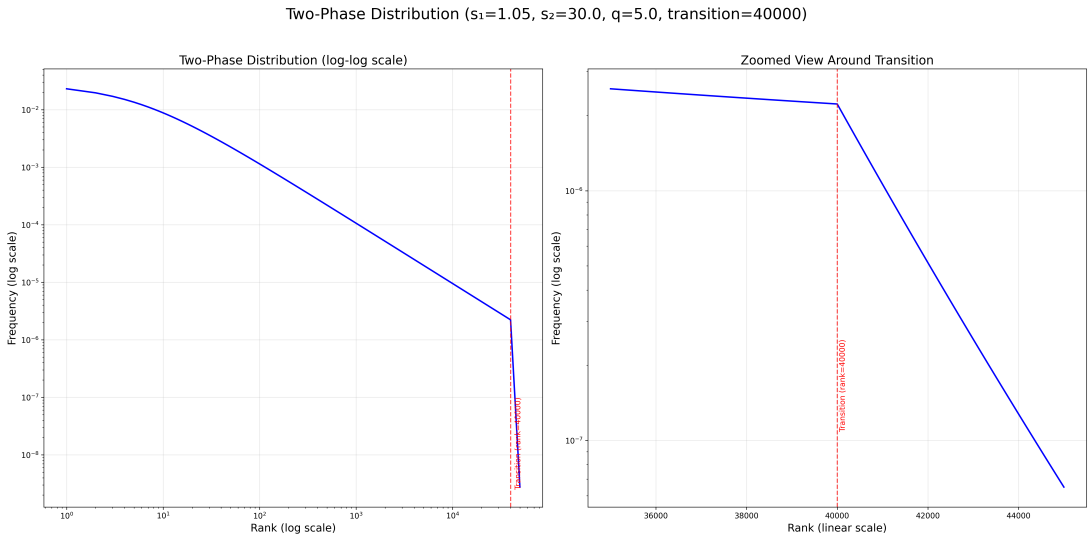

Figure 28: Two-phase cluster-probability distribution used in the synthetic model organism. A Mandelbrot–Zipf head ($s_1 = 1.05$) for ranks up to 40,000 transitions to a steeper power-law tail ($s_2 = 30.0$), approximating the head–tail structure of natural-language token statistics.

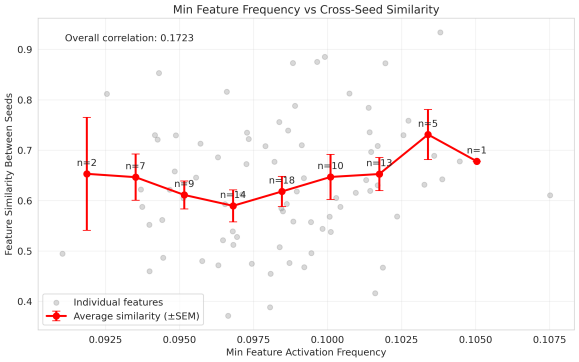

Figure 29: Two-phase model, $d_{\text{sae}} = 80$. Feature reproducibility shows only a weak positive relationship with activation frequency due to severe capacity limits.

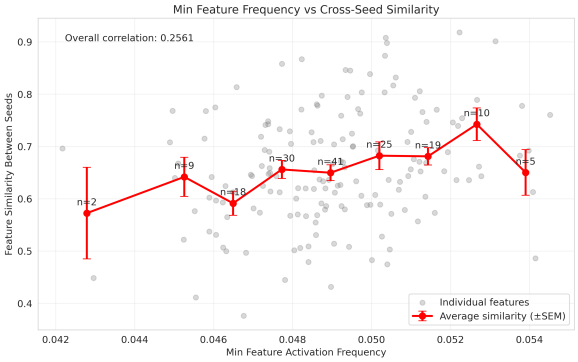

Figure 30: Two-phase model, $d_{\text{sae}} = 160$. The frequency–reproducibility trend is still weak but marginally clearer than at $d_{\text{sae}} = 80$.

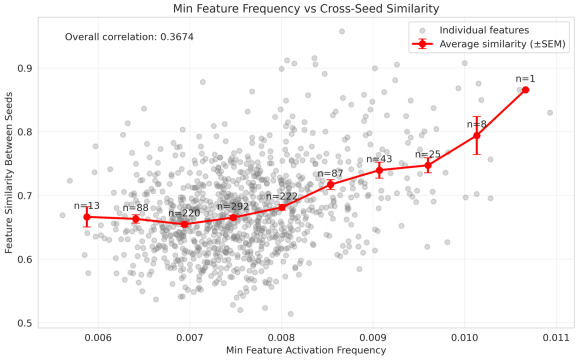

Figure 31: Two-phase model, $d_{\text{sae}} = 1{,}000$. A moderately strong positive correlation emerges, especially at higher activation frequencies, as increased capacity provides local redundancy for the more frequent clusters.

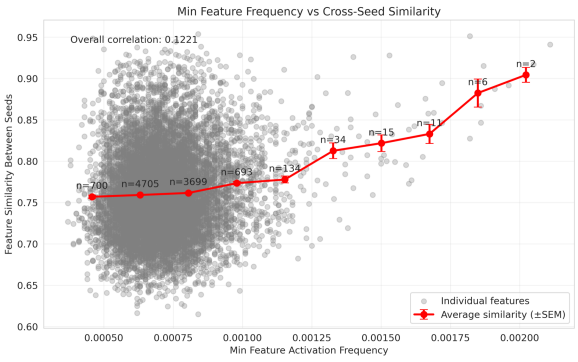

Figure 32: Two-phase model, $d_{\text{sae}} = 10{,}000$. With much larger capacity, reproducibility correlates strongly with activation frequency across a broad range.

Parameter	Value
TopK sparsity parameter (k)	8
Activation dimension (m)	20
SAE dictionary size (d_{sae})	$\{80, 160, 1{,}000, 10{,}000\}$
Number of clusters	50,000
Cluster dimension	8
Mandelbrot–Zipf exponent (s_1)	1.05
Mandelbrot–Zipf offset (q)	5.0
Long-tail exponent (s_2)	30.0
Transition rank	40,000
Training examples	100,000
Training steps	20,000
Batch size	4,096
Learning rate	0.04
Minimum learning rate	1e−5
LR decay factor	0.1
LR decay steps	[20,000]
Warmup steps	1,000
L_1 coefficient	0.1
Random seeds	3

Table 6: Hyperparameters for the two-phase Zipf synthetic experiments (Appendix D.4). The ground-truth cluster frequencies follow a Mandelbrot–Zipf distribution for ranks below the transition rank and a steeper power law beyond it, modeling the head and long tail of natural-language token statistics.

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.

E 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 nonzero 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.

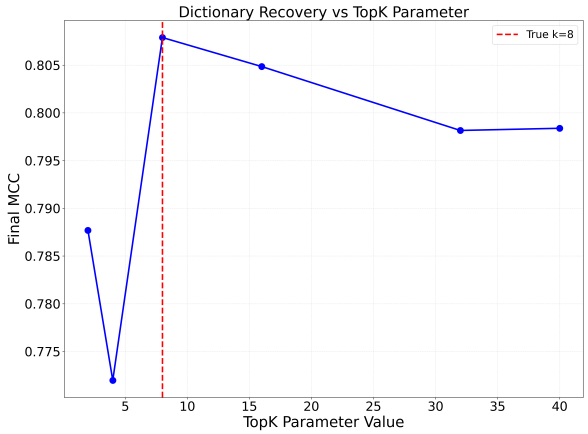

Figure 33: Effect of the TopK sparsity k in the matched regime ($d_{\text{gt}} = d_{\text{sae}} = 40$, true $s = 8$). Final GT-MCC (averaged over the last 100 training steps) peaks at $k = s = 8$; under-sparsity ($k < s$) degrades recovery more sharply than over-sparsity ($k > s$).

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 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.

F 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 architec-

tures 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.

F.1 Experimental Setup for Real Data Analysis

F.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.

F.1.2 Configuration for Pythia-160M Experiments

The detailed visualizations and quantitative comparisons presented in Section F.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 (Karvonen et al., 2024a). Each SAE was trained on a single A100 GPU in under 2 days.

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.

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

Additional experiments were conducted using activations from the `google/gemma-2-2B` model (Team et al., 2024a), targeting the residual stream output of layer 12 (`resid_post_layer_12`). Table 9 documents the pertinent hyperparameters for these larger-scale runs. Each SAE was trained on a single A100 GPU in under 2 days.

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 ar-

Parameter	Value
Activation dataset	`monology/pile-uncopyrighted`
Total training tokens	5×10^8
SAE batch size	2,048
Warmup steps	1,000
Sparsity warmup steps	5,000 (L_1-based and JumpReLU only)
Learning-rate decay start	0.8 of total training steps
Random seeds per configuration	3 (42, 43, 44)
Activation normalization	applied before SAE input
Autocast dtype	`torch.bfloat16`

Table 7: Common training parameters shared across all SAE architectures and language models in our LLM-activation experiments. Architecture-specific and model-specific parameters are listed in Tables 8 and 9.

Parameter	Value
Base LLM	`EleutherAI/pythia-160m-deduped`
Target layer	8 (residual stream output)
Activation dimension (m)	768
LLM batch size (activation generation)	32
Context length	1,024
LLM dtype	`torch.float32`
SAE dictionary width (d_{sae})	$2^{14} = 16{,}384$
SAE learning rate	3×10^{-4}
Architectures evaluated	Standard, TopK, BatchTopK, Gated, P-Anneal, JumpReLU, Matryoshka BatchTopK
Sparsity-penalty sweep (L_1-based):	
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 L_0 / k sweep (L_0-based)	$[20, 40, 80, 160, 320, 640]$

Table 8: SAE training parameters for experiments on Pythia-160M (`EleutherAI/pythia-160m-deduped`), layer 8. The sparsity-penalty and L_0/k sweeps follow the protocol of (Karvonen et al., 2024a).

chitectures 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.

F.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.

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

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

Parameter	Value
Base LLM	`google/gemma-2-2B`
Target layer	12 (residual stream output)
Activation dimension (m)	2,304
LLM batch size (activation generation)	4
Context length	1,024
LLM dtype	`torch.bfloat16`
SAE dictionary width (d_{sae})	$2^{16} = 65{,}536$
SAE learning rate	3×10^{-4}
Random seeds used	42, 43
Architectures evaluated	Standard, TopK, BatchTopK, Gated, P-Anneal, JumpReLU, Matryoshka BatchTopK
Sparsity-penalty sweep (L_1-based):	
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 L_0 / k sweep (L_0-based)	[20, 40, 80, 160, 320, 640]

Table 9: SAE training parameters for experiments on Gemma-2-2B (`google/gemma-2-2B`), layer 12. The larger SAE width (65,536 vs. 16,384 on Pythia-160M) makes PW-MCC computation more expensive, so we report results averaged over two seeds.

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.

A more granular view of these dynamics is presented in the detailed per-architecture analyses that follow (Figure 36, Figure 38, Figure 40, and Figure 42). The scatter plots in the lower-right panels consistently show a strong correlation between the decoder-vector similarity of matched feature pairs (cosine similarity) and their activation overlap across runs. This recurring pattern provides empirical evidence that PW-MCC, while being a dictionary-level metric, tracks consistency in activation behavior as well.

F.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 subpanel) tend to exhibit higher similarity. Figure 37 isolates and clearly depicts the positive relationship between shared activation frequency and pairwise similarity for this architecture.

F.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 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.

F.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 Fig-

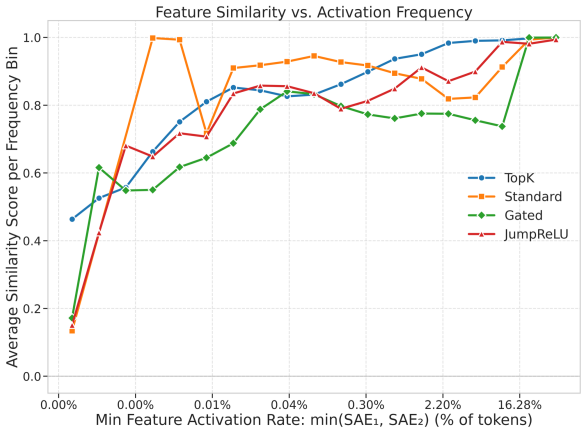

Figure 34: Average matched-pair similarity vs. shared activation rate across four SAE architectures (Pythia-160M, layer 8). Shared activation rate is $\min(\text{freq}_{\text{run 1}}, \text{freq}_{\text{run 2}})$. Higher shared activity corresponds to higher pairwise similarity across all architectures.

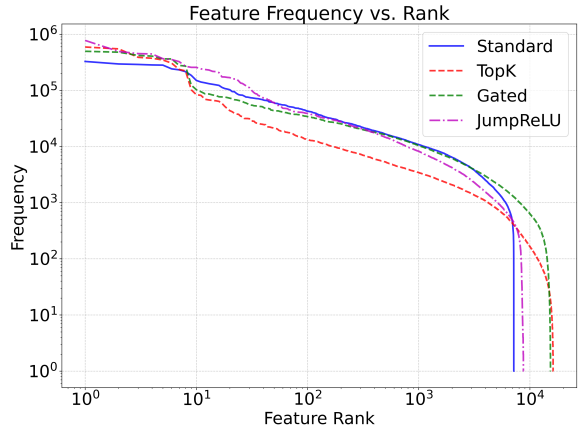

Figure 35: Feature activation frequency vs. rank for four SAE architectures (Pythia-160M, layer 8; log–log). All variants show power-law-like profiles but with different slopes and tail behavior; TopK and Gated have fewer dead features.

ure 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.

F.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 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.

F.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 (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

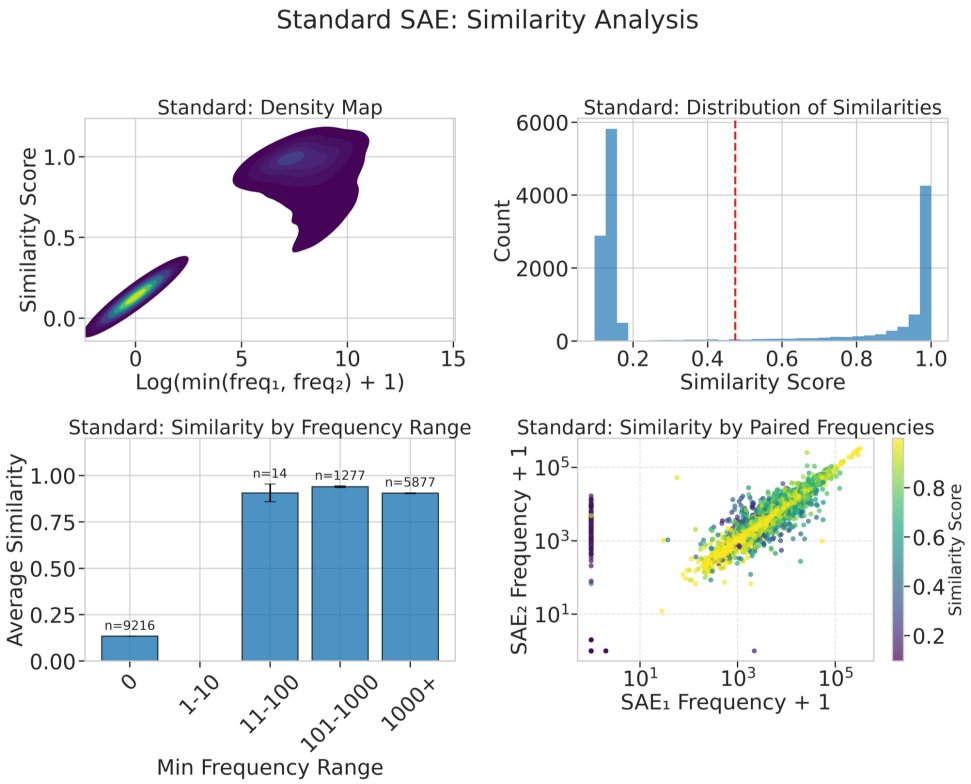

Figure 36: Feature similarity analysis for a Standard SAE (Pythia-160M, layer 8). Overall PW-MCC ≈ 0.47. *Top-left:* density map of pairwise similarity vs. log shared activation frequency. *Top-right:* histogram of pairwise similarity scores. *Bottom-left:* bucketed average similarity by log shared frequency. *Bottom-right:* scatter of feature activation frequencies from the two runs (colored by pairwise similarity); points near the diagonal indicate matched frequencies.

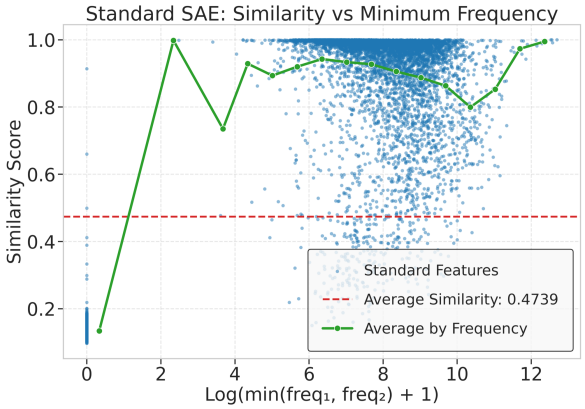

Figure 37: Average similarity vs. shared activation frequency (Standard SAE). Features with higher shared activation exhibit higher pairwise similarity, though overall consistency for this architecture is modest.

in determining the reliability and interpretability of features learned by SAEs from real-world LLM activations.

F.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 (Team et al., 2024b) 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 is 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 5×10^8 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 6. 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 (Ameisen et al., 2025; Lindsey

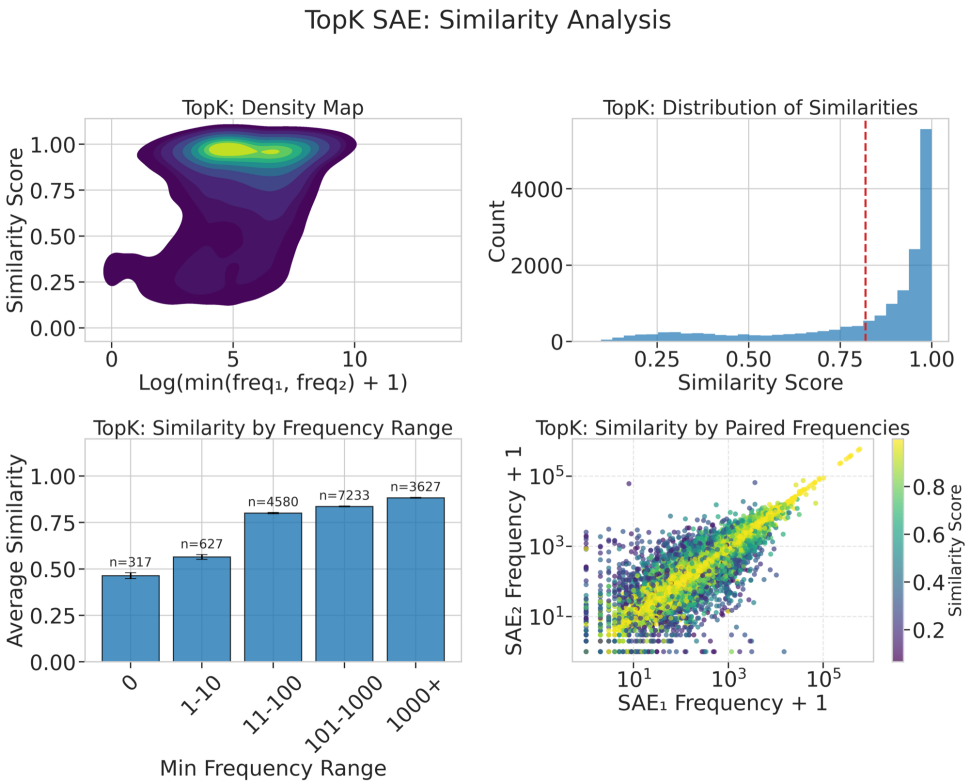

Figure 38: Feature similarity analysis for a TopK SAE (Pythia-160M, layer 8). Overall PW-MCC ≈ 0.82, the highest among the variants we evaluated. Panel layout is analogous to Fig. 36.

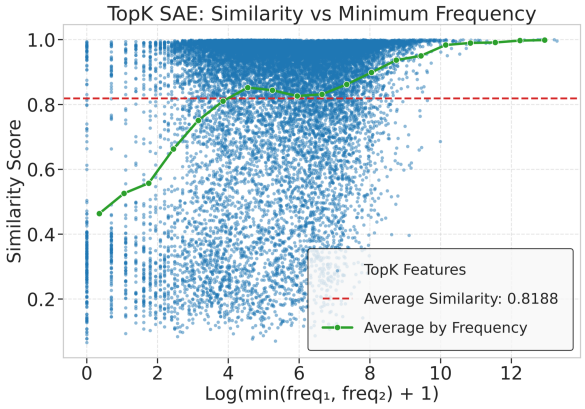

Figure 39: Average similarity vs. shared activation frequency (TopK SAE). High overall similarity and a strong positive correlation with shared activation frequency.

et al., 2025).

F.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 (Paulo et al., 2024), 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

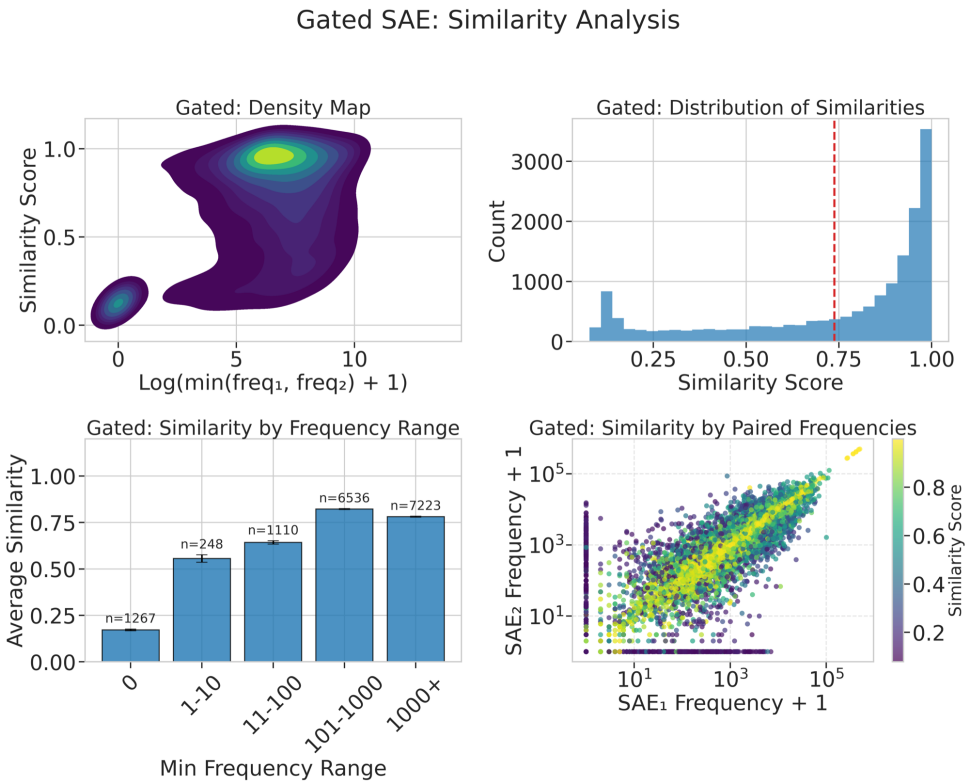

Figure 40: Feature similarity analysis for a Gated SAE (Pythia-160M, layer 8). Overall PW-MCC ≈ 0.74. Panel layout is analogous to Fig. 36.

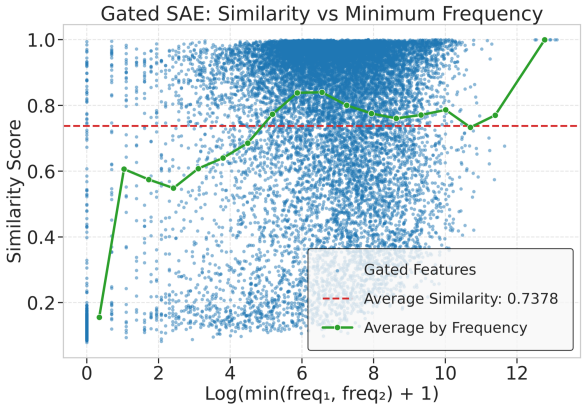

Figure 41: Average similarity vs. shared activation frequency (Gated SAE). Overall PW-MCC ≈ 0.74; a strong positive correlation between shared activation and similarity.

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 similarity between the two natural-language explanations of each matched pair, both explanations were provided to the same `gpt-4.1-2025-04-14` model. This model was then prompted to evaluate their semantic resemblance and assign an explanation-similarity score on a 1–10 scale. This score is presented as the **Explanation similarity** 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 similarity of their natural-language explanations produced by the automated interpretability pipeline. Feature pairs with low vector similarity (e.g., Buckets 1 and 2) typically receive divergent 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 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 is associated with stable natural-language explanations, though

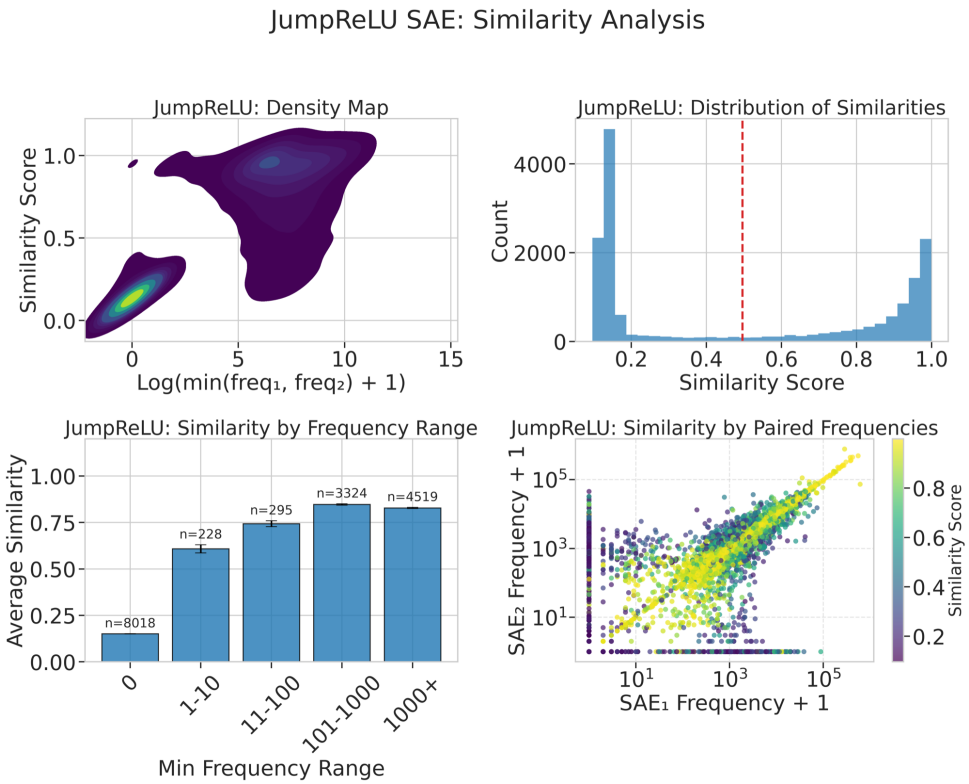

Figure 42: Feature similarity analysis for a JumpReLU SAE (Pythia-160M, layer 8). Overall PW-MCC ≈ 0.50. Panel layout is analogous to Fig. 36.

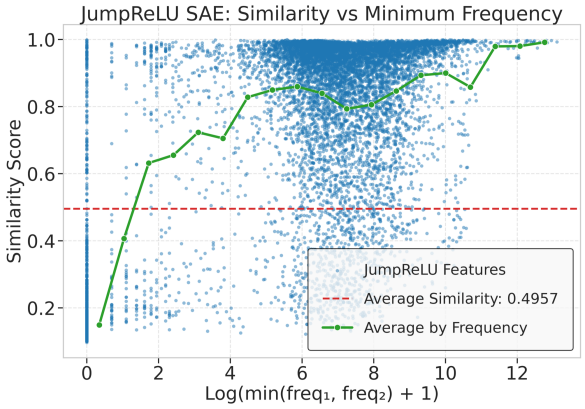

Figure 43: Average similarity vs. shared activation frequency (JumpReLU SAE). Similarity rises with shared activation frequency; two distinct clusters suggest subpopulations of features with differing learning or consistency characteristics.

this remains a partial proxy for downstream functional usefulness.

G 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(Paulo and Belrose, 2025). This view is bolstered by findings that multiple, incompatible mechanistic explanations can coexist for the same model behavior (Méloux et al., 2025; Makelov et al., 2023), 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(Lipton, 2018; Freiesleben et al., 2024). Initial exploration using single-run features certainly has value. However, for claims aspiring to scientific robustness—such as those underpinning causality (Marks et al., 2025; Geiger et al., 2023), safety verification (Abdaljalil et al.,

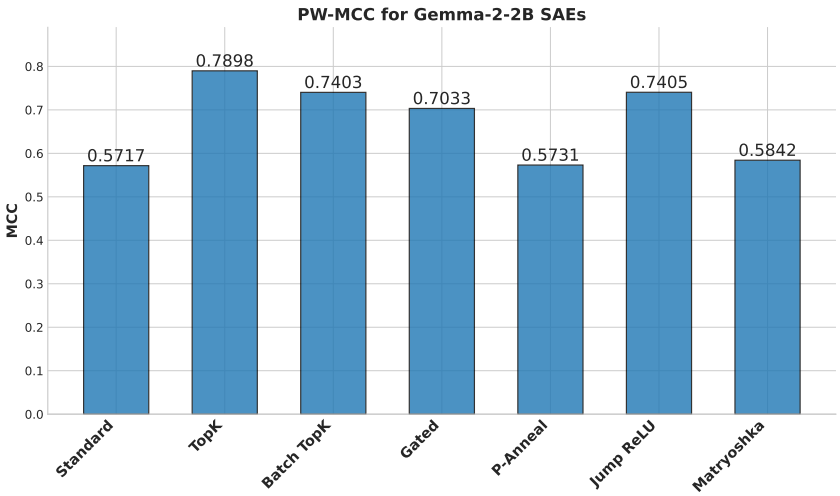

Figure 44: Final PW-MCC on Gemma-2-2B layer-12 activations. Seven SAE variants with dictionary width $d_{\text{sae}} = 65{,}536$; mean over two seeds. TopK again achieves the highest run-to-run feature consistency; JumpReLU is substantially more consistent here than on Pythia-160M.

2025; Härle et al., 2024), or canonical understanding (O'Brien et al., 2024; Olah, 2022)—sufficiently good stability must be quantitatively defined and verified. The current degree of instability in many applications is often underestimated (Paulo and Belrose, 2025). 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 (Olah et al., 2020). 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 (Olah et al., 2018). 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 (Fel et al., 2025), 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.

H Artifacts Used with License

Our training code is built upon the `dictionary_learning` package (Marks et al., 2024b), which is released under the MIT License. The models we used are Pythia-160M (Biderman et al., 2023) and Gemma-2-2B (Team et al., 2024b), which are licensed under the Apache 2.0 License and the Gemma License, respectively. Both licenses permit academic research use. The dataset we used is the uncopyrighted version of the Pile (Gao et al., 2020).

Bucket	Feat. IDs (SAE 1, SAE 2)	Expl. score (1–10)	SAE 1 feature explanation	SAE 2 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.

Table 10: Representative matched feature pairs across two independently trained TopK SAEs on Pythia-160M. Pairs are grouped into five buckets of increasing cosine similarity between decoder vectors. For each pair, we list feature indices in the two runs (SAE 1 and SAE 2), the LLM-rated similarity of their natural-language explanations on a 1–10 scale, and the explanations themselves. Low-similarity pairs (Buckets 1–2) tend to have unrelated behaviors, while high-similarity pairs (Buckets 4–5) describe nearly identical patterns across the two runs.