UNSTABLE FEATURES, REPRODUCIBLE SUBSPACES: UNDERSTANDING SEED DEPENDENCE IN SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse autoencoders (SAEs) are widely used to interpret neural network representations, but their utility depends on whether the learned features are reproducible across training runs. We study this question through *feature stability*: for each SAE feature, we estimate the probability that a similar feature reappears in an independently trained SAE. This yields a scalable per-feature signal that separates stable from unstable features. In a large-scale study across seeds, models, layers, dictionary sizes, and SAE variants, we find a pronounced functional asymmetry: stable features carry most of the reconstruction- and prediction-relevant signal, while unstable features have weak marginal impact and are dominated by low-frequency surface-form triggers in both activation statistics and automatic explanations. Geometrically, unstable features are individually non-reproducible but concentrate in reproducible lower-rank subspaces, suggesting that seed dependence often reflects basis ambiguity within a shared region of activation space rather than pure noise. A controlled synthetic model makes this mechanism explicit, showing that low-rank ground-truth features can be recovered at the subspace level while remaining non-identifiable as individual SAE latents across seeds. Finally, by pooling unique cross-seed features, we construct SAEs from stable features and find no evidence of a trade-off between stability and explained variance in this setting. Together, these results support cross-seed stability as a practical faithfulness filter and null check for SAE-based interpretability.

1 INTRODUCTION

Sparse autoencoders (SAEs) are a central tool in mechanistic interpretability because they aim to decompose model activations into sparse, human-interpretable features (Bricken et al., 2023; Cunningham et al., 2023; Gao et al., 2025). A basic question, however, is whether these features are reproducible: if SAEs trained on the same activations but with different random seeds learn substantially different dictionaries, then single-run interpretations become difficult to trust (Paulo & Belrose, 2025; Leask et al., 2025).

We study this problem through *feature stability*: the probability that a feature reappears in an independently trained SAE under a cosine-similarity matching rule. This gives a scalable per-feature signal that helps separate reproducible directions in activation space from seed-specific artifacts. Stability is not sufficient for faithfulness, but it is a useful filter and a stringent null check, especially given that automatic interpretation pipelines can assign plausible explanations even in settings where the underlying representations are random or non-canonical (Heap et al., 2025; Bhalla et al., 2026).

Using this signal, we compare stable and unstable features by their functional impact, token structure, and automatic explanations. We also analyze their geometry in decoder space, probe a low-rank mechanism in a controlled synthetic model, and test whether stable features pooled across runs can be used to construct seed-robust SAEs. We also compare pretrained-model SAEs against random-model controls, turning seed dependence from a nuisance into a measurable property that can be used both to evaluate interpretability claims and to construct more reproducible dictionaries.

Our contributions are as follows.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Functional asymmetry of stable and unstable features.** Stable features carry most of the reconstruction- and prediction-relevant signal, whereas unstable features have weak marginal impact and are dominated by low-frequency surface-form triggers in both activation statistics and automatic explanations.
- **Subspace recovery without feature identifiability.** Decoder-space analysis shows that unstable features are individually non-reproducible but collectively span reproducible lower-rank subspaces. In a controlled synthetic setting, low-rank ground-truth features are likewise recovered at the subspace level while failing to align with individual SAE latents.
- **No trade-off between stability and EV in feature-pool construction.** Using a deduplicated pool of unique features from multiple runs, we construct SAEs from stable cross-seed features and find that higher explained variance coincides with higher mean feature probability after tuning, suggesting that stability and reconstruction quality need not trade off in this construction setting.

2 Related Work

Feature stability. Paulo & Belrose (2025) showed that SAEs trained on the same model and data, differing only in random seed, can learn substantially different feature sets. Leask et al. (2025) argued that SAE latents are not canonical units of analysis, showing that features vary with dictionary size and can be merged across dictionaries. Fel et al. (2025) mitigate instability by anchoring dictionary atoms to data-derived archetypes. Gadgil et al. (2025) propose ensembling independently initialized SAEs via naive bagging and boosting. Our construction of an SAE from unique cross-seed features follows the same bagging intuition, but deduplicates the pooled feature set and selects a fixed-size subset of high-stability features rather than simply concatenating all latents from all runs. Chen et al. (2025) improve consistency by assigning features to activation-frequency groups and adapting encoder biases to enforce group-specific target frequencies. Cho et al. (2025) argue that external training data can be partially OOD for the base model and report that training on model-generated data improves cross-seed stability. Wang et al. (2025b) impose an ordered latent structure in which increasingly large prefixes of the feature set reconstruct the activation, reducing permutation non-identifiability and improving consistency. Heap et al. (2025) showed that SAEs trained on randomly initialized transformers can still yield plausible automatic interpretations, motivating random-model controls for SAE interpretability. Bhalla et al. (2026) give a complementary geometric account, arguing that multidimensional concepts can admit multiple valid SAE bases, making seed-dependent decompositions expected.

In this work, we estimate per-feature reappearance probabilities to study instability through seed-dependent basis choices within reproducible lower-rank decoder subspaces, probing this view with a controlled low-rank synthetic model and comparing stable and unstable features by reconstruction impact, downstream effects, and automatic explanations.

Finding similar features across SAEs. Balcells et al. (2024) track how individual SAE features evolve between adjacent layers, while Balagansky et al. (2025b) introduce a data-free method for matching SAE features across layers. Laptev et al. (2025) extend this line by constructing cross-layer feature-flow graphs. Wang et al. (2025a) identify corresponding SAE features across different language-model architectures. Anonymous Authors (2026) bridge the stability and matching literatures by benchmarking cross-seed SAE correspondence methods, including cosine and optimal-transport matchers, and evaluating them functionally via ablation effects and substitution tests.

3 Preliminaries

3.1 Sparse Autoencoders

A sparse autoencoder (SAE) is designed to represent hidden states as a sparse linear combination of feature embeddings (Bricken et al., 2023; Cunningham et al., 2023). Each coefficient is treated as an activation magnitude, or importance of this feature, which are assumed to be non-negative.

Architecturally, an SAE has the following form:

$$\begin{aligned} \boldsymbol{z} &= \sigma(\boldsymbol{W}_{\text{enc}}\boldsymbol{h} + \boldsymbol{b}_{\text{enc}}) \in \mathbb{R}^F, \\ \hat{\boldsymbol{h}} &= \boldsymbol{W}_{\text{dec}}\boldsymbol{z} + \boldsymbol{b}_{\text{dec}} \in \mathbb{R}^d, \end{aligned} \tag{1}$$

where $\boldsymbol{h}$ is a hidden state vector, $\boldsymbol{z}$ is a vector of feature activation magnitudes, and σ is a nonlinear activation function that induces sparsity and non-negativity. SAEs are trained on reconstruction loss plus optional regularization loss that controls sparsity level scaled by coefficient $\alpha \geq 0$:

$$\mathcal{L} = \|\boldsymbol{h} - \hat{\boldsymbol{h}}\|_2^2 + \alpha\mathcal{L}_{\text{reg}}(\boldsymbol{z}). \tag{2}$$

Vanilla SAE (Cunningham et al., 2023) simply uses ReLU as nonlinearity and $\mathcal{L}_{\text{reg}} = \|\boldsymbol{z}\|_1$. Widely used *TopK SAE* (Gao et al., 2025) imposes sparsity on $\boldsymbol{z}$ by zeroing out all entries whose values are outside the top-k elements, and *Batch TopK SAE* (Bussmann et al., 2024) operates on the batch level instead of single-entry and zeroes out elements smaller than top-(batch size $\times$ k). These activations allow to fix the desired sparsity level and drop the $\mathcal{L}_{\text{reg}}$ from loss function. ***JumpReLU SAE*** (Rajamanoharan et al., 2024) modifies Vanilla SAE by introducing learnable threshold parameter $\theta \in \mathbb{R}^F$ that independently controls the minimal activation value for each feature, and replacing nonlinearity with $\mathrm{JumpReLU}(\boldsymbol{x}) = \boldsymbol{x}\mathrm{H}(\boldsymbol{x} - \theta)$, where H is the Heaviside function.

We also evaluate *HierarchicalTopK SAE* (Balagansky et al., 2025a), which trains one dictionary across nested sparsity budgets so that increasingly large prefixes of the selected features are optimized to reconstruct the activation.

We denote $\boldsymbol{f}_i$ as the entity corresponding to the feature with index i, and $\boldsymbol{z}_i$ as the activation value of that feature. Its embedding $\boldsymbol{e}_i \in \mathbb{R}^d$ is the ith column of the decoder matrix, and we assume that all embeddings lie on a unit sphere (see Appendix A for normalization details).

3.2 Feature Matching and Stability

Two SAEs trained on similar hidden state distribution (e.g., on the same layer) are expected to converge towards similar representations of the hidden states and therefore similar feature representations. Since SAEs are invariant to feature ordering and two different initializations might produce same dictionaries, but with different indices (Balagansky et al., 2025b), a linear assignment problem is usually solved (Paulo & Belrose, 2025; Balagansky et al., 2025b; Fel et al., 2025): given two dictionaries $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, we seek for a permutation matrix $\boldsymbol{\Pi}$ so that discrepancy between dictionaries $\boldsymbol{D}_1$ and $\boldsymbol{\Pi}\boldsymbol{D}_2$ is minimized. This gives a one-to-one correspondence. The quality of resulting matching is a measure of SAE stability - if it is low, then features are mostly different. This is primary method for previous works on the problem we investigate in this paper.

In this work we adopt a many-to-one alternative: for each element from $\boldsymbol{D}_1$, we find the most similar element from $\boldsymbol{D}_2$, measured by maximum cosine similarity. This approach is much less computationally expensive and allows to compute stability of individual features, helping us understand how their individual properties correspond to the problem of instability.

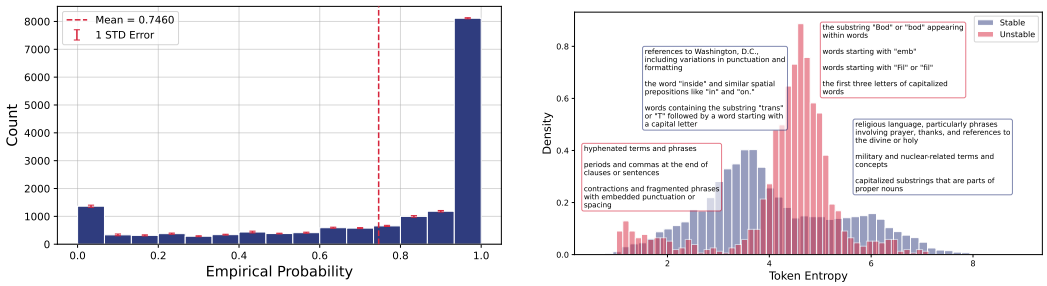
As a robustness check, replacing this argmax-cosine rule with one-to-one Hungarian matching yields nearly identical matched feature sets (IoU $= 0.978 \pm 0.001$).

4 Methodology

We study how reliably individual SAE features reappear across random initializations. Throughout, we represent each feature by its decoder vector, and ℓ_2-normalize all decoder columns so that cosine similarity reduces to a dot product.

Feature matching. Given two SAEs A and B with decoder columns $\{\boldsymbol{e}_i^{(A)}\}_{i=1}^F$ and $\{\boldsymbol{e}_j^{(B)}\}_{j=1}^F$, we say that features i and j *match* if $\cos\big(\boldsymbol{e}_i^{(A)}, \boldsymbol{e}_j^{(B)}\big) \geq \theta$, following Leask et al. (2025), $\theta = 0.7$ unless stated otherwise. See Appendix B.7 for results across reappearance probabilities.

Reappearance probability. We train $N{+}1$ SAEs on identical data and hyperparameters with different random seeds, and choose one run as an anchor ($k = 0$). For each anchor feature embedding $\boldsymbol{e}_i^{(0)}$, we



(a) Feature stability in the main setup. Empirical reappearance rates $\hat{p}$ across seeds. The dashed line marks the mean. Error bars show variability across five anchors.

(b) Token diversity in the main setup. Token entropy H_i for stable and unstable features, with representative feature interpretations.

Figure 1: **Feature stability and token diversity in the main setup.**

count how many of the other N SAEs contain a feature with cosine similarity at least θ:

$$X_{0,i} = \sum_{k=1}^{N} \mathbf{1}\left\{ \max_{j \in \{1,\dots,F\}} \cos\big(\boldsymbol{e}_i^{(0)}, \boldsymbol{e}_j^{(k)}\big) \geq \theta \right\}. \tag{3}$$

Let

$$p_i := \mathrm{P}\left(\max_j \cos(\boldsymbol{e}_i^{(0)}, \boldsymbol{e}_j^{(k)}) \geq \theta \right), \; k \neq 0.$$

Assuming independent seeds, $X_{0,i} \sim \mathrm{Binomial}(N, p_i)$, and we estimate p_i by $\hat{p}_i = \frac{X_{0,i}}{N}$. Across features, we use the empirical CDF $\hat{\mathbf{F}}(p) = \frac{1}{F}\sum_{i=1}^{F} \mathbf{1}\{\hat{p}_i \leq p\}$, which converges uniformly to $\mathbf{F}(p)$ by the theorem of Glivenko–Cantelli (Glivenko, 1933; Cantelli, 1933).

Endpoint stability. We focus on *endpoint* behavior: features that reappear in almost none or almost all runs. Fixing $\varepsilon = 0.05$, we call a feature *unstable* if $\hat{p}(\boldsymbol{f}_i^{(0)}) \leq \varepsilon$ and *stable* if $\hat{p}(\boldsymbol{f}_i^{(0)}) \geq 1-\varepsilon$. Our main task is therefore distilled into finding $\mathbf{F}(\varepsilon)$ and $\mathbf{F}(1-\varepsilon)$, via empirical distribution function.

Unique feature pool. For convenience, we can also introduce a Unique feature pool, which consists of pairs of SAE id S and feature index i,

$$\mathbb{U} \subseteq \{(S,i)\} \text{ s.t.} \quad \forall\, (S_1,i_1) \neq (S_2,i_2) \in \mathbb{U} : \cos\big(\boldsymbol{e}_{i_1}^{(S_1)}, \boldsymbol{e}_{i_2}^{(S_2)}\big) < \theta. \tag{4}$$

This set of features allows us to construct new SAEs from this pool, but with different feature probability distribution. See Section 5.3.

5 Quantitative Analysis

Experimental Setup. Unless stated otherwise, we report results in a single, well-instrumented configuration: we train 96 TopK SAEs (so $N = 95$ comparisons per anchor) on GPT-2 activations at the residual stream on layer 7, with TopK $= 64$ and dictionary size $F = 2^{14}$. Figure 1a summarizes the resulting empirical reappearance rates $\hat{p}$. The distribution is sharply concentrated near the endpoints: a large mass of features is recovered in nearly every run, while a smaller but clearly separated group is recovered in nearly none. The remaining features occupy a comparatively thin middle region (mean $\hat{p} \approx 0.75$ in this setting). Higher θ lowers the stable fraction and raises the unstable fraction. Vanilla remains most stable, followed by JumpReLU, with TopK and BatchTopK nearly identical. See Appendix B.2 for sensitivity results.

We compare stable and unstable features by activation statistics, token diversity, and functional impact.

Throughout, we use the partitions from Section 4 at $\varepsilon = 0.05$:

$$\mathcal{U}_\varepsilon = \{i \mid \hat{p}(\boldsymbol{f}_i^{(0)}) \leq \varepsilon\}, \qquad \mathcal{S}_\varepsilon = \{i \mid \hat{p}(\boldsymbol{f}_i^{(0)}) \geq 1-\varepsilon\}.$$

We evaluate on held-out tokens. See Appendix B for definitions and protocols.

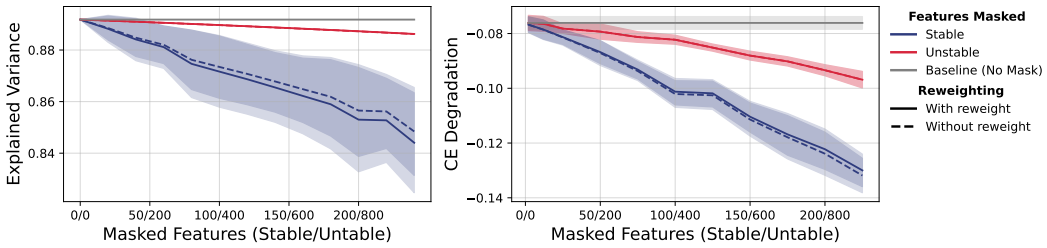


Figure 2: **Impact of masking stable vs. unstable features on reconstruction and next-token loss. Left**: explained variance (EV) under feature masking. **Right**: change in next-token loss under activation patching with masked-feature reconstructions. We mask N stable features and $4N$ unstable features to approximately match expected active mass. See Appendices B.5 and B.6 for protocol details.

5.1 Activation Statistics and Token Structure

Activation frequency and magnitude. We measure activation frequency ω_i and mean magnitude when active μ_i. Unstable features activate less often (0.18% vs. 0.44%), while stable features have a heavier high-magnitude tail. See Appendix B.3 for definitions and distributions.

Token entropy (token diversity). For each feature i, token entropy H_i measures the diversity of token IDs at positions where $z_{n,i} > 0$. Low entropy indicates concentrated lexical support. See Appendix B.4 for calculation details.

Crucially, *both* stable and unstable features exhibit two dominant entropy regimes, but with different content. For unstable features, the lowest-entropy mode is dominated by punctuation/formatting triggers, while a higher-entropy mode corresponds to short subword fragments and brittle substrings that match across many tokens. For stable features, a lower-entropy mode (typically higher than the unstable punctuation regime) corresponds to single words and tight synonym clusters, while the highest-entropy mode corresponds to higher-level concepts with many lexical realizations. Representative interpretations for these regimes are shown as callouts in Figure 1b.

Automatic interpretability. Stable features receive higher SAEBench detection scores. Their fraction of perfect scores is about $4.5\times$ that of unstable features. Their explanations emphasize phrases and semantic structure, whereas unstable-feature explanations emphasize substrings and surface form. See Appendix E for score distributions and examples. Among unstable versus stable explanations, `substring` occurs in 38.9% versus 11.3%, and `phrase` in 4.1% versus 32.0%. See Appendix E.5 for full statistics. GPT-5 predicts stability from explanation text with 0.88 accuracy. See Appendix E.6 for the protocol. Within matched log-frequency bins, stable features retain higher detection scores and usually larger masking effects. See Appendix B.8 for the full comparison.

5.2 Impact on Reconstruction and Next-Token Loss

Frequency-matched masking protocol. To approximately match active mass, we randomly mask N stable or $4N$ unstable features. See Appendix B.5 for sampling, uncertainty, and reweighting details.

Explained variance (EV). Masking $4N$ unstable features reduces EV only slightly, whereas masking N stable features causes a much larger drop (Figure 2, left). Stable features therefore account for most reconstruction-relevant variance. See Appendix B.5 for the EV definition.

Next-token loss under activation patching. We replace residual-stream activations with masked-feature reconstructions and measure $\mathrm{CE}_{\text{base}} - \mathrm{CE}_{\text{patched}}$. Masking unstable features produces a modest loss change, while masking fewer stable features causes much larger degradation (Figure 2, right). See Appendix B.6 for patching details.

Overall, unstable features behave like low-frequency lexical artifacts with limited functional impact, whereas stable features capture most of the reconstruction- and prediction-relevant structure. Across five equal-width $\hat{p}$ bins with matched active mass, the $[0.8, 1.0]$ bin has the largest impact. See Appendix B.7 for the full results.

5.3 Constructing SAE from Unique Features

We pool decoder features from n independently seeded SAEs, deduplicating them using Eq. 4. We estimate each pooled feature's reappearance probability $\hat{p}$ and initialize SAEs with $F = 16{,}384$ latents using the most probable features, the least probable features, or a uniform (*equiprobable*) sample. We then briefly tune each SAE. See Appendix B.12 for training costs.

Modest source pools yield dictionaries dominated by stable features (Figure 3). Brief tuning restores near-baseline EV for the most-probable initialization. The equiprobable baseline performs worse, and the least-probable initialization performs worst. Larger pools shift the most-probable selection toward higher reappearance rates and the least-probable selection toward seed-specific features. See Appendices B.9 and B.10 for tuning curves and probability distributions, and Appendix E.7 for interpretation results.

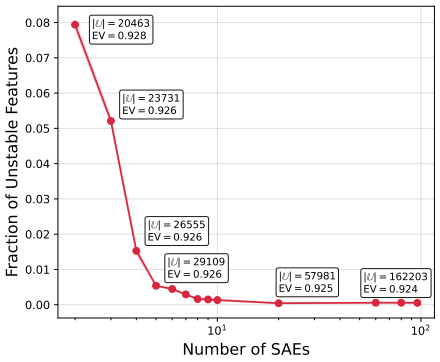
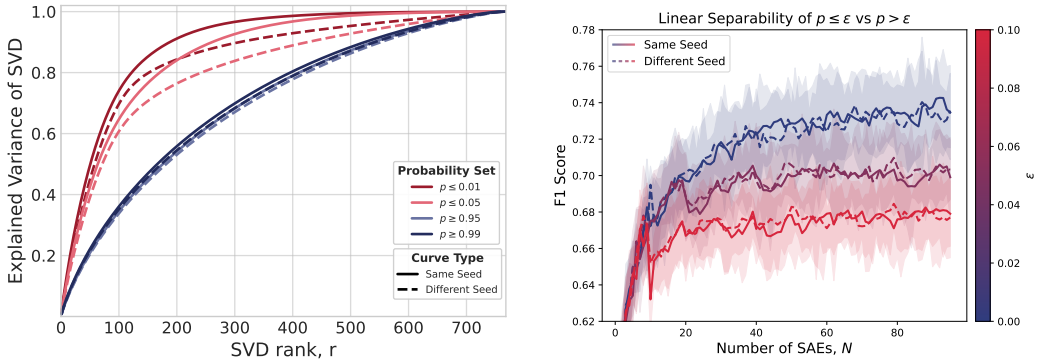


Figure 3: **Constructing a stable dictionary from multiple SAE runs.** Selecting the most probable features rapidly suppresses unstable features as the source pool grows. Boxes report pool size and explained variance.

The most-probable construction also remains competitive with a standard TopK SAE on SAEBench metrics (Appendix B.11, Table 5), indicating that the pooled construction improves feature stability without sacrificing downstream interpretability performance. Appendix C further shows that post-training a high-probability feature pool shifts some features toward lower reappearance probabilities, suggesting that lower-stability directions can be reconstruction-useful rather than mere random artifacts.

6 Geometric Analysis

We now investigate features geometrically at the level of decoder vectors $e_j^{(s)} \in \mathbb{R}^d$. Our goal is to show that: (i) unstable features are approximately linearly separable from the rest in decoder space, (ii) unstable features concentrate in a lower effective-dimensional subspace than stable features, and (iii) the *subspace* spanned by unstable features exhibits a high degree of cross-seed reproducibility.



(a) Singular-value explained variance within a seed (solid) and under cross-seed subspace transfer (dashed).

(b) Within-seed (solid) and cross-seed (dashed) F1 for separating unstable features from all others.

Figure 4: **Seed-consistent geometry of unstable features.** Left: feature subspaces transfer across seeds. Right: a linear separating direction also transfers across seeds.

6.1 Linear Separability in Decoder Space

We first ask whether the decoder vectors themselves carry a geometric signature of instability. As early, for each seed s let $\hat{p}(\boldsymbol{f}_i^{(s)})$ denote the probability of feature i and define the set of unstable features as $\mathcal{U}_\varepsilon^{(s)} = \{i : \hat{p}(\boldsymbol{f}_i^{(s)}) \le \varepsilon\}$.

We train a logistic regression classifier to predict the binary label $y_i^{(s)} = \mathbf{1}\{i \in \mathcal{U}_\varepsilon^{(s)}\}$ from the corresponding decoder vector $e_i^{(s)} \in \mathbb{R}^d$. In addition to within-seed evaluation, we test transferability by applying a classifier trained on one seed to the decoder vectors of another seed.

We evaluate performance using the F1 score (Figure 4b) as a function of N, the number of independently trained SAEs used to estimate the probabilities $\hat{p}(\boldsymbol{f}_i^{(s)})$. We find that the within-seed and cross-seed F1 scores are very close, and for sufficiently large N they saturate at approximately 0.73 for $\varepsilon = 0$ and 0.67 for $\varepsilon = 0.1$.

6.2 Overlap and Effective Rank of Subspaces

We test whether unstable features span subspaces that are low-dimensional and reproducible across seeds.

Let $X_{\mathcal{I}_\varepsilon}^{(s)} = \boldsymbol{W}_{\mathrm{dec}}^{(s)}[:, \mathcal{I}_\varepsilon^{(s)}]$ contain the decoder columns for stable or unstable features in seed s. The within-seed quantity $\mathrm{EV}_{\mathrm{SVD}}^{(s)}(r)$ measures variance captured by its top-r singular subspace. The cross-seed analogue $\mathrm{EV}_{\mathrm{SVD}}^{(a\to s)}(r)$ uses the subspace from seed a to explain seed s.

Figure 4a compares within-seed and cross-seed EV over all seed pairs among 96 SAEs, for stable and unstable sets at $\varepsilon \in \{0.01, 0.05\}$. Unstable features require fewer principal components and have lower effective rank. Singular subspaces also transfer accurately across seeds, including for unstable features. See Appendix F for formal definitions and size-matched rank estimates.

Taken together, although individual unstable features rarely reappear across seeds, the low-rank subspaces spanned by them are reproducible. This suggests that feature instability is primarily driven by seed-dependent mixing within a shared subspace, rather than changes in the subspace itself.

7 A Controlled Low-Rank Synthetic Model

To test whether shared low-rank structure alone can generate this pattern, we study a synthetic dictionary in which only a subset of the ground-truth features is constrained to a common low-dimensional subspace. Concretely, we define

$$W = \begin{bmatrix} D \\ UV \end{bmatrix}, \quad D \in \mathbb{R}^{n_{\mathrm{full}} \times d}, \quad U \in \mathbb{R}^{n_{\mathrm{low}} \times r}, \quad V \in \mathbb{R}^{r \times d}, \quad r < d,$$

where the rows of D are generic full-rank features and the rows of UV lie in the shared subspace $L = \mathrm{rowspan}(V) \subset \mathbb{R}^d$. We generate synthetic activations by sampling k rows of W uniformly without replacement, summing them with unit coefficients, and training TopK SAEs from multiple random seeds to reconstruct the resulting vectors.

This setup isolates a concrete mechanism for instability. If the full-rank features are identifiable as individual directions, they should be recovered consistently across seeds. By contrast, if the low-rank block is identifiable mainly at the level of its span, different seeds can learn different mixtures within L: individual low-rank features need not match, even though the subspace itself is reproducible.

Figure 5 confirms this prediction in the main synthetic setting ($d = 32$, $r = 2$, $k = 8$). The full-rank features are recovered stably, with reappearance probabilities close to 1 and cosine similarity to matched ground-truth features also close to 1. In contrast, the low-rank block has much lower mean reappearance probability (≈ 0.25) and substantially worse one-to-one recovery. This pattern persists across settings. The learned low-rank block has lower effective rank and subspace similarity above the random baseline. See Appendix D for additional results.

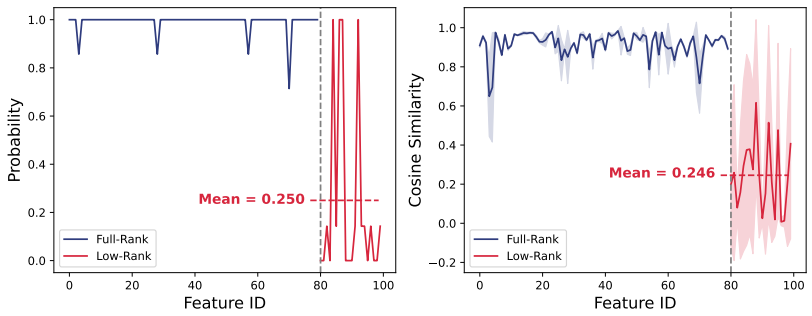


Figure 5: **Controlled low-rank model ($d = 32$, $r = 2$, $k = 8$).** Full-rank features (blue) are recovered consistently, whereas features in the shared low-rank subspace (red) are unstable as individual directions despite the reproducibility of their subspace.

This controlled model therefore reproduces the core geometric phenomenon from real SAEs: instability can arise from seed-dependent basis choices within a reproducible low-rank subspace, rather than from the absence of shared underlying structure.

8 Other Setups

We test stability across base models, layers, dictionary sizes, SAE nonlinearities, training budgets, and a random-model control.

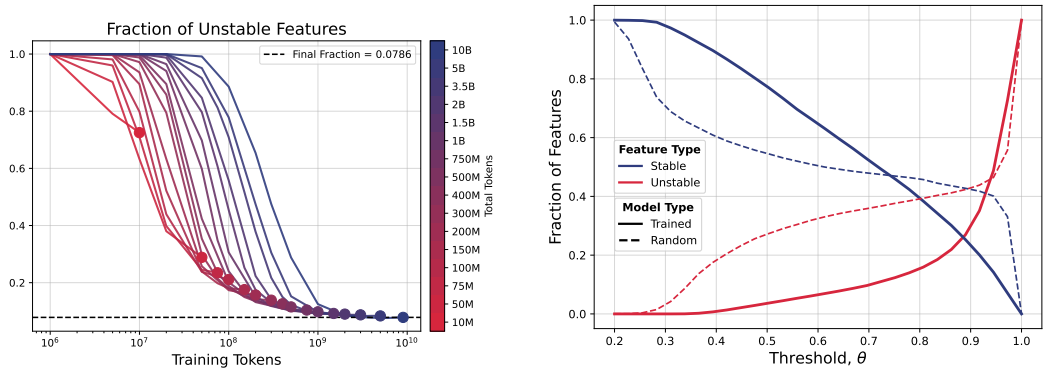
Across base models, layers, and dictionary sizes. TopK SAEs retain stable and unstable subsets across evaluated models, layers, and dictionary sizes. Instability generally decreases with depth in GPT-2 and Gemma-2, while some Pythia settings deviate. See Appendix G.2 for endpoint fractions. See Appendix G.1 for Qwen2.5-7B results and subspace-transfer replications. The full GPT-2 layer-wise trajectory and next-layer feature matching are shown in Appendix G.2

Effect of sparsity mechanism and reconstruction quality. Table 1 compares SAE variants on GPT-2 layer 7 at fixed $F = 2^{14}$, including EV. TopK and BatchTopK are nearly indistinguishable both in stability and EV (≈ 0.892), suggesting that the dominant source of seed dependence is the hard k-sparsity constraint itself rather than whether it is imposed per-token or per-batch. Within TopK SAEs, increasing k improves EV but monotonically increases the unstable fraction and decreases the stable fraction, consistent with stronger sparsity constraints reducing seed-dependent degrees of freedom. Vanilla (ReLU+ℓ_1) is extremely stable (near-zero unstable fraction), but at comparable sparsity it has substantially lower EV than TopK, highlighting a clear tradeoff between stability and reconstruction quality when moving along the choice of sparsity/activation mechanism. HierarchicalTopK also improves stability relative to TopK but lowers EV, consistent with additional structure in the sparsity objective reducing seed dependence at some reconstruction cost. JumpReLU provides an intermediate point on this tradeoff curve: it improves stability relative to TopK while achieving EV close to TopK at higher sparsities.

Table 1: SAE type comparison on GPT-2 at layer 7.

SAE type	Sparsity	EV	Unstable	Stable
Vanilla (ReLU + ℓ_1)	62	0.833	0.003	0.810
	91	0.856	0.006	0.764
TopK	32	0.862	0.053	0.619
	48	0.882	0.080	0.560
	64	0.892	0.098	0.522
	80	0.902	0.118	0.499
BatchTopK	64	0.892	0.097	0.523
HierarchicalTopK	64	0.876	0.028	0.716
JumpReLU	29	0.841	0.049	0.624
	58	0.884	0.034	0.620
	93	0.902	0.038	0.602

More SAE training does not remove instability. Figure 6a shows the unstable fraction in the main TopK setting as a function of total SAE training tokens from 10M to 10B. The unstable fraction



(a) Unstable-feature fraction versus SAE training tokens.

(b) Stable and unstable fractions for pretrained and random GPT-2.

Figure 6: **Training dynamics and the random-model control.** Instability plateaus with additional SAE training (left), while the stability distribution sharply distinguishes pretrained from random-model activations (right).

decreases early in training and then approaches a clear non-zero plateau ($\approx 8\%$ at 10B tokens in this setting). The stable fraction rises with training but shows diminishing returns. See Appendix G.3 for the curve and setup.

Dead-salmon baseline: trained vs. random transformers. Finally, we repeat the same SAE training procedure on activations from a randomly initialized GPT-2 (same architecture and SAE hyperparameters). Our stability metric sharply distinguishes these cases: Figure 6b compares stability as a function of the cosine matching threshold θ for SAEs trained on a trained GPT-2 versus a random GPT-2. Across a wide range of θ, the trained-model SAEs retain a large stable subset, whereas in the random-model setting the stable fraction collapses and instability dominates. Importantly, automatic interpretation alone would not reliably flag this failure mode: SAEs trained on the random model can still achieve high detection scores (see Appendix G.4 for score distributions). Together, these results motivate cross-seed recurrence as a stringent null check that helps separate reproducible directions in a trained model's activation space from seed-dependent artifacts of the SAE objective.

9 Conclusion

Our results establish feature stability, measured via cross-run reappearance probability, as a critical lens for sparse autoencoder interpretability. Stability cleanly partitions features into two regimes: a high-impact stable subset that captures structural and compositional patterns essential for reconstruction and prediction, and a low-impact unstable subset that behaves as brittle, seed-sensitive artifacts with surface-level triggers. Crucially, instability is not random noise. Unstable features concentrate in a reproducible low-rank subspace, suggesting seed dependence often reflects arbitrary mixing within a shared region of activation space. Our controlled toy model makes this mechanism explicit, showing that low-rank ground-truth features can be recovered at the subspace level while remaining non-identifiable as individual SAE latents across seeds. This insight motivates a methodological shift. Researchers should report stability distributions rather than single-run artifacts, prioritize stable features for mechanistic claims and interventions, and treat unstable features as a distinct low-rank regime rather than interchangeable discoveries. Dead-salmon controls demonstrate that automatic interpretability pipelines can yield compelling but unstable features even in random models, underscoring that cross-seed recurrence is a necessary complement to plausibility-based validation. Stability provides a scalable, model-agnostic filter that increases the reliability of sparse autoencoder conclusions. By pooling stable features across seeds and applying brief post-training, researchers can construct seed-robust dictionaries without sacrificing reconstruction quality. While stability does not replace causal validation, it offers a practical foundation for reproducible science, enabling shareable feature dictionaries and ensuring that mechanistic insights reflect properties of the model rather than artifacts of optimization.

AI USE STATEMENT

We used generative AI for coding assistance and to edit and polish individual paragraphs. We did not use it to generate experimental ideas, prepare data, interpret experimental results, or add related work. Our experimental use of LLMs is described in Appendix E. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 defines feature matching and stability estimation. Appendices A and B describe normalization and evaluation protocols. Appendices E and F document interpretation and geometric analyses. Upon acceptance, we plan to publicly release a dataset of trained SAE checkpoints across different random seeds, layers, activation functions, and base models.

REFERENCES

Anonymous Authors. Benchmarking cross-seed feature correspondence in sparse autoencoders. Under review for TMLR, OpenReview, 2026. URL https://openreview.net/forum?id=5cy6WtSC8f. OpenReview submission.

Nikita Balagansky, Yaroslav Aksenov, Daniil Laptev, Vadim Kurochkin, Gleb Gerasimov, Nikita Koriagin, and Daniil Gavrilov. Train one sparse autoencoder across multiple sparsity budgets to preserve interpretability and accuracy. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10171–10179. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.emnlp-main.515. URL https://aclanthology.org/2025.emnlp-main.515/.

Nikita Balagansky, Ian Maksimov, and Daniil Gavrilov. Mechanistic permutability: Match features across layers. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=MDvecs7EvO.

Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. Evolution of SAE features across layers in LLMs, 2024. URL https://arxiv.org/abs/2410.08869. Presented at the Attributing Model Behavior at Scale (ATTRIB) workshop at NeurIPS 2024.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Owen Lewis, Thomas McGrath, Jack Merullo, Atticus Geiger, and Ekdeep Singh Lubana. Do sparse autoencoders capture concept manifolds?, 2026. URL https://arxiv.org/abs/2604.28119.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders, 2024. URL https://arxiv.org/abs/2412.06410.

Francesco Paolo Cantelli. Sulla determinazione empirica delle leggi di probabilità. *Giornale dell'Istituto Italiano degli Attuari*, 4:421–424, 1933.

Siyu Chen, Heejune Sheen, Xuyuan Xiong, Tianhao Wang, and Zhuoran Yang. Taming polysemanticity in LLMs: Provable feature recovery via sparse autoencoders, 2025. URL https://arxiv.org/abs/2506.14002.

Seonglae Cho, Harryn Oh, Donghyun Lee, Luis Eduardo Rodrigues Vieira, Andrew Bermingham, and Ziad El Sayed. FaithfulSAE: Towards capturing faithful features with sparse autoencoders without external dataset dependencies, 2025. URL https://arxiv.org/abs/2506.17673.

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

Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E. Ba, and Talia Konkle. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9v1eW8HgMU.

Soham Gadgil, Chris Lin, and Su-In Lee. Ensembling sparse autoencoders, 2025. URL https://arxiv.org/abs/2505.16077.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Valery Glivenko. Sulla determinazione empirica delle leggi di probabilità. *Giornale dell'Istituto Italiano degli Attuari*, 4:92–99, 1933.

Thomas Heap, Tim Lawson, Lucy Farnik, and Laurence Aitchison. Automated interpretability metrics do not distinguish trained and random transformers, 2025. URL https://arxiv.org/abs/2501.17727.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *arXiv preprint arXiv: 2503.09532*, 2025. URL https://arxiv.org/abs/2503.09532.

Daniil Laptev, Nikita Balagansky, Yaroslav Aksenov, and Daniil Gavrilov. Analyze feature flow to enhance interpretation and steering in language models, 2025. URL https://arxiv.org/abs/2502.03032.

Patrick Leask, Bart Bussmann, Michael T. Pearce, Joseph Isaac Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9ca9eHNrdH.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features, 2025. URL https://arxiv.org/abs/2501.16615.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024. URL https://arxiv.org/abs/2407.14435.

Ashwin Saraswatula and David Klindt. Data whitening improves sparse autoencoder learning, 2025. URL https://arxiv.org/abs/2511.13981.

Junxuan Wang, Xuyang Ge, Wentao Shu, Qiong Tang, Yunhua Zhou, Zhengfu He, and Xipeng Qiu. Towards universality: Studying mechanistic similarity across language model architectures. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=2J18i8T0oI.

Sophie L. Wang, Alex Quach, Nithin Parsan, and John J. Yang. Enforcing orderedness to improve feature consistency, 2025b. URL https://arxiv.org/abs/2512.02194.

A SAE TRAINING DETAILS

A.1 SAE VARIANTS

Vanilla SAE (Cunningham et al., 2023) uses ReLU as the nonlinearity and $\mathcal{L}_{\text{reg}} = \|\boldsymbol{z}\|_1$. *TopK SAE* (Gao et al., 2025) imposes sparsity on $\boldsymbol{z}$ by zeroing out all entries outside the top-k values. *Batch TopK SAE* (Bussmann et al., 2024) applies the same idea at the batch level, zeroing out elements below the top-(batch size $\times$ k) threshold; these activations fix the desired sparsity level and remove the need for an explicit regularization term. *HierarchicalTopK SAE* (Balagansky et al., 2025a) trains a single dictionary across nested sparsity budgets, so increasingly large prefixes of the selected features are optimized to reconstruct the activation. *JumpReLU SAE* (Rajamanoharan et al., 2024) modifies the Vanilla SAE by learning a threshold $\theta_i \in \mathbb{R}$ for each feature and using $\mathrm{JumpReLU}(\boldsymbol{x}) = \boldsymbol{x}\mathrm{H}(\boldsymbol{x} - \theta)$, where H is the Heaviside function.

A.2 SAE TRAINING SETUP

Unless otherwise stated, each SAE is trained on 1B token activations sampled from the `sample-10BT` split of FineWeb. We train all seeds with the same activation sampling order: at each optimization step, we maintain an activation store containing 8192×32 token activations and sample 8192 token positions uniformly at random from this store. The random seed controlling this sampling procedure is shared across SAE runs, so differences between runs are due to SAE initialization only.

We use an auxiliary loss for dead features, following Gao et al. (2025). A feature is marked as *dead* if it has not activated for more than 20 consecutive batches. The auxiliary loss gives such dead features a learning signal by asking them to help reconstruct the residual error left by the main active features. Concretely, after computing the standard SAE reconstruction $\hat{\boldsymbol{x}}$, we form the residual

$$\boldsymbol{r} = \boldsymbol{x} - \hat{\boldsymbol{x}}.$$

The auxiliary loss selects dead features with large activations on the current batch and uses their decoder directions to reconstruct this residual. This encourages inactive features to specialize to currently unexplained directions, reducing the chance that they remain permanently unused. The auxiliary term is then added to the standard SAE training objective during training.

A.3 DECODER NORMALIZATION

During training, we normalize decoder weights so that feature embeddings and gradients have unit norm. Namely, suppose that w_i and g_i is a decoder column and its gradient correspondingly, then the procedure is:

1. Compute row-normalized decoder weights: $u_i = \frac{w_i}{\|w_i\|_2}$.
2. Remove the component parallel to u_i from the gradient: $g_i \leftarrow g_i - (g_i^T u_i)u_i$, so that gradient step will be performed on the tangent space of the unit sphere.
3. Set weights to normalized: $w_i \leftarrow u_i$.

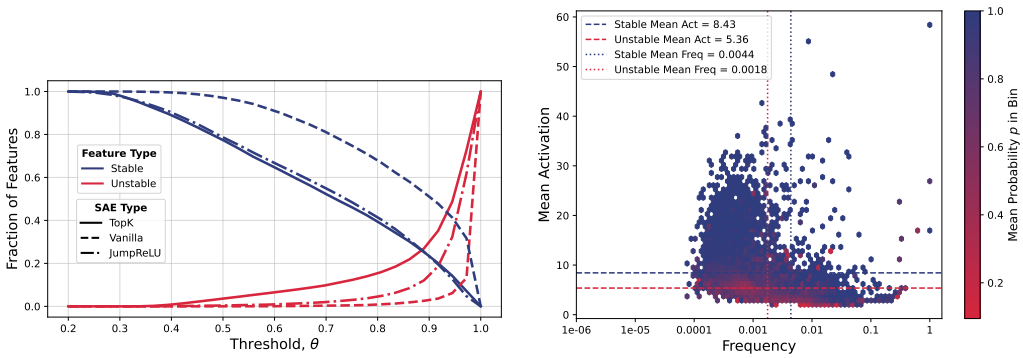
After training, we fold the mean and the standard deviation of the initial hidden state buffer into the weights so that no explicit normalization of the input and rescaling of the output is required, then normalize the decoder and scale encoder weights and bias to counteract this normalization, so that SAE do not change the output.

B ADDITIONAL DETAILS FOR QUANTITATIVE ANALYSIS

B.1 EVALUATION TOKENS AND ACTIVATION COLLECTION

All quantitative statistics (frequencies, magnitudes, entropy, EV, and CE under patching) are computed over a held-out collection of token positions. Let B_{eval} be the number of evaluation batches and T the context length, and define

$$N_{\text{tok}} \;=\; B_{\text{eval}} \cdot T.$$



(a) **Endpoint fractions vs. cosine threshold θ.** Fractions of *stable* and *unstable* features as a function of cosine matching threshold θ for several SAE types in the same base-model setting.

(b) **Activation frequency and conditional mean magnitude for stable vs. unstable features.** Unstable features are concentrated at lower frequencies, while stable features exhibit a heavier high-magnitude tail.

Figure 7: **Additional quantitative appendix results.** Left: threshold sensitivity of endpoint fractions across SAE families. Right: joint distribution of activation frequency ω_i and conditional mean magnitude μ_i for stable and unstable features.

For token index $n \in \{1, \ldots, N_{\text{tok}}\}$, let $\boldsymbol{h}_n \in \mathbb{R}^d$ denote the base-model activation at the SAE training location (residual stream), and let $\boldsymbol{z}_n \in \mathbb{R}^F_{\geq 0}$ denote SAE feature activations. We use the SAE reconstruction

$$\hat{\boldsymbol{h}}_n = \boldsymbol{W}_{\text{dec}} \boldsymbol{z}_n + \boldsymbol{b}_{\text{dec}}. \tag{5}$$

All evaluation batches are disjoint from SAE training batches.

B.2 Cosine-threshold sensitivity across SAE families

We vary the cosine matching threshold θ used to declare cross-seed feature matches and recompute the endpoint fractions for each SAE family in the main GPT-2 setting. Figure 7a shows that increasing θ makes matching stricter, so the stable fraction falls and the unstable fraction rises across all SAE types. The qualitative ordering is preserved throughout the sweep: TopK and BatchTopK remain nearly identical, JumpReLU stays intermediate, and Vanilla exhibits the strongest endpoint stability.

B.3 Usage statistics: activation frequency and mean magnitude

We quantify how often each feature is used and how strongly it fires when active via:

$$\omega_i = \frac{1}{N_{\text{tok}}} \sum_{n=1}^{N_{\text{tok}}} \mathbf{1}\{z_{n,i} > 0\}, \tag{6}$$

$$\mu_i = \frac{\sum_{n=1}^{N_{\text{tok}}} z_{n,i}}{\sum_{n=1}^{N_{\text{tok}}} \mathbf{1}\{z_{n,i} > 0\}}. \tag{7}$$

Here ω_i is the fraction of evaluated token positions where feature i activates, and μ_i is the mean activation magnitude conditioned on activation.

B.4 Token entropy computation

Let $v_n \in \{1, \ldots, |\mathcal{V}|\}$ be the vocabulary token id at token position n. For each feature i, define token-conditional activation counts and the induced token distribution

$$c_i(v) = \sum_{n=1}^{N_{\text{tok}}} \mathbf{1}\{(v_n = v) \wedge (z_{n,i} > 0)\}, \tag{8}$$

$$\pi_i(v) = \frac{c_i(v)}{\sum_{v' \in \mathcal{V}} c_i(v')}, \tag{9}$$

and token entropy

$$H_i = -\sum_{v \in \mathcal{V}} \pi_i(v) \log \pi_i(v). \tag{10}$$

We use the natural logarithm. Features that never activate on the evaluation set are excluded from entropy plots.

B.5 Masking protocol, uncertainty, and reweighting

For a masked index set $M \subseteq \{1, \dots, F\}$, define

$$\boldsymbol{z}_n^{(-M)} = \boldsymbol{z}_n \odot (\mathbf{1} - \mathbf{1}_M), \tag{11}$$

$$\hat{\boldsymbol{h}}_n^{(-M)} = \boldsymbol{W}_{\mathrm{dec}} \boldsymbol{z}_n^{(-M)} + \boldsymbol{b}_{\mathrm{dec}}. \tag{12}$$

We measure reconstruction quality by explained variance

$$\mathrm{EV}(M) = 1 - \frac{\mathbb{E}_n\left[\|\boldsymbol{h}_n - \hat{\boldsymbol{h}}_n^{(-M)}\|_2^2\right]}{\mathbb{E}_n\left[\|\boldsymbol{h}_n - \bar{\boldsymbol{h}}\|_2^2\right]}, \qquad \bar{\boldsymbol{h}} = \mathbb{E}_n[\boldsymbol{h}_n]. \tag{13}$$

For each masking budget N, we sample masks as follows:

- Sample M_S by drawing N features uniformly without replacement from $\mathcal{S}_\varepsilon$.
- Sample M_U by drawing $4N$ features uniformly without replacement from $\mathcal{U}_\varepsilon$.

We repeat this sampling procedure $R = 10$ times independently for each N. Plots report the mean across these R draws, and the shaded band indicates one standard deviation across draws.

Reweighting. In addition to direct masked reconstructions $\hat{\boldsymbol{h}}_n^{(-M)}$, we evaluate a reweighted variant that matches the ℓ_2 norm of the unmasked SAE reconstruction $\hat{\boldsymbol{h}}_n$ at each token position:

$$\tilde{\boldsymbol{h}}_n^{(-M)} = \hat{\boldsymbol{h}}_n^{(-M)} \cdot \frac{\|\hat{\boldsymbol{h}}_n\|_2}{\|\hat{\boldsymbol{h}}_n^{(-M)}\|_2 + \delta}, \tag{14}$$

where $\delta > 0$ is a small constant for numerical stability.

B.6 Next-token loss under activation patching

To measure downstream impact, we patch the residual stream at the SAE training location. For each token position n, we run a clean forward pass and compute cross-entropy $\mathrm{CE}_{\text{base}}$ on the evaluation tokens. We then rerun the forward pass where $\boldsymbol{h}_n$ is replaced by either $\hat{\boldsymbol{h}}_n^{(-M)}$ (no reweighting) or $\tilde{\boldsymbol{h}}_n^{(-M)}$ (with reweighting), and compute $\mathrm{CE}_{\text{patched}}(M)$ on the same evaluation tokens. We report

$$\Delta\mathrm{CE}(M) = \mathrm{CE}_{\text{base}} - \mathrm{CE}_{\text{patched}}(M), \tag{15}$$

so more negative values indicate larger degradation (as in Figure 2).

B.7 Threshold-free robustness across reappearance probability

The endpoint sets $\mathcal{U}_\varepsilon$ and $\mathcal{S}_\varepsilon$ summarize a strongly bimodal distribution. We also evaluate effective rank and masking directly as functions of $\hat{p}$. Each anchor uses $N = 95$ independently seeded comparison SAEs, regardless of activation frequency. For distinct normalized rows within one SAE, the probability that the largest pairwise cosine exceeds 0.7 is approximately 0.009 in GPT-2 and 0.003 in Gemma-2. This makes accidental conflation rare at $\theta = 0.7$. $\varepsilon = 0.05$ lies above these within-dictionary collision rates.

For effective rank, we divide the features into 14 population-balanced bins, draw size-matched subsets of 768 features, and average over 50 draws per bin. The $\hat{p} = 0$ population has $\mathrm{ER}/d = 0.541$. The remaining bins lie mainly between 0.739 and 0.765 (Table 2). This isolates the strongest rank

Table 2: Normalized effective rank across population-balanced reappearance-probability bins. Each value averages 50 size-matched draws.

$\hat{p}$	0.00	0.07	0.22	0.38	0.51	0.63	0.72	0.80	0.86	0.91	0.95	0.97	0.99	1.00
ER/d	0.541	0.739	0.759	0.763	0.765	0.765	0.762	0.763	0.760	0.761	0.762	0.753	0.752	0.744

Table 3: Active-mass-matched masking across equal-width reappearance-probability bins.

$\hat{p}$ bin	$-\Delta\mathrm{CE}$ ($\times 10^{-3}$)	$-\Delta\mathrm{EV}$ ($\times 10^{-3}$)
$[0.0, 0.2)$	12.2	6.2
$[0.2, 0.4)$	9.6	5.3
$[0.4, 0.6)$	7.3	5.4
$[0.6, 0.8)$	14.1	9.2
$[0.8, 1.0]$	60.5	17.6

concentration at the zero-reappearance endpoint and shows the behavior over the complete probability range.

We next split features into five equal-width $\hat{p}$ bins and scale the number of masked features by each bin's median activation frequency. The resulting active masses agree within 0.2. Table 3 reports three-draw averages over 200k tokens. The highest-probability bin produces 4.3 to $8.3\times$ the next-token-loss change and 1.9 to $3.3\times$ the EV change of the other bins.

B.8 Frequency-controlled functional and interpretability comparisons

We group features by activation frequency and compare stable and unstable features inside each bin. Stable features receive higher automatic-interpretation scores in all five bins (Table 4). Their CE and EV effects are also larger in most bins. The highest-frequency bin contains few unstable features and consequently has higher variance. Frequency-stratified token entropy preserves the separation between stable and unstable features.

B.9 Explained variance during tuning of constructed SAEs

The most-probable construction rapidly recovers baseline EV, while the equiprobable construction approaches it more gradually. Random and least-probable initializations improve but remain below baseline (Figure 8).

B.10 Probability distributions of selected features in constructed SAEs

From $n = 4$ to $n = 96$, mean probability rises from 0.793 to 0.830 for the most-probable set, falls from 0.294 to 0.051 for the least-probable set, and changes little (0.743 to 0.727) for the equiprobable set (Figure 9). Four SAEs already separate the sets. Additional runs mainly isolate the near-zero-probability tail.

B.11 SAEBench metrics for standard and most-probable construction SAEs

Table 5 shows that the most-probable construction SAE remains broadly competitive with standard TopK SAEs on SAEBench: SCR and TPP decrease modestly, while Sparse Probing Top-1 and the Autointerp score slightly improve.

B.12 Cost of the pooled-feature construction

The 96-seed experiment estimates $\hat{p}$, but roughly 4 to 10 source SAEs already yield a dictionary dominated by stable features with EV near 0.92 (Figure 3). Source SAEs share an activation stream and can train in parallel. Each uses one H100 for approximately 2.5 hours on GPT-2 or 12 hours on Qwen2.5-7B. Post-training uses 2M tokens, about 0.2% of a standard run. Total accelerator-hours scale linearly with pool size.

Table 4: Stable and unstable features compared within matched log-frequency bins. Causal columns give stable-to-unstable impact ratios.

$\log_{10}$ frequency	Detection, unstable/stable	CE ratio	EV ratio
-3.63	0.811 / 0.865	$3.0\times$	$3.7\times$
-3.24	0.791 / 0.874	$6.3\times$	$1.3\times$
-2.96	0.780 / 0.867	$4.8\times$	$4.1\times$
-2.58	0.748 / 0.809	$1.9\times$	$1.9\times$
-1.25	0.659 / 0.740	$23.4\times$	$0.45\times$

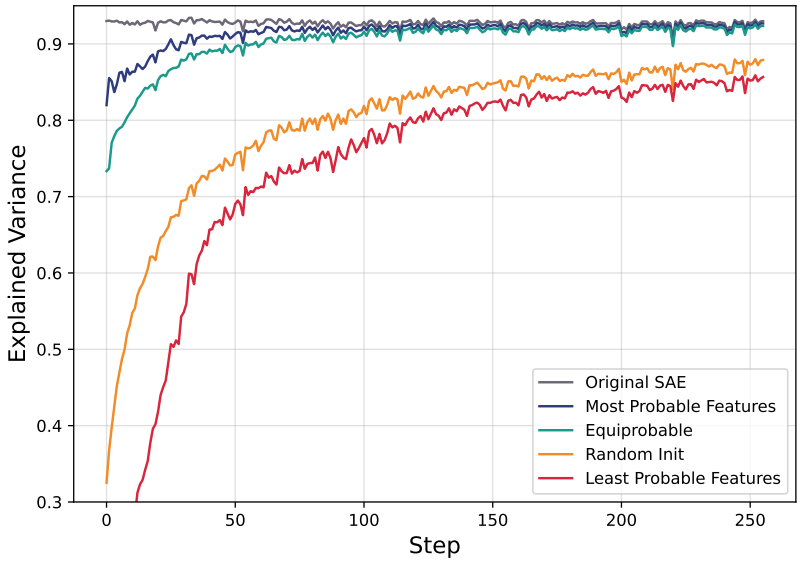


Figure 8: **Explained variance during brief tuning of SAEs constructed from different feature subsets.** Initialization from the most probable features recovers nearly the same explained variance as the original SAE, the equiprobable construction performs somewhat worse, and the least-probable construction lags substantially behind.

C Additional Discussion on Stability-Inducing Methods

Our main analysis studies what unstable features are like and why they arise under a fixed SAE objective. A related but distinct question is how to make learned SAE dictionaries more stable. We group stability-inducing approaches into three broad axes: explicit or implicit regularization, metric reweighting, and bagging-style feature pooling.

Regularization and architectural constraints. Regularization or additional constraints can reduce the space of near-equivalent solutions and thereby increase cross-seed agreement. In our experiments, Vanilla ReLU+ℓ_1 SAEs are substantially more stable than TopK SAEs at comparable sparsity, but have lower EV. JumpReLU SAEs also include an ℓ_1-style sparsity regularization term, which may partly explain their improved stability relative to TopK. Archetypal SAEs constrain the optimization by tying learnable features to archetypes derived from training data (Fel et al., 2025); we do not include them in our quantitative comparison because the original method was evaluated on vision models, and in our text-activation experiments we were unable to obtain competitive reconstruction performance. HierarchicalTopK is different: it does not add an explicit ℓ_1 penalty, but can be viewed as an implicit regularizer because a single dictionary must support good reconstruction across nested sparsity budgets (Balagansky et al., 2025a). This may reduce the degeneracy of the fixed-k TopK objective and make seed-dependent basis choices less likely. This is also consistent with the TopK sparsity sweep in Table 1: lower k yields lower EV but higher stability, suggesting that stronger sparsity constraints can reduce seed-dependent basis choices.

Data whitening and Mahalanobis losses. Our results show that unstable features contribute less to reconstruction, suggesting that they may lie in directions that are downweighted by the standard

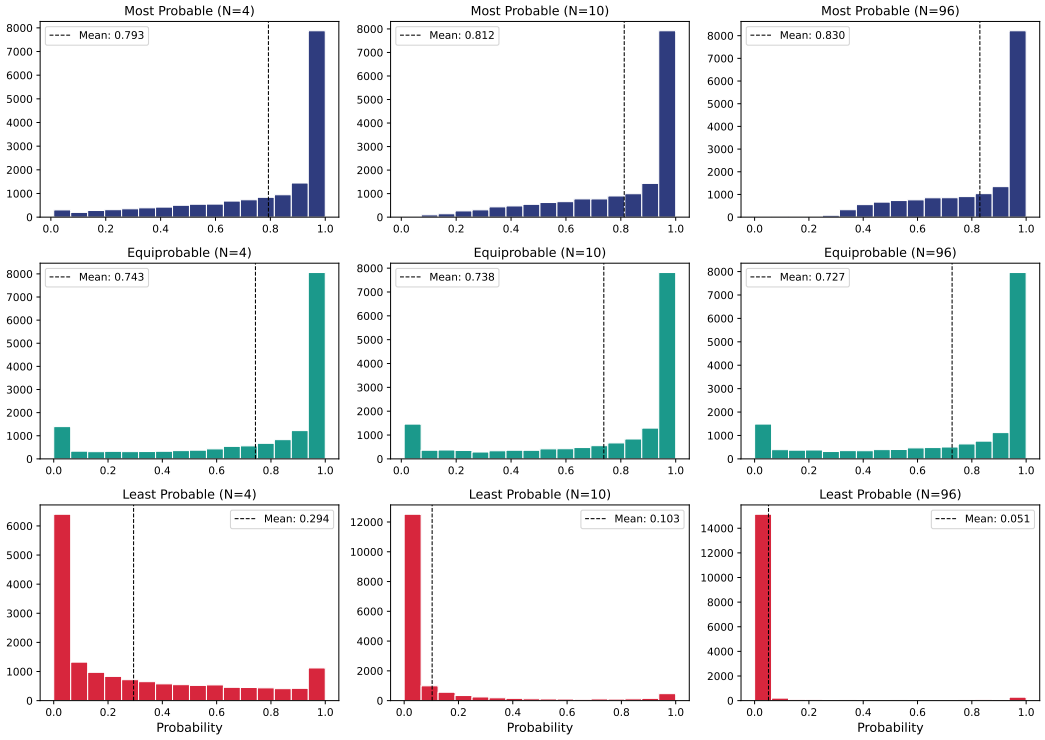


Figure 9: **Probability distributions of selected features for different source-pool sizes.** Rows correspond to the most-probable, equiprobable, and least-probable constructions. Columns use $n = 4$, 10, and 96 source SAEs. Even small source pools already make the most-probable construction concentrate on high-reappearance features, while the least-probable construction remains dominated by near-zero-probability features.

Metric	Standard	Most-probable
Sparse Probing (Top-1)	0.656	0.673
AutoInterp (mean)	0.818	0.850
SCR (Top 20)	0.269	0.258
TPP (Top 20)	0.107	0.090

Table 5: SAEBench metrics for standard and most-probable construction SAEs.

MSE objective. This motivates reweighting the reconstruction metric. Let $\boldsymbol{x} \in \mathbb{R}^d$ be a hidden state, $\hat{\boldsymbol{x}}$ its SAE reconstruction, $\boldsymbol{r} = \hat{\boldsymbol{x}} - \boldsymbol{x}$ the residual, and $\Sigma \in \mathbb{R}^{d \times d}$ the activation covariance. Standard MSE uses $\boldsymbol{r}^\top \boldsymbol{r}$, while a regularized Mahalanobis loss uses

$$\boldsymbol{r}^\top (\Sigma + \lambda I)^{-1} \boldsymbol{r}.$$

This is closely related to training on whitened activations (Saraswatula & Klindt, 2025). To interpolate continuously between MSE and this Mahalanobis objective, we train with

$$\boldsymbol{r}^\top (\Sigma + \lambda I)^{-\alpha} \boldsymbol{r}, \qquad \alpha \in [0, 1],$$

where $\alpha = 0$ recovers MSE and $\alpha = 1$ recovers regularized Mahalanobis loss. Figure 10 shows that increasing α can reduce instability, but substantially worsens EV. This is expected: reweighting low-variance directions makes the training objective less aligned with standard reconstruction quality.

Constructing SAEs from feature pools. A third route is to aggregate reproducible structure across runs rather than changing the single-run objective. Our most-probable feature-pool construction deduplicates decoder features across independently trained SAEs, selects high-probability features, and briefly post-trains the resulting dictionary. As shown in Figure 10, this approach can substantially

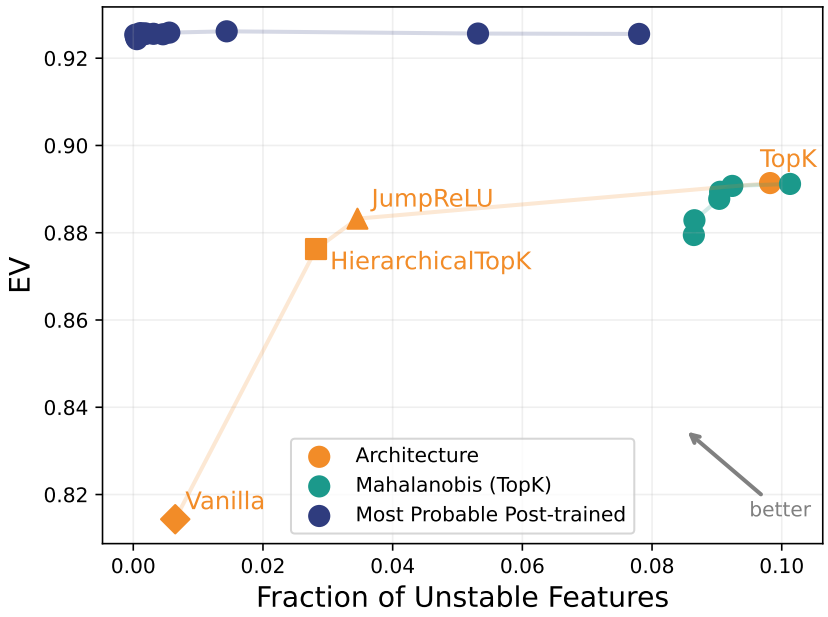


Figure 10: **Routes to SAE stability in the EV–instability plane.** Orange points compare SAE architecture/objective variants, including TopK, JumpReLU, Vanilla, and HierarchicalTopK. Green points vary the Mahalanobis/whitening interpolation for TopK SAEs. Blue points show most-probable feature-pool SAEs after post-training with different numbers of source SAEs; the rightmost blue point corresponds to the post-trained TopK SAE (i.e., a feature pool with a single SAE). Architectural and loss-level changes can reduce instability but trade off against EV, whereas feature pooling can reach low instability while preserving high EV.

reduce instability while preserving high EV. Interestingly, before post-training the selected features all have reappearance probability bounded away from zero (around 0.3 in the shown run), but after post-training some features shift toward lower reappearance probabilities (Figure 11). This supports the view that lower-stability directions can be useful for reconstruction, rather than being mere random artifacts.

D Additional Results for the Controlled Low-Rank Synthetic Model

We report additional synthetic-model results beyond the main setting from Section 7. Across multiple choices of low-rank dimension r and sparsity level k, the same qualitative pattern persists: full-rank features are consistently stable and well recovered, while features confined to the shared low-rank block are much less stable as individual directions.

D.0.1 Residual-Based Diagnostic for Low-Rank Feature Recovery

The toy model also admits a diagnostic for checking whether the low-rank ground-truth atoms are present in the residual structure, even when they are not recovered as individual SAE latents. Let an activation with exactly one low-rank ground-truth feature be

$$h^{(t)} = \sum_{i \in F_t} f_i + g_{r(t)},$$

where f_i are full-rank ground-truth features and $g_{r(t)}$ is a low-rank ground-truth feature. Suppose the SAE reconstruction activates the correctly identified full-rank features and exactly one unstable

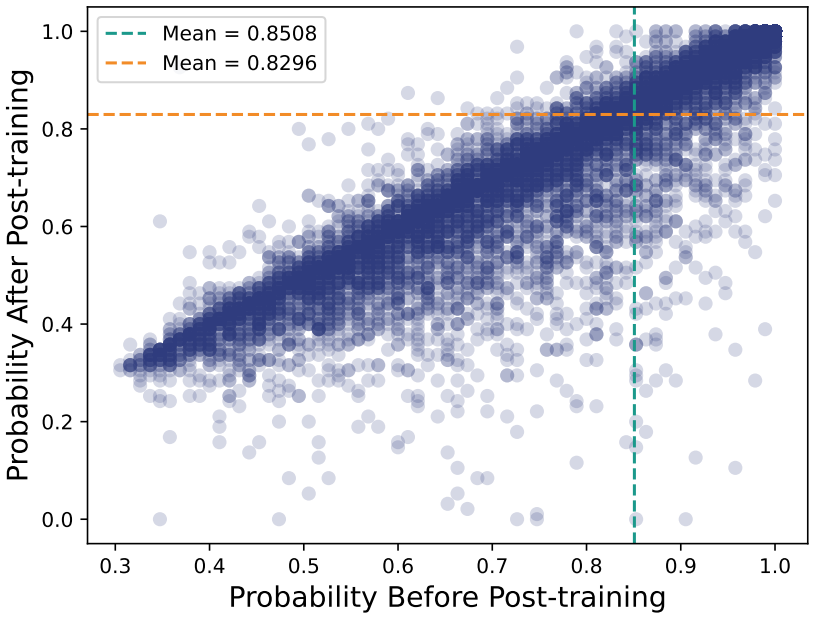
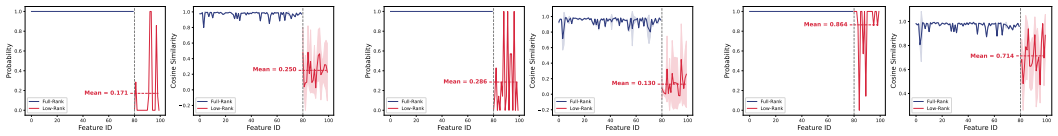


Figure 11: **Feature reappearance probabilities before and after post-training.** The most-probable construction initializes from relatively high-probability pooled features, but post-training shifts some features toward lower reappearance probabilities. Lower-stability directions can therefore be reconstruction-useful rather than mere random artifacts.



(a) $d = 32, r = 1, k = 4$. (b) $d = 32, r = 1, k = 8$. (c) $d = 32, r = 4, k = 4$.

Figure 12: **Additional synthetic settings.** The same full-rank versus low-rank split appears across multiple values of the subspace rank r and sparsity level k: full-rank features retain high reappearance probabilities and high cosine similarity to ground truth, whereas low-rank features do not.

feature $\hat{g}_{j(t)}$:

$$\hat{h}^{(t)} \approx \sum_{i \in F_t} f_i + \hat{g}_{j(t)}.$$

Then the residual-corrected vector

$$h^{(t)} - \hat{h}^{(t)} + \hat{g}_{j(t)} \approx g_{r(t)}.$$

Let $z^{(t)}$ denote the vector of unstable-feature activations for example t. Then

$$R = \left\{ h^{(t)} - \hat{h}^{(t)} + \hat{g}_{j(t)} : \|z^{(t)}\|_0 = 1 \right\}$$

should cluster around the ground-truth low-rank atoms $g_1, \ldots, g_M$, assuming each low-rank atom appears in at least one selected activation. In the idealized case where the full-rank features are exactly identified and there is no reconstruction error, the elements of R are exactly ground-truth low-rank atoms; with approximate full-rank recovery, they are noisy samples around those atoms.

We therefore run k-means on R with M clusters, replace the unstable decoder rows by the resulting centroids, and recompute cosine similarity to the ground-truth dictionary. Figure 14 shows that this

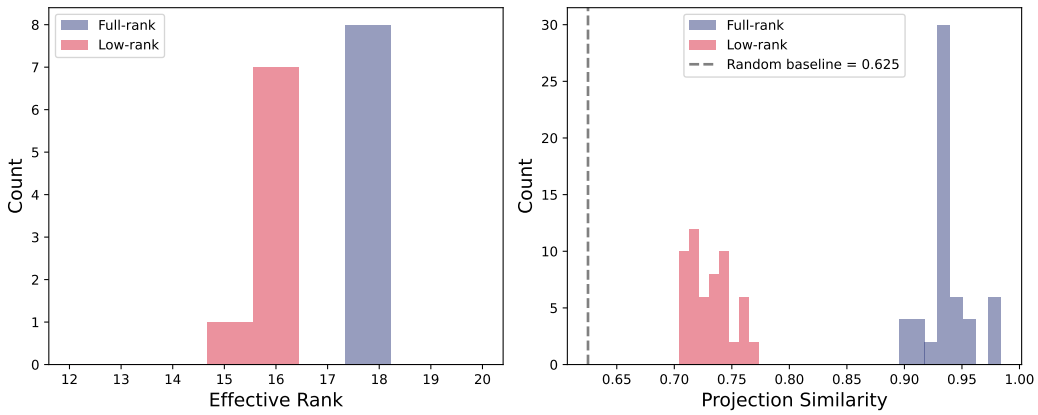


Figure 13: **Effective rank and cross-seed subspace similarity in the synthetic model ($d = 32$, $r = 2$, $k = 8$).** The learned low-rank block has smaller effective rank than the full-rank block, yet its cross-seed subspace similarity remains clearly above the random baseline. Thus the low-rank features are unstable as individual vectors but still reconstruct a reproducible shared subspace.

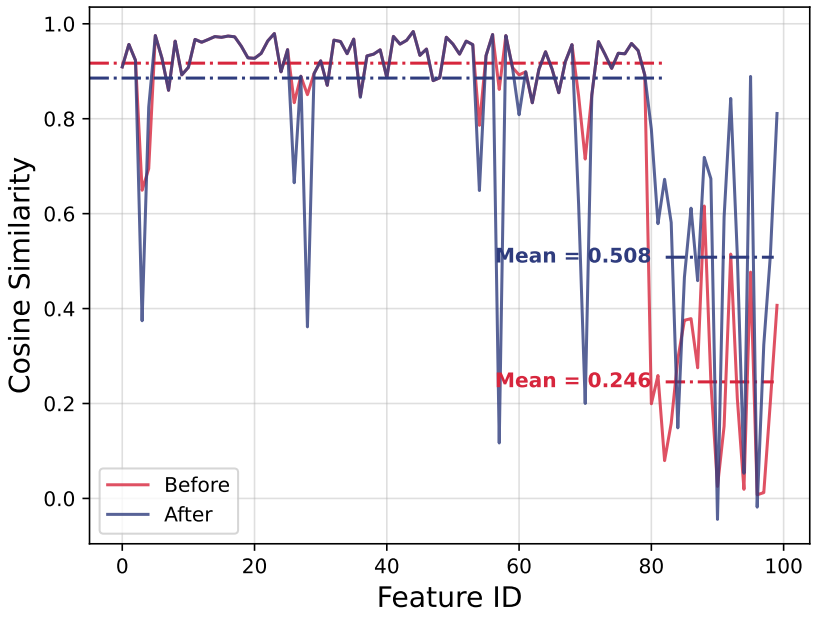
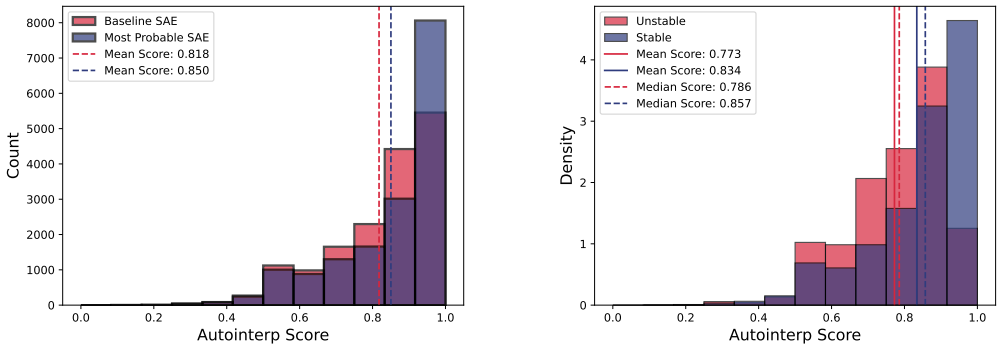


Figure 14: **Residual-based recovery of low-rank ground-truth features in the toy model.** Cosine similarity between decoder rows and matched ground-truth features before and after replacing unstable decoder rows with centroids obtained from residual-corrected activations. The procedure partially recovers the low-rank block while leaving the already identified full-rank block largely unchanged.

substantially improves alignment for the low-rank block in the $d = 32, r = 2, k = 8$ toy setting. This is a diagnostic rather than a practical training method: it relies on the toy setup's known number of low-rank atoms and selected one-unstable examples, and freezing the replaced decoder rows followed by encoder retraining did not improve EV.



(a) **Automatic interpretation scores: baseline SAE vs. SAE constructed from the most probable features.** Histogram of detection scores. Dashed lines show mean scores for each SAE.

(b) **Automatic interpretation (detection) scores for stable vs. unstable features.** Stable features achieve higher scores, with a substantially larger mass at score $= 1$.

Figure 15: **Automatic-interpretation comparisons.** Left: the most-probable-feature SAE preserves high detection scores relative to the baseline SAE. Right: stable features are substantially easier to detect automatically than unstable ones.

E Additional Details for Qualitative Analysis

This appendix contains the detailed qualitative results summarized in the token-entropy discussion of Section 5. We compare stable and unstable features using automatic feature interpretation and explanation text analysis.

E.1 Qualitative results from automatic interpretation

We use the same stable/unstable split as elsewhere in the paper (with $\varepsilon = 0.05$ and a single anchor SAE). We run the SAEBench auto-interpretation pipeline (Karvonen et al., 2025) with `Qwen/Qwen3-32B` as the interpreter/evaluator model. Full settings are given below in Appendix E.2.

Stable features are more interpretable. SAEBench assigns each feature a *detection score*, defined as the evaluator model's accuracy at predicting whether a given context contains *any* token position on which the feature activates (Appendix E.3). Figure 15b shows the distribution of detection scores for stable and unstable features. Stable features achieve noticeably higher scores overall. This gap is driven primarily by the mass at score $= 1$ (perfect detection on held-out contexts): in our setting, the fraction of stable features with score 1 is $\approx 4.5\times$ larger than for unstable features.

Explanations differ systematically. Inspecting the generated explanations reveals a consistent qualitative split. Unstable features are frequently described in terms of *surface form*: letter patterns, capitalization, prefixes, punctuation, and short substrings. Stable features more often describe *how words function in context*: phrases and constructions, syntactic roles (e.g., pronouns, conjunctions), or broader semantic groupings. This aligns with Section 5, where unstable features concentrate on lower token-entropy regimes dominated by punctuation/subword triggers, while stable features extend to higher-entropy lexical and conceptual patterns.

Keyword evidence for surface vs. structural patterns. To quantify the surface-vs-structural distinction in explanation text, we measure how often explanations contain keywords such as `substring`, `starting`, `letter` (surface-form) versus `phrase`, `expression`, `construction` (structural/compositional), across bins of $\hat{p}(\boldsymbol{f}_i^{(0)})$. We restrict to explanations with detection score > 0.7 to reduce noise from low-quality explanations. A simple summary already shows a large separation: in the unstable features' explanations, `substring` appears in 38.9% of explanations, while for the stable ones it appears in 11.3%. Conversely, `phrase` increases from 4.1% to 32.0% from unstable to stable features. The full table (with confidence intervals and bin counts) is given in Appendix E.5.

Table 6: **Representative automatic interpretations.** Three illustrative explanations per class are shown for unstable vs. stable features.

Unstable features (examples)	Stable features (examples)
words starting with "ve" followed by a letter that continues a root or morpheme	the phrase "so as to" and similar constructions indicating purpose or intent
proper nouns, especially those starting with capital letters	reflexive pronouns and words referring to oneself or others
the substring "Tor" at the start of words	technical terms related to servers, connections, and computing components

Explanations alone predict stability. As an auxiliary test, we ask a frontier LLM to classify features as stable vs. unstable using *only* their auto-interp explanations, with labels anonymized during prompting. On a balanced evaluation set, the model achieves 0.88 accuracy. Full prompts and the exact protocol are given in Appendix E.6.

E.2 Auto-interpretation setup (SAEBench)

We use the SAEBench auto-interpretation pipeline (Karvonen et al., 2025) with `Qwen/Qwen3-32B` as both interpreter and evaluator model. Interpretation/evaluation contexts are drawn from a held-out slice of FineWeb consisting of 2M tokens. For each feature, we build prompt contexts from token windows around activations and cap the model's generation length at 128 tokens.

Example selection. For each feature, we construct two disjoint sets of contexts:

- **Explanation set (used to generate the explanation):** 10 contexts with the largest activation value, plus 5 additional contexts sampled via importance sampling.
- **Evaluation set (used to score the explanation):** 2 additional top-activation contexts (not overlapping with the explanation set), plus 10 importance-sampled contexts, plus 2 random contexts.

All sampling is performed independently per feature. The resulting prompts follow SAEBench defaults, aside from the local inference adjustments described above.

E.3 Detection score definition

Given a feature i and a context window c (a sequence of tokens), define the binary label

$$y_i(c) = \mathbf{1}\{\exists \text{ a token position in } c \text{ such that } z_{t,i} > 0\},$$

i.e., whether the context contains *any* position where the feature activates. Given the interpreter-produced explanation text $\mathcal{E}_i$, the evaluator model predicts $\hat{y}_i(c) \in \{0, 1\}$. The *detection score* for feature i is the accuracy on the held-out evaluation set:

$$\mathrm{DS}_i \;=\; \frac{1}{|\mathcal{C}_i|} \sum_{c \in \mathcal{C}_i} \mathbf{1}\{\hat{y}_i(c) = y_i(c)\},$$

where $\mathcal{C}_i$ is the evaluation context set described in Appendix E.2.

E.4 Additional representative interpretations

Table 6 above shows a small set of representative explanations illustrating the surface-form vs. structural distinction. Here we provide additional examples to support the qualitative claims in Appendix E.

E.5 Keyword analysis across stability bins

We bin features by their empirical reappearance rate $\hat{p}(\boldsymbol{f}_i^{(0)})$ and compute, for each bin and each keyword k, the empirical fraction $\hat{q}$ of explanations that contain k as a substring match (case-sensitive,

Table 7: **Additional automatic interpretations.** More examples of unstable (surface-form) vs. stable (structural/compositional) explanations. Only interpretations with the score > 0.7 are shown.

Unstable features	Stable features
words starting with "Div" or containing the substring "div" in various contexts	*phrases involving positional prepositions like "in the," "right up to," and "next to."*
the substring "wr" at the start of words	*phrases indicating reference to previously mentioned information, such as "as described above" or "mentioned earlier"*
the substring "Tro" at the start of words	*phrases describing characters and their roles in stories or films*
words starting with the substrings "wa", "por", "po", or similar letter combinations	*words related to peace and calmness, including variations like "peaceable" and "peacefully"*
partial words with double letters in the middle, such as "Perr", "Phys", "Ol", "Bonni", "feat", "iodin", and "Coleman"	*the word "firm" and similar capitalized terms in business or organizational contexts*
words containing the substring "cran" followed by a letter, often related to "cranberries" or "cranium."	*words starting with the letter "L" (case-insensitive)*
first names, especially when preceded by a description or title	*words ending in the substring "-el"*
words starting with "Rum", "mist", and "of" as well as variations like "Rumor", "misten", and "of"	*the word "lock" and variations with attached punctuation or nearby words*
words containing the substring "br" followed by "on" or "o" in medical or anatomical contexts	*the phrase "no longer" and related negated continuations*
words starting with "El" or "el"	*names and parts of names starting with "Mc" or "McK" or "Mac"*
surnames and partial surnames, especially those starting with "Hart"	*words related to negative outcomes or effects, especially when paired with terms like "impact," "effects," or "feedback."*
the word "batch" and similar capitalized substrings like "Dow" and "Alabama"	*words indicating uncertainty or hypothetical situations, such as "potential," "possible," and similar variations*
names or parts of names with a specific stylistic or phonetic pattern, particularly those beginning with <<Alfred>>, <<styl>>, <<transc>>, or <<University>>	*phrases involving change with quantity or degree, especially those using "the… the… " or "increases/decreases with."*
words related to storage containers, especially "tank," "tanks," and variations like "gallons," "empty," and "tankers."	*the word "healthy" and related concepts about diet and well-being*
words starting with "pie" or "piez" and related technical or compound terms	*variations of the word "dig" and related actions or contexts*
the surname "Gordon" and related proper nouns	*adjectives and phrases starting with "un-" that convey a sense of freedom or lack of restriction*
words starting with the substring "Whit" or "Wal"	*coordinating conjunctions "and" and "or" and their surrounding context*
government-related terms such as "Cabinet," "Prime," and "bills."	*words related to teenagers and specific substrings like "teen", "teens", "Teen", and "agers"*
words related to substances, fluids, or materials in contexts of waste, water, or bodily fluids	*the word "course" and related educational concepts*
capitalized names and the substring "Marshall" with variations	*language names and related terms such as English, Spanish, Arabic, and their variations*

Table 8: **Keyword frequencies in feature explanations across reappearance bins (Wald 95% CI).** Each entry reports the fraction of explanations containing the given keyword, shown as $\hat{q}\pm$ (95% CI). Column headers give bin ranges for $\hat{p}(\boldsymbol{f}_i^{(0)})$ and the number of features n in that bin after restricting to detection score > 0.7.

Keyword	$[0.00, 0.05)$ (n=1020)	$[0.05, 0.10)$ (n=220)	$[0.90, 0.95)$ (n=1053)	$[0.95, 1.00]$ (n=6547)
`substring`	0.389±0.030	0.209±0.054	0.121±0.020	0.113±0.008
`starting`	0.276±0.027	0.082±0.036	0.050±0.013	0.052±0.005
`letter`	0.108±0.019	0.068±0.033	0.036±0.011	0.037±0.005
`phrase`	0.041±0.012	0.205±0.053	0.313±0.028	0.320±0.011
`expression`	0.010±0.006	0.036±0.025	0.056±0.014	0.058±0.006
`construction`	0.008±0.005	0.032±0.023	0.066±0.015	0.060±0.006

using the raw explanation text). To reduce noise from low-quality explanations, we restrict to features with detection score $\mathrm{DS}_i > 0.7$.

We report confidence intervals for each proportion. Below we include two versions of the same table: a Wald interval (symmetric and matches the common $\hat{q} \pm 1.96\sqrt{\hat{q}(1-\hat{q})/n}$ form) and a Wilson interval (typically better calibrated near 0/1).

E.6 LLM-ONLY CLASSIFICATION FROM EXPLANATION TEXT

We test whether the *auto-interp explanation text alone* contains enough signal to predict feature stability. Concretely, we run a blind classification experiment where a frontier LLM (GPT-5) receives only short natural-language explanations produced by the auto-interpretation pipeline summarized in Appendix E and must predict a binary label. No activation traces, token examples, decoder vectors, or $\hat{\rho}$ values are shown.

Data and split. We construct a balanced dataset of explanations from endpoint features with $\varepsilon = 0.05$ using the same anchor SAE as in the main text. We sample 50 explanations from each class for in-context training (100 total) and a disjoint 50 explanations from each class for evaluation (100 total). During prompting, labels are anonymized as $\{0, 1\}$ and the model is not told what the labels correspond to.

Classification prompt. We use a single in-context learning prompt: first provide labeled training examples (balanced across labels), then ask for labels on the test set, requesting one label per line and no additional text. We report accuracy on the balanced test set.

```
You will see short explanations of patterns that activate a feature.
Each explanation has a label 0 or 1. Learn what distinguishes the labels.

TRAINING EXAMPLES:
Label: 0  Explanation: <EXPLANATION_1>
Label: 0  Explanation: <EXPLANATION_2>
...
Label: 1  Explanation: <EXPLANATION_51>
Label: 1  Explanation: <EXPLANATION_52>
...

Now predict labels for the TEST explanations.
Output one label (0 or 1) per line, in order, with no extra text.

TEST EXPLANATIONS:
1) <TEST_EXPLANATION_1>
2) <TEST_EXPLANATION_2>
...
```

Result. On the balanced test set (50 explanations per class), GPT-5 achieves accuracy 0.88. Since the model sees only explanation text, this indicates that stability is strongly reflected in the linguistic descriptions produced by auto-interpretation.

Qualitative class summaries prompt. In addition to classification, we ask the same LLM to summarize typical patterns in each class, using explanations only. We provide a small random subset from each class (without mentioning stability) and request concise descriptions.

```
Here are explanations from two classes (A and B). Summarize typical patterns in each
Be concise and concrete.

Class A:
- <EXPLANATION_A1>
- <EXPLANATION_A2>
...

Class B:
- <EXPLANATION_B1>
- <EXPLANATION_B2>
...
```

Model-produced summaries. The model consistently describes the two classes in terms of surface-form versus structural/compositional language. In a representative run it summarized:

Table 9: Effective rank for size-matched probability sets.

Probability Set	Effective Rank
$p \leq 0.01$	450.5 ± 4.4
$p \leq 0.05$	498.5 ± 4.4
$p \geq 0.95$	620.8 ± 3.2
$p \geq 0.99$	617.7 ± 3.3

- **Unstable:** explanations emphasize what the word looks like on the surface (substrings, prefixes, capitalization, punctuation) and narrow lexical triggers.
- **Stable:** explanations emphasize how words function in sentences (phrases, constructions, discourse roles) and broader compositional or semantic groupings.

Notes and limitations. This experiment is intended as a lightweight diagnostic rather than a primary result. Because it relies on a particular evaluator model and prompt format, we treat it as supportive evidence that the surface-vs-structural distinction is present in explanation text, consistent with Table 6 and the keyword analysis in Appendix E.5.

E.7 Automatic interpretation for baseline vs. most-probable-feature SAE

The SAE built from the most probable features retains high detection scores relative to the baseline (Figure 15a).

F Effective Rank and SVD of the Decoder Submatrices

As in the main text, given an index set $\mathcal{I}_\varepsilon^{(s)} \subseteq \{1, \ldots, F\}$ (either $\mathcal{U}_\varepsilon^{(s)}$ for unstable features or $\mathcal{S}_\varepsilon^{(s)}$ for stable features), define the corresponding submatrix

$$X_{\mathcal{I}_\varepsilon}^{(s)} := \boldsymbol{W}_{\text{dec}}^{(s)}[:, \mathcal{I}_\varepsilon^{(s)}] \in \mathbb{R}^{d \times m_s}, \qquad m_s := |\mathcal{I}_\varepsilon^{(s)}|.$$

We consider its singular value decomposition

$$X_{\mathcal{I}_\varepsilon}^{(s)} = V_\varepsilon^{(s)} \, \Sigma_\varepsilon^{(s)} \, \big(U_\varepsilon^{(s)}\big)^\top,$$

where $V_\varepsilon^{(s)} \in \mathbb{R}^{d \times d}$ and $U_\varepsilon^{(s)} \in \mathbb{R}^{m_s \times m_s}$ are orthogonal matrices, and $\Sigma_\varepsilon^{(s)} \in \mathbb{R}^{d \times m_s}$ is rectangular, with diagonal entries $\sigma_1^{(s)} \geq \sigma_2^{(s)} \geq \cdots \geq \sigma_d^{(s)} \geq 0$, that are singular values, and all remaining diagonal entries are zero.

We define the effective rank of submatrix $X_{\mathcal{I}_\varepsilon}^{(s)}$ as

$$\text{ER}(X_{\mathcal{I}_\varepsilon}^{(s)}) = \exp\Big(- \sum_{k=1}^{\min(d, m_s)} p_k^{(s)} \, \log p_k^{(s)} \Big),$$

where $p_k^{(s)} = \sigma_k^{(s)} / \sum_{j=1}^{\min(d,m_s)} \sigma_j^{(s)}$. To fairly compare the effective ranks of submatrices with different numbers of features, we calculate them under a size-matched protocol: for each seed for all feature sets we select

$$k^{(s)} = \min\big(|\mathcal{U}_{0.01}^{(s)}|, \, |\mathcal{S}_{0.01}^{(s)}|, |\mathcal{U}_{0.05}^{(s)}|, \, |\mathcal{S}_{0.05}^{(s)}|\big),$$

and for sets whose number of features differs from $k^{(s)}$, we estimate ER by averaging over $B = 50$ random subsamples of size $k^{(s)}$.

We define the explained variance $\text{EV}_{\text{SVD}}^{(s)}(r)$ of singular values of matrix $X_{\mathcal{I}_\varepsilon}^{(s)}$ as

$$\text{EV}_{\text{SVD}}^{(s)}(r) = \sum_{i=1}^{r} \big(\sigma_i^{(s)}\big)^2 / \sum_{i=1}^{d} \big(\sigma_i^{(s)}\big)^2.$$

Let $V_\varepsilon^{(s)}[:, 1:r] \in \mathbb{R}^{d\times r}$ denotes the matrix, whose r columns are the top-r left singular vectors, and define the projector onto their span

$$P_\varepsilon^{(s)}(r) \;:=\; V_\varepsilon^{(s)}[:, 1:r]\,(V_\varepsilon^{(s)}[:, 1:r])^\top.$$

Then $\mathrm{EV}_{\mathrm{SVD}}^{(s)}(r)$ admits an alternative representation

$$\mathrm{EV}_{\mathrm{SVD}}^{(s)}(r) \;=\; \frac{\left\|P_\varepsilon^{(s)}(r) X_{\mathcal{I}_\varepsilon}^{(s)}\right\|_F^2}{\left\|X_{\mathcal{I}_\varepsilon}^{(s)}\right\|_F^2}.$$

This suggests a natural cross-seed analogue: for an anchor seed a and another seed $s \neq a$, define

$$\mathrm{EV}_{\mathrm{SVD}}^{(a\to s)}(r) \;:=\; \frac{\left\|P_\varepsilon^{(a)}(r) X_{\mathcal{I}_\varepsilon}^{(s)}\right\|_F^2}{\left\|X_{\mathcal{I}_\varepsilon}^{(s)}\right\|_F^2}.$$

Intuitively, $\mathrm{EV}_{\mathrm{SVD}}^{(a\to s)}(r)$ measures how well the top-r singular subspace learned in seed a explains the feature subspace in seed s.

G Additional Details for Other Setups

G.1 Large-model replication and cross-seed subspace transfer

We train eight TopK SAEs with independent seeds on Qwen2.5-7B layers 4, 7, 14, and 18, using dictionary size 32,768 and $k = 128$. Across the four layers, the unstable fraction ranges from 1.3% to 2.7%, and its dependence on the cosine threshold follows the profile observed in the smaller models. At layer 14, masking 128 stable features with active mass approximately 46 produces $2.1\times$ the EV degradation and $2.3\times$ the next-token-loss degradation of an unstable set with the same active mass.

We also evaluate cross-seed SVD transfer for GPT-2, Pythia-160M, and Gemma-2 2B at three layers per model. At rank 64, transfer for unstable-feature subspaces exceeds the Haar-random baseline by 2.7 to $5.8\times$ for GPT-2, 3.1 to $3.4\times$ for Pythia, and 3.2 to $16.7\times$ for Gemma-2. The effective-rank gap becomes smaller in later layers and in the eight-seed Qwen experiment, while cross-seed transfer remains positive across the evaluated configurations. The detailed functional and geometric experiments use GPT-2 layer 7. These experiments provide independent replications of the central stability and subspace results.

G.2 Endpoint fractions across models, layers, and dictionary sizes

Section 8 summarizes how endpoint stability varies across base models, layers, and dictionary sizes for TopK SAEs. Table 10 reports the full endpoint fractions for all configurations considered.

The left panel of Figure 16 reports the layer-wise fractions of stable and unstable features in TopK SAEs with $K = 64$ and $F = 2^{14}$, trained on GPT-2 residual-stream activations, using 21 independently trained SAEs per layer. The right panel presents the fraction of stable and unstable source features with a next-layer decoder-cosine match at threshold $\tau_{\mathrm{layer}} = 0.7$. Overall, stable SAE features become more prevalent with layer depth and are much more likely than unstable features to admit a next-layer decoder-cosine match.

G.3 Training-token sweeps: setup and measurement details

In Section 8 we study how endpoint stability evolves as SAE training progresses. We sweep the total number of SAE training tokens from 10M to 10B while keeping the SAE architecture and hyperparameters fixed to the main TopK setting (GPT-2, `resid_post` layer 7, TopK=64, $F = 2^{14}$). For each training budget, we train the same number of independent SAEs (different random seeds) and estimate feature reappearance probabilities using the matching procedure in Section 4. We then report the endpoint fractions (unstable and stable) using the same ε and cosine threshold θ as in the main setting.

Figure 6a reports the fraction of unstable features as a function of training tokens. Figure 17a reports the corresponding stable fraction curve. Together, these curves show that instability decreases early and then approaches a non-zero plateau, while the stable fraction continues to increase with diminishing returns.

Table 10: Fractions of *unstable* ($\hat{p} \leq \varepsilon$) and *stable* ($\hat{p} \geq 1 - \varepsilon$) features for TopK SAEs across dictionary sizes and layers.

Base model	Layer	$F = 2^{14}$		$F = 2^{15}$		$F = 2^{16}$	
		Unstable	Stable	Unstable	Stable	Unstable	Stable
GPT-2	4	0.219	0.339	0.161	0.356	0.088	0.387
	7	0.080	0.447	0.066	0.440	0.051	0.432
	10	0.031	0.484	0.035	0.470	0.040	0.433
Pythia-160M-deduped	4	0.136	0.421	0.109	0.558	0.049	0.634
	7	0.062	0.513	0.096	0.569	0.468	0.316
	10	0.064	0.461	0.077	0.527	0.114	0.456
Gemma-2 2B	7	0.042	0.653	0.027	0.647	0.021	0.588
	12	0.013	0.563	0.012	0.548	0.024	0.482
	18	0.005	0.681	0.008	0.629	0.020	0.509

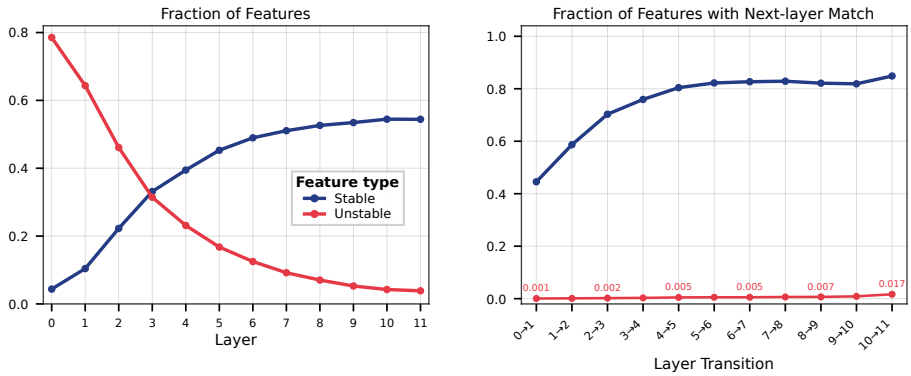
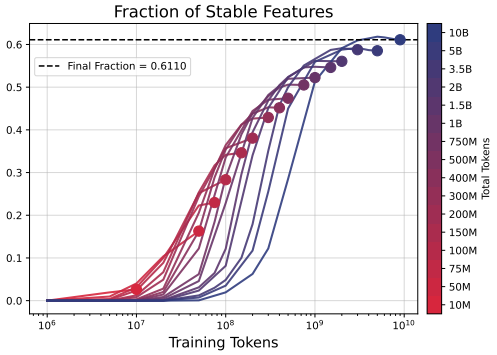


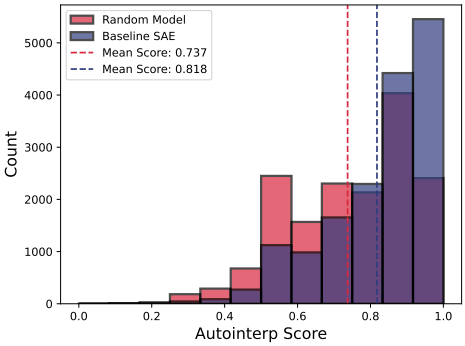
Figure 16: **Evolution of feature stability across layers.** Left: layer-wise fraction of stable (blue) and unstable (red) SAE features, showing how reproducible and non-reproducible dictionary elements evolve with depth. Right: for stable (blue) and unstable (red) source features separately, the fraction with a next-layer decoder-cosine match at threshold $\tau_{\text{layer}} = 0.7$, measuring cross-layer feature matching.

G.4 Automatic interpretation on trained vs. random transformers

Figure 17b compares automatic interpretation (detection) scores for SAEs trained on activations from a trained GPT-2 versus a randomly initialized GPT-2 (same SAE architecture and hyperparameters). This shows that high detection scores can occur even in the random-model setting, motivating stability as a complementary faithfulness signal.



(a) **Stable fraction vs. SAE training tokens.** Fraction of stable features in the main TopK setting as a function of total SAE training tokens.



(b) **Automatic interpretation scores for SAEs trained on trained vs. random base models.** High detection scores are achievable even on random-model activations, despite the absence of cross-seed reproducibility.

Figure 17: **Additional controls and training dynamics.** Left: the stable fraction increases with more SAE training, but with diminishing returns. Right: high automatic-interpretation scores remain possible even for SAEs trained on random-model activations.