Beyond SAEs: Consistent Concept Discovery with Sparse Coding

Anonymous authors
Paper under double-blind review

Abstract

Sparse autoencoders (SAEs) explain foundation model representations by decomposing activations into sparse combinations of learned dictionary atoms. However, the learned dictionaries can vary across training runs, limiting the reliability and reproducibility of SAE-based analyses. We propose Consistent Sparse Dictionary Learning (ConSDL), which replaces the standard linear encoder with per-sample sparse coding. This is motivated by our synthetic analysis showing that coefficients from a single linear encoder pass are suboptimal compared with those obtained by per-sample optimization, suggesting that sparse coding is a reliable mechanism for dictionary learning. Across 16 synthetic settings, ConSDL achieves near-perfect dictionary and coefficient recovery. These gains translate to state-of-the-art cross-seed consistency across four vision and language settings, where ConSDL improves dictionary and coefficient consistency by 17.6% and 35.9%, respectively, over the average of six SAE baselines.

1 Introduction

Sparse autoencoders (SAEs) (Cunningham et al., 2023) have become a widely used approach for analyzing the internal representations of foundation models. SAEs decompose each model activation into a sparse linear combination of vectors from a learned overcomplete dictionary. Empirically, these dictionary vectors often align with human-interpretable concepts, such as colors, textures, and objects in vision models or characters, words, and syntactic patterns in language models (Elhage et al., 2022). Importantly, these concepts can be learned without a predefined vocabulary, enabling scalable explanations of model behavior in vision (Lim et al., 2025) and language (Templeton et al., 2026), audits of hidden biases in models and datasets (Ahsan & Wallace, 2025; Choi et al., 2026), and identification of candidate factors in scientific and medical data (Dasdelen et al., 2025; Nerrise et al., 2026).

Yet, a critical limitation remains before SAEs can serve as practical tools for analysis. SAEs trained on the same data with the same configuration can learn substantially different dictionaries across random seeds (Paulo & Belrose, 2025). This limits reproducibility and reliability. More fundamentally, it raises the question of whether the features learned by SAEs genuinely reflect the model's internal representations or instead produces artifacts of the training procedure. Recent work has further heightened this concern by showing that SAEs can fail to recover known ground-truth features in controlled synthetic settings (Korznikov et al., 2026; Pacela et al., 2026).

In this paper, we propose Consistent Sparse Dictionary Learning (ConSDL) to make sparse decompositions of model activations consistent across independent training runs, enabling reliable and reproducible analysis. To this end, ConSDL replaces the linear encoder of SAEs with per-sample sparse coding. The algorithm design is motivated by prior work showing that coefficients predicted by a single encoder pass can be suboptimal compared with those obtained by per-sample optimization (O'Neill et al., 2024). We hypothesize that updating the dictionary with these suboptimal coefficients contributes to inconsistency across training runs. ConSDL therefore uses per-sample sparse coding throughout training to obtain coefficients before each dictionary update. To make this procedure stable in practice, we

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.

additionally use coefficient refitting to correct shrinkage from the sparsity penalty and a co-active similarity regularizer to reduce dictionary redundancy.

Across diverse synthetic and real-world settings, ConSDL consistently outperforms existing SAEs. In controlled synthetic experiments, ConSDL achieves near-perfect recovery of the ground-truth dictionary and coefficients. It remains robust in challenging settings where existing methods degrade, supporting our hypothesis that improved coefficient estimation leads to more consistent dictionary learning. On vision and language foundation models, ConSDL substantially improves dictionary and coefficient consistency over existing methods.

Our contributions are threefold. First, we introduce ConSDL, which incorporates per-sample sparse coding to produce consistent sparse decompositions across training runs. Second, we systematically study SAE recovery under synthetic conditions and identify when existing methods fail and what factors contribute to these failures. Third, we show that ConSDL improves cross-seed dictionary and coefficient consistency on foundation-model activations across multiple vision and language backbones, dictionary widths, and sparsity levels.

2 Related work

Consistency of SAEs. Recent work has drawn attention to the inconsistency of SAEs, showing that independent training runs can learn substantially different dictionaries (Paulo & Belrose, 2025). Fel et al. (2025) improves dictionary consistency by constraining dictionary atoms to lie within the convex hull of the activation data. Nelson et al. (2026b) instead regularizes the dictionary geometry to reduce ambiguity among sparse codes, improving dictionary and coefficient consistency. However, these studies differ in key experimental choices, including dictionary size, sparsity, and training procedure, making direct comparison difficult. Song et al. (2025) further show that SAE consistency can vary substantially across such configurations. It therefore remains unclear whether the reported gains hold under matched settings. We address this by comparing existing methods across a broad range of settings using the same evaluation protocol.

Sparse coding and dictionary learning. Given a fixed dictionary, sparse coding infers a sparse coefficient vector for every input, such that a linear combination of dictionary atoms with the inferred coefficients reconstructs the input (Olshausen & Field, 1997). A common formulation uses an ℓ_1 penalty and can be solved with ISTA (Daubechies et al., 2004) or its accelerated variant FISTA (Beck & Teboulle, 2009). Alternatively, matching pursuit greedily selects atoms that best explain the current residual (Mallat & Zhang, 1993). Dictionary learning jointly learns the dictionary by alternating between sparse coefficient estimation and dictionary updates (Aharon et al., 2006; Mairal et al., 2010). Our proposed ConSDL adopts this alternating formulation, using FISTA for per-sample sparse inference before each dictionary update. Under suitable assumptions, such iterative approaches are theoretically guaranteed to recover the underlying dictionary and coefficients (Agarwal et al., 2016).

From amortized inference to per-sample sparse coding. Standard SAEs use a shared learned encoder to predict sparse coefficients rather than optimizing them separately for each input. This amortized inference (Marino et al., 2018) avoids costly per-sample optimization and enables efficient large-scale training, but at a cost to coefficient quality. O'Neill et al. (2024) show that optimizing coefficients directly under a fixed dictionary substantially improves recovery. Sun et al. (2026) connect amortized inference to feature absorption, splitting, and dead atoms. Joung & Bae (2026) retain an amortized encoder but iteratively refine its predicted coefficients during dictionary learning. Pacela et al. (2026) instead remove the learned encoder and use per-sample sparse coding throughout training, but report degraded recovery for larger dictionaries. Our ConSDL builds on this non-amortized approach while improving its stability and scalability to foundation-model activations. Additional related work is discussed in Appendix A.

3 Method

We propose Consistent Sparse Dictionary Learning (ConSDL) to improve the consistency of dictionaries and coefficients across training runs (Fig. 1). ConSDL replaces the learned

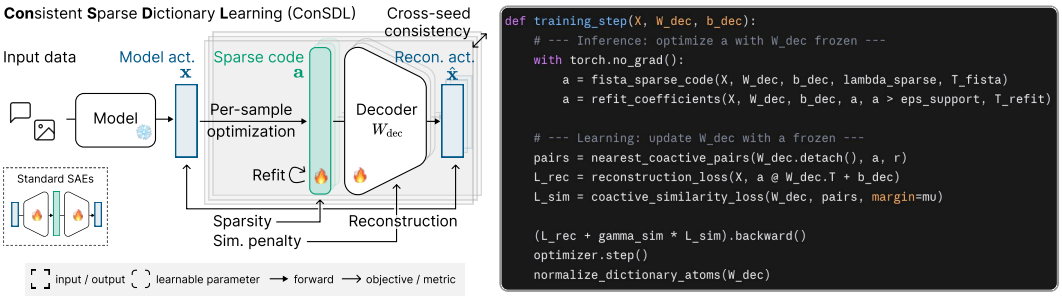

Figure 1: **Method. Left**: Consistent Sparse Dictionary Learning (ConSDL) alternates between per-sample sparse coefficient inference with refitting and decoder updates under a reconstruction objective with co-active similarity regularization, holding the other component fixed at each stage. Standard SAEs (inset) instead use a learned encoder trained jointly with the decoder. Stacked panels indicate independent training seeds, across which ConSDL aims to recover consistent codes and dictionaries. **Right**: Python-style pseudocode.

encoder with per-input coefficient optimization and updates the dictionary from these estimates. It then refits the magnitudes of the selected coefficients without the sparsity penalty to correct shrinkage from sparse coding, and encourages co-active atoms to remain distinct by penalizing the similarity of the most similar co-active pair.

3.1 Per-sample Sparse Coding

Let $\mathbf{x} \in \mathbb{R}^d$ denote a model activation with dimension d. Let $W_{\text{dec}} \in \mathbb{R}^{d \times m}$ denote the learned dictionary, corresponding to the decoder weight in an SAE, where m is the number of dictionary atoms. Its j-th column $\mathbf{w}_j \in \mathbb{R}^d$ represents a unit-norm dictionary atom. Let $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$ denote the decoder bias, and $\mathbf{a} \in \mathbb{R}^m_{\geq 0}$ denote the nonnegative coefficient vector, with $a_j \in \mathbb{R}$ specifying the contribution of atom j to the reconstruction.

Standard SAEs use a linear encoder followed by a non-linearity to predict the sparse coefficients $\mathbf{a}$ from each input in a single forward pass. These coefficients directly influence the gradients used to update each dictionary atom during training, yet coefficients produced by a single encoder pass can be suboptimal compared with those optimized for the current dictionary (O'Neill et al., 2024). We therefore replace the encoder with per-sample sparse coding and infer the coefficients for each input under the current dictionary with a sparsity penalty. Specifically, we solve

$$\mathbf{a}^{\text{sc}}(\mathbf{x}) = \arg\min_{\mathbf{a} \geq 0} \frac{1}{2} \|\mathbf{x} - W_{\text{dec}}\mathbf{a} - \mathbf{b}_{\text{dec}}\|_2^2 + \lambda \|\mathbf{a}\|_1, \tag{1}$$

where $\lambda > 0$ controls the sparsity. We approximate this solution using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) (Beck & Teboulle, 2009), initialized from zero. Each FISTA iteration updates an estimate of the coefficients by taking a gradient step on the reconstruction term followed by nonnegative soft thresholding for the ℓ_1 penalty.

3.2 Coefficient Refitting

While the ℓ_1 penalty promotes sparsity by penalizing coefficient magnitude, it also introduces *shrinkage bias*: even after the active atoms are correctly identified, the optimization objective favors smaller nonzero coefficients to reduce the penalty (Zou, 2006). Using these shrunken coefficients directly for decoder updates can underestimate the contributions of the selected atoms. We therefore retain the support identified by FISTA and refit the coefficients on this fixed support without the ℓ_1 penalty, following post-selection refitting procedures in sparse estimation (Figueiredo et al., 2007; Chzhen et al., 2019). Let $S^{\text{sc}}(\mathbf{x}) = \{j : a_j^{\text{sc}}(\mathbf{x}) > 0\}$ denote the support selected by FISTA. Given this support, we compute

$$\mathbf{a}^{\text{refit}}(\mathbf{x}) = \arg\min_{a_j = 0 \ \forall j \notin S^{\text{sc}}(\mathbf{x})} \frac{1}{2} \|\mathbf{x} - W_{\text{dec}}\mathbf{a} - \mathbf{b}_{\text{dec}}\|_2^2, \tag{2}$$

which is a nonnegative least-squares problem over the selected support. We approximate its solution using projected gradient descent initialized from $\mathbf{a}^{\text{sc}}(\mathbf{x})$, constraining the coefficients to remain nonnegative and supported on $S^{\text{sc}}(\mathbf{x})$. Thus, refitting adjusts only the magnitudes of the selected coefficients without activating new atoms. Additional details on the FISTA optimization and refitting procedure are provided in Appendix B.

3.3 Co-active Similarity Regularization

Per-sample sparse coding improves coefficient estimation, but applying it throughout dictionary learning can still lead multiple atoms to converge to similar directions, resulting in redundant dictionary elements. To stabilize dictionary learning in practice, we introduce a co-active similarity regularizer that penalizes high similarity only between atoms active for the same sample, rather than between all atom pairs in the dictionary.

For each input $\mathbf{x}_i$, let $A_i = \{\mathbf{w}_j : a_{i,j}^{\text{refit}} > 0\}$ denote the set of active dictionary atoms, where j refers to the atom index. For each $\mathbf{w}_j \in A_i$, we define its most similar co-active atom as

$$\widetilde{\mathbf{w}}_{i,j} = \underset{\mathbf{w} \in A_i \setminus \{\mathbf{w}_j\}}{\arg\max}\ \mathbf{w}_j^\top \mathbf{w}, \qquad s_{i,j} = \mathbf{w}_j^\top \widetilde{\mathbf{w}}_{i,j}. \tag{3}$$

where $s_{i,j}$ is their cosine similarity, since the atoms are unit normalized. To focus the regularization on highly similar pairs, we follow margin-based contrastive objectives (Hadsell et al., 2006) and formulate the co-active similarity penalty as

$$\mathcal{L}_{\text{sim}} = \frac{1}{|\mathcal{V}_\mu|} \sum_{(i,j) \in \mathcal{V}_\mu} (s_{i,j} - \mu)^2, \tag{4}$$

where $\mu \in [0, 1)$ is a similarity margin and $\mathcal{V}_\mu = \{(i, j) : \mathbf{w}_j \in A_i,\ s_{i,j} > \mu\}$ is the set of matched co-active pairs whose similarity exceeds the margin. Implementation details of the regularizer are provided in Appendix B.

3.4 Training Objective and Alternating Optimization

Given the refitted coefficients, we define the reconstruction loss as

$$\mathcal{L}_{\text{rec}} = \frac{1}{2} \left\| \mathbf{x} - W_{\text{dec}} \mathbf{a}^{\text{refit}}(\mathbf{x}) - \mathbf{b}_{\text{dec}} \right\|_2^2. \tag{5}$$

The overall training objective is $\mathcal{L} = \mathcal{L}_{\text{rec}} + \gamma_{\text{sim}} \mathcal{L}_{\text{sim}}$, where $\gamma_{\text{sim}} > 0$ controls the strength of co-active similarity regularization. For each update, we first infer and refit the coefficients under the current dictionary and identify the co-active atom pairs. These quantities are then held fixed while optimizing W_{dec} and $\mathbf{b}_{\text{dec}}$. Thus, gradients do not propagate through coefficient estimation or pair selection, and training alternates between per-sample sparse coding and dictionary updates. Full optimization details are provided in Appendix B, with hyperparameter settings in Appendix D.

4 Synthetic Analysis of Dictionary and Coefficient Recovery

To study the conditions under which existing SAE methods recover the underlying dictionary and coefficients, and whether ConSDL improves recovery where they fail, we use controlled synthetic data with a known ground-truth decomposition.

4.1 Experimental Setup

Synthetic data generation. We generate synthetic data as $\mathbf{x} = \sum_{i=1}^{m} a_i \mathbf{w}_i$, where $\mathbf{x} \in \mathbb{R}^d$ is an observed activation, $\mathbf{w}_i \in \mathbb{R}^d$ denotes a ground-truth dictionary atom, and $a_i \in \mathbb{R}_{\geq 0}$ is its nonnegative coefficient. In the baseline setting, we use activation dimension $d = 768$, dictionary width $m = 3072$, and an expected k=60 active atoms per sample (Fig. 2A). Following prior synthetic SAE studies (Korznikov et al., 2026; O'Neill et al., 2024), we sample dictionary atoms independently from an isotropic Gaussian distribution and normalize them

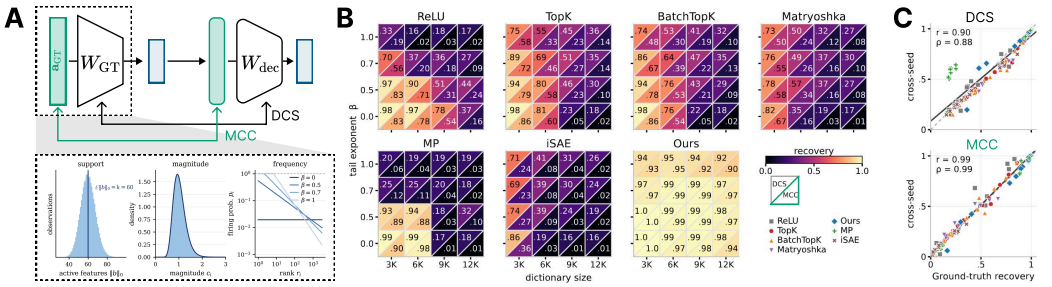

Figure 2: **Synthetic recovery and cross-seed consistency. (A)** Activations are generated from a ground-truth dictionary W_{GT} with sparse codes a_{GT}. Insets show the sampling distributions for the active-set size, coefficient magnitudes, and atom selection frequencies. **(B)** Dictionary (DCS; upper-left) and coefficient (MCC; lower-right) recovery across dictionary sizes and tail exponents β. **(C)** Cross-seed consistency correlates with ground-truth recovery for both dictionaries (top) and coefficients (bottom). Each point is one method at a given dictionary size and β. r and ρ denote the Pearson and Spearman correlation coefficients.

to unit norm. For each sample, atoms are selected independently with equal probability, yielding random sparse supports over an approximately incoherent dictionary, as commonly assumed in sparse coding analyses (Arora et al., 2015). Each nonzero coefficient is drawn from a log-normal distribution with expected value 1.

We extend the baseline setting along two axes motivated by properties of foundation-model representations: dictionary size and atom-selection imbalance. First, we vary the ground-truth dictionary width over 4×, 8×, 12×, and 16× the activation dimension, corresponding to $m = 3072$–12288. This range reflects evidence that larger dictionaries continue to uncover additional concepts in foundation model activations (Templeton et al., 2026). Second, we vary atom-selection imbalance using Zipf exponents $\beta \in \{0, 0.5, 0.7, 1.0\}$, where $\beta = 0$ corresponds to uniform selection and larger values produce increasingly long-tailed frequencies, reflecting the highly imbalanced concept frequencies in web-scale pretraining data (Udandarao et al., 2024). This results in 16 synthetic configurations.

Baselines. We compare a standard ReLU SAE (Cunningham et al., 2023) with TopK (Gao et al., 2025), BatchTopK (Bussmann et al., 2024), Matryoshka SAE (Bussmann et al., 2025), MP-SAE (Costa et al., 2026), and iSAE (Hindupur et al., 2026). For each synthetic configuration, we generate 10^6 observations and train each method for five epochs, reducing the likelihood that recovery failures arise from insufficient data or training. We repeat each experiment with three random seeds.

Metrics. We evaluate dictionary recovery using decoder cosine similarity (DCS) (Paulo & Belrose, 2025) after Hungarian matching to the ground-truth dictionary. Coefficient recovery is measured by mean coefficient correlation (MCC) (Pacela et al., 2026) under the same matching. For cross-seed consistency, we apply the same metrics between independently trained runs and average over all seed pairs. Higher values indicate better recovery or consistency. Full metric definitions and experimental details are provided in Appendix D.

4.2 Findings

Recovery strongly depends on the data setting. Despite differences across SAE variants, dictionary and coefficient recovery are generally high for smaller dictionaries and less long-tailed atom selection distributions (lower-left region of each heatmap in Fig. 2B). As dictionary size grows and atom selection becomes more long-tailed, both decline sharply (upper-right region). This sensitivity may help explain why prior synthetic studies have reported both successful (Song et al., 2025) and failed (Pacela et al., 2026) SAE recovery. Since it is unclear which setting best reflects real model activations, these results motivate methods that remain reliable across a wider range of conditions.

Amortization gap in coefficient inference. Even in settings with high dictionary recovery, the encoder does not fully recover the corresponding coefficients. One possible explanation is the amortization gap identified in prior work (O'Neill et al., 2024). We examine this by re-optimizing the coefficients for each sample while keeping the learned dictionary fixed. This improves both coefficient recovery and input reconstruction (Table 5), showing that the encoder leaves recoverable error in coefficient inference. However, when dictionary recovery is poor, coefficient re-optimization yields limited gains, suggesting that a more fundamental solution requires reliable dictionary learning.

ConSDL remains robust under challenging synthetic conditions. Across all 16 synthetic settings, ConSDL achieves near-perfect recovery of both the ground-truth coefficients and dictionary, including conditions where existing SAE methods degrade (Fig. 2B). This shows that incorporating per-sample sparse coding into dictionary learning not only addresses the coefficient-inference gap, but also improves dictionary recovery when incorporated into dictionary learning. By recovering nearly the same ground-truth decomposition across runs, ConSDL also achieves high cross-seed consistency. Full results are provided in Appendix E (Tables 9 – 12).

Cross-seed consistency is associated with ground-truth recovery. Across the synthetic sweeps, higher cross-seed consistency corresponds to stronger ground-truth recovery (Fig. 2C). Decoder consistency is strongly correlated with dictionary recovery (Pearson $r = 0.90$, Spearman $\rho = 0.88$), while coefficient consistency shows an even stronger correlation with coefficient recovery ($r = 0.99$, $\rho = 0.98$). Although cross-seed consistency does not by itself establish ground-truth recovery, these strong correlations support its use as a complementary measure for real model activations, where the ground truth is unavailable.

5 Evaluation on Foundation Model Activations

We next evaluate whether the gains observed in synthetic settings extend to foundation-model activations in vision and language domains. Since ground-truth decompositions are unavailable, we use cross-seed consistency as the primary evaluation criterion.

5.1 Experimental Setup and Evaluation

Foundation model activations. For vision, we train SAEs on penultimate-layer activations from CLIP ViT-B/16 (Radford et al., 2021) and DINOv2 (Oquab et al., 2023). We use ImageNet (Russakovsky et al., 2015) and include both class-token and spatial-token activations following prior work (Lim et al., 2025), training each SAE for one epoch over approximately 1.2M images. For language, we train SAEs on layer-9 activations from GPT-2 Small (Radford et al., 2019) and Pythia-160M (Biderman et al., 2023). We train each SAE on 500M FineWeb tokens (Penedo et al., 2024) with a context length of 1024. The activation dimension is 768 for all four models. We evaluate the trained SAEs on the ImageNet validation set for vision and on 20M FineWeb tokens for language. Additional model, data, and training details are provided in Appendix D.4.

Baselines and metrics. We use the same SAE baselines as in the synthetic evaluation. We evaluate dictionary widths of 4×, 8×, and 16× the activation dimension, each at three sparsity levels, $\ell_0 \approx 32$, 64, and 128. Because ground-truth decompositions are unavailable for foundation models, we evaluate cross-seed consistency of the dictionary and coefficients by training each configuration with three random seeds and computing `DCS` and `MCC` between independently trained runs, as in the synthetic evaluation. We further report dictionary utilization metrics to assess whether improved consistency comes at the cost of dictionary capacity. We measure the usable atom fraction, defined as the fraction of atoms that are neither dead nor duplicated. Dead atoms are never activated, while duplicated atoms point in highly similar decoder directions, reducing the diversity of concepts represented by the dictionary.

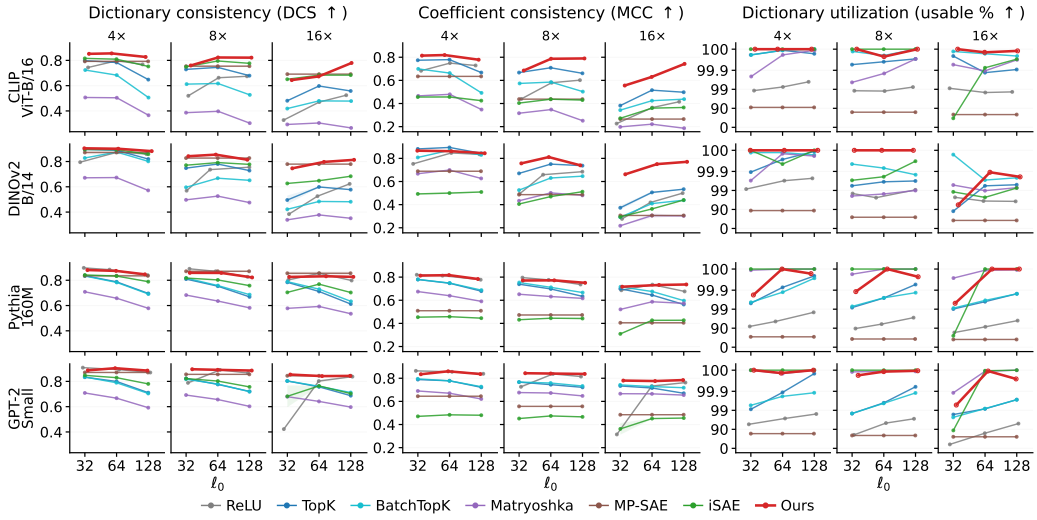

Figure 3: **Consistency and dictionary utilization of foundation-model activations.** We compare seven methods across two vision and two language backbones, three expansion factors, and three sparsity levels ℓ_0. Cross-seed dictionary and coefficient consistency are measured by DCS and MCC, and dictionary utilization by the fraction of atoms that are neither dead nor duplicated. ConSDL achieves consistently higher cross-seed consistency, with the largest gains at 16×, while maintaining high dictionary utilization.

5.2 Results

ConSDL is more consistent than baselines across vision and language backbones. Across 36 configurations spanning four backbones, three dictionary sizes, and three sparsity levels, ConSDL improves both dictionary and coefficient consistency over existing SAE baselines (Fig. 3). Averaged across backbones, ConSDL achieves a DCS of 0.84 and an MCC of 0.78, corresponding to relative gains of 17.6% and 35.9% over the baseline average (Table 6). Compared with the strongest baseline for each metric, ConSDL improves dictionary and coefficient consistency by 1.9% and 13.3% on average. No single baseline is competitive on both metrics: MP-SAE achieves comparable DCS but substantially lower MCC, while ReLU and TopK achieve stronger coefficient consistency among the baselines but lower DCS. ConSDL maintains high consistency in both the learned dictionaries and coefficients across vision and language backbones.[1] Detailed results for each backbone and configuration are provided in Appendix E (Tables 13 – 16).

Consistency gains do not come at the cost of dictionary quality. Duplicate atoms appear across several baselines, particularly ReLU and MP-SAE, while TopK-based methods and iSAE contain many dead atoms in some settings. As shown in the dictionary utilization panel of Fig. 3, ConSDL maintains high usable atom fractions across backbones, dictionary sizes, and sparsity levels. Averaged across backbones, ConSDL achieves a usable atom fraction of 1.00, a 5.6% improvement over the baseline average and matching the strongest baseline (Table 6). Reconstruction quality remains at a similar level to the baselines across configurations. We also report input R^2 in Tables 13 – 16.

5.3 Ablation Study

Setup. We ablate coefficient refitting and co-active similarity regularization to identify how each component contributes to the cross-seed consistency of ConSDL. We evaluate 16× dictionaries at $\ell_0 \approx 128$ on CLIP ViT-B/16 and Pythia-160M (Table 1). Because removing either component changes the resulting sparsity, we tune the sparsity penalty separately for each variant to obtain comparable ℓ_0.

[1] We also compare DCS and MCC under activation-footprint matching instead of decoder cosine similarity, and obtain the same ranking across methods. See Table 4.

Table 1: **Ablation study at 16× expansion and matched sparsity ($\ell_0 \approx 128$).** Starting from per-sample sparse coding, we add coefficient refitting or similarity regularization individually, while **Full** combines both. Since these components affect sparsity, we compare configurations matched by ℓ_0. Values are mean ± standard deviation over three seeds.

Backbone	Variant	Sparsity		Reconstruction	Cross-seed Consistency		Dictionary Utilization		
		λ	ℓ_0	R^2	DCS	MCC	Usable atom frac.	# Dup.	# Dead
CLIP ViT-B/16	Sparse coding	1.2	116 ±5.36	0.87 ±0.00	0.84 ±0.00	**0.72** ±0.00	0.25 ±0.00	9,270 ±35.3	0 ±0.00
	+ Refitting	0.8	116 ±1.88	**0.92** ±0.00	**0.85** ±0.00	0.70 ±0.00	0.55 ±0.00	5,564 ±74.9	0 ±0.00
	+ Similarity reg.	1.0	110 ±2.56	0.89 ±0.00	0.70 ±0.00	0.49 ±0.00	1.00 ±0.00	56 ±1.76	0 ±0.00
	Full	0.75	130 ±2.13	0.92 ±0.00	0.78 ±0.00	0.66 ±0.00	**1.00** ±0.00	**1** ±0.90	0 ±0.00
Pythia-160M	Sparse coding	3.0	113 ±2.45	0.69 ±0.00	0.29 ±0.00	0.17 ±0.00	0.03 ±0.01	2,144 ±52.4	9,767 ±103
	+ Refitting	1.5	108 ±0.32	0.92 ±0.02	**0.84** ±0.00	**0.73** ±0.00	0.74 ±0.00	3,191 ±26.7	0 ±0.00
	+ Similarity reg.	2.9	133 ±2.48	0.72 ±0.00	0.30 ±0.00	0.16 ±0.00	0.28 ±0.02	239 ±26.0	8,652 ±345
	Full	1.0	110 ±0.47	**0.93** ±0.00	0.81 ±0.00	0.70 ±0.00	**1.00** ±0.00	**0** ±0.00	0 ±0.00

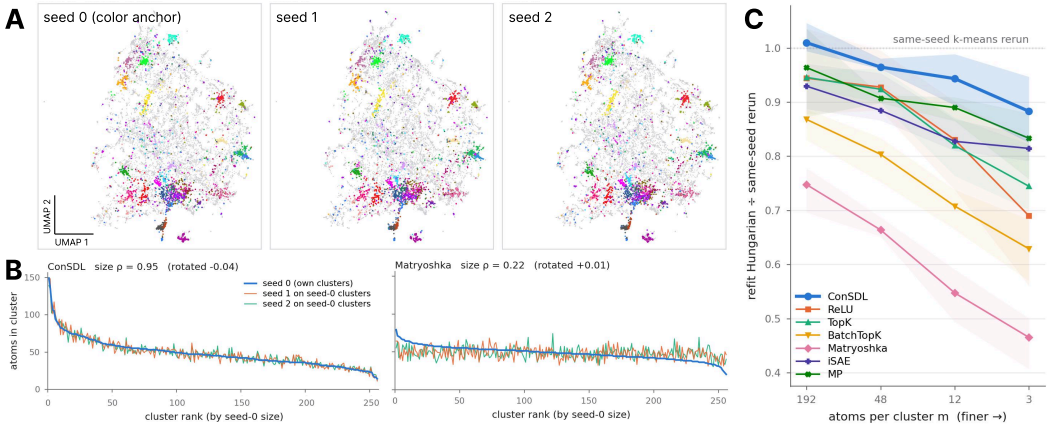

Figure 4: **Dictionary cluster consistency across training runs. (A)** Joint UMAP of ConSDL decoder atoms at 16×, $\ell_0 \approx 128$, colored by top 30 clusters from a reference run. **(B)** Cluster-size profiles for ConSDL and Matryoshka. ρ denotes the Spearman correlation of cluster sizes across runs. **(C)** Normalized cluster-centroid agreement across granularities. Lines show the mean over nine CLIP settings, and shading shows the interquartile range.

Coefficient refitting stabilizes dictionary learning. Comparing sparse coding (SC) with + Refitting, SC requires a larger λ to reach a similar sparsity level, especially on Pythia (3.0 vs.1.5). This stronger sparsity penalty is accompanied by 9,767 dead atoms and substantially lower reconstruction quality ($R^2 = 0.69$). This severe failure of SC on Pythia is consistent with prior reports that sparse-coding-based dictionary learning can degrade at larger scales (Pacela et al., 2026). On CLIP, SC does not produce dead atoms but still requires a stronger sparsity penalty. Refitting removes the dead atoms on Pythia and raises R^2 to 0.92 on both backbones, from 0.69 on Pythia and 0.87 on CLIP.

Similarity regularization reduces redundancy. However, + Refitting still leaves thousands of duplicated atoms on both backbones. Comparing + Refitting with Full ConSDL, similarity regularization reduces duplicated atoms from 5,564 to 1 on CLIP and from 3,191 to 0 on Pythia. The same effect appears without refitting. Applying similarity regularization without refitting (+ Similarity Reg.) also substantially reduces duplication, but does not resolve the instability observed on Pythia. These results indicate that both components are necessary for stable dictionary learning.

5.4 Analysis

Consistency extends beyond individual dictionary atoms. We next examine whether atom-level consistency extends to the overall structure of the dictionary. We focus on CLIP ViT-B/16 and cluster atoms based on decoder direction similarity. We select one training run as a reference and evaluate whether other runs recover similar cluster sizes and centroids. For cluster sizes, we assign atoms from each run to the nearest reference centroid and

compare the resulting cluster sizes using Spearman correlation. For cluster centroids, we independently cluster each run, match its centroids one-to-one to the reference centroids, and measure their cosine similarity. We compare this agreement with that obtained by rerunning k-means on the reference dictionary to account for clustering variability.

As shown in Fig. 4, ConSDL reproduces both cluster sizes and cluster centroids more consistently across training runs. At 16× expansion and $\ell_0 \approx 128$, its cluster-size rank correlation reaches 0.95, compared with 0.22 for Matryoshka (Fig. 4B). ConSDL also achieves the highest normalized centroid agreement at every granularity, with scores of 1, 0.96, 0.94, and 0.88 at 192, 48, 12, and 3 atoms per cluster, respectively (Fig. 4C). A score of 1 indicates that cross-run agreement matches that obtained by rerunning k-means on the same dictionary. The gap between methods widens at finer granularities, indicating that ConSDL more reliably preserves fine-grained dictionary structure across training runs.

ConSDL preserves sparse probing performance. We next use sparse probing (Gurnee et al., 2023) to test whether class relevant information can be captured by a small number of dictionary atoms. Concentrating such information in a few atoms can make the learned representation easier to inspect and interpret. Across vision and language models, ConSDL performs close to the strongest SAE baselines, achieving the best results in several settings and otherwise differing only marginally (Tables 7 and 8). This suggests that the improved cross-seed consistency of ConSDL do not come at the cost of information localized to a small number of dictionary atoms. Additional setup details are provided in Appendix E.3. Additional analyses are provided in Appendix E.4, including qualitative examples of consistent concepts in CLIP (Fig. 6) and Pythia-160M (Fig. 7).

6 Discussion and Limitations

Although ConSDL improves cross-seed consistency over existing SAEs on foundation-model activations, it remains lower than the recovery achieved in synthetic settings. This gap may reflect the greater complexity of real activations, including a larger number of underlying concepts, stronger frequency imbalance, and deviations from the assumption that activations are sparse linear combinations of concept directions. Recent work has also questioned whether all concepts can be represented by individual linear directions, suggesting that some may instead exhibit nonlinear or manifold-like structure (Bhalla et al., 2026). However, some meaningful structure may still be representable by linear directions (Park et al., 2023). ConSDL therefore focuses on improving cross-seed consistency within this sparse linear assumption, rather than addressing whether all concepts can be represented linearly.

ConSDL replaces a single encoder pass with iterative per-sample sparse coding, increasing training cost relative to standard SAE baselines. We analyze training and inference costs in Appendix C. This additional computation reflects a trade-off between efficiency and improved consistency. Future work could reduce this overhead through more efficient sparse solvers. Furthermore, our experiments focus on relatively small foundation models because evaluating multiple methods across dictionary widths, sparsity, and random seeds is computationally demanding. We therefore do not establish whether the gains in cross-seed consistency extend to models with billions of parameters or substantially larger dictionaries. Evaluating ConSDL at larger scales remains an important direction for future work.

7 Conclusion

In this work, we introduce ConSDL, which revisits classical sparse coding to improve the cross-seed consistency over existing SAEs. By replacing the linear encoder with per-sample sparse coding and incorporating coefficient refitting and regularization against redundancy, ConSDL learns more consistent dictionaries and coefficients. It achieves substantially better ground-truth recovery across diverse synthetic settings. These benefits also transfer to foundation-model activations, where ConSDL improves cross-seed dictionary and coefficient consistency across multiple vision and language backbones, dictionary widths, and sparsity levels, while improving dictionary utilization. Future work will investigate scaling ConSDL to larger foundation models and extending it beyond linear feature representations.

AI use statement

In this work, we used generative AI tools to assist with research ideation and experimental design, implementation of methods and experimental pipelines, retrieval and discovery of related work, drafting portions of the manuscript, and editing for English grammar and clarity. We did not use generative AI tools for other tasks requiring disclosure, and the remaining required-disclosure tasks were not applicable to this work. All AI-assisted research decisions, code, retrieved references, and manuscript text were reviewed, validated, and revised by the authors as appropriate. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

This work does not involve human subjects, collection of personal data, or deployment in safety-critical settings. We are not aware of any additional ethical concerns specific to this work.

Reproducibility statement

We provide detailed descriptions of the proposed method, experimental setup, evaluation protocols, and training hyperparameters in the main text and appendix. All datasets and pretrained model checkpoints used in our experiments are publicly available and cited in the corresponding sections. We are also preparing a public release of the source code and experiment configurations, which will be made available upon publication.

References

Alekh Agarwal, Animashree Anandkumar, Prateek Jain, and Praneeth Netrapalli. Learning sparsely used overcomplete dictionaries via alternating minimization. *SIAM Journal on Optimization*, 26(4):2775–2799, 2016.

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

Hiba Ahsan and Byron C Wallace. Can saes reveal and mitigate racial biases of llms in healthcare? *arXiv preprint arXiv:2511.00177*, 2025.

Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. In *Conference on learning theory*, pp. 113–149. PMLR, 2015.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Jack Merullo, et al. Do sparse autoencoders capture concept manifolds? *arXiv preprint arXiv:2604.28119*, 2026.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O' Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu,

Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

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

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

Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8): 1207–1223, 2006.

Jinho Choi, Hyesu Lim, Steffen Schneider, and Jaegul Choo. Conceptscope: Characterizing dataset bias via disentangled visual concepts. *Advances in Neural Information Processing Systems*, 38:9604–9639, 2026.

Evgenii Chzhen, Mohamed Hebiri, and Joseph Salmon. On lasso refitting strategies. *Bernoulli*, 25(4A):3175–3200, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Valérie Costa, Thomas Fel, Ekdeep S Lubana, Bahareh Tolooshams, and Demba Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. *Advances in Neural Information Processing Systems*, 38:34381–34424, 2026.

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.

Muhammed Furkan Dasdelen, Hyesu Lim, Michele Buck, Katharina S Götze, Carsten Marr, and Steffen Schneider. Cytosae: Interpretable cell embeddings for hematology. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 77–86. Springer, 2025.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

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

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

Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing*, 1(4):586–597, 2007.

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 *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset, 2007.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Sai Sumedh R Hindupur, Ekdeep S Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *Advances in Neural Information Processing Systems*, 38:11649–11699, 2026.

Kyung-In Joung and Jang Pyo Bae. Fista-refined dictionary learning with learned topk initialization for sparse decomposition of transformer residual streams. *Algorithms*, 19(8): 697, 2026.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*, 2025.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, et al. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *arXiv preprint arXiv:2503.09532*, 2025.

Anton Korznikov, Andrey Galichin, Alexey Dontsov, Oleg Rogov, Ivan Oseledets, and Elena Tutubalina. Sanity checks for sparse autoencoders: Do saes beat random baselines? *arXiv preprint arXiv:2602.14111*, 2026.

Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2852–2861, 2017.

Hyesu Lim, Jinho Choi, Jaegul Choo, and Steffen Schneider. Sparse autoencoders reveal selective remapping of visual concepts during adaptation. In *International Conference on Learning Representations*, volume 2025, pp. 24444–24469, 2025.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of machine learning research*, 11(1), 2010.

Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pp. 3403–3412. PMLR, 2018.

Anish Mudide, Josh Engels, Eric Michaud, Max Tegmark, and Christian Schroeder de Witt. Efficient dictionary learning with switch sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 101830–101844, 2025.

Walter Nelson, Marco Fumero, Theofanis Karaletsos, and Francesco Locatello. Statistical and structural identifiability in representation learning. *arXiv preprint arXiv:2603.11970*, 2026a.

Walter Nelson, Theofanis Karaletsos, and Francesco Locatello. Toward identifiable sparse autoencoders. *arXiv preprint arXiv:2605.31245*, 2026b.

Favour Nerrise, Lucy Yin, Mohammad H Abbasi, Kilian M Pohl, and Ehsan Adeli. Geosae: Geometric prior-guided layer-wise sparse autoencoder annotation of brain mri foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6741–6749, 2026.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

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

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Vitória Barin Pacela, Shruti Joshi, Isabela Camacho, Simon Lacoste-Julien, and David Klindt. Stop probing, start coding: Why linear probes and sparse autoencoders fail at compositional generalisation. *arXiv preprint arXiv:2603.28744*, 2026.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

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

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37: 30811–30849, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Xiangchen Song, Aashiq Muhamed, Yujia Zheng, Lingjing Kong, Zeyu Tang, Mona T Diab, Virginia Smith, and Kun Zhang. Position: Mechanistic interpretability should prioritize feature consistency in saes. *arXiv preprint arXiv:2505.20254*, 2025.

Wenjie Sun, Di Wang, and Lijie Hu. The price of amortized inference in sparse autoencoders. In *The Fourteenth International Conference on Learning Representations*, 2026.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *arXiv preprint arXiv:2605.29358*, 2026.

Vishaal Udandarao, Ameya Prabhu, Adhiraj Ghosh, Yash Sharma, Philip H Torr, Adel Bibi, Samuel Albanie, and Matthias Bethge. No" zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. *Advances in Neural Information Processing Systems*, 37:61735–61792, 2024.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Xudong Zhu, Mohammad Mahdi Khalili, and Zhihui Zhu. Abstopk: Rethinking sparse autoencoders for bidirectional features. In *International Conference on Learning Representations*, volume 2026, pp. 50325–50352, 2026.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

Appendix

A Extended Related Work **16**

B Implementation Details of ConSDL **16**

C Computational Cost Analysis **18**

D Experimental Details **19**

D.1 Baselines 19

D.2 Evaluation metrics 20

D.2.1 Reconstruction 20

D.2.2 Dictionary utilization 20

D.2.3 cross-seed consistency 21

D.2.4 Matching basis 21

D.3 Details of Synthetic experimental 21

D.4 Details of Foundation Model Experiments 22

D.4.1 Model and Data Details 22

D.4.2 Optimization and hyperparameters 22

E Full Experimental Results **23**

E.1 Synthetic Results 23

E.2 Results on Foundation Model Activations 24

E.3 Sparse Probing Evaluation 24

E.4 Analysis: Region carve and cross-seed latent matching 26

A Extended Related Work

Improving SAE architectures and dictionaries. SAEs gained growing attention in mechanistic interpretability as their application to large language model activations demonstrated the potential to uncover large numbers of interpretable concepts (Bricken et al., 2023). Subsequent work initially focused on improving reconstruction while maintaining sparse activations. JumpReLU SAEs (Rajamanoharan et al., 2024) use learned activation thresholds to optimize sparsity more directly, while TopK SAEs (Gao et al., 2025) and BatchTopK SAEs (Bussmann et al., 2024) enforce sparsity through explicit feature selection. More recent work has shifted attention beyond reconstruction toward the quality and stability of the learned dictionary. SAEBench evaluates feature-level pathologies such as splitting and absorption (Karvonen et al., 2025), while Matryoshka SAEs explicitly target these failures through nested dictionaries (Bussmann et al., 2025). MP-SAE (Costa et al., 2026) instead predicts coefficients iteratively to better capture correlated concepts that may be missed by a single encoder pass.

Ground-Truth Recovery in Synthetic Settings. The inconsistency of SAE dictionaries raises a more fundamental question of identifiability (Nelson et al., 2026a), namely whether SAEs recover the true underlying decomposition. This cannot be determined from real model activations because the ground truth is unavailable. Prior work has therefore used synthetic data with known dictionaries and sparse coefficients to measure recovery directly, and report substantial recovery failures. (Korznikov et al., 2026; O'Neill et al., 2024; Pacela et al., 2026). However, as with prior consistency studies, the reported results are based on specific data distributions and training settings. It therefore remains unclear when SAE recovery succeeds and when it breaks down. We address this through a broader synthetic study spanning a range of difficulty levels.

Recovery guarantees for sparse coding and dictionary learning Sparse recovery theory characterizes when the underlying coefficients can be recovered from observations generated by a known dictionary (Donoho & Elad, 2003; Candes et al., 2006). Exact recovery is possible when the coefficient vector is sufficiently sparse and the dictionary atoms are sufficiently distinct, so that different sparse coefficient vectors do not produce the same observation (Donoho & Elad, 2003). Restricted isometry provides a stronger formal condition by requiring the dictionary mapping to approximately preserve the geometry of sparse vectors (Candes et al., 2006). When the dictionary is also unknown, the problem additionally requires identifying the dictionary from multiple sparse observations. Under suitable assumptions, including sufficiently many observations and an initialization close to the ground-truth dictionary, alternating minimization can recover both an overcomplete dictionary and its coefficients (Agarwal et al., 2016).

B Implementation Details of ConSDL

We provide additional implementation details for the sparse coding, coefficient refitting, and co-active repulsion procedures described in Sections 3.1 and 3.3.

FISTA optimization. Let $\mathbf{r} = \mathbf{x} - \mathbf{b}_{\text{dec}}$. We use the standard FISTA iterations (Beck & Teboulle, 2009), initialized with $\mathbf{a}_0 = \mathbf{a}_{-1} = \mathbf{0}$ and $t_0 = 1$. The momentum update is

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \qquad \mathbf{y}_k = \mathbf{a}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{a}_k - \mathbf{a}_{k-1}).$$

The proximal-gradient update is then

$$\mathbf{u}_k = \mathbf{y}_k - \eta W_{\text{dec}}^\top \left(W_{\text{dec}}\mathbf{y}_k - \mathbf{r}\right), \qquad \mathbf{a}_{k+1} = [\mathbf{u}_k - \eta\lambda]_+,$$

where $\mathbf{r} = \mathbf{x} - \mathbf{b}_{\text{dec}}$, λ controls the sparsity of the inferred coefficients, and $[\cdot]_+$ denotes element-wise projection onto nonnegative values.

The step size is set to

$$\eta = \frac{1}{\sigma_{\max}(W_{\text{dec}})^2},$$

```python
def training_step(X, W_dec, b_dec):
    # Sparse coding
    A_sc = fista_sparse_code(
        X, W_dec, b_dec, lambda_sparse, T_fista
    )

    # Coefficient refitting
    support = A_sc > eps_support
    A_refit = refit_coefficients(
        X, W_dec, b_dec, A_sc, support, T_refit
    )
    A_refit = stop_gradient(A_refit)

    # Co-active similarity
    neighbors = nearest_coactive_pairs(
        W_dec.detach(), A_refit, shortlist_size=r
    )

    X_hat = A_refit @ W_dec.T + b_dec
    L_rec = reconstruction_loss(X, X_hat)
    L_sim = coactive_similarity_loss(
        W_dec, neighbors, margin=mu
    )

    # Dictionary update
    loss = L_rec + gamma_sim * L_sim
    loss.backward()
    optimizer.step()
    normalize_dictionary_atoms(W_dec)
```

Figure 5: Python pseudocode for one training step.

where $\sigma_{\max}(W_{\text{dec}})$ is estimated from the current decoder using 32 power iterations. Since the decoder changes during training, the step size is recomputed under the current dictionary.

Coefficient refitting. Let $\mathbf{a}^{\text{sc}}(\mathbf{x})$ denote the coefficient vector returned by the FISTA solve above. Using its selected support $S^{\text{sc}}(\mathbf{x})$ defined in Section 3.1, we initialize $\mathbf{a}^{(0)} = \mathbf{a}^{\text{sc}}(\mathbf{x})$ and perform projected gradient descent on the reconstruction objective without the ℓ_1 penalty. Each iteration computes

$$\widetilde{\mathbf{a}}^{(q+1)} = \mathbf{a}^{(q)} - \eta W_{\text{dec}}^{\top}\left(W_{\text{dec}}\mathbf{a}^{(q)} - \mathbf{r}\right),$$

followed by

$$a_j^{(q+1)} = \begin{cases} \max(\widetilde{a}_j^{(q+1)}, 0), & j \in S^{\text{sc}}(\mathbf{x}), \\ 0, & j \notin S^{\text{sc}}(\mathbf{x}). \end{cases}$$

We use the same step size η as in the FISTA solve and perform 5–30 refitting iterations depending on the experimental setting. In implementation, a numerical threshold of 10^{-8} is used to determine the support returned by FISTA.

Co-active similarity loss. Section 3.3 presents the regularizer as a squared margin penalty for clarity. In implementation, we normalize the similarity above the margin and apply a logarithmic transformation. For each matched co-active pair with $s_{i,j} > \mu$, we use

$$\phi_\mu(s_{i,j}) = -\log\left[1 - \frac{1}{1+\delta}\left(\frac{s_{i,j}-\mu}{1-\mu}\right)^2\right],$$

where $\delta > 0$ is a small constant that keeps the loss finite as $s_{i,j} \to 1$. We then compute

$$\mathcal{L}_{\text{sim}} = \frac{1}{|\mathcal{V}_\mu|}\sum_{(i,j)\in\mathcal{V}_\mu}\phi_\mu(s_{i,j}),$$

Table 2: Per-token inference and training cost across SAE architectures and dictionary expansion factors. We report latency in μs with MFLOPs in parentheses, using matched sparsity budgets of $\ell_0 = 32$, 64, and 128 for 4×, 8×, and 16×, respectively.

Architecture	4×		8×		16×	
	Inference	**Training**	**Inference**	**Training**	**Inference**	**Training**
Pythia-160M						
ReLU	0.04 (4.7)	0.32 (28.3)	0.08 (9.4)	0.53 (56.6)	0.14 (18.9)	0.97 (113.2)
TopK	0.12 (4.7)	0.36 (28.3)	0.21 (9.4)	0.59 (56.6)	0.37 (18.9)	1.05 (113.2)
BatchTopK	0.07 (4.7)	0.48 (28.3)	0.13 (9.4)	0.75 (56.6)	0.24 (18.9)	1.40 (113.2)
Matryoshka	0.07 (4.7)	0.81 (33.0)	0.13 (9.4)	1.20 (66.1)	0.24 (18.9)	2.02 (132.1)
MP	2.38 (302.0)	2.23 (287.8)	8.38 (1,208.0)	3.78 (575.7)	31.06 (4,831.8)	7.14 (1,151.3)
iSAE	0.13 (4.7)	0.37 (28.3)	0.22 (9.4)	0.59 (56.6)	0.39 (18.9)	1.03 (113.2)
Ours	2.28 (325.6)	3.00 (338.6)	4.26 (651.2)	5.30 (684.2)	8.22 (1,302.3)	10.45 (1,396.7)
CLIP ViT-B/16						
ReLU	0.03 (4.7)	0.21 (28.3)	0.07 (9.4)	0.37 (56.6)	0.14 (18.9)	0.76 (113.2)
TopK	0.10 (4.7)	0.24 (28.3)	0.18 (9.4)	0.42 (56.6)	0.35 (18.9)	0.84 (113.2)
BatchTopK	0.06 (4.7)	0.31 (28.3)	0.12 (9.4)	0.58 (56.6)	0.23 (18.9)	1.18 (113.2)
Matryoshka	0.06 (4.7)	0.59 (33.0)	0.12 (9.4)	0.94 (66.1)	0.23 (18.9)	1.68 (132.1)
MP	1.88 (302.0)	1.61 (287.8)	7.22 (1,208.0)	2.89 (575.7)	29.01 (4,831.8)	6.15 (1,151.3)
iSAE	0.11 (4.7)	0.24 (28.3)	0.19 (9.4)	0.41 (56.6)	0.36 (18.9)	0.81 (113.2)
Ours	2.03 (325.6)	2.25 (335.6)	4.00 (651.2)	4.35 (672.3)	8.04 (1,302.3)	8.77 (1,349.3)

Table 3: Breakdown of ConSDL training-step cost across dictionary expansion factors. Values report latency in milliseconds per minibatch, with the fraction of total training time in parentheses.

Backbone	**Expansion**	**Sparse coding**	**Refitting**	**Similarity reg.**	**Other update**	**Total**
Pythia-160M	4×	7.7 (62%)	1.7 (14%)	2.3 (19%)	0.6 (5%)	12.3
	8×	14.3 (66%)	3.2 (15%)	3.5 (16%)	0.7 (3%)	21.7
	16×	27.6 (64%)	6.1 (14%)	8.0 (19%)	1.2 (3%)	42.8
CLIP ViT-B/16	4×	42.0 (74%)	9.3 (16%)	3.8 (7%)	1.6 (3%)	56.7
	8×	82.6 (75%)	18.2 (17%)	6.6 (6%)	2.3 (2%)	109.7
	16×	166.2 (75%)	36.6 (17%)	13.9 (6%)	4.4 (2%)	221.1

where $\mathcal{V}_\mu = \{(i,j) : \mathbf{w}_j \in A_i,\ s_{i,j} > \mu\}$. The normalization maps similarities between the margin μ and the maximum similarity of 1 to a common range, while the logarithmic transformation increases the penalty as two atoms approach identical directions. When $\mathcal{V}_\mu$ is empty, the similarity loss is set to zero.

Co-active neighbor search. The nearest-neighbor search in Section 3.3 is restricted to the r most similar decoder atoms for computational efficiency. At every training step, we compute the candidate neighbors from the current normalized decoder using cosine similarity and retain the top $r = 32$ neighbors of each atom, excluding the atom itself. For each input and active atom, we keep only candidates that are also active for that input and select the candidate with the highest cosine similarity. If none of the candidate neighbors is co-active, that atom does not contribute to the repulsion loss.

Full algorithm. Fig. 5 provides Python code summarizing the complete training step and how the components described above are combined. The code also includes renormalization after each dictionary update to maintain the unit-norm constraint on the dictionary atoms throughout training.

C Computational Cost Analysis

Measurement setup. To compare the computational cost of ConSDL with existing SAE architectures, we measure per-token inference cost, complete training-step cost, and

a component-wise breakdown of the training step. Measurements are performed on an NVIDIA H100 80GB GPU using PyTorch 2.2.2 with CUDA 12.1, reporting the median of 20 timed iterations after 5 warm-up iterations. We use matched sparsity budgets of $\ell_0 = 32$, 64, and 128 for 4×, 8×, and 16× expansion, respectively. Each step processes 4,096 activation tokens for Pythia-160M and 25,216 activation tokens for CLIP ViT-B/16, corresponding to 128 images with 197 tokens each. All costs are normalized per token, and the frozen backbone forward pass is excluded because it is identical across SAE methods.

Inference and training cost. Table 2 compares the per-token test-time inference and training costs of ConSDL with existing SAE architectures across dictionary expansion factors. Because ConSDL performs sparse coding separately for each sample, it requires more computation than conventional SAEs that obtain sparse coefficients with a single encoder pass. MP-SAE also performs iterative computation during both training and inference. In our measurements, we use 10 matching-pursuit iterations during MP-SAE training, while at test time we continue the iterative selection until the target sparsity is reached. Since each MP-SAE iteration selects a latent based on the current residual, its test-time cost increases with sparsity and exceeds that of ConSDL at higher sparsity levels. ConSDL uses a fixed budget of $T = 30$ FISTA iterations and $T_{\text{refit}} = 5$ refitting iterations across all expansion factors. Although ConSDL has higher training cost than MP-SAE and single-pass SAE variants, this additional computation represents a trade-off for substantially higher cross-seed consistency and better dictionary utilization.

Compute cost breakdown. Table 3 breaks down the training cost of ConSDL into sparse coding, coefficient refitting, similarity regularization, and the remaining update. Sparse coding accounts for most of the computation, averaging approximately 69% of the training step across settings. In contrast, coefficient refitting contributes only about 16% on average, despite its substantial effect on consistency observed in the ablation study. Similarity regularization accounts for a smaller fraction of the overall cost, while the remaining decoding and optimization steps contribute only a few percent. The relative cost of each component also remains similar as the dictionary expansion increases, indicating that the additional computation of ConSDL is dominated by the fixed iterative sparse-coding procedure rather than by refitting or similarity regularization.

D Experimental Details

This section provides experimental details, including baselines (Section D.1) and evaluation metrics (Section D.2) shared across all experiments, followed by details of the synthetic settings (Section D.3) and foundation model experiments (Section D.4).

D.1 Baselines

Standard and sparsity-controlled SAEs. *ReLU SAE* (Cunningham et al., 2023) produces nonnegative coefficients using a ReLU encoder and an ℓ_1 sparsity penalty. *TopK SAE* (Gao et al., 2025) removes the ℓ_1 penalty and retains the k largest encoder coefficients for each input. *BatchTopK SAE* (Bussmann et al., 2024) performs this selection jointly across a minibatch, constraining the average number of active coefficients while allowing sparsity to vary across individual inputs. *Matryoshka SAE* (Bussmann et al., 2025) trains a single ordered dictionary by applying reconstruction objectives to multiple nested prefixes, so that shorter prefixes are also optimized to reconstruct the input rather than relying on the full dictionary (Bussmann et al., 2025). TopK, BatchTopK, and Matryoshka use the AuxK loss (Gao et al., 2025), which provides learning signal to inactive latents by using them to explain the residual reconstruction error. We use an auxiliary-loss coefficient of 1/32 and set $k_{\text{aux}} = d_{\text{in}}/2$.

Iterative sparse encoding. *MP-SAE* (Costa et al., 2026) replaces single-pass coefficient inference with an iterative matching-pursuit procedure that repeatedly selects dictionary atoms based on the current reconstruction residual. We include MP-SAE as an alternative approach to iterative sparse coefficient inference.

Consistency-focused SAE. *iSAE* (Nelson et al., 2026b) modifies TopK using two components designed to improve dictionary and coefficient identifiability. First, it adopts AbsTopK (Zhu et al., 2026), which selects coefficients by magnitude while preserving their signs. This allows a single dictionary atom to contribute with either sign rather than requiring separate atoms for opposite signed contributions. Second, iSAE regularizes the decoder toward an approximate restricted-isometry property (Candes & Tao, 2005; Nelson et al., 2026b), discouraging distinct sparse coefficient vectors from producing nearly identical reconstructions. We use an aRIP regularization weight of 0.1

Related baselines excluded from quantitative comparison. We also considered two closely related methods but exclude them from the quantitative comparisons because we could not obtain stable operating points under our experimental settings. *Archetypal SAE* (Fel et al., 2025) is designed to improve cross-run dictionary consistency by first summarizing the activation distribution with K-means centroids and then learning dictionary atoms from these anchors. *DL-FISTA* (Pacela et al., 2026) removes the amortized encoder and uses alternating optimization. The original study reports a severe loss of dictionary recovery as the latent dimension increases. We attempted to reproduce both methods across our evaluation settings but were unable to obtain sufficiently stable results for reliable comparison, and therefore omit them from the reported quantitative results.

D.2 Evaluation metrics

We evaluate ConSDL and SAE baselines in terms of input reconstruction, dictionary utilization, and cross-seed consistency.

D.2.1 Reconstruction

A useful sparse representation should retain enough information to reconstruct the original activation. We measure reconstruction quality using input R^2,

$$R^2 = 1 - \frac{\sum_{n=1}^{N} \|\mathbf{x}_n - \hat{\mathbf{x}}_n\|_2^2}{\sum_{n=1}^{N} \|\mathbf{x}_n - \bar{\mathbf{x}}\|_2^2},$$

where $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$ denote the original and reconstructed activations, respectively, and $\bar{\mathbf{x}}$ denotes the mean activation.

D.2.2 Dictionary utilization

Dead atoms. These represent dictionary capacity that is effectively unused. For atom j, we define its activation frequency as

$$p_j = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left[a_{nj} \neq 0\right].$$

We classify an atom as dead when $p_j < 10^{-6}$ and report the number of dead atoms.

Duplicated atoms. These represent redundant dictionary capacity, with multiple atoms pointing in nearly identical decoder directions. Prior work measures duplication using pairwise decoder cosine similarity at a single threshold (Mudide et al., 2025), making the result sensitive to the particular threshold and without explicitly identifying groups of mutually similar atoms. We instead exclude dead atoms and apply complete-linkage clustering to the remaining normalized decoder atoms over cosine-similarity thresholds $\tau \in \{0.6, 0.7, 0.8, 0.9\}$. Complete linkage ensures that every pair of atoms within a duplicate cluster satisfies the similarity criterion. At each threshold, we classify all atoms in clusters of size greater than one as duplicated and report the duplicated-atom count averaged across thresholds.

Usable atom fraction. We summarize dictionary utilization as the fraction of atoms that are neither dead nor duplicated. Let N_{dead} denote the number of dead atoms and $\bar{N}_{\text{dup}}$ the duplicated-atom count averaged across the thresholds above. We define

$$\text{Usable Atom Fraction} = 1 - \frac{N_{\text{dead}} + \bar{N}_{\text{dup}}}{m}.$$

Higher values indicate that more of the dictionary capacity is both active and non-redundant.

Table 4: Three cross-seed consistency metrics on CLIP ViT-B/16 ($16\times$, $\ell_0 \approx 128$, 3 seeds), each evaluated under two matchings: the absolute decoder cosine Hungarian used throughout the paper (π_{dec}), and an independent Hungarian computed from activation-magnitude footprints over the 50,000-image ImageNet validation split (π_{fp}). Higher is better and best per column in bold. The ranking is unchanged across the two matchings.

Architecture	Decoder cosine (DCS)		Footprint cosine (50k images)		Code Pearson (MCC)	
	π_{dec}	π_{fp}	π_{dec}	π_{fp}	π_{dec}	π_{fp}
ReLU	0.525	0.496	0.595	0.615	0.417	0.416
TopK	0.558	0.525	0.696	0.715	0.499	0.498
BatchTopK	0.480	0.428	0.524	0.575	0.437	0.458
Matryoshka	0.271	0.189	0.243	0.290	0.184	0.199
MP-SAE	0.694	0.644	0.653	0.672	0.276	0.288
iSAE	0.683	0.658	0.624	0.634	0.588	0.588
Ours	**0.780**	**0.769**	**0.838**	**0.844**	**0.744**	**0.745**

D.2.3 CROSS-SEED CONSISTENCY

Decoder consistency. We measure cross seed decoder consistency using one-to-one Hungarian matching between normalized decoder atoms based on cosine similarity, following prior work (Paulo & Belrose, 2025). We refer to the resulting score as decoder cosine similarity (`DCS`). For each seed pair, `DCS` is the mean cosine similarity of the matched atoms,

$$\text{DCS}_{r,s} = \frac{1}{m} \sum_{j=1}^{m} \cos\left(\mathbf{d}_j^{(r)}, \mathbf{d}_{\pi(j)}^{(s)}\right),$$

where π denotes the optimal Hungarian assignment. We report `DCS` averaged over all seed pairs.

Coefficient consistency. We use the same decoder-based assignment to evaluate whether matched atoms exhibit similar coefficient patterns across inputs. For each matched atom pair, we compute the Pearson correlation coefficient between their coefficients across evaluation samples. We report the mean coefficient correlation (`MCC`), averaged over all matched atom pairs and then over all seed pairs. For computational efficiency, coefficient correlations are computed on a fixed subset of sampled tokens rather than the full evaluation set.

D.2.4 MATCHING BASIS

The cross-seed ranking is consistent under both geometric and activation-based matching. Throughout the paper we align dictionaries across seeds with a Hungarian assignment on absolute decoder cosine similarity (π_{dec}). To verify that our conclusions are not an artifact of this geometric choice, we construct a second, independent assignment from activation footprints: for each latent we record its mean activation magnitude on every image of the ImageNet validation split (50K images, 985K tokens), yielding a footprint in $\mathbb{R}^{50\text{K}}$, and match seeds by footprint cosine similarity (π_{fp}). We then score three consistency metrics (`DCS`, `MCC`, and footprint cosine similarity) on CLIP ViT-B/16 ($16\times$, $\ell_0 \approx 128$, 3 seeds) under both assignments. Table 4 shows that the two assignments induce the identical ranking for every metric, and Ours ranks first in all six columns. Since π_{dec} requires no forward passes over the dataset, we adopt it as the default matching basis in the main text.

D.3 DETAILS OF SYNTHETIC EXPERIMENTAL

This section provides additional details for Section 4.

Support and coefficient distributions. We write each ground-truth coefficient as $a_i = b_i c_i$, where $b_i \in 0, 1$ determines whether feature i is active and $c_i > 0$ its activation magnitude. In the default setting, supports are sampled independently as $b_i \sim \text{Bernoulli}(p_i)$

with $p_i = k/m$. Consequently, the number of active features varies across observations, with expected sparsity $k = 60$. Conditional on activation, coefficient magnitudes are sampled independently from LogNormal($\mu = 0, \sigma = 0.25$).

Long-tailed firing probabilities. To vary atom-frequency imbalance independently of the ground-truth dictionary, we randomly permute the atom indices, assign each atom a rank $r_i \in 1, \ldots, m$, and set $p_i = \min(1, Cr_i^{-\beta})$, where C normalizes the expected number of active atoms to $k = 60$ before clipping. We use the four Zipf exponents specified in the main text. Supports are otherwise sampled independently, so this sweep changes marginal atom selection frequencies without introducing explicit co-occurrence correlations.

Training details. We train each SAE for five epochs over a fixed pool of 10^6 observations per configuration. Each configuration is trained with three random seeds, $\{0, 1, 2\}$. Unless otherwise stated, all models use a batch size of 128, a learning rate of 10^{-4}, and 500 warmup steps followed by a constant learning-rate schedule. For TopK, BatchTopK, Matryoshka, and iSAE, we sweep $k \in \{60, 90, 120\}$ and report the best-performing setting for each synthetic configuration. MP-SAE uses 60 matching-pursuit iterations during both training and inference. For ReLU SAE and ConSDL, the sparsity-controlling coefficient is selected separately for each synthetic configuration to obtain comparable sparsity levels. Exact sparsity parameters and experiment-specific hyperparameters are reported in Table 18.

D.4 Details of Foundation Model Experiments

We provide additional details on model backbones and SAE training data (Section D.4.1), training settings (Section D.4.2), and sparse probing protocols (Section E.3).

D.4.1 Model and Data Details

Vision models. For CLIP, we use `openai/clip-vit-base-patch16` (https://huggingface.co/openai/clip-vit-base-patch16). Its ViT-B/16 vision encoder has approximately 86M parameters, 12 Transformer blocks, a hidden dimension of 768, 12 attention heads, and a patch size of 16×16 for 224×224 inputs. For DINOv2, we use `facebook/dinov2-base` (https://huggingface.co/facebook/dinov2-base), which is a ViT-B/14 model with approximately 86M parameters, 12 Transformer blocks, a hidden dimension of 768, 12 attention heads, and a patch size of 14×14.

Vision dataset. For the vision data, we use `evanarlian/imagenet_1k_resized_256` (https://huggingface.co/datasets/evanarlian/imagenet_1k_resized_256). This release preserves the original ImageNet-1K splits while resizing the shorter side of each image to 256 pixels. Its training split contains 1,281,167 images and validation split contains 500,000 images

Language models. We use the pretrained Pythia-160M checkpoint `EleutherAI/pythia-160m` (https://huggingface.co/EleutherAI/pythia-160m). Pythia-160M contains 162M parameters and 12 Transformer layers, with a hidden dimension of 768 and 12 attention heads. For GPT-2 Small, we use `openai-community/gpt2` (https://huggingface.co/openai-community/gpt2). GPT-2 Small contains 124M parameters and 12 Transformer layers, with a hidden dimension of 768, 12 attention heads, and a maximum context length of 1024.

Language dataset. We use FineWeb (`HuggingFaceFW/fineweb`; https://huggingface.co/datasets/HuggingFaceFW/fineweb), a CommonCrawl-based corpus containing more than 18.5T tokens. For each language backbone, we tokenize and cache a fixed subset of 500M tokens using the corresponding model tokenizer. The same token set is reused across random seeds, with only the sequence order shuffled between runs.

D.4.2 Optimization and hyperparameters

Training details. Unless otherwise stated, vision and language experiments use the same optimization settings. We use a learning rate of 3×10^{-4} with 500 linear warmup steps, followed by a constant learning-rate schedule. We train all models with three random

Table 5: **Per-sample re-optimization improves coefficient recovery under a fixed dictionary.** For each trained decoder, we compare amortized coefficients with per-sample FISTA re-optimization. For dictionaries with DCS ≥ 0.5, re-optimized cells are shaded by the improvement over amortized inference, with darker green indicating larger gains up to $+0.2$. The gains are larger when dictionary recovery is stronger.

Architecture	β	# Atoms	DCS	ℓ_0		MCC		Input R^2	
				Amortized	Re-optimized	Amortized	Re-optimized	Amortized	Re-optimized
ReLU	0.0	3K	0.983	67.4	63.8	0.833	0.975	0.659	0.885
	0.0	6K	0.970	76.5	75.6	0.780	0.962	0.633	0.872
	0.0	9K	0.777	62.5	92.8	0.555	0.672	0.434	0.585
	0.0	12K	0.367	86.9	70.6	0.157	0.168	0.268	0.249
	0.5	3K	0.967	113	69.6	0.832	0.954	0.755	0.869
	0.5	6K	0.899	116	66.0	0.713	0.830	0.659	0.668
	0.5	9K	0.509	76.8	55.9	0.314	0.332	0.336	0.290
	0.5	12K	0.437	103	66.6	0.239	0.256	0.350	0.278
	0.7	3K	0.702	120	69.2	0.564	0.605	0.512	0.476
	0.7	6K	0.369	73.0	107	0.204	0.223	0.176	0.305
	0.7	9K	0.359	110	61.5	0.183	0.197	0.260	0.180
	0.7	12K	0.270	101	68.5	0.102	0.112	0.175	0.154
	1.0	3K	0.334	102	72.2	0.182	0.207	-0.067	-0.061
	1.0	6K	0.163	56.5	99.5	0.024	0.037	-0.381	-0.147
	1.0	9K	0.181	125	53.9	0.034	0.045	-0.297	-0.246
	1.0	12K	0.167	108	62.9	0.020	0.030	-0.281	-0.223
TopK	0.0	3K	0.982	60.0	60.0	0.862	0.947	0.812	0.920
	0.0	6K	0.810	60.0	60.0	0.611	0.669	0.646	0.699
	0.0	9K	0.227	60.0	60.0	0.048	0.048	0.403	0.427
	0.0	12K	0.181	60.0	60.0	0.023	0.023	0.400	0.423
	0.5	3K	0.943	90.0	90.0	0.791	0.910	0.786	0.908
	0.5	6K	0.803	90.0	90.0	0.582	0.655	0.671	0.743
	0.5	9K	0.463	60.0	60.0	0.236	0.246	0.460	0.486
	0.5	12K	0.297	60.0	60.0	0.100	0.101	0.414	0.433
	0.7	3K	0.886	90.0	90.0	0.716	0.816	0.686	0.813
	0.7	6K	0.695	90.0	90.0	0.462	0.512	0.553	0.643
	0.7	9K	0.512	90.0	90.0	0.268	0.285	0.484	0.551
	0.7	12K	0.354	60.0	60.0	0.146	0.149	0.352	0.402
	1.0	3K	0.755	60.0	60.0	0.578	0.649	0.388	0.568
	1.0	6K	0.551	60.0	60.0	0.328	0.356	0.267	0.370
	1.0	9K	0.445	60.0	60.0	0.230	0.238	0.113	0.271
	1.0	12K	0.356	60.0	60.0	0.141	0.144	0.143	0.196

seeds, $\{0, 1, 2\}$, where the seed controls both parameter initialization and data ordering. Vision models use batches of 128 images, whereas language models use 4,096 tokens per optimization step.

Architecture-specific settings. For TopK, BatchTopK, Matryoshka, and iSAE, k is selected according to the target sparsity. MP-SAE uses 10 or 32 matching-pursuit iterations during training. At inference time, matching pursuit is continued until the target L_0 is reached. For ReLU SAE and ConSDL, sparsity is controlled indirectly through the ℓ_1 coefficient and sparse-coding penalty, respectively. We therefore consider operating points within $\pm 15\%$ of the target L_0 and select the setting with the highest DCS $\times$ usable atom fraction within this range. Exact sparsity parameters and experiment-specific hyperparameters are reported in Table 17.

E Full Experimental Results

This section provides full results for the synthetic experiments from Section 4 in Section E.1, and for the foundation model experiments from Section 5 in Section E.2.

E.1 Synthetic Results

Frozen dictionary coefficient re-optimization. To separate coefficient-inference error from dictionary-recovery error, we re-optimize coefficients while keeping each trained decoder fixed (Table 5). For each sample, we replace the amortized encoder output with coefficients obtained by running 50 FISTA iterations against the same learned dictionary,

Table 6: **Cross-seed consistency and dictionary utilization across foundation-model backbones.** For each backbone and method, we report the mean and standard deviation across nine combinations of dictionary expansion (4×, 8×, 16×) and sparsity ($\ell_0 \approx$ 32, 64, 128). The variation therefore reflects differences across these settings rather than training seeds. The last two columns report the relative improvement of ConSDL over the average of the six SAE baselines and the strongest baseline for each backbone and metric. The final three rows report averages over the four backbones with equal weighting.

Backbone	Metric	ReLU	TopK	BatchTopK	Matryoshka	MP-SAE	iSAE	ConSDL	vs. Avg.	vs. Best
CLIP ViT-B/16	DCS	0.61 ±0.15	0.67 ±0.10	0.56 ±0.10	0.37 ±0.08	0.75 ±0.04	0.75 ±0.06	**0.78 ±0.07**	+26.7%	+4.8%
	MCC	0.53 ±0.17	0.63 ±0.13	0.52 ±0.11	0.31 ±0.10	0.45 ±0.15	0.40 ±0.06	**0.73 ±0.09**	+54.7%	+16.8%
	Usable atom frac.	0.99 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.85 ±0.05	0.96 ±0.10	**1.00 ±0.00**	+3.5%	+0.0%
DINOv2-B/14	DCS	0.68 ±0.16	0.73 ±0.13	0.65 ±0.15	0.50 ±0.12	0.83 ±0.04	0.77 ±0.10	**0.84 ±0.05**	+21.4%	+1.6%
	MCC	0.61 ±0.19	0.69 ±0.17	0.61 ±0.19	0.47 ±0.16	0.49 ±0.16	0.44 ±0.07	**0.78 ±0.06**	+42.3%	+14.1%
	Usable atom frac.	0.98 ±0.01	0.98 ±0.04	**1.00 ±0.00**	0.99 ±0.01	0.75 ±0.11	0.99 ±0.01	0.99 ±0.02	+4.5%	-0.6%
Pythia-160M	DCS	0.85 ±0.04	0.74 ±0.07	0.75 ±0.06	0.62 ±0.05	**0.85 ±0.01**	0.78 ±0.05	0.85 ±0.02	+10.8%	-0.8%
	MCC	0.76 ±0.05	0.69 ±0.06	0.70 ±0.05	0.61 ±0.04	0.46 ±0.04	0.43 ±0.04	**0.77 ±0.03**	+26.0%	+1.1%
	Usable atom frac.	0.93 ±0.04	1.00 ±0.00	1.00 ±0.00	**1.00 ±0.00**	0.65 ±0.05	0.97 ±0.08	1.00 ±0.00	+8.1%	-0.1%
GPT-2 Small	DCS	0.81 ±0.14	0.77 ±0.05	0.77 ±0.05	0.65 ±0.04	0.86 ±0.01	0.78 ±0.05	**0.88 ±0.02**	+13.7%	+2.3%
	MCC	0.75 ±0.16	0.74 ±0.03	0.75 ±0.02	0.66 ±0.02	0.56 ±0.07	0.46 ±0.03	**0.82 ±0.03**	+25.8%	+9.6%
	Usable atom frac.	0.87 ±0.18	0.99 ±0.01	0.99 ±0.01	**1.00 ±0.00**	0.79 ±0.03	0.99 ±0.04	1.00 ±0.00	+6.5%	-0.1%
All backbones	**DCS**	0.74	0.73	0.68	0.53	0.82	0.77	**0.84**	**+17.6%**	**+1.9%**
	MCC	0.66	0.69	0.65	0.51	0.49	0.43	**0.78**	**+35.9%**	**+13.3%**
	Usable atom frac.	0.94	0.99	1.00	1.00	0.76	0.98	**1.00**	**+5.6%**	**+0.0%**

without updating the decoder. We then recompute coefficient recovery (MCC) and input reconstruction R^2. When the learned dictionary remains accurate, per-sample re-optimization substantially improves MCC and generally improves reconstruction, showing that the amortized encoder can leave recoverable coefficient information unused. As dictionary recovery degrades with larger dictionaries and more long-tailed atom selection frequencies, these gains become much smaller. Better coefficient inference alone therefore cannot compensate for an inaccurate dictionary, motivating improvements to the dictionary learning process itself.

Full results of synthetic experiments. As a supplement to Fig. 2, we report the full synthetic results across all baselines, including input reconstruction, dictionary utilization (# Dead and # Dup.), ground-truth recovery, and cross-seed consistency, in Tables 9, 10, 11, and 12. Existing baselines generally show degraded ground-truth recovery and cross-seed consistency as the settings become more challenging, often accompanied by poorer dictionary utilization. In contrast, our method remains substantially more robust across the sweep, including the largest dictionaries and most strongly long-tailed activation distributions.

E.2 Results on Foundation Model Activations

Full results on foundation-model activations. We report complete results for three dictionary expansion factors (4×, 8×, and 16×) and three matched sparsity levels ($\ell_0 \approx$ 32, 64, and 128). For each setting, we report input reconstruction quality using R^2, cross-seed dictionary consistency using decoder cosine similarity (DCS), coefficient consistency using mean coefficient correlation (MCC), and dictionary utilization using the usable atom fraction. Results are provided in Table 13 for CLIP ViT-B/16, Table 14 for DINOv2 ViT-B/14, Table 15 for GPT-2 Small, and Table 16 for Pythia-160M.

E.3 Sparse Probing Evaluation

We provide the full sparse probing protocol and results for vision and language models. Following prior work (Gurnee et al., 2023; Karvonen et al., 2025), we identify dictionary atoms that best distinguish positive from negative examples and evaluate probes restricted to the Top-1 or Top-5 selected atoms.

Atom selection. We follow the class-conditional activation criterion used in sparse probing (Gurnee et al., 2023). We first normalize each atom by its mean nonzero activation magnitude and then rank atoms by the absolute difference between their mean normalized activations on positive and negative training examples. Atom selection is performed using only the training split.

Table 7: Sparse probing of vision dictionaries. AUPRC from the k most discriminative dictionary atoms per class, $k \in \{1, 5\}$, at each expansion and its matched sparsity ($4\times$ at $\ell_0 \approx 32$, $8\times$ at $\ell_0 \approx 64$, $16\times$ at $\ell_0 \approx 128$). Avg. is the unweighted mean over the six evaluation datasets. $\pm$ is the standard deviation across training seeds of the reported score.

Backbone	Cell	Method	ImageNet *Object*		SUN397 *Scene*		Food101 *Food*		DTD *Texture*		Caltech256 *Object*		RAF-DB *Emotion*		Avg.	
			@1	@5	@1	@5	@1	@5	@1	@5	@1	@5	@1	@5	@1	@5
CLIP ViT-B/16	$4\times$ $\ell_0 \approx 32$	ReLU	**0.33** ±0.00	0.39 ±0.00	0.35 ±0.00	0.43 ±0.00	0.42 ±0.01	0.46 ±0.00	0.51 ±0.01	0.61 ±0.01	0.70 ±0.00	0.77 ±0.00	0.45 ±0.04	0.51 ±0.01	0.46 ±0.01	0.53 ±0.00
		TopK	0.28 ±0.00	0.36 ±0.00	0.34 ±0.00	**0.45** ±0.00	0.42 ±0.00	**0.50** ±0.00	0.49 ±0.01	**0.61** ±0.00	0.67 ±0.00	0.76 ±0.00	**0.48** ±0.00	**0.52** ±0.01	0.45 ±0.00	**0.54** ±0.00
		BatchTopK	0.28 ±0.00	0.36 ±0.00	0.34 ±0.00	0.45 ±0.00	0.42 ±0.01	0.50 ±0.00	0.49 ±0.01	0.61 ±0.00	0.67 ±0.00	0.76 ±0.00	0.46 ±0.04	0.51 ±0.01	0.44 ±0.01	0.53 ±0.00
		Matryoshka	0.26 ±0.00	0.33 ±0.00	0.31 ±0.00	0.42 ±0.00	0.39 ±0.01	0.46 ±0.01	0.48 ±0.01	0.59 ±0.00	0.61 ±0.01	0.71 ±0.01	0.42 ±0.04	0.49 ±0.00	0.41 ±0.01	0.50 ±0.00
		MP-SAE	0.28 ±0.00	0.36 ±0.00	0.34 ±0.00	0.45 ±0.00	0.42 ±0.00	0.50 ±0.01	0.51 ±0.01	0.60 ±0.01	0.67 ±0.00	0.76 ±0.00	0.42 ±0.03	0.50 ±0.02	0.44 ±0.00	0.53 ±0.00
		iSAE	0.15 ±0.00	0.29 ±0.00	0.17 ±0.01	0.37 ±0.00	0.25 ±0.04	0.41 ±0.01	0.25 ±0.02	0.58 ±0.00	0.36 ±0.02	0.68 ±0.00	0.24 ±0.05	0.42 ±0.01	0.24 ±0.02	0.46 ±0.00
		ConSDL	0.32 ±0.00	0.39 ±0.00	**0.36** ±0.00	0.44 ±0.00	**0.43** ±0.00	0.46 ±0.00	**0.52** ±0.01	0.61 ±0.00	**0.72** ±0.00	**0.77** ±0.00	0.48 ±0.01	0.51 ±0.02	**0.47** ±0.00	0.53 ±0.00
	$8\times$ $\ell_0 \approx 64$	ReLU	**0.36** ±0.00	**0.40** ±0.00	0.37 ±0.00	0.45 ±0.00	**0.46** ±0.01	0.50 ±0.00	0.51 ±0.00	0.61 ±0.00	0.72 ±0.00	0.79 ±0.00	0.41 ±0.04	0.53 ±0.00	0.47 ±0.01	0.55 ±0.00
		TopK	0.29 ±0.00	0.36 ±0.00	0.35 ±0.00	0.45 ±0.00	0.44 ±0.00	0.52 ±0.00	0.48 ±0.01	0.61 ±0.00	0.68 ±0.00	0.76 ±0.00	0.47 ±0.02	0.51 ±0.01	0.45 ±0.00	0.54 ±0.00
		BatchTopK	0.29 ±0.00	0.36 ±0.00	0.35 ±0.00	0.44 ±0.00	0.43 ±0.00	0.51 ±0.00	0.50 ±0.00	0.60 ±0.01	0.68 ±0.00	0.76 ±0.00	**0.48** ±0.01	0.50 ±0.00	0.45 ±0.00	0.53 ±0.00
		Matryoshka	0.26 ±0.00	0.32 ±0.00	0.33 ±0.00	0.41 ±0.00	0.42 ±0.00	0.48 ±0.00	0.49 ±0.00	0.59 ±0.00	0.62 ±0.00	0.71 ±0.00	0.43 ±0.01	0.47 ±0.01	0.42 ±0.00	0.50 ±0.00
		MP-SAE	0.34 ±0.00	0.39 ±0.00	0.37 ±0.00	**0.46** ±0.00	0.44 ±0.00	**0.54** ±0.00	0.52 ±0.00	**0.61** ±0.00	0.72 ±0.00	**0.79** ±0.00	0.40 ±0.01	0.51 ±0.00	0.46 ±0.00	**0.55** ±0.00
		iSAE	0.17 ±0.00	0.29 ±0.00	0.18 ±0.01	0.38 ±0.00	0.21 ±0.03	0.45 ±0.00	0.25 ±0.02	0.58 ±0.00	0.33 ±0.02	0.69 ±0.00	0.27 ±0.05	0.47 ±0.01	0.23 ±0.01	0.48 ±0.00
		ConSDL	0.36 ±0.00	0.40 ±0.00	**0.38** ±0.00	0.44 ±0.00	0.46 ±0.01	0.47 ±0.00	**0.54** ±0.00	0.61 ±0.00	**0.74** ±0.00	0.78 ±0.00	0.48 ±0.01	**0.54** ±0.02	**0.49** ±0.00	0.54 ±0.00
	$16\times$ $\ell_0 \approx 128$	ReLU	0.37 ±0.00	0.41 ±0.00	0.38 ±0.00	0.45 ±0.00	0.49 ±0.01	0.50 ±0.00	0.50 ±0.01	0.61 ±0.00	0.72 ±0.01	0.79 ±0.00	0.41 ±0.03	0.52 ±0.01	0.48 ±0.01	0.55 ±0.00
		TopK	0.31 ±0.00	0.34 ±0.00	0.36 ±0.00	0.44 ±0.00	0.46 ±0.01	0.51 ±0.00	0.48 ±0.01	0.59 ±0.00	0.69 ±0.00	0.74 ±0.00	0.45 ±0.01	0.52 ±0.01	0.46 ±0.01	0.53 ±0.00
		BatchTopK	0.28 ±0.00	0.34 ±0.00	0.35 ±0.00	0.44 ±0.00	0.44 ±0.00	0.50 ±0.00	0.48 ±0.00	0.59 ±0.00	0.68 ±0.00	0.74 ±0.00	0.44 ±0.03	0.53 ±0.01	0.45 ±0.00	0.52 ±0.00
		Matryoshka	0.26 ±0.00	0.29 ±0.00	0.33 ±0.00	0.37 ±0.00	0.40 ±0.01	0.44 ±0.01	0.48 ±0.01	0.57 ±0.01	0.62 ±0.01	0.66 ±0.01	0.43 ±0.03	0.46 ±0.00	0.42 ±0.01	0.46 ±0.00
		MP-SAE	**0.42** ±0.00	**0.45** ±0.00	0.40 ±0.01	**0.49** ±0.00	0.47 ±0.01	**0.56** ±0.01	0.48 ±0.01	**0.62** ±0.01	0.77 ±0.01	**0.82** ±0.01	0.39 ±0.05	0.48 ±0.02	0.49 ±0.01	**0.57** ±0.00
		iSAE	0.19 ±0.00	0.30 ±0.00	0.19 ±0.01	0.39 ±0.00	0.19 ±0.02	0.46 ±0.00	0.25 ±0.02	0.58 ±0.01	0.36 ±0.01	0.70 ±0.00	0.27 ±0.06	**0.55** ±0.02	0.24 ±0.01	0.50 ±0.00
		ConSDL	0.38 ±0.00	0.40 ±0.00	**0.41** ±0.00	0.44 ±0.00	0.49 ±0.01	0.48 ±0.01	**0.54** ±0.01	0.61 ±0.01	0.77 ±0.00	0.79 ±0.00	**0.51** ±0.03	0.55 ±0.01	**0.52** ±0.01	0.55 ±0.00
DINOv2-B/14	$4\times$ $\ell_0 \approx 32$	ReLU	**0.31** ±0.00	0.39 ±0.00	0.31 ±0.00	0.42 ±0.00	0.36 ±0.01	0.46 ±0.01	0.51 ±0.01	0.67 ±0.01	**0.68** ±0.00	0.78 ±0.00	0.37 ±0.01	0.41 ±0.00	0.42 ±0.00	0.52 ±0.00
		TopK	0.29 ±0.00	**0.39** ±0.00	0.31 ±0.00	**0.45** ±0.00	0.36 ±0.00	**0.50** ±0.00	**0.53** ±0.00	**0.69** ±0.00	0.67 ±0.00	0.79 ±0.00	**0.40** ±0.00	0.42 ±0.01	**0.43** ±0.00	**0.54** ±0.00
		BatchTopK	0.29 ±0.00	0.39 ±0.00	**0.31** ±0.00	0.45 ±0.00	**0.36** ±0.00	0.50 ±0.00	0.51 ±0.01	0.68 ±0.00	0.68 ±0.00	0.79 ±0.00	0.37 ±0.00	0.44 ±0.01	0.42 ±0.00	0.54 ±0.00
		Matryoshka	0.27 ±0.00	0.37 ±0.00	0.29 ±0.00	0.42 ±0.00	0.35 ±0.00	0.47 ±0.00	0.53 ±0.01	0.66 ±0.00	0.63 ±0.01	0.76 ±0.00	0.37 ±0.02	**0.44** ±0.01	0.41 ±0.00	0.52 ±0.00
		MP-SAE	0.28 ±0.00	0.38 ±0.00	0.29 ±0.00	0.43 ±0.00	0.35 ±0.01	0.49 ±0.00	0.52 ±0.01	0.67 ±0.00	0.65 ±0.00	0.77 ±0.00	0.38 ±0.02	0.41 ±0.01	0.41 ±0.00	0.53 ±0.00
		iSAE	0.13 ±0.00	0.31 ±0.00	0.16 ±0.02	0.35 ±0.00	0.20 ±0.02	0.38 ±0.00	0.29 ±0.03	0.64 ±0.00	0.34 ±0.02	0.70 ±0.01	0.24 ±0.07	0.40 ±0.01	0.23 ±0.01	0.46 ±0.00
		ConSDL	0.29 ±0.00	0.38 ±0.00	0.30 ±0.00	0.42 ±0.00	0.35 ±0.01	0.48 ±0.00	0.48 ±0.01	0.67 ±0.01	0.66 ±0.01	0.78 ±0.00	0.37 ±0.01	0.41 ±0.01	0.41 ±0.00	0.52 ±0.00
	$8\times$ $\ell_0 \approx 64$	ReLU	**0.33** ±0.00	**0.41** ±0.00	**0.33** ±0.00	0.44 ±0.00	**0.38** ±0.01	0.48 ±0.00	0.53 ±0.00	0.68 ±0.00	**0.71** ±0.00	**0.80** ±0.00	0.37 ±0.01	0.40 ±0.01	**0.44** ±0.00	0.54 ±0.00
		TopK	0.30 ±0.00	0.40 ±0.00	0.32 ±0.00	**0.46** ±0.00	0.37 ±0.00	0.49 ±0.00	0.53 ±0.01	0.69 ±0.00	0.70 ±0.00	0.79 ±0.00	**0.40** ±0.01	**0.42** ±0.01	0.44 ±0.00	**0.54** ±0.00
		BatchTopK	0.30 ±0.00	0.40 ±0.00	0.32 ±0.00	0.46 ±0.00	0.37 ±0.00	0.50 ±0.00	0.52 ±0.01	0.69 ±0.00	0.69 ±0.00	0.79 ±0.00	0.39 ±0.00	0.42 ±0.01	0.43 ±0.00	0.54 ±0.00
		Matryoshka	0.28 ±0.00	0.37 ±0.00	0.30 ±0.00	0.43 ±0.00	0.36 ±0.00	0.48 ±0.00	0.53 ±0.00	0.67 ±0.01	0.66 ±0.01	0.77 ±0.00	0.36 ±0.01	0.41 ±0.01	0.42 ±0.00	0.52 ±0.00
		MP-SAE	0.30 ±0.00	0.39 ±0.00	0.32 ±0.00	0.45 ±0.00	0.37 ±0.00	**0.51** ±0.00	0.52 ±0.01	0.68 ±0.00	0.69 ±0.00	0.79 ±0.00	0.37 ±0.00	0.39 ±0.01	0.43 ±0.00	0.54 ±0.00
		iSAE	0.14 ±0.00	0.32 ±0.00	0.15 ±0.01	0.37 ±0.00	0.20 ±0.00	0.39 ±0.00	0.28 ±0.05	0.63 ±0.00	0.31 ±0.02	0.72 ±0.00	0.20 ±0.10	0.38 ±0.00	0.21 ±0.02	0.47 ±0.00
		ConSDL	0.31 ±0.00	0.40 ±0.00	0.32 ±0.00	0.43 ±0.00	0.37 ±0.00	0.46 ±0.00	0.50 ±0.02	0.68 ±0.01	0.68 ±0.01	0.79 ±0.00	0.36 ±0.01	0.40 ±0.01	0.42 ±0.00	0.53 ±0.00
	$16\times$ $\ell_0 \approx 128$	ReLU	**0.35** ±0.00	0.41 ±0.00	**0.34** ±0.00	0.44 ±0.00	0.38 ±0.01	0.47 ±0.00	0.52 ±0.01	0.67 ±0.01	**0.72** ±0.01	0.79 ±0.00	0.37 ±0.01	0.40 ±0.00	**0.45** ±0.00	0.53 ±0.00
		TopK	0.31 ±0.00	0.39 ±0.00	0.32 ±0.00	0.44 ±0.00	0.37 ±0.00	0.48 ±0.00	0.52 ±0.00	0.69 ±0.00	0.69 ±0.00	0.78 ±0.00	0.38 ±0.02	0.40 ±0.00	0.43 ±0.00	0.53 ±0.00
		BatchTopK	0.30 ±0.00	0.39 ±0.00	0.32 ±0.00	0.44 ±0.00	0.36 ±0.00	0.48 ±0.00	0.52 ±0.00	**0.69** ±0.01	0.69 ±0.00	0.78 ±0.00	0.38 ±0.01	0.40 ±0.00	0.43 ±0.00	0.53 ±0.00
		Matryoshka	0.28 ±0.00	0.37 ±0.00	0.31 ±0.00	0.42 ±0.00	0.37 ±0.01	0.47 ±0.00	0.51 ±0.01	0.66 ±0.01	0.66 ±0.01	0.76 ±0.00	0.37 ±0.00	0.40 ±0.00	0.42 ±0.00	0.51 ±0.00
		MP-SAE	0.34 ±0.00	**0.42** ±0.00	0.33 ±0.00	**0.45** ±0.00	0.37 ±0.00	**0.50** ±0.01	0.52 ±0.00	0.68 ±0.01	0.72 ±0.00	**0.81** ±0.00	0.34 ±0.01	0.38 ±0.00	0.44 ±0.00	**0.54** ±0.00
		iSAE	0.16 ±0.00	0.33 ±0.00	0.17 ±0.00	0.36 ±0.00	0.21 ±0.01	0.39 ±0.00	0.30 ±0.06	0.63 ±0.01	0.37 ±0.02	0.73 ±0.00	0.25 ±0.09	0.38 ±0.01	0.24 ±0.02	0.47 ±0.00
		ConSDL	0.33 ±0.00	0.38 ±0.00	0.33 ±0.00	0.39 ±0.00	**0.39** ±0.00	0.42 ±0.00	**0.53** ±0.01	0.66 ±0.02	0.71 ±0.01	0.77 ±0.00	0.35 ±0.00	0.40 ±0.02	0.44 ±0.00	0.50 ±0.01

Table 8: **Sparse probing performance of language-model dictionaries.** We report AUROC using the k most discriminative dictionary atoms per probe for $k \in \{1, 5\}$ across three matched expansion and sparsity settings: $4\times$ with $\ell_0 \approx 32$, $8\times$ with $\ell_0 \approx 64$, and $16\times$ with $\ell_0 \approx 128$. Scores are averaged over the 113 binary probes in SAE-Probes. $\pm$ is the standard deviation across training seeds of the reported score.

Backbone	Method	**4**×, $\ell_0 \approx 32$		**8**×, $\ell_0 \approx 64$		**16**×, $\ell_0 \approx 128$	
		@1	@5	@1	@5	@1	@5
Pythia-160M	ReLU	0.69 ±0.00	0.75 ±0.00	0.70 ±0.00	0.77 ±0.00	0.70 ±0.00	0.78 ±0.00
	TopK	0.70 ±0.00	0.77 ±0.00	0.71 ±0.00	0.78 ±0.00	0.72 ±0.00	0.80 ±0.00
	BatchTopK	0.70 ±0.00	0.77 ±0.00	**0.72** ±0.01	0.79 ±0.00	0.73 ±0.00	**0.80** ±0.00
	Matryoshka	**0.71** ±0.00	0.77 ±0.00	0.72 ±0.01	0.79 ±0.00	0.72 ±0.00	0.79 ±0.00
	MP-SAE	0.70 ±0.00	**0.78** ±0.00	0.68 ±0.00	0.76 ±0.00	0.67 ±0.00	0.74 ±0.00
	iSAE	0.70 ±0.00	0.77 ±0.00	0.71 ±0.00	0.78 ±0.00	**0.73** ±0.00	0.79 ±0.00
	ConSDL	0.70 ±0.00	0.76 ±0.00	0.71 ±0.00	0.77 ±0.00	0.73 ±0.00	0.78 ±0.00
GPT-2 Small	ReLU	0.71 ±0.00	0.77 ±0.00	0.73 ±0.00	0.79 ±0.00	0.74 ±0.00	0.79 ±0.00
	TopK	**0.72** ±0.00	**0.78** ±0.00	**0.73** ±0.00	0.79 ±0.00	0.75 ±0.00	0.80 ±0.00
	BatchTopK	0.72 ±0.00	0.78 ±0.00	0.73 ±0.00	0.79 ±0.00	0.75 ±0.00	0.80 ±0.00
	Matryoshka	0.71 ±0.00	0.78 ±0.00	0.73 ±0.00	0.79 ±0.00	0.74 ±0.00	0.79 ±0.00
	MP-SAE	0.70 ±0.00	0.77 ±0.00	0.69 ±0.00	0.76 ±0.00	0.67 ±0.00	0.75 ±0.00
	iSAE	0.72 ±0.00	0.77 ±0.00	0.73 ±0.00	0.79 ±0.00	0.74 ±0.00	0.80 ±0.00
	ConSDL	0.71 ±0.00	0.77 ±0.00	0.73 ±0.00	0.78 ±0.00	0.74 ±0.00	0.79 ±0.00

Probe construction. We evaluate the Top-1 and Top-5 selected atoms. For Top-1, we use the absolute activation of the selected atom directly as the prediction score, without fitting an additional classifier. For Top-5, we fit a logistic regression classifier using the raw activations of the five selected atoms on the training split and evaluate its decision scores on the evaluation split.

Language benchmarks. For language models, we use the SAE-Probes benchmark (Kantamneni et al., 2025), which consists of 113 binary classification tasks spanning a diverse range of linguistic and semantic concepts. For each task, examples are labeled as positive or negative according to the target concept, and the learned SAE activations are used as probe inputs. We follow the benchmark protocol for constructing the training and evaluation sets and report the area under the receiver operating characteristic curve (AUROC).

Vision benchmarks. For vision models, following the evaluation strategy of ConceptScope (Choi et al., 2026), we evaluate sparse probing across datasets covering diverse types of visual concepts. We use ImageNet (Russakovsky et al., 2015) with 1,000 object classes, SUN397 (Xiao et al., 2010) with 397 scene categories, Food-101 (Bossard et al., 2014)with 101 food categories, Caltech-256 (Griffin et al., 2007) with 256 object categories, DTD (Cimpoi et al., 2014) with 47 texture categories, and RAF-DB (Li et al., 2017) with 7 facial emotion categories. Each category is treated as a one-vs-rest binary prediction task. We use 50 training images per category for atom selection, and evaluate on the full validation or test split. All available negative examples are used for evaluation. We report the area under the precision–recall curve (AUPRC) because the resulting one-vs-rest tasks are highly imbalanced.

E.4 Analysis: Region carve and cross-seed latent matching

This section examines whether independently trained SAEs decompose the same input in the same way, and whether individual latents recover the same selectivity across seeds.

Region-level decomposition. Figure 6A and Fig. 7A test whether different seeds decompose the same input similarly. We cluster patches or tokens based on their coefficient vectors. At each position, coefficients below 1% of the maximum are discarded and the remainder renormalized. We compare positions using half the L_1 distance between these vectors, ranging from 0 for identical coefficients to 1 for no overlap. Average-linkage hierarchical clustering is then applied separately for each seed to form $k = 10$ regions. Regions are colored by their mean position in the image or text, and agreement across seeds is measured by the adjusted Rand index (ARI).

Organization of matched latents across seeds. Figures 6B–C and 7B–C examine whether latents matched across seeds occupy similar regions of their learned dictionaries and exhibit similar activation behavior. We match latents by decoder cosine similarity and visualize the three dictionaries side by side in a shared UMAP space, fitted on one seed and used to project the other two. We also compare the regions where matched latents activate on the same input and their top-activating samples.

Vision. We analyze CLIP ViT-B/16 at 16× expansion and $\ell_0 \approx 128$ using an ImageNet validation image. In Fig. 6A, the three seeds produce nearly identical region decompositions, with pairwise ARI values of $\{0.98, 0.98, 0.97\}$, consistently separating the handler, fence and background, and the animals. Panels B and C examine two matched latents active on a patch on the animal's head. The first responds to animals in close physical contact, such as nuzzling, grooming, and huddling, with activation concentrated around the contact region rather than a particular species. Its top-activating images show the same relation across unrelated classes. The second captures a heart or romance-related pattern and is similarly reproduced across seeds. These examples show that a single patch can be decomposed into distinct latents that recur across independent runs.

Language. We analyze Pythia-160M at 16× expansion and $\ell_0 \approx 128$ using an example from the 20M-token FineWeb evaluation set. In Fig. 7A, the three seeds produce broadly similar region decompositions, with pairwise ARI values of $\{0.83, 0.65, 0.74\}$. We exclude position 0 because its unusually large activation would otherwise form a separate region. Panels B and C examine two matched latents. The first responds to animal nouns such as *tiger*, *lion*, *pig*, *rat*, and *bear* across unrelated contexts and has high decoder similarity across seeds ($\cos = 0.97$ to both). The second responds to sentence-final punctuation and the preceding token. Despite lower decoder similarity ($\cos = 0.65$ and 0.79), its top-activating contexts show the same pattern across all three seeds. This suggests that similar activation behavior can persist even when decoder similarity is moderate.

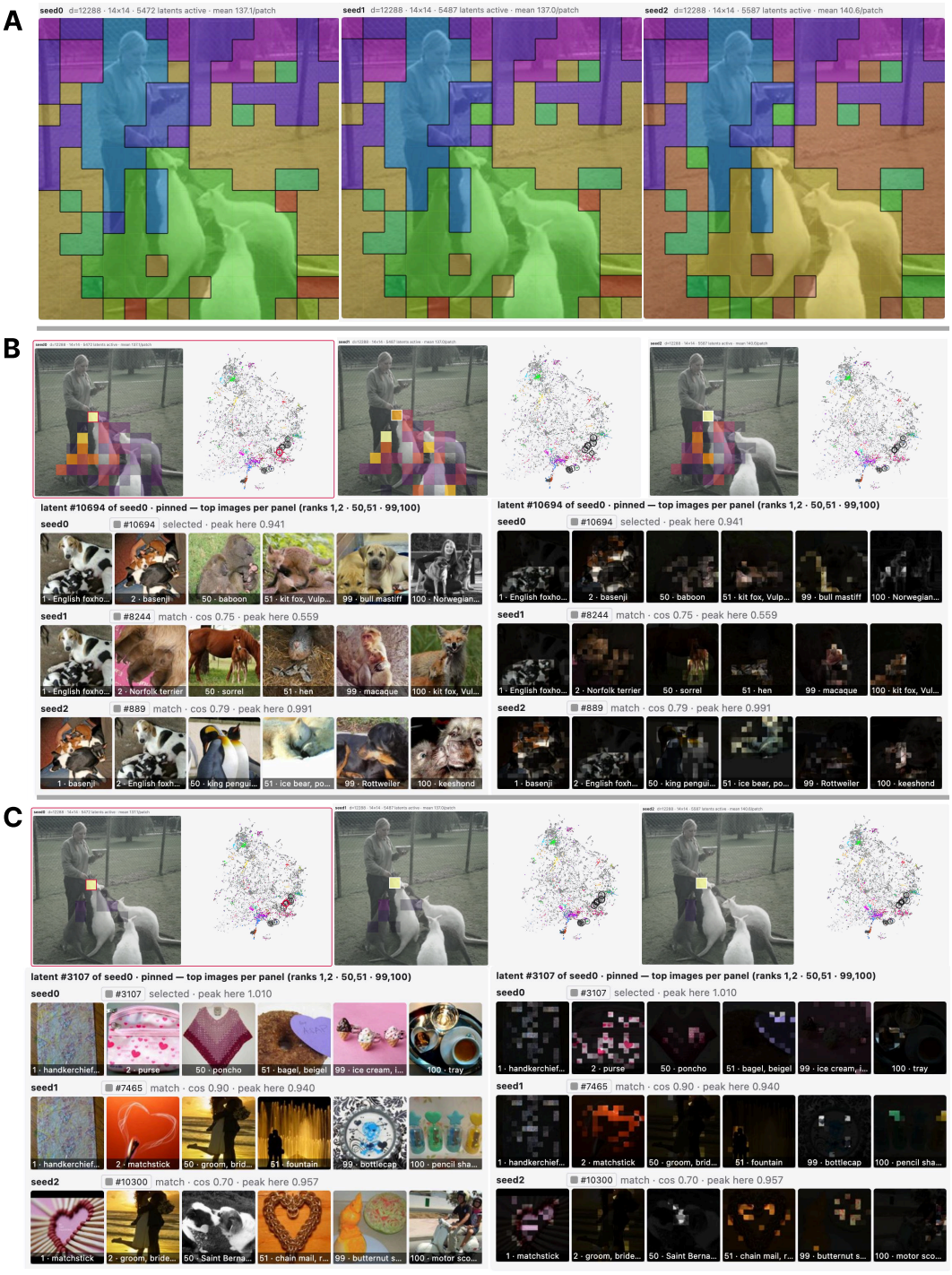

Figure 6: **Region decomposition and cross-seed latent analysis in CLIP ViT-B/16.** 16× dictionary ($m = 12288$), $\ell_0 \approx 128$, three seeds. **(A)** Each 14×14 patch grid is independently clustered into $k = 10$ regions from its coefficient vectors. Hue encodes region position, and headers report active latents and their mean per patch. **(B)** One seed-0 patch is selected (red outline). Top: activation map and shared UMAP, fitted on seed 0 with seeds 1–2 projected into the same space. Black rings mark latents active at the patch, red the selected latent, and gray its nearest matches. Latent #10694 matches #8244 ($\cos = 0.75$) and #889 ($\cos = 0.79$). Bottom: top-activating images at ranks $1, 2, 50, 51$, and $99, 100$, shown unmasked and activation-masked, reveal animals in close contact. **(C)** Same analysis for a second latent at the patch. #3107 matches #7465 ($\cos = 0.90$) and #10300 ($\cos = 0.70$), showing a heart/romance motif.



Figure 7: **Region decomposition and cross-seed latent analysis in Pythia-160M.** 16× dictionary ($m = 12288$), three seeds, using a 96-token window with approximately 112 active latents per token. **(A)** Tokens are independently clustered into $k = 10$ regions for each seed. Hue encodes position in the text rather than content. **(B)** One seed-0 token is selected (red outline). For each seed, we show the passage with latent activations highlighted, a shared UMAP fitted on seed 0 with seeds 1–2 projected into the same space, and top-activating contexts at ranks $1, 2,\ 50, 51$, and $99, 100$. Black rings mark latents active at the token, red the selected latent, and gray its nearest matches. Latent #12079 matches #12005 (cos = 0.97) and #7854 (cos = 0.97), capturing animal nouns. **(C)** Same analysis for a second latent. #5107 matches #6110 (cos = 0.65) and #10394 (cos = 0.79), capturing sentence-final punctuation and the preceding token.

Table 9: **Full synthetic results with a 3K-atom ground-truth dictionary across tail exponents β.** We evaluate all methods across $\beta \in \{0, 0.5, 0.7, 1.0\}$. We report sparsity ℓ_0, reconstruction (R^2), ground-truth recovery (DCS, MCC), cross-seed consistency, and dictionary utilization measured by the usable atom fraction. The numbers of dead and duplicated atoms are also reported. Metrics are averaged across training seeds, while cross-seed DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Setting	Architecture	Sparsity	Reconstruction	Ground-truth Recovery		Cross-seed Consistency		Dictionary Utilization		
		ℓ_0	R^2	DCS	MCC	DCS	MCC	# Dup	# Dead	Usable atom frac.
$\beta = 0.0$	ReLU	67.0 ±0.1	0.67 ±0.00	0.98 ±0.00	0.83 ±0.00	0.97 ±0.00	0.97 ±0.00	10 ±2	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.82 ±0.00	0.98 ±0.00	0.86 ±0.00	0.97 ±0.00	0.93 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	BatchTopK	61.0 ±0.0	0.81 ±0.00	0.98 ±0.00	0.86 ±0.00	0.96 ±0.00	0.93 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	Matryoshka	64.0 ±0.0	0.69 ±0.00	0.82 ±0.00	0.67 ±0.00	0.69 ±0.00	0.58 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	MP-SAE	57.7 ±0.0	0.95 ±0.00	0.99 ±0.00	0.89 ±0.00	0.99 ±0.00	0.95 ±0.00	20 ±4	0 ±0	1.00 ±0.00
	iSAE	60.0 ±0.0	0.72 ±0.00	0.86 ±0.00	0.36 ±0.01	0.75 ±0.00	0.32 ±0.01	0 ±0	440 ±16	0.86 ±0.01
	Ours	61.0 ±0.0	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.5$	ReLU	113 ±1	0.77 ±0.00	0.97 ±0.00	0.83 ±0.00	0.95 ±0.00	0.94 ±0.00	23 ±9	0 ±0	1.00 ±0.00
	TopK	90.0 ±0.0	0.79 ±0.00	0.94 ±0.00	0.79 ±0.00	0.91 ±0.00	0.86 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	113 ±1	0.79 ±0.00	0.93 ±0.00	0.77 ±0.00	0.89 ±0.00	0.84 ±0.00	0 ±1	0 ±0	1.00 ±0.00
	Matryoshka	66.0 ±0.3	0.63 ±0.00	0.78 ±0.00	0.59 ±0.00	0.68 ±0.00	0.56 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	MP-SAE	59.0 ±0.0	0.90 ±0.00	0.93 ±0.00	0.88 ±0.00	0.91 ±0.00	0.84 ±0.00	294 ±20	0 ±0	0.95 ±0.00
	iSAE	60.0 ±0.0	0.62 ±0.00	0.74 ±0.00	0.27 ±0.00	0.64 ±0.00	0.25 ±0.00	0 ±0	737 ±11	0.76 ±0.00
	Ours	62.0 ±0.0	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.7$	ReLU	120 ±2	0.57 ±0.01	0.70 ±0.00	0.56 ±0.00	0.75 ±0.00	0.61 ±0.00	15 ±2	13 ±1	0.99 ±0.00
	TopK	90.0 ±0.0	0.69 ±0.01	0.89 ±0.00	0.72 ±0.00	0.83 ±0.00	0.76 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	65.0 ±0.4	0.59 ±0.01	0.84 ±0.00	0.66 ±0.00	0.78 ±0.00	0.68 ±0.00	2 ±1	0 ±0	1.00 ±0.00
	Matryoshka	66.0 ±0.9	0.54 ±0.01	0.71 ±0.00	0.51 ±0.00	0.63 ±0.00	0.51 ±0.00	1 ±0	3 ±2	1.00 ±0.00
	MP-SAE	59.7 ±0.0	0.09 ±0.01	0.25 ±0.00	0.12 ±0.00	0.60 ±0.00	0.12 ±0.00	718 ±12	0 ±0	0.88 ±0.00
	iSAE	60.0 ±0.0	0.54 ±0.01	0.69 ±0.00	0.23 ±0.01	0.61 ±0.00	0.24 ±0.00	0 ±0	955 ±5	0.69 ±0.00
	Ours	71.0 ±1.4	**0.96** ±0.01	**0.97** ±0.00	**0.97** ±0.00	**0.96** ±0.00	**0.95** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 1.0$	ReLU	102 ±4	-0.07 ±0.00	0.33 ±0.01	0.18 ±0.01	0.39 ±0.00	0.21 ±0.00	56 ±8	275 ±34	0.90 ±0.01
	TopK	60.0 ±0.0	0.52 ±0.04	0.75 ±0.00	0.58 ±0.00	0.69 ±0.00	0.58 ±0.00	2 ±1	22 ±3	0.99 ±0.00
	BatchTopK	69.0 ±1.2	0.13 ±0.13	0.74 ±0.00	0.57 ±0.00	0.68 ±0.00	0.57 ±0.01	4 ±0	5 ±3	**1.00** ±0.00
	Matryoshka	67.0 ±1.5	0.44 ±0.01	0.70 ±0.00	0.51 ±0.00	0.64 ±0.00	0.52 ±0.00	6 ±0	139 ±8	0.95 ±0.00
	MP-SAE	59.6 ±0.0	-0.31 ±0.01	0.20 ±0.00	0.06 ±0.00	0.56 ±0.00	0.06 ±0.00	780 ±4	**4** ±1	0.86 ±0.00
	iSAE	60.0 ±0.0	0.47 ±0.01	0.71 ±0.00	0.24 ±0.00	0.66 ±0.00	0.26 ±0.00	0 ±0	1035 ±10	0.66 ±0.00
	Ours	103 ±1	**0.98** ±0.00	**0.94** ±0.00	**0.93** ±0.00	**0.93** ±0.00	**0.90** ±0.00	0 ±0	13 ±1	1.00 ±0.00

Table 10: **Full synthetic results with a 6K-atom ground-truth dictionary across tail exponents β.** We evaluate all methods across $\beta \in \{0, 0.5, 0.7, 1.0\}$. We report sparsity ℓ_0, reconstruction (R^2), ground-truth recovery (DCS, MCC), cross-seed consistency, and dictionary utilization measured by the usable atom fraction. The numbers of dead and duplicated atoms are also reported. Metrics are averaged across training seeds, while cross-seed DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Setting	Architecture	Sparsity	Reconstruction	Ground-truth Recovery		Cross-seed Consistency		Dictionary Utilization		
		ℓ_0	R^2	DCS	MCC	DCS	MCC	# Dup	# Dead	Usable atom frac.
$\beta = 0.0$	ReLU	76.0 ±0.3	0.64 ±0.00	0.97 ±0.00	0.78 ±0.00	0.94 ±0.00	0.94 ±0.00	44 ±5	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.65 ±0.00	0.81 ±0.00	0.61 ±0.00	0.67 ±0.00	0.53 ±0.00	2 ±1	0 ±0	1.00 ±0.00
	BatchTopK	71.0 ±0.0	0.63 ±0.00	0.76 ±0.00	0.55 ±0.00	0.59 ±0.00	0.44 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	Matryoshka	75.0 ±0.0	0.48 ±0.00	0.35 ±0.00	0.17 ±0.00	0.20 ±0.00	0.05 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	MP-SAE	58.0 ±0.0	**0.95** ±0.00	0.99 ±0.00	**0.98** ±0.00	0.98 ±0.00	0.98 ±0.00	51 ±12	0 ±0	1.00 ±0.00
	iSAE	60.0 ±0.0	0.41 ±0.00	0.19 ±0.00	0.03 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	60.0 ±0.0	0.92 ±0.00	**0.99** ±0.00	0.97 ±0.00	**0.99** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.5$	ReLU	116 ±0	0.67 ±0.00	0.90 ±0.00	0.71 ±0.00	0.85 ±0.00	0.81 ±0.00	30 ±2	0 ±0	1.00 ±0.00
	TopK	90.0 ±0.0	0.67 ±0.00	0.80 ±0.00	0.58 ±0.00	0.72 ±0.00	0.61 ±0.00	1 ±0	0 ±0	1.00 ±0.00
	BatchTopK	119 ±0	0.66 ±0.00	0.76 ±0.00	0.53 ±0.00	0.67 ±0.00	0.56 ±0.00	1 ±0	0 ±0	1.00 ±0.00
	Matryoshka	75.0 ±0.5	0.51 ±0.00	0.57 ±0.00	0.34 ±0.00	0.47 ±0.00	0.33 ±0.00	5 ±1	0 ±0	1.00 ±0.00
	MP-SAE	59.2 ±0.1	0.91 ±0.00	0.94 ±0.00	0.89 ±0.00	0.92 ±0.00	0.83 ±0.00	532 ±17	0 ±0	0.95 ±0.00
	iSAE	60.0 ±0.0	0.45 ±0.00	0.39 ±0.00	0.09 ±0.00	0.35 ±0.00	0.09 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	61.0 ±0.3	**0.95** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.7$	ReLU	73.0 ±3.6	0.17 ±0.00	0.37 ±0.00	0.20 ±0.00	0.43 ±0.00	0.22 ±0.00	50 ±5	20 ±1	0.99 ±0.00
	TopK	90.0 ±0.0	0.55 ±0.00	0.70 ±0.00	0.46 ±0.00	0.62 ±0.00	0.50 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	77.0 ±0.5	0.47 ±0.00	0.59 ±0.00	0.35 ±0.00	0.52 ±0.00	0.39 ±0.00	1 ±0	0 ±0	1.00 ±0.00
	Matryoshka	77.0 ±0.3	0.43 ±0.00	0.49 ±0.00	0.26 ±0.00	0.43 ±0.00	0.29 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	MP-SAE	59.9 ±0.0	0.19 ±0.01	0.25 ±0.00	0.11 ±0.00	0.64 ±0.00	0.10 ±0.00	1479 ±31	0 ±0	0.87 ±0.00
	iSAE	60.0 ±0.0	0.40 ±0.00	0.39 ±0.00	0.08 ±0.00	0.35 ±0.00	0.09 ±0.00	0 ±0	3746 ±21	0.39 ±0.00
	Ours	67.0 ±1.5	**0.99** ±0.00	**0.99** ±0.00	**1.00** ±0.00	**0.99** ±0.00	**0.99** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 1.0$	ReLU	56.0 ±12.1	-0.38 ±0.01	0.16 ±0.00	0.02 ±0.00	0.17 ±0.00	0.03 ±0.00	8 ±2	135 ±101	0.98 ±0.02
	TopK	60.0 ±0.0	0.35 ±0.11	0.55 ±0.00	0.33 ±0.00	0.51 ±0.00	0.36 ±0.00	1 ±1	200 ±10	0.97 ±0.00
	BatchTopK	72.0 ±1.0	0.14 ±0.02	0.53 ±0.00	0.26 ±0.00	0.48 ±0.00	0.37 ±0.00	2 ±0	62 ±4	0.99 ±0.00
	Matryoshka	75.0 ±0.8	0.24 ±0.01	0.43 ±0.00	0.19 ±0.00	0.40 ±0.00	0.27 ±0.00	5 ±1	202 ±18	0.97 ±0.00
	MP-SAE	59.7 ±0.0	-0.18 ±0.01	0.19 ±0.00	0.04 ±0.00	0.61 ±0.01	0.05 ±0.00	1490 ±32	**11** ±3	0.87 ±0.00
	iSAE	60.0 ±0.0	0.25 ±0.01	0.41 ±0.00	0.08 ±0.00	0.39 ±0.00	0.11 ±0.00	0 ±0	3994 ±10	0.35 ±0.00
	Ours	106 ±4	**0.98** ±0.00	**0.95** ±0.00	**0.95** ±0.00	**0.95** ±0.00	**0.93** ±0.00	0 ±0	23 ±7	**1.00** ±0.00

Table 11: **Full synthetic results with a 9K-atom ground-truth dictionary across tail exponents β.** We evaluate all methods across $\beta \in \{0, 0.5, 0.7, 1.0\}$. We report sparsity ℓ_0, reconstruction (R^2), ground-truth recovery (DCS, MCC), cross-seed consistency, and dictionary utilization measured by the usable atom fraction. The numbers of dead and duplicated atoms are also reported. Metrics are averaged across training seeds, while cross-seed DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Setting	Architecture	Sparsity	Reconstruction	Ground-truth Recovery		Cross-seed Consistency		Dictionary Utilization		
		ℓ_0	R^2	DCS	MCC	DCS	MCC	# Dup	# Dead	Usable atom frac.
$\beta = 0.0$	ReLU	63.0 ±0.0	0.44 ±0.00	0.78 ±0.00	0.55 ±0.00	0.62 ±0.00	0.53 ±0.00	66 ±5	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.41 ±0.00	0.23 ±0.00	0.05 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	89.0 ±0.0	0.45 ±0.00	0.22 ±0.00	0.05 ±0.00	0.15 ±0.00	0.02 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	80.0 ±0.4	0.44 ±0.00	0.19 ±0.00	0.03 ±0.00	0.14 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.47 ±0.00	0.17 ±0.00	0.01 ±0.00	0.18 ±0.00	0.00 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	iSAE	60.0 ±0.0	0.40 ±0.00	0.16 ±0.00	0.01 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	63.0 ±0.0	**0.93** ±0.00	**0.99** ±0.00	**0.98** ±0.00	**0.99** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.5$	ReLU	76.0 ±0.8	0.33 ±0.00	0.51 ±0.00	0.31 ±0.00	0.48 ±0.00	0.33 ±0.00	30 ±4	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.46 ±0.00	0.46 ±0.00	0.23 ±0.00	0.39 ±0.00	0.23 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	81.0 ±0.0	0.48 ±0.00	0.43 ±0.00	0.21 ±0.00	0.36 ±0.00	0.21 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	79.0 ±0.0	0.45 ±0.00	0.37 ±0.00	0.15 ±0.00	0.31 ±0.00	0.16 ±0.00	1 ±0	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.42 ±0.00	0.18 ±0.00	0.03 ±0.00	0.50 ±0.00	0.02 ±0.00	146 ±4	0 ±0	0.99 ±0.00
	iSAE	60.0 ±0.0	0.41 ±0.00	0.24 ±0.00	0.03 ±0.00	0.22 ±0.00	0.03 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	64.0 ±0.0	**0.96** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**1.00** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.7$	ReLU	109 ±8	0.26 ±0.01	0.36 ±0.00	0.18 ±0.00	0.42 ±0.00	0.21 ±0.00	21 ±1	9 ±1	1.00 ±0.00
	TopK	90.0 ±0.0	0.48 ±0.00	0.51 ±0.00	0.27 ±0.00	0.45 ±0.00	0.30 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	78.0 ±0.8	0.41 ±0.01	0.44 ±0.00	0.21 ±0.00	0.39 ±0.00	0.24 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	78.0 ±0.4	0.39 ±0.00	0.37 ±0.00	0.16 ±0.00	0.33 ±0.00	0.18 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.28 ±0.00	0.20 ±0.00	0.04 ±0.00	0.61 ±0.00	0.04 ±0.00	1504 ±13	0 ±0	0.92 ±0.00
	iSAE	60.0 ±0.0	0.37 ±0.00	0.30 ±0.00	0.04 ±0.00	0.28 ±0.00	0.05 ±0.00	0 ±0	6826 ±50	0.26 ±0.01
	Ours	70.0 ±0.4	**0.98** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**0.99** ±0.00	**0.99** ±0.00	0 ±0	1 ±1	1.00 ±0.00
$\beta = 1.0$	ReLU	126 ±20	-0.30 ±0.01	0.18 ±0.00	0.03 ±0.00	0.20 ±0.00	0.05 ±0.00	37 ±2	68 ±29	0.99 ±0.00
	TopK	60.0 ±0.0	0.11 ±0.02	0.45 ±0.00	0.23 ±0.00	0.43 ±0.00	0.26 ±0.00	2 ±2	573 ±71	0.94 ±0.01
	BatchTopK	75.0 ±4.3	0.16 ±0.05	0.41 ±0.00	0.18 ±0.00	0.38 ±0.00	0.22 ±0.00	2 ±1	**1** ±2	**1.00** ±0.00
	Matryoshka	77.0 ±1.1	0.20 ±0.00	0.32 ±0.00	0.11 ±0.00	0.31 ±0.00	0.15 ±0.00	2 ±1	7 ±4	1.00 ±0.00
	MP-SAE	59.8 ±0.0	-0.05 ±0.01	0.19 ±0.00	0.03 ±0.00	0.61 ±0.00	0.05 ±0.00	1906 ±4	10 ±2	0.89 ±0.00
	iSAE	60.0 ±0.0	0.23 ±0.01	0.31 ±0.00	0.04 ±0.00	0.29 ±0.00	0.07 ±0.00	0 ±0	7082 ±5	0.23 ±0.00
	Ours	128 ±6	**0.93** ±0.01	**0.94** ±0.00	**0.92** ±0.00	**0.92** ±0.00	**0.90** ±0.00	0 ±0	83 ±6	0.99 ±0.00

Table 12: **Full synthetic results with a 12K-atom ground-truth dictionary across tail exponents β.** We evaluate all methods across $\beta \in \{0, 0.5, 0.7, 1.0\}$. We report sparsity ℓ_0, reconstruction (R^2), ground-truth recovery (DCS, MCC), cross-seed consistency, and dictionary utilization measured by the usable atom fraction. The numbers of dead and duplicated atoms are also reported. Metrics are averaged across training seeds, while cross-seed DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Setting	Architecture	Sparsity	Reconstruction	Ground-truth Recovery		Cross-seed Consistency		Dictionary Utilization		
		ℓ_0	R^2	DCS	MCC	DCS	MCC	# Dup	# Dead	Usable atom frac.
$\beta = 0.0$	ReLU	87.0 ±0.0	0.27 ±0.00	0.37 ±0.00	0.16 ±0.00	0.22 ±0.00	0.05 ±0.00	1 ±1	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.40 ±0.00	0.18 ±0.00	0.02 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	88.0 ±0.6	0.45 ±0.00	0.17 ±0.00	0.02 ±0.00	0.15 ±0.00	0.02 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	85.0 ±0.0	0.44 ±0.00	0.17 ±0.00	0.02 ±0.00	0.14 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.57 ±0.00	0.18 ±0.00	0.02 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	iSAE	60.0 ±0.0	0.40 ±0.00	0.16 ±0.00	0.01 ±0.00	0.15 ±0.00	0.01 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	63.0 ±0.0	**0.82** ±0.00	**0.98** ±0.00	**0.94** ±0.00	**0.96** ±0.00	**0.98** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.5$	ReLU	102 ±0	0.36 ±0.00	0.44 ±0.00	0.24 ±0.00	0.39 ±0.00	0.25 ±0.00	29 ±3	0 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.42 ±0.00	0.30 ±0.00	0.10 ±0.00	0.25 ±0.00	0.09 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	86.0 ±0.0	0.45 ±0.00	0.29 ±0.00	0.09 ±0.00	0.24 ±0.00	0.09 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	82.0 ±0.0	0.44 ±0.00	0.27 ±0.00	0.08 ±0.00	0.23 ±0.00	0.08 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.59 ±0.00	0.32 ±0.00	0.11 ±0.00	0.28 ±0.00	0.08 ±0.00	352 ±8	0 ±0	0.98 ±0.00
	iSAE	60.0 ±0.0	0.40 ±0.00	0.19 ±0.00	0.02 ±0.00	0.18 ±0.00	0.02 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	67.0 ±1.2	**0.90** ±0.01	**0.98** ±0.00	**0.97** ±0.00	**0.97** ±0.00	**0.98** ±0.00	0 ±0	0 ±0	1.00 ±0.00
$\beta = 0.7$	ReLU	100 ±9.7	0.22 ±0.01	0.27 ±0.00	0.09 ±0.00	0.30 ±0.00	0.11 ±0.00	28 ±7	8 ±0	1.00 ±0.00
	TopK	60.0 ±0.0	0.38 ±0.02	0.35 ±0.00	0.14 ±0.00	0.32 ±0.00	0.16 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	BatchTopK	87.0 ±1.0	0.34 ±0.00	0.35 ±0.00	0.13 ±0.00	0.31 ±0.00	0.16 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Matryoshka	85.0 ±0.0	0.37 ±0.00	0.30 ±0.00	0.10 ±0.00	0.28 ±0.00	0.13 ±0.00	1 ±0	0 ±0	1.00 ±0.00
	MP-SAE	60.0 ±0.0	0.38 ±0.00	0.18 ±0.00	0.02 ±0.00	0.52 ±0.00	0.02 ±0.00	261 ±4	0 ±0	0.99 ±0.00
	iSAE	60.0 ±0.0	0.36 ±0.00	0.24 ±0.00	0.02 ±0.00	0.24 ±0.00	0.03 ±0.00	0 ±0	0 ±0	1.00 ±0.00
	Ours	90.0 ±1.0	**0.94** ±0.00	**0.97** ±0.00	**0.97** ±0.00	**0.96** ±0.00	**0.96** ±0.00	0 ±0	8 ±2	1.00 ±0.00
$\beta = 1.0$	ReLU	109 ±25	-0.29 ±0.03	0.17 ±0.00	0.02 ±0.00	0.18 ±0.00	0.03 ±0.00	31 ±3	198 ±93	0.98 ±0.01
	TopK	60.0 ±0.0	0.27 ±0.08	0.36 ±0.00	0.14 ±0.00	0.36 ±0.00	0.17 ±0.00	1 ±0	1723 ±91	0.86 ±0.01
	BatchTopK	80.0 ±1.2	0.11 ±0.01	0.32 ±0.00	0.09 ±0.00	0.31 ±0.00	0.15 ±0.00	0 ±0	1002 ±46	0.92 ±0.00
	Matryoshka	78.0 ±2.5	0.09 ±0.00	0.27 ±0.00	0.07 ±0.00	0.26 ±0.00	0.11 ±0.00	1 ±0	541 ±38	0.96 ±0.00
	MP-SAE	59.9 ±0.0	-0.01 ±0.01	0.19 ±0.00	0.03 ±0.00	0.60 ±0.00	0.05 ±0.00	2217 ±33	**4** ±1	0.90 ±0.00
	iSAE	60.0 ±0.0	0.17 ±0.01	0.26 ±0.00	0.02 ±0.00	0.25 ±0.00	0.05 ±0.00	0 ±0	10167 ±17	0.17 ±0.00
	Ours	117 ±5	**0.93** ±0.00	**0.92** ±0.00	**0.90** ±0.01	**0.91** ±0.00	**0.86** ±0.00	0 ±0	154 ±17	**0.99** ±0.00

Table 13: **Full results on CLIP ViT-B/16 across dictionary widths and sparsity levels.** We evaluate three expansion factors (4×, 8×, 16×) and three target sparsity levels ($\ell_0 \approx 32, 64, 128$). We report achieved ℓ_0, reconstruction (R^2), cross-seed consistency (DCS, MCC), and dictionary utilization, including usable atom fraction and numbers of dead and duplicated atoms. Metrics are averaged across training seeds, while DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Expansion	ℓ_0 band	Method	Sparsity	Reconstruction	Consistency		Dictionary Utilization		
			ℓ_0	R^2	DCS	MCC	Usable atom frac.	# Dup.	# Dead
4×	$\ell_0 \approx 32$	ReLU	34 ±0.4	0.89 ±0.00	0.74 ±0.00	0.68 ±0.00	0.99 ±0.00	36 ±4.1	0 ±0.00
		TopK	32 ±0.0	**0.93** ±0.00	0.80 ±0.00	0.77 ±0.00	1.00 ±0.00	1 ±0.58	0 ±0.00
		BatchTopK	32 ±0.0	0.93 ±0.00	0.72 ±0.00	0.70 ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
		Matryoshka	32 ±0.0	0.92 ±0.00	0.51 ±0.00	0.47 ±0.01	1.00 ±0.00	6 ±0.87	0 ±0.00
		MP-SAE	32 ±0.0	0.93 ±0.00	0.79 ±0.01	0.63 ±0.00	0.91 ±0.00	275 ±9	0 ±0.00
		iSAE	32 ±0.0	0.92 ±0.00	0.82 ±0.00	0.46 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	35 ±0.5	0.91 ±0.00	**0.85** ±0.00	**0.81** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	65 ±0.8	0.91 ±0.00	0.80 ±0.00	0.75 ±0.00	0.99 ±0.00	23 ±2.6	0 ±0.00
		TopK	64 ±0.0	0.94 ±0.00	0.78 ±0.00	0.78 ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
		BatchTopK	64 ±0.0	0.94 ±0.00	0.68 ±0.01	0.66 ±0.01	1.00 ±0.00	0 ±0.29	0 ±0.00
		Matryoshka	64 ±0.2	0.93 ±0.00	0.50 ±0.00	0.48 ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
		MP-SAE	64 ±0.0	**0.94** ±0.00	0.79 ±0.01	0.63 ±0.00	0.91 ±0.00	275 ±9	0 ±0.00
		iSAE	64 ±0.0	0.94 ±0.00	0.81 ±0.00	0.46 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	57 ±0.8	0.92 ±0.00	**0.86** ±0.00	**0.82** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	113 ±1	0.92 ±0.00	0.77 ±0.00	0.73 ±0.00	1.00 ±0.00	12 ±2.1	0 ±0.00
		TopK	128 ±0	0.96 ±0.00	0.65 ±0.00	0.67 ±0.00	1.00 ±0.00	1 ±0.76	0 ±0.00
		BatchTopK	129 ±0	0.96 ±0.00	0.51 ±0.00	0.49 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		Matryoshka	129 ±0	0.95 ±0.00	0.37 ±0.00	0.35 ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
		MP-SAE	128 ±0	**0.96** ±0.00	0.79 ±0.01	0.63 ±0.00	0.91 ±0.00	275 ±9	0 ±0.00
		iSAE	128 ±0	0.95 ±0.00	0.75 ±0.00	0.42 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	121 ±0	0.96 ±0.00	**0.83** ±0.00	**0.78** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
8×	$\ell_0 \approx 32$	ReLU	34 ±1.8	0.89 ±0.00	0.52 ±0.01	0.44 ±0.01	0.99 ±0.00	73 ±4.5	0 ±0.00
		TopK	32 ±0.0	0.94 ±0.00	0.73 ±0.00	0.67 ±0.00	1.00 ±0.00	4 ±2.00	0 ±0.00
		BatchTopK	32 ±0.4	0.94 ±0.00	0.61 ±0.00	0.57 ±0.00	1.00 ±0.00	1 ±0.76	0 ±0.00
		Matryoshka	32 ±0.0	0.92 ±0.00	0.39 ±0.00	0.32 ±0.00	1.00 ±0.00	27 ±4.1	0 ±0.00
		MP-SAE	32 ±0.0	0.93 ±0.00	0.75 ±0.01	0.44 ±0.00	0.84 ±0.01	983 ±80	0 ±0.00
		iSAE	32 ±0.0	0.93 ±0.00	0.75 ±0.00	0.40 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	35 ±0.4	**0.94** ±0.00	**0.76** ±0.00	**0.69** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	65 ±3.2	0.91 ±0.00	0.66 ±0.00	0.58 ±0.00	0.99 ±0.00	76 ±7.3	0 ±0.00
		TopK	64 ±0.0	0.95 ±0.00	0.74 ±0.00	0.71 ±0.00	1.00 ±0.00	4 ±1.04	0 ±0.00
		BatchTopK	64 ±0.4	0.95 ±0.00	0.62 ±0.00	0.59 ±0.00	1.00 ±0.00	2 ±1.04	0 ±0.00
		Matryoshka	65 ±0.0	0.94 ±0.00	0.40 ±0.00	0.35 ±0.00	1.00 ±0.00	9 ±2.00	0 ±0.00
		MP-SAE	64 ±0.0	**0.95** ±0.00	0.75 ±0.01	0.44 ±0.00	0.84 ±0.01	983 ±80	0 ±0.00
		iSAE	64 ±0.0	0.94 ±0.00	0.80 ±0.00	0.44 ±0.01	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	64 ±0.7	0.91 ±0.00	**0.82** ±0.00	**0.79** ±0.00	1.00 ±0.00	2 ±0.58	0 ±0.00
	$\ell_0 \approx 128$	ReLU	121 ±0	0.93 ±0.00	0.67 ±0.00	0.60 ±0.00	0.99 ±0.00	47 ±11.0	0 ±0.00
		TopK	128 ±0	0.96 ±0.00	0.68 ±0.00	0.66 ±0.00	1.00 ±0.00	3 ±1.76	0 ±0.00
		BatchTopK	129 ±0	0.96 ±0.00	0.53 ±0.00	0.50 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		Matryoshka	129 ±0	0.95 ±0.00	0.30 ±0.00	0.25 ±0.00	1.00 ±0.00	3 ±2.31	0 ±0.00
		MP-SAE	128 ±0	**0.97** ±0.00	0.75 ±0.01	0.44 ±0.00	0.84 ±0.01	983 ±80	0 ±0.00
		iSAE	128 ±0	0.96 ±0.00	0.78 ±0.00	0.43 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	134 ±1	0.94 ±0.00	**0.82** ±0.00	**0.79** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
16×	$\ell_0 \approx 32$	ReLU	29 ±2.1	0.89 ±0.00	0.33 ±0.00	0.23 ±0.01	0.99 ±0.00	109 ±8	0 ±0.00
		TopK	32 ±0.0	**0.94** ±0.00	0.48 ±0.00	0.38 ±0.00	1.00 ±0.00	4 ±2.50	0 ±0.00
		BatchTopK	32 ±0.0	0.93 ±0.00	0.42 ±0.00	0.34 ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
		Matryoshka	32 ±0.0	0.93 ±0.00	0.29 ±0.00	0.20 ±0.00	1.00 ±0.00	9 ±4.77	0 ±0.00
		MP-SAE	32 ±0.0	0.94 ±0.00	**0.69** ±0.00	0.27 ±0.00	0.79 ±0.00	2623 ±67	0 ±0.00
		iSAE	32 ±0.0	0.93 ±0.00	0.65 ±0.00	0.27 ±0.00	0.67 ±0.00	1589 ±6	2517 ±51
		ConSDL	35 ±0.7	0.92 ±0.00	0.64 ±0.00	**0.56** ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	64 ±1.4	0.91 ±0.00	0.47 ±0.00	0.36 ±0.00	0.99 ±0.00	181 ±7	0 ±0.00
		TopK	64 ±0.0	**0.95** ±0.00	0.60 ±0.00	0.52 ±0.00	1.00 ±0.00	16 ±0.3	0 ±0.00
		BatchTopK	64 ±0.0	0.95 ±0.00	0.48 ±0.00	0.42 ±0.00	1.00 ±0.00	3 ±1.26	0 ±0.00
		Matryoshka	64 ±0.0	0.94 ±0.00	0.30 ±0.00	0.22 ±0.00	1.00 ±0.00	13 ±2.8	0 ±0.00
		MP-SAE	64 ±0.0	0.95 ±0.00	**0.69** ±0.00	0.27 ±0.00	0.79 ±0.00	2623 ±67	0 ±0.00
		iSAE	64 ±0.0	0.95 ±0.00	0.68 ±0.00	0.36 ±0.00	1.00 ±0.00	11 ±3.3	0 ±0.58
		ConSDL	63 ±0.7	0.90 ±0.01	0.67 ±0.00	**0.63** ±0.00	1.00 ±0.00	**2** ±0.72	0 ±0.00
	$\ell_0 \approx 128$	ReLU	116 ±1	0.93 ±0.00	0.52 ±0.00	0.42 ±0.00	0.99 ±0.00	168 ±9	0 ±0.00
		TopK	128 ±0	0.96 ±0.00	0.56 ±0.00	0.50 ±0.00	1.00 ±0.00	12 ±0.6	0 ±0.00
		BatchTopK	129 ±0	0.96 ±0.00	0.48 ±0.00	0.44 ±0.00	1.00 ±0.00	4 ±1.00	0 ±0.00
		Matryoshka	129 ±0	0.95 ±0.00	0.27 ±0.00	0.19 ±0.00	1.00 ±0.00	6 ±3.01	0 ±0.00
		MP-SAE	128 ±0	**0.97** ±0.00	0.69 ±0.00	0.27 ±0.00	0.79 ±0.00	2623 ±67	0 ±0.00
		iSAE	128 ±0	0.96 ±0.00	0.68 ±0.00	0.37 ±0.00	1.00 ±0.00	6 ±0.76	0 ±0.00
		ConSDL	130 ±2	0.92 ±0.00	**0.78** ±0.00	**0.74** ±0.00	1.00 ±0.00	**1** ±0.90	0 ±0.00

Table 14: **Full results on DINOv2-B/14 across dictionary widths and sparsity levels.** We evaluate three expansion factors (4×, 8×, 16×) and three target sparsity levels ($\ell_0 \approx 32, 64, 128$). We report achieved ℓ_0, reconstruction (R^2), cross-seed consistency (DCS, MCC), and dictionary utilization, including usable atom fraction and numbers of dead and duplicated atoms. Metrics are averaged across training seeds, while DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Expansion	ℓ_0 band	Method	Sparsity	Reconstruction	Consistency		Dictionary Utilization		
			ℓ_0	R^2	DCS	MCC	Usable atom frac.	# Dup.	# Dead
4×	$\ell_0 \approx 32$	ReLU	29 ±0.0	0.93 ±0.00	0.79 ±0.00	0.75 ±0.00	0.99 ±0.00	26 ±2.6	0 ±0.00
		TopK	32 ±0.0	0.96 ±0.00	0.90 ±0.00	**0.88** ±0.00	1.00 ±0.00	3 ±2.02	0 ±0.00
		BatchTopK	32 ±0.0	0.96 ±0.00	0.83 ±0.00	0.81 ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
		Matryoshka	32 ±0.0	0.95 ±0.00	0.67 ±0.00	0.67 ±0.00	1.00 ±0.00	9 ±2.84	0 ±0.00
		MP-SAE	32 ±0.0	0.96 ±0.00	0.87 ±0.02	0.69 ±0.01	0.89 ±0.01	352 ±28	0 ±0.00
		iSAE	32 ±0.0	0.95 ±0.00	0.90 ±0.00	0.49 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.58
		ConSDL	32 ±0.0	0.95 ±0.00	**0.90** ±0.00	0.87 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	66 ±0.5	0.95 ±0.00	0.88 ±0.00	0.85 ±0.00	1.00 ±0.00	9 ±1.42	0 ±0.00
		TopK	64 ±0.0	0.97 ±0.00	0.89 ±0.00	**0.89** ±0.00	1.00 ±0.00	1 ±0.76	0 ±0.00
		BatchTopK	64 ±0.2	0.97 ±0.00	0.88 ±0.00	0.87 ±0.00	1.00 ±0.00	0 ±0.58	0 ±0.00
		Matryoshka	64 ±0.0	0.96 ±0.00	0.67 ±0.00	0.70 ±0.00	1.00 ±0.00	0 ±0.50	0 ±0.00
		MP-SAE	64 ±0.0	0.97 ±0.00	0.87 ±0.02	0.69 ±0.01	0.89 ±0.01	352 ±28	0 ±0.00
		iSAE	64 ±0.0	0.96 ±0.00	0.89 ±0.00	0.50 ±0.01	1.00 ±0.00	0 ±0.00	2 ±1.00
		ConSDL	66 ±1.4	**0.97** ±0.00	**0.90** ±0.00	0.86 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	123 ±0	0.96 ±0.00	0.86 ±0.00	0.84 ±0.00	1.00 ±0.00	7 ±0.63	0 ±0.00
		TopK	128 ±0	0.97 ±0.00	0.82 ±0.00	0.84 ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
		BatchTopK	128 ±0	0.97 ±0.00	0.80 ±0.00	0.83 ±0.00	1.00 ±0.00	1 ±0.76	0 ±0.00
		Matryoshka	128 ±0	0.96 ±0.00	0.57 ±0.00	0.63 ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
		MP-SAE	127 ±0	0.98 ±0.00	0.87 ±0.02	0.69 ±0.01	0.89 ±0.01	352 ±28	0 ±0.00
		iSAE	128 ±0	0.97 ±0.00	0.86 ±0.00	0.51 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	138 ±0	**0.98** ±0.00	**0.88** ±0.00	0.84 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
8×	$\ell_0 \approx 32$	ReLU	32 ±0.4	0.94 ±0.00	0.57 ±0.00	0.49 ±0.00	0.99 ±0.00	89 ±11.5	0 ±0.00
		TopK	32 ±0.0	0.96 ±0.00	0.75 ±0.00	0.67 ±0.01	0.99 ±0.00	35 ±9.6	0 ±0.00
		BatchTopK	32 ±0.0	0.96 ±0.00	0.60 ±0.00	0.53 ±0.00	1.00 ±0.00	4 ±1.80	0 ±0.00
		Matryoshka	32 ±0.0	0.95 ±0.00	0.50 ±0.00	0.43 ±0.00	0.98 ±0.00	121 ±12	0 ±0.00
		MP-SAE	32 ±0.0	**0.96** ±0.00	0.83 ±0.00	0.49 ±0.00	0.74 ±0.01	1577 ±64	0 ±0.00
		iSAE	32 ±0.0	0.96 ±0.00	0.77 ±0.00	0.41 ±0.00	1.00 ±0.00	4 ±1.50	14 ±5.1
		ConSDL	34 ±1.3	0.96 ±0.00	**0.84** ±0.00	**0.76** ±0.00	**1.00** ±0.00	**0** ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	54 ±0.6	0.95 ±0.00	0.74 ±0.00	0.66 ±0.00	0.98 ±0.00	148 ±5	0 ±0.00
		TopK	64 ±0.0	0.97 ±0.00	0.78 ±0.00	0.75 ±0.00	1.00 ±0.00	22 ±3.1	0 ±0.00
		BatchTopK	64 ±0.0	0.97 ±0.00	0.67 ±0.00	0.63 ±0.00	1.00 ±0.00	5 ±1.44	0 ±0.00
		Matryoshka	64 ±0.0	0.96 ±0.00	0.53 ±0.00	0.50 ±0.00	0.98 ±0.00	94 ±8.4	0 ±0.00
		MP-SAE	64 ±0.0	**0.97** ±0.00	0.83 ±0.00	0.49 ±0.00	0.74 ±0.01	1577 ±64	0 ±0.00
		iSAE	64 ±0.0	0.97 ±0.00	0.79 ±0.00	0.47 ±0.00	1.00 ±0.00	2 ±0.58	10 ±2.6
		ConSDL	61 ±0.4	0.96 ±0.00	**0.85** ±0.00	**0.81** ±0.00	**1.00** ±0.00	**0** ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	128 ±0	0.96 ±0.00	0.75 ±0.00	0.68 ±0.00	0.99 ±0.00	57 ±4.9	0 ±0.00
		TopK	128 ±0	0.98 ±0.00	0.73 ±0.00	0.74 ±0.00	1.00 ±0.00	20 ±1.3	0 ±0.00
		BatchTopK	128 ±0	0.98 ±0.00	0.65 ±0.00	0.65 ±0.00	1.00 ±0.00	9 ±0.58	0 ±0.00
		Matryoshka	128 ±0	0.97 ±0.00	0.48 ±0.00	0.48 ±0.00	0.99 ±0.00	62 ±6.9	0 ±0.00
		MP-SAE	127 ±0	**0.98** ±0.00	**0.83** ±0.00	0.49 ±0.00	0.74 ±0.01	1577 ±64	0 ±0.00
		iSAE	128 ±0	0.97 ±0.00	0.78 ±0.00	0.51 ±0.01	1.00 ±0.00	0 ±0.29	3 ±1.53
		ConSDL	122 ±2	0.98 ±0.00	0.81 ±0.00	**0.74** ±0.00	**1.00** ±0.00	**0** ±0.00	0 ±0.00
16×	$\ell_0 \approx 32$	ReLU	33 ±0.0	0.94 ±0.00	0.38 ±0.00	0.28 ±0.00	0.98 ±0.00	187 ±10	96 ±35.2
		TopK	32 ±0.0	0.96 ±0.00	0.50 ±0.04	0.37 ±0.02	0.88 ±0.08	218 ±116	1300 ±1126
		BatchTopK	32 ±0.0	0.96 ±0.00	0.42 ±0.00	0.29 ±0.00	**1.00** ±0.00	2 ±1.32	0 ±0.00
		Matryoshka	32 ±0.0	0.95 ±0.00	0.34 ±0.00	0.22 ±0.00	0.99 ±0.00	64 ±6.9	0 ±0.58
		MP-SAE	32 ±0.0	**0.96** ±0.00	**0.78** ±0.00	0.31 ±0.00	0.62 ±0.00	4640 ±39	0 ±0.00
		iSAE	32 ±0.0	0.96 ±0.00	0.63 ±0.00	0.29 ±0.00	0.99 ±0.00	52 ±5.3	94 ±29.8
		ConSDL	36 ±0.0	0.95 ±0.00	0.75 ±0.00	**0.66** ±0.00	0.94 ±0.00	**0** ±0.00	704 ±36
	$\ell_0 \approx 64$	ReLU	62 ±0.7	0.95 ±0.00	0.53 ±0.00	0.42 ±0.00	0.96 ±0.00	414 ±35	29 ±23.3
		TopK	64 ±0.0	0.97 ±0.00	0.60 ±0.00	0.51 ±0.01	0.99 ±0.00	70 ±8.4	0 ±0.00
		BatchTopK	64 ±0.0	0.97 ±0.00	0.48 ±0.00	0.41 ±0.00	1.00 ±0.00	35 ±7.0	0 ±0.00
		Matryoshka	64 ±2.1	0.96 ±0.00	0.38 ±0.00	0.31 ±0.02	0.99 ±0.00	126 ±31	0 ±0.00
		MP-SAE	64 ±0.0	**0.97** ±0.00	0.78 ±0.00	0.31 ±0.00	0.62 ±0.00	4640 ±39	0 ±0.00
		iSAE	64 ±0.0	0.97 ±0.00	0.65 ±0.00	0.36 ±0.00	0.98 ±0.00	233 ±6	52 ±24.7
		ConSDL	71 ±0.0	0.95 ±0.00	**0.80** ±0.00	**0.75** ±0.00	**1.00** ±0.00	**14** ±1.4	0 ±0.00
	$\ell_0 \approx 128$	ReLU	125 ±1	0.96 ±0.00	0.62 ±0.00	0.50 ±0.00	0.96 ±0.00	462 ±11	0 ±0.00
		TopK	128 ±0	0.98 ±0.00	0.58 ±0.00	0.53 ±0.00	0.99 ±0.00	62 ±4.6	0 ±0.00
		BatchTopK	128 ±0	0.98 ±0.00	0.48 ±0.00	0.44 ±0.00	1.00 ±0.00	27 ±5.6	0 ±0.00
		Matryoshka	128 ±0	0.97 ±0.00	0.35 ±0.00	0.30 ±0.00	0.99 ±0.00	88 ±3.3	0 ±0.00
		MP-SAE	128 ±0	**0.98** ±0.00	0.78 ±0.00	0.31 ±0.00	0.62 ±0.00	4640 ±39	0 ±0.00
		iSAE	128 ±0	0.98 ±0.00	0.68 ±0.00	0.44 ±0.00	0.99 ±0.00	88 ±2.3	4 ±0.58
		ConSDL	138 ±0	0.95 ±0.00	**0.81** ±0.00	**0.77** ±0.00	1.00 ±0.00	**24** ±3.6	0 ±0.00

Table 15: **Full results on GPT-2 Small across dictionary widths and sparsity levels.** We evaluate three expansion factors (4×, 8×, 16×) and three target sparsity levels ($\ell_0 \approx 32, 64, 128$). We report achieved ℓ_0, reconstruction (R^2), cross-seed consistency (DCS, MCC), and dictionary utilization, including usable atom fraction and numbers of dead and duplicated atoms. Metrics are averaged across training seeds, while DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Expansion	ℓ_0 band	Method	Sparsity	Reconstruction	Consistency		Dictionary Utilization		
			ℓ_0	R^2	DCS	MCC	Usable atom frac.	# Dup.	# Dead
4×	$\ell_0 \approx 32$	ReLU	31 ±0.2	0.87 ±0.02	**0.91** ±0.00	**0.86** ±0.00	0.95 ±0.00	161 ±8	6 ±3.00
		TopK	32 ±0.0	0.89 ±0.00	0.83 ±0.00	0.79 ±0.00	0.99 ±0.00	27 ±3.4	0 ±0.00
		BatchTopK	32 ±0.1	0.90 ±0.00	0.83 ±0.00	0.79 ±0.00	0.99 ±0.00	17 ±3.6	0 ±0.00
		Matryoshka	32 ±0.1	0.88 ±0.00	0.71 ±0.00	0.69 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	32 ±0.0	**0.98** ±0.00	0.87 ±0.00	0.64 ±0.01	0.83 ±0.00	517 ±13	0 ±0.00
		iSAE	32 ±0.0	0.89 ±0.00	0.85 ±0.00	0.47 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	34 ±0.1	0.90 ±0.00	0.89 ±0.01	0.83 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	68 ±0.2	0.89 ±0.00	0.89 ±0.00	0.86 ±0.00	0.97 ±0.00	82 ±2.0	0 ±0.00
		TopK	64 ±0.0	0.92 ±0.00	0.80 ±0.00	0.78 ±0.00	1.00 ±0.00	4 ±0.50	0 ±0.00
		BatchTopK	64 ±0.1	0.92 ±0.00	0.79 ±0.00	0.78 ±0.00	1.00 ±0.00	6 ±1.61	0 ±0.00
		Matryoshka	64 ±0.1	0.91 ±0.00	0.67 ±0.00	0.67 ±0.00	1.00 ±0.00	0 ±0.87	0 ±0.00
		MP-SAE	64 ±0.0	**0.98** ±0.00	0.87 ±0.00	0.64 ±0.01	0.83 ±0.00	517 ±13	0 ±0.00
		iSAE	64 ±0.0	0.91 ±0.00	0.83 ±0.00	0.48 ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	62 ±1.7	0.90 ±0.00	**0.90** ±0.00	**0.86** ±0.00	1.00 ±0.00	0 ±0.38	0 ±0.00
	$\ell_0 \approx 128$	ReLU	134 ±1	0.91 ±0.00	0.87 ±0.00	**0.84** ±0.00	0.98 ±0.00	48 ±0.5	0 ±0.00
		TopK	128 ±0	0.94 ±0.00	0.71 ±0.00	0.72 ±0.00	1.00 ±0.00	0 ±0.87	0 ±0.00
		BatchTopK	128 ±0	0.94 ±0.00	0.70 ±0.00	0.73 ±0.00	1.00 ±0.00	4 ±0.76	0 ±0.00
		Matryoshka	128 ±0	0.93 ±0.00	0.59 ±0.00	0.62 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	127 ±0	**0.99** ±0.00	0.87 ±0.00	0.64 ±0.01	0.83 ±0.00	517 ±13	0 ±0.00
		iSAE	128 ±0	0.94 ±0.00	0.78 ±0.00	0.48 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	126 ±0	0.94 ±0.00	**0.89** ±0.00	0.84 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
8×	$\ell_0 \approx 32$	ReLU	33 ±0.3	0.87 ±0.00	0.79 ±0.01	0.73 ±0.01	0.80 ±0.04	357 ±40	845 ±365
		TopK	32 ±0.0	0.91 ±0.00	0.82 ±0.00	0.77 ±0.00	0.99 ±0.00	91 ±6.0	0 ±0.00
		BatchTopK	32 ±0.1	0.91 ±0.00	0.82 ±0.00	0.77 ±0.00	0.99 ±0.00	91 ±10.3	0 ±0.00
		Matryoshka	32 ±0.1	0.90 ±0.00	0.69 ±0.00	0.68 ±0.00	1.00 ±0.00	0 ±0.50	0 ±0.00
		MP-SAE	32 ±0.0	**0.98** ±0.00	0.86 ±0.00	0.56 ±0.00	0.79 ±0.00	1292 ±8	0 ±0.00
		iSAE	32 ±0.0	0.90 ±0.00	0.82 ±0.00	0.45 ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	37 ±0.2	0.90 ±0.00	**0.90** ±0.00	**0.84** ±0.00	1.00 ±0.00	2 ±0.43	0 ±0.00
	$\ell_0 \approx 64$	ReLU	67 ±0.2	0.91 ±0.01	0.89 ±0.00	0.84 ±0.00	0.95 ±0.00	288 ±7	0 ±0.00
		TopK	64 ±0.0	0.93 ±0.00	0.78 ±0.00	0.74 ±0.00	1.00 ±0.00	25 ±3.4	0 ±0.00
		BatchTopK	64 ±0.1	0.93 ±0.00	0.78 ±0.00	0.76 ±0.00	1.00 ±0.00	26 ±1.1	0 ±0.00
		Matryoshka	64 ±0.1	0.92 ±0.00	0.66 ±0.00	0.67 ±0.00	1.00 ±0.00	0 ±0.58	0 ±0.00
		MP-SAE	64 ±0.0	**0.99** ±0.00	0.86 ±0.00	0.56 ±0.00	0.79 ±0.00	1292 ±8	0 ±0.00
		iSAE	64 ±0.0	0.92 ±0.00	0.80 ±0.00	0.47 ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	64 ±0.3	0.93 ±0.01	0.89 ±0.00	**0.84** ±0.00	1.00 ±0.00	0 ±0.72	0 ±0.00
	$\ell_0 \approx 128$	ReLU	124 ±1	0.92 ±0.00	0.87 ±0.00	0.81 ±0.00	0.97 ±0.00	171 ±7	0 ±0.00
		TopK	128 ±0	0.95 ±0.00	0.72 ±0.00	0.72 ±0.00	1.00 ±0.00	5 ±2.36	0 ±0.00
		BatchTopK	128 ±0	0.95 ±0.00	0.72 ±0.00	0.73 ±0.00	1.00 ±0.00	7 ±1.50	0 ±0.00
		Matryoshka	128 ±0	0.94 ±0.00	0.60 ±0.00	0.65 ±0.00	1.00 ±0.00	0 ±0.58	0 ±0.00
		MP-SAE	127 ±0	**0.99** ±0.00	0.86 ±0.00	0.56 ±0.00	0.79 ±0.00	1292 ±8	0 ±0.00
		iSAE	128 ±0	0.94 ±0.00	0.76 ±0.00	0.47 ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	134 ±0	0.94 ±0.00	**0.89** ±0.00	**0.84** ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
16×	$\ell_0 \approx 32$	ReLU	30 ±0.9	0.86 ±0.01	0.42 ±0.01	0.31 ±0.01	0.38 ±0.06	366 ±112	7243 ±1086
		TopK	32 ±0.0	0.92 ±0.00	0.80 ±0.00	0.73 ±0.00	0.98 ±0.00	205 ±6	0 ±0.00
		BatchTopK	32 ±0.1	0.92 ±0.00	0.80 ±0.00	0.74 ±0.00	0.98 ±0.00	288 ±1	0 ±0.00
		Matryoshka	32 ±0.0	0.91 ±0.00	0.68 ±0.00	0.67 ±0.00	**1.00** ±0.00	**15** ±2.8	0 ±0.00
		MP-SAE	32 ±0.0	**0.98** ±0.00	0.84 ±0.00	0.48 ±0.00	0.75 ±0.00	3033 ±5	0 ±0.00
		iSAE	32 ±0.0	0.91 ±0.00	0.68 ±0.09	0.36 ±0.05	0.89 ±0.16	222 ±384	1177 ±2038
		ConSDL	34 ±0.1	0.90 ±0.00	**0.85** ±0.00	**0.78** ±0.00	0.99 ±0.00	33 ±2.5	32 ±6.5
	$\ell_0 \approx 64$	ReLU	64 ±0.9	0.91 ±0.00	0.80 ±0.01	0.73 ±0.01	0.84 ±0.04	779 ±66	1195 ±618
		TopK	64 ±0.0	0.94 ±0.00	0.76 ±0.00	0.72 ±0.00	0.99 ±0.00	103 ±9	0 ±0.00
		BatchTopK	64 ±0.2	0.94 ±0.00	0.76 ±0.00	0.73 ±0.00	0.99 ±0.00	100 ±3	0 ±0.00
		Matryoshka	64 ±0.1	0.93 ±0.00	0.64 ±0.00	0.67 ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
		MP-SAE	64 ±0.0	**0.99** ±0.00	**0.84** ±0.00	0.48 ±0.00	0.75 ±0.00	3033 ±5	0 ±0.00
		iSAE	64 ±0.0	0.93 ±0.00	0.76 ±0.00	0.45 ±0.00	1.00 ±0.00	**0** ±0.29	0 ±0.00
		ConSDL	68 ±0.1	0.93 ±0.00	0.84 ±0.00	**0.78** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	133 ±1	0.93 ±0.01	0.84 ±0.00	0.76 ±0.00	0.95 ±0.00	494 ±13	119 ±85
		TopK	128 ±0	0.95 ±0.00	0.69 ±0.00	0.67 ±0.00	1.00 ±0.00	34 ±8.0	0 ±0.00
		BatchTopK	128 ±0	0.95 ±0.00	0.71 ±0.00	0.72 ±0.00	1.00 ±0.00	34 ±6.1	0 ±0.00
		Matryoshka	128 ±0	0.95 ±0.00	0.60 ±0.00	0.65 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	128 ±0	**0.99** ±0.00	**0.84** ±0.00	0.48 ±0.00	0.75 ±0.00	3033 ±5	0 ±0.00
		iSAE	128 ±0	0.95 ±0.00	0.70 ±0.00	0.46 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	126 ±0	0.94 ±0.00	0.84 ±0.00	**0.78** ±0.00	1.00 ±0.00	5 ±0.95	0 ±0.00

Table 16: **Full results on Pythia-160M across dictionary widths and sparsity levels.** We evaluate three expansion factors (4×, 8×, 16×) and three target sparsity levels ($\ell_0 \approx 32, 64, 128$). We report achieved ℓ_0, reconstruction (R^2), cross-seed consistency (DCS, MCC), and dictionary utilization, including usable atom fraction and numbers of dead and duplicated atoms. Metrics are averaged across training seeds, while DCS and MCC are averaged across seed pairs. ± denotes the corresponding standard deviation.

Expansion	ℓ_0 band	Method	Sparsity	Reconstruction	Consistency		Dictionary Utilization		
			ℓ_0	R^2	DCS	MCC	Usable atom frac.	# Dup.	# Dead
4×	$\ell_0 \approx 32$	ReLU	31 ±0.3	0.84 ±0.00	**0.90** ±0.00	**0.82** ±0.01	0.92 ±0.00	247 ±5	1 ±1.15
		TopK	32 ±0.0	0.90 ±0.00	0.84 ±0.00	0.78 ±0.00	1.00 ±0.00	15 ±1.8	0 ±0.00
		BatchTopK	32 ±0.1	0.91 ±0.02	0.83 ±0.00	0.78 ±0.00	1.00 ±0.00	13 ±1.0	0 ±0.00
		Matryoshka	32 ±0.0	0.90 ±0.02	0.71 ±0.00	0.67 ±0.00	1.00 ±0.00	0 ±0.29	0 ±0.00
		MP-SAE	32 ±0.0	**0.96** ±0.00	0.84 ±0.01	0.51 ±0.00	0.72 ±0.00	874 ±7	0 ±0.00
		iSAE	32 ±0.0	0.89 ±0.00	0.84 ±0.00	0.45 ±0.01	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	34 ±0.3	0.88 ±0.00	0.88 ±0.00	0.81 ±0.00	1.00 ±0.00	6 ±0.00	0 ±0.00
	$\ell_0 \approx 64$	ReLU	55 ±0.5	0.89 ±0.02	**0.88** ±0.00	0.81 ±0.00	0.96 ±0.00	136 ±4	0 ±0.00
		TopK	64 ±0.0	0.93 ±0.02	0.79 ±0.00	0.75 ±0.00	1.00 ±0.00	3 ±1.15	0 ±0.00
		BatchTopK	64 ±0.2	0.93 ±0.02	0.78 ±0.00	0.75 ±0.00	1.00 ±0.00	4 ±0.52	0 ±0.00
		Matryoshka	64 ±0.1	0.92 ±0.02	0.66 ±0.00	0.64 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	64 ±0.1	**0.97** ±0.00	0.84 ±0.01	0.51 ±0.00	0.72 ±0.00	874 ±7	0 ±0.00
		iSAE	64 ±0.0	0.91 ±0.00	0.83 ±0.00	0.46 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	63 ±0.6	0.88 ±0.00	0.87 ±0.00	**0.82** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	126 ±1	0.90 ±0.00	0.84 ±0.00	0.78 ±0.00	0.99 ±0.00	45 ±3.2	0 ±0.00
		TopK	128 ±0	0.95 ±0.01	0.70 ±0.00	0.68 ±0.00	1.00 ±0.00	1 ±0.00	0 ±0.00
		BatchTopK	128 ±0	0.95 ±0.01	0.69 ±0.00	0.69 ±0.00	1.00 ±0.00	1 ±0.58	0 ±0.00
		Matryoshka	128 ±0	0.94 ±0.01	0.58 ±0.00	0.59 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	127 ±0	**0.98** ±0.00	0.84 ±0.01	0.51 ±0.00	0.72 ±0.00	874 ±7	0 ±0.00
		iSAE	128 ±0	0.94 ±0.00	0.79 ±0.01	0.45 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	119 ±0	0.86 ±0.00	**0.85** ±0.00	**0.78** ±0.00	1.00 ±0.00	1 ±0.29	0 ±0.00
8×	$\ell_0 \approx 32$	ReLU	34 ±0.3	0.86 ±0.00	**0.89** ±0.00	**0.80** ±0.00	0.90 ±0.00	640 ±15	1 ±1.00
		TopK	32 ±0.0	0.91 ±0.00	0.81 ±0.00	0.74 ±0.00	0.99 ±0.00	51 ±3.6	0 ±0.58
		BatchTopK	32 ±0.1	0.92 ±0.02	0.82 ±0.00	0.75 ±0.00	0.99 ±0.00	44 ±4.5	0 ±0.00
		Matryoshka	32 ±0.0	0.91 ±0.02	0.68 ±0.00	0.65 ±0.00	1.00 ±0.00	2 ±0.50	0 ±0.00
		MP-SAE	32 ±0.0	**0.97** ±0.00	0.87 ±0.01	0.47 ±0.01	0.63 ±0.00	2273 ±29	0 ±0.00
		iSAE	32 ±0.0	0.90 ±0.00	0.82 ±0.00	0.43 ±0.00	1.00 ±0.00	**0** ±0.00	0 ±0.00
		ConSDL	35 ±0.2	0.90 ±0.00	0.86 ±0.00	0.77 ±0.00	1.00 ±0.00	7 ±1.26	0 ±0.00
	$\ell_0 \approx 64$	ReLU	61 ±0.3	0.90 ±0.02	0.87 ±0.00	**0.78** ±0.00	0.94 ±0.00	375 ±2	0 ±0.00
		TopK	64 ±0.0	0.93 ±0.01	0.75 ±0.00	0.70 ±0.00	1.00 ±0.00	16 ±1.7	0 ±0.00
		BatchTopK	64 ±0.2	0.94 ±0.01	0.76 ±0.00	0.71 ±0.00	1.00 ±0.00	15 ±2.6	0 ±0.00
		Matryoshka	64 ±0.1	0.93 ±0.01	0.64 ±0.00	0.63 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	64 ±0.0	**0.97** ±0.00	0.87 ±0.01	0.47 ±0.01	0.63 ±0.00	2273 ±29	0 ±0.00
		iSAE	64 ±0.0	0.92 ±0.00	0.80 ±0.00	0.44 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	69 ±0.1	0.88 ±0.00	0.86 ±0.00	0.77 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	123 ±0	0.91 ±0.00	0.83 ±0.00	0.74 ±0.00	0.97 ±0.00	172 ±3	0 ±0.00
		TopK	128 ±0	0.95 ±0.01	0.67 ±0.00	0.63 ±0.00	1.00 ±0.00	4 ±1.52	0 ±0.00
		BatchTopK	128 ±0	0.96 ±0.01	0.69 ±0.00	0.67 ±0.00	1.00 ±0.00	8 ±1.04	0 ±0.00
		Matryoshka	128 ±0	0.95 ±0.01	0.58 ±0.00	0.62 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		MP-SAE	127 ±0	**0.98** ±0.00	**0.87** ±0.01	0.47 ±0.01	0.63 ±0.00	2273 ±29	0 ±0.00
		iSAE	128 ±0	0.94 ±0.00	0.76 ±0.01	0.44 ±0.01	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	135 ±0	0.87 ±0.00	0.82 ±0.00	**0.75** ±0.00	1.00 ±0.00	2 ±0.75	0 ±0.00
16×	$\ell_0 \approx 32$	ReLU	33 ±0.3	0.86 ±0.00	0.80 ±0.00	0.69 ±0.00	0.83 ±0.00	1369 ±20	709 ±34
		TopK	32 ±0.0	0.92 ±0.02	0.78 ±0.00	0.70 ±0.00	0.99 ±0.00	121 ±10	0 ±0.00
		BatchTopK	32 ±0.2	0.92 ±0.02	0.79 ±0.00	0.72 ±0.00	0.99 ±0.00	108 ±5	0 ±0.00
		Matryoshka	32 ±0.1	0.92 ±0.02	0.58 ±0.00	0.52 ±0.00	**1.00** ±0.00	**5** ±2.38	0 ±0.00
		MP-SAE	32 ±0.0	**0.97** ±0.00	**0.86** ±0.01	0.41 ±0.01	0.61 ±0.00	4809 ±26	0 ±0.00
		iSAE	32 ±0.0	0.90 ±0.00	0.70 ±0.00	0.31 ±0.00	0.75 ±0.02	466 ±43	2618 ±232
		ConSDL	33 ±0.1	0.91 ±0.00	0.83 ±0.00	**0.72** ±0.00	1.00 ±0.00	8 ±2.29	52 ±7.2
	$\ell_0 \approx 64$	ReLU	64 ±0.5	0.91 ±0.02	0.85 ±0.00	**0.73** ±0.00	0.92 ±0.00	1040 ±5	0 ±0.00
		TopK	64 ±0.0	0.94 ±0.01	0.71 ±0.00	0.65 ±0.00	1.00 ±0.00	50 ±2.6	0 ±0.00
		BatchTopK	64 ±0.3	0.94 ±0.01	0.73 ±0.00	0.68 ±0.00	1.00 ±0.00	42 ±1.3	0 ±0.00
		Matryoshka	64 ±0.2	0.94 ±0.01	0.59 ±0.00	0.59 ±0.00	1.00 ±0.00	1 ±0.76	0 ±0.00
		MP-SAE	64 ±0.0	**0.98** ±0.00	**0.86** ±0.01	0.41 ±0.01	0.61 ±0.00	4809 ±26	0 ±0.00
		iSAE	64 ±0.0	0.93 ±0.00	0.77 ±0.00	0.43 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	75 ±0.3	0.91 ±0.00	0.83 ±0.00	0.73 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
	$\ell_0 \approx 128$	ReLU	130 ±0	0.92 ±0.00	0.80 ±0.00	0.68 ±0.00	0.96 ±0.00	490 ±2	0 ±0.00
		TopK	128 ±0	0.96 ±0.01	0.61 ±0.00	0.56 ±0.00	1.00 ±0.00	19 ±1.3	0 ±0.00
		BatchTopK	129 ±1	0.96 ±0.01	0.63 ±0.00	0.60 ±0.00	1.00 ±0.00	19 ±4.2	0 ±0.00
		Matryoshka	128 ±0	0.95 ±0.01	0.54 ±0.00	0.57 ±0.00	1.00 ±0.00	0 ±0.50	0 ±0.00
		MP-SAE	127 ±0	**0.99** ±0.00	**0.86** ±0.01	0.41 ±0.01	0.61 ±0.00	4809 ±26	0 ±0.00
		iSAE	128 ±0	0.95 ±0.00	0.70 ±0.00	0.43 ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00
		ConSDL	135 ±0	0.92 ±0.00	0.83 ±0.00	**0.74** ±0.00	1.00 ±0.00	0 ±0.00	0 ±0.00

Table 17: **Hyperparameters for sparsity matching and ConSDL training.** For each backbone, expansion, and target sparsity level ($\ell_0 \approx 32, 64, 128$), we report the ReLU ℓ_1 coefficient and ConSDL sparse-coding penalty λ used for sparsity matching. For ConSDL, we additionally report the number of FISTA iterations T_{FISTA}, coefficient-refitting steps T_{refit}, and the co-active similarity weight γ_{sim}.

Backbone	**Expansion**	**ReLU**			**ConSDL**											
		ℓ_1			λ			T_{FISTA}			T_{refit}			γ_{sim}		
		32	64	128	32	64	128	32	64	128	32	64	128	32	64	128
CLIP ViT-B/16	×4	1.4e-4	1.0e-4	7.5e-5	0.8	0.6	0.35	30	30	30	10	5	5	1e-3	1e-3	1e-3
	×8	1.5e-4	1.2e-4	8.4e-5	0.8	0.8	0.5	30	30	30	60	10	5	5e-3	1e-3	1e-3
	×16	1.6e-4	1.3e-4	1.0e-4	1	1.25	0.75	50	30	30	50	10	10	5e-3	5e-3	2e-3
DINOv2-B/14	×4	1.5e-4	8.5e-5	5.6e-5	1.76	1.08	0.74	30	30	30	10	10	10	5e-3	5e-3	1e-3
	×8	1.5e-4	1.2e-4	6.8e-5	2	1.65	0.88	50	30	30	30	10	10	5e-3	5e-3	1e-3
	×16	1.5e-4	1.2e-4	8.5e-5	2.8	2.5	1.6	50	30	30	40	22	5	5e-3	5e-3	5e-3
Pythia-160M	×4	1.2e-4	8.0e-5	4.5e-5	1.5	1.2	1.2	30	30	30	13	8	4	1e-3	5e-3	5e-3
	×8	1.2e-4	8.0e-5	5.0e-5	1.5	1.5	1.5	50	50	30	22	15	8	1e-3	5e-3	5e-3
	×16	1.3e-4	8.0e-5	5.0e-5	1.7	1.5	1	50	50	50	50	26	14	1e-3	5e-3	5e-3
GPT-2 Small	×4	1.2e-4	7.0e-5	4.5e-5	7.1	4.6	2.95	30	30	30	15	5	5	5e-3	5e-3	5e-3
	×8	1.2e-4	7.5e-5	5.0e-5	8.38	5.35	3.4	30	30	30	15	10	5	5e-3	5e-3	5e-3
	×16	1.3e-4	8.0e-5	5.0e-5	12.5	6.5	4.5	30	30	30	30	18	10	5e-3	5e-3	2e-3

Table 18: **Hyperparameters for the synthetic experiments.** For each ground-truth dictionary size and tail exponent β, we report the ReLU sparsity coefficient ℓ_1 and the ConSDL sparse-coding penalty λ and number of refitting steps T_{refit}. ConSDL uses $T_{\text{FISTA}} = 50$, and $\gamma_{\text{sim}} =$ `1e-3` in all settings.

β	**3K**			**6K**			**9K**			**12K**		
	ReLU ℓ_1	Ours λ	Ours T_{refit}	ReLU ℓ_1	Ours λ	Ours T_{refit}	ReLU ℓ_1	Ours λ	Ours T_{refit}	ReLU ℓ_1	Ours λ	Ours T_{refit}
0.0	1.5e-4	0.3	30	1.4e-4	0.5	10	1.8e-4	0.4	10	1.8e-4	0.5	5
0.5	1e-4	0.3	30	1.2e-4	0.4	10	1.8e-4	0.3	10	1.6e-4	0.4	10
0.7	1.5e-4	0.35	20	2.1e-4	0.25	30	1.8e-4	0.35	30	2e-4	0.3	10
1.0	2.1e-4	0.15	30	2.3e-4	0.15	20	2.1e-4	0.25	30	2.1e-4	0.3	20