DIMINISHING RETURNS OF DATA IN SPARSE AUTOENCODERS: FUNCTIONAL BEHAVIOR DOES NOT GUARANTEE FEATURE REPRODUCIBILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse autoencoders (SAEs) are typically trained on hundreds of millions to billions of activation tokens, yet little work has tested whether this scale is necessary. We study how much activation data SAEs need by training four SAE architectures on three language models with budgets ranging from 5M tokens to a 500M-token reference. We measure two notions of convergence: functional behavior on standard evaluations and one-to-one reproducibility of learned features. Functional data requirements are strongly metric-dependent: reconstruction often approaches the full-data reference at substantially smaller budgets, whereas feature reproducibility continues to improve with data. For Gemma-2-2B BatchTopK, the fraction of strictly matched features rises from 2% at 5M tokens to 29% at 50M. Data selection also matters: random sampling better preserves the original activation distribution and generally favors reconstruction, whereas diversity-based sampling frequently improves AutoInterp. A theoretical analysis separates statistical consistency of subsampled training from the stronger identifiability and optimization conditions required for one-to-one feature recovery. Our results show that SAE training can often be made far more data-efficient when budgets are matched to the goal, and feature-level claims are backed by explicit reproducibility checks, paving the way toward cheaper and more reliable SAE research.

1 INTRODUCTION

Sparse autoencoders (SAEs) have become a widely used tool for mechanistic interpretability, aiming to decompose dense model activations into sparse, interpretable features Bricken et al. (2023); Cunningham et al. (2024); Sharkey et al. (2025). A central motivation is *polysemanticity*: individual neurons can respond to multiple unrelated concepts, making neuron-level interpretation difficult (Olah et al., 2020; Elhage et al., 2022). SAEs address this by learning an encoder that maps activations to sparse latent features and a decoder that reconstructs original activations (Olshausen & Field, 1996). Recent work has scaled SAEs to frontier models such as GPT-4 (Gao et al., 2025) and Claude 3 Sonnet (Templeton et al., 2024), motivating training on increasingly **large collections of activations**.

This raises a basic question: *how much activation data does an SAE actually need?* We study this *unexplored* question in an online training setting by varying activation exposure and measuring two notions of convergence: *functional behavior* asks how reconstruction and downstream SAE metrics change with activation budget, while *feature convergence* asks when the learned features themselves become reproducible under one-to-one matching. To clarify the distinction, we develop a theoretical framework of when additional data will lead to feature recovery and show that reducing the amount of activation data alone is not sufficient to guarantee dictionary convergence. This distinction matters because a central motivation for SAEs is not merely good reconstruction but reliable feature discovery. Prior work has questioned whether independently trained SAEs recover the same latent structure (Paulo & Belrose, 2026; Gerasimov et al., 2026; Lan et al., 2024), with Paulo & Belrose (2026) using Hungarian matching (Kuhn, 1955) and encoder-decoder agreement to quantify feature correspondence across dictionaries.

Our work suggests that large activation budgets are not universally required for high-quality SAE behavior. Our main contributions are: 1) We derive a theoretical characterization of feature recov-

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.

ery under data reduction, and separate statistical consistency from the additional requirements of feature identifiability and successful optimization; 2) across three language models, four SAE architectures, multiple configurations, and activation budgets we present a systematic study of online SAE data requirements; 3) we demonstrate a distinction between functional behavior and feature convergence, where functional SAE metrics exhibit distinct data-scaling behavior, while one-to-one feature reproducibility continues to improve substantially with additional activation exposure; and 4) we show that data selection changes SAE quality in metric-dependent ways: random sampling better preserves reconstruction and intervention-level metrics, while diversity-based sampling shifts training toward high-norm activations and frequently improves Autointerp Paulo et al. (2025).

To the best of our knowledge, our findings provide the **first systematic evidence** that SAE training can be **substantially more** data-efficient than current large-scale practice. For reconstruction and several functional evaluations, small activation budgets (5M-token) recover much of the behavior of a 500M-token SAE. In contrast, reproducible feature recovery becomes increasingly reliable with larger activation budgets, highlighting SAE's data requirements depend on the evaluation objective.

2 Related Work

Sparse autoencoders and evaluation. Since the standard SAE (Bricken et al., 2023; Cunningham et al., 2024), several architectural variants have been proposed, including JumpReLU (Rajamanoharan et al., 2024), TopK and BatchTopK (Gao et al., 2025; Bussmann et al., 2024), and Matryoshka SAEs (Bussmann et al., 2025). Evaluation has likewise moved beyond reconstruction loss to feature absorption (Chanin et al., 2025), sparse probing (Gurnee et al., 2023; Gao et al., 2025), unlearning (Farrell et al., 2024), and automated interpretability (Bills et al., 2023; Paulo et al., 2025), which Karvonen et al. (2025) compiled into SAEBench, now the standard benchmark for SAE evaluation.

Feature stability. A separate line of work asks whether the features SAEs recover are stable. Paulo & Belrose (2026) find that SAEs trained on the same data with different seeds learn substantially different features, casting doubt on whether any single SAE identifies a canonical set. Yet Lan et al. (2024) show that the spaces these features span remain similar, even across language models. Gerasimov et al. (2026) reconcile these findings: Seed-unstable features carry little reconstruction or predictive signal and cluster in reproducible low-rank subspaces, suggesting basis ambiguity rather than noise. We complement this work by asking how feature reproducibility depends on the amount of activation data.

Data selection. Outside interpretability, data selection is a well-studied route to efficiency: coreset methods seek a small subset that preserves the training signal of the full dataset (Moser et al., 2025), and Coleman et al. (2020) show that cheap proxy models can select such subsets an order of magnitude faster without hurting final accuracy. A common diversity-based strategy is farthest point sampling, which greedily adds the point farthest from the current selection (Gonzalez, 1985; Eldar et al., 1997) and has been applied to core-set selection for deep networks (Sener & Savarese, 2018). Closest to our setting, Yang et al. (2025b) use SAE features to prune instruction-tuning data, but they select the language model's training data rather than the SAE's. We instead ask whether the choice of activations matters beyond their quantity.

Research gap: Standard practice trains SAEs on hundreds of millions of tokens (Karvonen et al., 2025; Lieberum et al., 2024). To our knowledge, no prior work systematically tests whether this scale is necessary, or whether a selected subset of activations can match SAEs trained on far larger activation budgets on established metrics.

3 When Should Feature Matching Agree?

Let $\widehat{\theta}_{1,N}$ denote an SAE trained on the full activation sample and $\widehat{\theta}_{q,N}$ denote an SAE trained on a subsample of value-independent random activations with fraction $q \in (0, 1)$. Both runs use the same SAE training objective. Let $R(\theta) = \mathbb{E}_{X\sim P_X}[\ell_\theta(X)]$ denote the corresponding population objective, where $\ell_\theta(X)$ represents the training loss the SAE computes on a single activation X, using parameters θ, and $\widehat{R}_{1,N}(\theta) = \frac{1}{N}\sum_{t=1}^{N} \ell_\theta(X_t)$ denote full sample empirical risk. We define the indices retained by value-independent random sampling using $S_{q,N} \subseteq \{1, \ldots, N\}$ and the subsample empirical risk using $\widehat{R}_{q,N}(\theta) = \frac{1}{|S_{q,N}|}\sum_{t\in S_{q,N}} \ell_\theta(X_t)$. Finally, $E(\theta) = [e_1(\theta), \ldots, e_p(\theta)]$ and

$D(\theta) = [d_1(\theta), \ldots, d_p(\theta)]$ denote the encoder- and decoder-side feature system, respectively. We assume that features are normalized before cosine-based comparison. Let $\mathcal{S}_p$ denote the set of $p \times p$ permutation matrices, so that for $P \in \mathcal{S}_p$, $\bar{E}(\theta')P$ is $\bar{E}(\theta')$ with its columns reordered. For two SAE parameter settings θ and θ', define the encoder and decoder one-to-one distances as

$$d_E(\theta, \theta') = \min_{P \in \mathcal{S}_p} \left\| \bar{E}(\theta) - \bar{E}(\theta')P \right\|_F, \tag{1}$$

$$d_D(\theta, \theta') = \min_{P \in \mathcal{S}_p} \left\| \bar{D}(\theta) - \bar{D}(\theta')P \right\|_F, \tag{2}$$

where $\| \cdot \|_F$ is the Frobenius norm, which is exactly zero when the two systems have the same feature directions.

Assumptions. We require four conditions: *(i) Common population objective:* full- and reduced-data runs use the same architecture, preprocessing, sparsity mechanism, loss weights, and population weighting; *(ii) Statistical consistency:* both empirical objectives converge uniformly to the same population risk; *(iii) Optimization consistency:* training approaches the empirical optimum as the number of observed activations increases; and *(iv) Feature identifiability:* all population minimizers recover the same encoder and decoder feature systems up to a common latent permutation, with positive separation from alternative feature systems.

Theorem 1 (Asymptotic encoder and decoder feature alignment)**.** *Under the assumptions above, for every fixed $q > 0$,*

$$d_E(\widehat{\theta}_{q,N}, \widehat{\theta}_{1,N}) \xrightarrow{p} 0, \tag{3}$$

$$d_D(\widehat{\theta}_{q,N}, \widehat{\theta}_{1,N}) \xrightarrow{p} 0. \tag{4}$$

Equivalently,

$$\min_{P_E \in \mathcal{S}_p} \left\| \bar{E}_{q,N} - \bar{E}_{1,N} P_E \right\|_F \xrightarrow{p} 0, \tag{5}$$

$$\min_{P_D \in \mathcal{S}_p} \left\| \bar{D}_{q,N} - \bar{D}_{1,N} P_D \right\|_F \xrightarrow{p} 0. \tag{6}$$

*Proof sketch.*Under assumptions (i)–(ii), the full- and reduced-data empirical risks converge uniformly to R. By (iii), both trained SAEs approach the minimizers of R, and by (iv) those minimizers share a single feature system up to permutation. The result follows by the triangle inequality.

Corollary 1 (Agreement of encoder and decoder matchings)**.** *Let $P_E^{(N)}$ and $P_D^{(N)}$ denote the permutations that maximize total cosine similarity between the two SAEs' encoder and decoder directions, respectively:*

$$P_E^{(N)} = \arg\max_{P \in \mathcal{S}_p} \sum_{j=1}^{p} \cos\big(e_{q,N,j},\, e_{1,N,P(j)}\big), \tag{7}$$

$$P_D^{(N)} = \arg\max_{P \in \mathcal{S}_p} \sum_{j=1}^{p} \cos\big(d_{q,N,j},\, d_{1,N,P(j)}\big). \tag{8}$$

Suppose that at the population optimum the encoder and decoder directions of each feature share the same index, and that the optimal one-to-one matching is unique for both encoder and decoder cosine sums, exceeding every other permutation by a fixed margin $\delta > 0$. Then, under the assumptions of Theorem 1,

$$\Pr\big(P_E^{(N)} = P_D^{(N)}\big) \longrightarrow 1 \qquad \text{as } N \to \infty.$$

Corollary 2 (Assignment agreement is weaker than strong feature recovery)**.** *Encoder–decoder assignment agreement does not imply high encoder and decoder cosine similarity at a finite data budget. In particular,*

$$P_E^{(N)}(j) = P_D^{(N)}(j) = k$$

only states that feature k is independently selected as the optimal one-to-one counterpart under both encoder and decoder matching. It does not imply $\cos(e_{q,N,j}, e_{1,N,k}) > \tau$ or $\cos(d_{q,N,j}, d_{1,N,k}) > \tau$ for a fixed threshold τ. Thus, substantial agreement between encoder- and decoder-based Hungarian assignments can coexist with a much smaller fraction of strongly aligned features.

Interpretation. Theorem 1 characterizes conditions under which reducing activation data preserves the asymptotic feature system. It also clarifies that, at finite budgets, assignment agreement is weaker than strong directional recovery: encoder and decoder matchings may select the same reference counterpart even when their cosine similarities remain moderate. This distinction motivates our empirical comparison of counterpart agreement and strict shared-feature recovery in Sec. 6.

4 Experimental Setup

We study SAE data requirements along two axes: *activation quantity* and *activation selection*. Unless stated otherwise, we follow the corresponding SAEBench (Karvonen et al., 2025) training and evaluation configurations so that small-data runs are directly comparable to the large-data reference.

Models and SAE Configurations. We evaluate three large language models (LLMs) - Gemma-2-2B (Gemma Team, 2024) (layer 12), Pythia-160M (Biderman et al., 2023) (layer 6), and Qwen3-8B (Yang et al., 2025a)(layer 17), and four SAE architectures: BatchTopK (Bussmann et al., 2024), Matryoshka BatchTopK (Bussmann et al., 2025), JumpReLU (Rajamanoharan et al., 2024), and ReLU(Bricken et al., 2023; Cunningham et al., 2024), across multiple configurations. Within each comparison, we hold the LLM, activation site, dictionary width, sparsity setting, and optimization hyperparameters fixed, varying only the activation budget or selection strategy.

Online SAE Training. SAEs are trained on FineWeb (Penedo et al., 2024) in an *online* activation-generation setting Marks et al. (2024). Text is streamed through a frozen LLM, target-layer activations are buffered, and SAE mini-batches are drawn from the selected activations before the buffer is refreshed with new text. Selected activations are used once, avoiding a fixed precomputed dataset. Across budgets, we fix the 500M training configuration and optimization steps Marks et al. (2024), varying only the retained activation fraction to isolate data scale from training exposure.

Training-Data Budgets. The reference configuration is designed for 500M activation tokens using Marks et al. (2024) framework. For reduced-data experiments, we retain a fraction $q \in \{0.01, 0.05, 0.10, 1.0\}$ of the activations generated at each buffer refresh. Thus, each fresh activation pool contributes 1%, 5%, 10%, or 100% of its activations to SAE training. The selected activations are consumed once before a new activation pool is generated from the online data stream. Therefore, q controls the amount of fresh activation data retained from each online buffer, rather than the size of a fixed replay dataset.

Data sampling. We reduce the activation set using two sampling strategies: random sampling and farthest-point sampling (FPS), which explicitly promotes geometric diversity.

Evaluation. We evaluate all SAEs using SAEBench (Karvonen et al., 2025) default configurations and metrics, including reconstruction, sparse probing (Gurnee et al., 2023; Gao et al., 2025), absorption (Chanin et al., 2025), SCR (Marks et al., 2025), TPP (Karvonen et al., 2025), RAVEL (Huang et al., 2024), automated interpretability (Bills et al., 2023; Paulo et al., 2025), and unlearning (Farrell et al., 2024). We measure feature reproducibility against the 500M reference using one-to-one Hungarian matching of encoder and decoder features (Paulo & Belrose, 2026; Kuhn, 1955). Matching similarity and encoder-decoder assignment agreement distinguish *feature convergence* from *functional convergence*

5 Functional Data Requirements Are Metric Dependent

Here, we show how much activation data is required to reproduce the functional behavior of a large-data SAE. We compare SAEs trained on randomly sampled activation budgets of 5M, 25M, and 50M tokens against corresponding 500M-token references. We focus on two representative configurations: a BatchTopK SAE with 16k latents and a standard ReLU SAE with 65k latents trained on layer 12 residual-stream activations of Gemma-2-2B. Figs. 1-2 report the corresponding SAEBench metrics across activation budgets. Results for additional SAE architectures, dictionary widths, and sparsities are provided in Appendix A.

a. Reconstruction. The loss measures how well the SAE reconstruction preserves the model's next-token prediction behavior relative to the original model and a zero-ablation baseline. *Observation: The information required for reconstruction is learned early.* In Figs. 1-2, we find that all reduced-



Figure 1: **Functional data scaling for a BatchTopK SAE with 16k latents.** SAEBench metrics are reported for SAEs trained with {5M, 25M, 50M, 500M} activation tokens. While smaller activation budgets approaches the 500M reference for reconstruction, probing, AutoInterp, and intervention-metrics (e, h) exhibit heterogeneous, non-monotonic dependence on activation exposure.



Figure 2: **Functional data scaling for a standard ReLU SAE with 65k latents.** SAEBench metrics are reported for SAEs trained with {5M, 25M, 50M, 500M} activation tokens. Reconstruction remains close to the 500M reference at smaller activation budgets, whereas feature-level metrics show metric-dependent and frequently non-monotonic responses to increasing activation exposure.

data SAEs remain close to the 500M reference. While the 5M setting shows only a modest gap, 25M and 50M SAEs are generally very close to the reference, highlighting that substantially less than 500M activation tokens are sufficient to retain the LLM's behavior captured by SAE reconstruction.

b. k-Sparse Probing (SP). It asks whether a pre-specified concept can be recovered from a small number of SAE latents. Following SAEBench, we report $k = 1$, so the metric specifically probes

whether concept information is concentrated in an individual latent. *Observation: Concept localization is largely preserved at smaller data budgets.* Across sparsity settings and dictionary widths, reduced-data SAEs remain close to the 500M reference. Sparse-probing accuracy is broadly consistent across 5M-500M budgets, with smaller budgets sometimes outperforming the 500M reference, showing that concept localization is well preserved with substantially less data.

c. Feature Absorption (FA). It measures a complementary failure mode: whether a concept that should be represented by its main feature is instead partially captured by other latents. We report $1-$ mean absorption; higher is better. *Observation: More data does not systematically reduce concept fragmentation.* FA is more sensitive to data budget than SP, yet reduced-data SAEs frequently outperform the 500M reference. Together, SP and FA show that strong concept localization and feature alignment can be achieved with substantially smaller data budgets.

d. Automated Interpretability. AutoInterp asks an LLM to infer a semantic description of a latent from its activating examples and evaluates whether that description predicts activation on held-out sequences, probing the semantic description of individual features rather than reconstruction quality. *Observation: Feature description is not monotonically determined by activation quantity.* AutoInterp varies moderately with activation budget, but its ordering depends on SAE configuration, including sparsity and dictionary width. Reduced-data SAEs outperform the 500M reference in many regimes and underperform it in others. In several settings, the trend is even reversed, with smaller activation budgets yielding higher AutoInterp scores. Thus, increasing activation exposure **does not** consistently improve the semantic description of individual features.

e. Spurious Correlation Removal (SCR). It evaluates whether ablating a small set of concept-related latents removes a spurious correlation while preserving the intended classification signal. *Observation: SCR shows moderate but non-monotonic sensitivity to data.* SCR varies more with activation budget than reconstruction, but increasing data does not consistently improve intervention specificity, and the performance of reduced-data SAEs depends on the width and the architecture.

f. Targeted Probe Perturbation (TPP). It asks whether ablating latents associated with one class primarily affects that class while leaving others unchanged. *Observation: TPP is remarkably insensitive to activation budget* Across both dictionary sizes, TPP remains close to the 500M reference despite the large difference in activation exposure. The ordering of data budgets is non-monotonic, but the absolute deviations are small, suggesting that this measure of class-specific feature separation requires relatively little additional data.

g. RAVEL. It evaluates whether an intervention successfully changes a targeted attribute while preserving unrelated attributes. *Observation: RAVEL is non-monotonic across activation budgets.* It varies with activation budget and architecture, indicating that effective intervention can be achieved without requiring consistently larger activation exposure.

h. Unlearning. It suppresses latents associated with a target domain and measures whether target knowledge can be removed while unrelated performance is preserved. *Observation: Unlearning is the most configuration-sensitive evaluation* It varies substantially with activation budget, architecture, and sparsity, highlighting the importance of these factors for unlearning performance. Reduced-data SAEs span a wide performance range relative to the 500M reference, highlighting that effective unlearning can emerge even with substantially smaller activation budgets. To strengthen all the observations above, we additonally add a paired statistical test in Table 26.

In total, these results raise a natural next question: if all other hyperparameters are held constant, are the observed volatile differences primarily a consequence of using less data, or do they also depend on ***which*** activations are randomly selected? To separate data quantity from data selection, we next compare random sampling with diversity-based activation selection at matched budgets.

Does Data Selection Matter? Here, we ask whether SAE data requirements are determined only by *how much* data is used, or also by *which* activations are selected. To separate these effects, we compare uniform random sampling (RS) with Farthest Point Sampling (FPS) (Gonzalez, 1985; Eldar et al., 1997) at matched activation budgets. While RS retains activations independently of their values and therefore preserves the frequency structure of the activation stream in expectation, FPS selects activations to increase geometric coverage, preferentially retaining points that are far from those already selected. We follow the evaluation configuration of Figs. 1-2 and compare both selection strategies. Results for additional SAE architectures are reported in Appendix A.

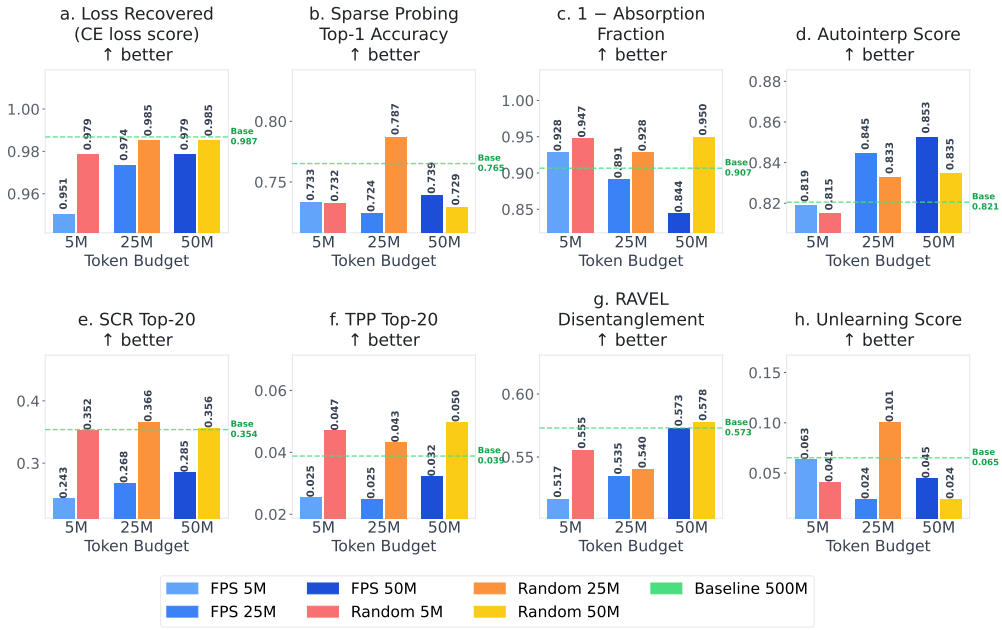


Figure 3: **Random sampling (RS) vs. FPS for a BatchTopK SAE with 16k latents.** At each activation budget, paired bars compare FPS and RS, while the green dashed line denotes the corresponding 500M reference. Selection strategy induces metric-dependent tradeoffs: RS better preserves reconstruction and several intervention metrics, whereas FPS often improves AutoInterp.

Fig. 3 shows that the choice of activation subset matters even at a fixed budget: RS and FPS do not produce uniformly better or worse SAEs, but instead favor different properties of the learned representation. ***i) Random sampling preserves reconstruction.*** Across dictionary sizes and budgets, loss recovered under RS stays closer to the 500M reference than under FPS, most visibly at small budgets. Geometric coverage alone thus preserves the model's predictive behavior less faithfully than representative sampling. ***ii) Concept- and intervention-level metrics favor RS.*** SCR, TPP, and RAVEL also favor RS across budgets, in some cases exceeding the 500M reference, while SP and FA show no consistent winner. ***iii) FPS yields higher AutoInterp.*** FPS outperforms RS on AutoInterp at every budget in both configurations, and in 79% of matched comparisons across all architectures, models, and budgets. FPS shifts training toward geometrically distinct, high-norm activations and is associated with higher AutoInterp but lower reconstruction scores. FPS also outperforms RS on unlearning at some budgets (5M and 50M in Fig. 3), though less consistently than for AutoInterp.

Data Quality Is Objective Dependent. The matched-budget comparison shows that reducing SAE training data is not characterized by a single notion of data quality. RS better preserves several properties of the full activation stream, most clearly reconstruction, while FPS can improve metrics like AutoInterp and, in some settings of unlearning. Thus, **which activations are selected matters in addition to how many are selected.** A subset that is representative of the original activation distribution and a subset that maximizes geometric diversity optimize *different* notions of SAE quality.

Why does FPS produce this tradeoff? To characterize what each selector retains, we compared the L2-norm distribution of residual-stream activations selected for SAE training by RS vs. FPS, with all selectors applied to identical activation pools drawn from the 500M-token stream. RS reproduces the stream's distribution within sampling error at every budget, so its effect is purely a reduction in data quantity. FPS instead concentrates training on a higher-norm region of activation space, shifting the distribution further right as the subset shrinks from 10% to 1% of 500M reference (see Table 1).

6 Functional Similarity Does Not Imply Feature Reproducibility

So far, increasing activation budget does not uniformly improve SAE function: reconstruction often approaches the 500M reference early, while other metrics remain heterogeneous and non-monotonic. We now ask a stricter question: *does functional similarity imply recovery of the same features?*

Table 1: **FPS reshapes the training-data distribution; random selection does not.*** Random selection is distribution-neutral at every budget (KL ≤ 0.0003 bits); FPS-induced divergence grows with subset selectivity. Results for Qwen3-8B and Pythia-160M are in Table 24 in the appendix.

			Selected vs. 500M stream		
Model	**Selection**	**Coreset**	**Δ mean norm (%)**	**KL (bits)**	**Mean-dir. cosine**
Gemma-2-2B	FPS	5M	$+28.8 \pm 0.9$	2.28	0.941 ± 0.003
		25M	$+22.7 \pm 0.3$	1.51	0.954 ± 0.001
		50M	$+18.9 \pm 0.2$	1.11	0.961 ± 0.001
	Random	5M	0.0 ± 0.4	0.000	1.000 ± 0.000
		25M	0.0 ± 0.2	0.000	1.000 ± 0.000
		50M	0.0 ± 0.1	0.000	1.000 ± 0.000

*Δ *mean norm* is the relative shift in mean activation L_2 norm; *KL* is the base-2 divergence between binned norm distributions; and *Mean-dir. cosine* measures mean-direction alignment. All selectors use identical activation pools. Values are mean ± SD across pool refreshes (50 for norm statistics; 15 for direction statistics); values shown as ± 0.000 have SD $< 10^{-4}$.

Table 2: **Counterpart agreement and strict feature correspondence.** *Enc.-dec. agree* is the fraction of features for which independently computed encoder and decoder Hungarian matchings select the same 500M reference latent. *Shared* additionally requires both encoder and decoder cosine similarities to exceed $\tau = 0.7$. We find a similar observation for both Qwen3-8b and Pythia-160m, shown in Table 25 in the appendix.

Model	**SAE**	**Budget**	**Counterpart (%)**	**Shared, $\tau \geq 0.7$ (%)**
Gemma-2-2B	ReLU	5M	76.15	28.42
		25M	83.06	41.13
		50M	**85.28**	**47.26**
	BatchTopK	5M	66.61	2.20
		25M	80.15	16.94
		50M	**83.47**	**28.86**

Following prior work on SAE feature reproducibility Paulo & Belrose (2026), we compare reduced-data SAEs against the corresponding 500M-token reference using one-to-one Hungarian matching. Encoder and decoder features are matched independently by cosine similarity. A latent is *shared* at the threshold τ if both matchings select the same reference latent and both cosine similarities exceed τ. Unless otherwise stated, we use $\tau = 0.7$ as a strict one-to-one feature-recovery criterion.

Counterpart Agreement Emerges Before Strong Feature Recovery. Even at small activation budgets, reduced-data SAEs are not learning completely unrelated dictionaries. For Gemma-2-2B BatchTopK at 5M tokens, the mean matched decoder cosine similarity is 0.454, and independently computed encoder and decoder matchings identify the same reference counterpart for 66.6% of latents. However, only 2.20% of latents satisfy the stricter shared-feature criterion at $\tau = 0.7$.

This gap distinguishes two different notions of correspondence. *Counterpart agreement* asks whether the encoder and decoder identify the same reference feature, whereas *strong feature recovery* additionally requires the corresponding feature directions themselves to be highly aligned. Reduced-data SAEs can therefore recover recognizable counterparts well before those counterparts become strongly reproducible.

Feature Reproducibility Improves Systematically with Data. Table 2 shows that feature recovery strengthens consistently with activation budget. For Gemma-2-2B BatchTopK, encoder-decoder agreement rises from 66.61% at 5M to 83.47% at 50M, while strict shared-feature recovery at $\tau = 0.7$ increases from 2.20% to 28.86%. The same trend holds for ReLU SAEs, showing that counterpart agreement emerges earlier than strong one-to-one feature recovery. Additional results are in Table 25 in the appendix.

Qualitative Examples Illustrate the Same Distinction. Table 3 presents representative Gemma-2-2B ReLU examples spanning low, medium, and high mean encoder-decoder alignment. For each 5M-500M SAE pair, Hungarian matching aligns features using encoder-decoder cosine similarity, while Gemma-based AutoInterp provides balanced-accuracy scores. We retain pairs for which the encoder- and decoder-based matches agree, and both AutoInterp scores are positive and select one representative pair from each of the alignment bins for qualitative analysis. At low and

Table 3: Representative matched Gemma-2-2B ReLU features between the 5M SAE and the 500M reference across increasing mean encoder-decoder alignment. The matched latents are (2782 ↔ 2782), (411 ↔ 2629),(1507 ↔ 1507).

Regime	Mean alignment	5M SAE feature	500M matched feature
low	0.0–0.2	The word *average* and surrounding mathematical or statistical expressions	Mathematical formatting, LaTeX code, variable subscripts, or mathematical expressions
Medium	0.4–0.6	Words related to medical topics, particularly cancer, diseases, and research	Mathematical expressions, code snippets, formatting symbols, or sequential lists
high	0.8–1.0	Comparative adjectives and adverbs, often followed by *than* or *as*	Comparative adjectives and adverbs, often followed by words such as *than* or *more*

medium alignment, matched features can have substantially different semantic descriptions despite both matching criteria identifying them as corresponding. In contrast, the highly aligned example captures essentially the same semantic pattern across the two SAEs: *Comparative adjectives and adverbs, of- ten followed by than*. This qualitative behavior is consistent with the quantitative results above. Counterpart identity alone is a weak notion of feature recovery; strongly aligned matched pairs provide a much stricter indication that the same feature has been recovered across data budgets.

Functional Similarity Does Not Imply Feature Reproducibility. By 25M–50M tokens, reconstruction and several functional SAE metrics are already close to their 500M reference values in many regimes, yet feature recovery remains incomplete. Table 2 shows that increasing activation exposure consistently strengthens one-to-one feature correspondence across models and SAE architectures, even when functional metrics do not improve monotonically with data. Thus, functional similarity can emerge at smaller budgets, while feature reproducibility continues to improve with additional activation exposure, making them distinct notions of SAE convergence.

7 Conclusion

In this work, we ask a simple but practically important question: *How much activation data does an SAE actually need?* We study this across four SAE architectures and three language models, analyzing data requirements along three axes: theory, functional performance, and feature reproducibility. First, we develop a theoretical framework characterizing when reduced-data and full-data SAEs should recover the same feature system, showing that convergence to the same population objective is not sufficient for one-to-one feature recovery; identifiability and optimization conditions are also required. Second, training SAEs on 1%, 5%, 10%, and 100% of the available online activation data, we find that many SAEBench metrics approach the large-data reference well before the full budget. Third, feature-level analysis reveals a different regime: encoder- and decoder-based Hungarian matchings agree for a substantial fraction of features even at small budgets, but stricter one-to-one similarity criteria show continued improvement with more data. Counterpart agreement thus emerges earlier than strong feature reproducibility.

Together, these results suggest that SAE data requirements are objective-dependent. Smaller activation budgets can often preserve reconstruction and aggregate functional behavior, while reproducible recovery of individual features may require substantially more data. We therefore recommend matching SAE training scale to the intended use: avoid defaulting to the largest available activation corpus for functional evaluation, but explicitly assess feature reproducibility when individual latents are used as scientific objects.

Limitations. Our study relies heavily on the training and evaluation implementations provided by SAEBench. All functional metrics are computed using the corresponding SAEBench evaluation scripts, and our SAE training pipeline follows the same underlying framework. Consequently, our conclusions inherit any assumptions, implementation choices, and metric-specific limitations of these tools. Additionally, the current SAEBench evaluation framework supports RAVEL and unlearning evaluations only for Gemma models. Consequently, we are currently unable to evaluate the Qwen and Pythia models using these metrics.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, and interpret results. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, or support qualitative and thematic data analysis. Generating synthetic data sets, assisting with translation, and cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft parts of a research paper, brainstorm, search for information, and edit a research paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artefacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All training hyperparameters are detailed in Section 4 and Appendix A. The proof of our theorem is provided in Appendix B. The code will be released upon acceptance.

REFERENCES

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2304.01373.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. OpenAI, 2023. URL https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

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. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK sparse autoencoders. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. In *International Conference on Machine Learning (ICML)*, 2025.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1906.11829.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.08600.

Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, et al. Toy models of superposition. Transformer Circuits Thread, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. Applying sparse autoencoders to unlearn knowledge in language models. *arXiv preprint arXiv:2410.19278*, 2024.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.04093.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Gleb Gerasimov, Timofei Rusalev, Nikita Balagansky, Daniil Laptev, Vadim Kurochkin, and Daniil Gavrilov. Unstable features, reproducible subspaces: Understanding seed dependence in sparse autoencoders. *arXiv preprint arXiv:2606.12138*, 2026.

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research (TMLR)*, 2023.

Jing Huang, Zhengxuan Wu, Christopher Potts, Mor Geva, and Atticus Geiger. RAVEL: Evaluating interpretability methods on disentangling language model representations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *International Conference on Machine Learning (ICML)*, 2025.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

Michael Lan, Philip Torr, Austin Meek, Ashkan Khakzar, David Krueger, and Fazl Barez. Quantifying feature space universality across large language models via sparse autoencoders. *arXiv preprint arXiv:2410.06981*, 2024.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open sparse autoencoders everywhere all at once on Gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

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

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2403.19647.

Brian B. Moser, Arundhati S. Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. A coreset selection of coreset selection literature: Introduction and recent advances. *arXiv preprint arXiv:2505.17799*, 2025.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. In *International Conference on Learning Representations (ICLR)*, 2026.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024. arXiv:2406.17557.

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.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. Transformer Circuits Thread, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xianjun Yang, Shaoliang Nie, Lijuan Liu, Suchin Gururangan, Ujjwal Karn, Rui Hou, Madian Khabsa, and Yuning Mao. Diversity-driven data selection for language model tuning through sparse autoencoder. *arXiv preprint arXiv:2502.14050*, 2025b.

A EXPERIMENTAL DETAILS AND ADDITIONAL FUNCTIONAL DATA-SCALING RESULTS

We evaluate four SAE architectures: standard ReLU, BatchTopK, JumpReLU, and Matryoshka BatchTopK. Across all configurations, SAE batch size 2048, dictionary widths 16,384 and 65,536, 1000 warmup steps, and 5000 sparsity-warmup steps. Decay begins after 80% of training, TopK-style k-annealing ends after 1% of the total training steps, BOS tokens are removed, and activations exceeding $10\times$ the reference activation norm are filtered. The default full-data training budget is 500M activation tokens. For activation extraction, Pythia-160M uses LM batch size 32, context length 1024, and `float32`; Gemma-2-2B uses batch size 4, context length 1024, and `bfloat16`; and Qwen3-8B uses batch size 16, context length 2048, and `bfloat16`. For Gemma-2-2B and Pythia-160M, we evaluate both 16K and 65K dictionary widths. For Qwen3-8B, we use only the 65K width across all SAE architectures.

The trainer index used in our evaluation tables identifies the architecture-specific sparsity operating point. We use the L_1 sparsity penalty over$\{0.012,\ 0.015,\ 0.020,\ 0.030,\ 0.040,\ 0.060\}$. At dictionary width 16K, trainers 0-5 correspond to these values in order; at width 64K, trainers 6-11 follow the same ordering.

For BatchTopK, we use $k \in \{80, 160\}$. Trainers 0 and 1 correspond to $k = 80$ and $k = 160$ at width 16K, while trainers 2 and 3 correspond to the same two operating points at width 64K. The trainer uses `auxk_alpha` $= 1/32$, `threshold_beta` $= 0.999$, and begins threshold tracking after 1000 steps.

For JumpReLU, we use the target activation count $L_0^{\text{target}} \in \{80, 160\}$. Trainers 0 and 1 correspond to target $L_0 = 80$ and 160 at width 16K, while trainers 2 and 3 correspond to the same targets at width 64K. The sparsity penalty is fixed to 1.0 and the JumpReLU bandwidth to 0.001.

For Matryoshka BatchTopK, we use $k \in \{80, 160\}$. Trainers 0 and 1 correspond to $k = 80$ and 160 at width 16K, and trainers 2 and 3 correspond to the same values at width 64K. The nested groups use fractions $\left\{\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{17}{32}\right\}$, with `auxk_alpha` $= 1/32$, `threshold_beta` $= 0.999$, and threshold tracking beginning after 1000 steps.

Table 4: **SAE evaluation across activation budgets and selection strategies for Gemma-2-2B, ReLU (16k). Random and FPS are compared at 5M,25M, and 50M activation budgets against the corresponding 500M baseline.**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
0	5M	Random	0.993	0.959	0.703	0.253	0.029	0.756	**0.657**	0.006	1400.2
		FPS	0.985	0.974	0.703	0.194	0.021	0.750	0.639	**0.009**	1298.8
	25M	Random	**0.995**	0.974	0.704	0.261	**0.031**	0.787	0.653	0.002	1353.2
		FPS	0.992	**0.983**	**0.718**	0.223	0.022	0.802	0.634	0.004	1265.1
	50M	Random	**0.995**	0.979	0.715	0.264	0.030	0.797	0.656	**0.009**	1346.5
		FPS	0.993	0.955	0.712	0.230	0.024	0.772	0.625	0.004	**1204.0**
	500M	Baseline	**0.995**	0.947	0.712	**0.268**	0.030	**0.805**	0.641	**0.009**	1289.5
1	5M	Random	0.990	0.950	0.696	0.253	0.025	0.791	0.632	0.006	997.1
		FPS	0.979	**0.971**	0.705	0.191	0.016	0.750	0.622	0.002	907.3
	25M	Random	0.992	0.968	0.713	**0.275**	0.028	0.775	0.622	0.024	962.3
		FPS	0.988	0.955	0.720	0.216	0.024	0.791	0.613	**0.034**	882.4
	50M	Random	0.990	0.946	0.731	0.262	0.029	0.805	0.628	0.011	960.8
		FPS	0.990	0.921	**0.736**	0.237	0.025	0.805	0.614	0.004	**840.4**
	500M	Baseline	**0.993**	0.942	0.734	0.261	**0.030**	**0.808**	**0.637**	0.008	914.0
2	5M	Random	0.982	**0.963**	0.735	0.253	0.024	**0.807**	0.570	**0.015**	600.4
		FPS	0.970	0.954	0.736	0.197	0.012	0.769	0.565	0.006	534.2
	25M	Random	0.988	0.904	0.735	0.263	0.024	0.793	0.596	0.002	590.1
		FPS	0.980	0.870	**0.756**	0.212	0.021	0.800	0.561	0.006	526.8
	50M	Random	0.988	0.925	0.745	0.272	0.025	0.799	**0.597**	0.011	590.0
		FPS	0.984	0.842	0.747	0.239	0.020	0.796	0.587	0.009	**501.0**
	500M	Baseline	**0.990**	0.890	0.749	**0.277**	**0.027**	0.805	0.594	**0.015**	563.8
3	5M	Random	0.975	0.876	0.770	0.248	0.018	0.793	0.543	0.004	269.5
		FPS	0.952	**0.884**	0.774	0.174	0.010	0.746	0.514	**0.024**	**230.7**
	25M	Random	0.982	0.840	0.778	0.259	0.020	0.782	0.548	0.006	275.9
		FPS	0.969	0.749	0.786	0.188	0.014	0.788	0.549	0.009	242.6
	50M	Random	0.979	0.799	0.766	**0.283**	**0.023**	**0.803**	0.522	0.017	279.0
		FPS	0.972	0.762	**0.794**	0.206	0.016	0.791	**0.557**	0.008	234.9
	500M	Baseline	**0.984**	0.791	0.785	0.283	0.021	0.796	0.554	0.023	271.4
4	5M	Random	0.969	**0.827**	0.800	0.221	0.015	0.763	0.489	0.015	157.7
		FPS	0.934	0.815	0.807	0.145	0.008	0.768	0.475	**0.073**	**133.4**
	25M	Random	0.975	0.769	0.801	0.238	0.015	0.763	0.501	0.032	163.3
		FPS	0.957	0.732	**0.812**	0.176	0.009	0.769	0.488	0.023	143.8
	50M	Random	0.974	0.773	0.807	0.237	0.016	0.772	0.503	0.023	165.9
		FPS	0.962	0.719	0.811	0.194	0.012	0.785	0.476	0.049	141.1
	500M	Baseline	**0.977**	0.746	0.802	**0.273**	**0.017**	**0.800**	**0.521**	0.045	162.6
5	5M	Random	0.956	0.835	0.829	0.197	0.009	0.761	0.374	0.026	83.4
		FPS	0.906	**0.844**	0.816	0.111	0.007	0.759	0.419	0.041	**71.1**
	25M	Random	0.959	0.771	0.829	0.190	0.011	0.762	0.434	0.038	84.0
		FPS	0.939	0.736	0.826	0.141	0.006	0.744	0.457	**0.143**	75.0
	50M	Random	**0.964**	0.763	0.826	0.189	**0.013**	0.764	0.411	0.028	85.4
		FPS	0.946	0.738	0.844	0.155	0.008	0.731	0.447	0.139	73.0
	500M	Baseline	**0.964**	0.726	**0.849**	**0.207**	0.010	**0.806**	**0.484**	0.041	84.0

Table 5: **Per-trainer SAE evaluation — Gemma-2-2B, ReLU (64K).** Each row is one trainer run. Missing evaluations shown as "–".

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
6	5M	Random	0.995	**0.902**	0.648	0.191	0.011	0.775	**0.690**	0.024	1700.5
		FPS	0.988	0.894	0.663	0.125	0.008	0.792	0.664	0.013	**1439.5**
	25M	Random	0.995	0.865	0.669	**0.192**	**0.012**	0.811	0.685	0.011	1932.7
		FPS	0.993	0.758	0.695	0.155	0.009	0.805	0.665	0.009	1686.4
	50M	Random	**0.997**	0.819	0.673	0.183	0.012	**0.813**	0.685	0.021	1987.0
		FPS	0.995	0.729	**0.702**	0.159	0.009	0.812	0.681	**0.051**	1678.4
	500M	Baseline	**0.997**	0.788	0.689	0.184	0.012	0.807	0.683	0.006	1923.7
7	5M	Random	0.993	**0.881**	0.659	**0.196**	0.012	0.792	0.661	0.019	1176.4
		FPS	0.982	0.865	0.672	0.146	0.010	0.792	0.646	0.017	**976.9**
	25M	Random	0.993	0.838	0.686	0.190	0.011	0.796	0.660	0.009	1340.8
		FPS	0.990	0.695	0.698	0.160	0.009	0.803	0.644	0.008	1149.1
	50M	Random	**0.995**	0.791	0.692	0.193	**0.013**	**0.812**	**0.681**	0.009	1381.5
		FPS	0.992	0.673	0.704	0.159	0.010	0.791	0.643	**0.045**	1145.6
	500M	Baseline	**0.995**	0.721	**0.706**	0.180	0.011	0.806	0.642	0.006	1332.1
8	5M	Random	0.990	**0.838**	0.701	**0.211**	0.013	0.793	0.637	0.023	685.9
		FPS	0.977	0.764	0.712	0.139	0.007	0.793	0.603	0.024	**553.8**
	25M	Random	**0.992**	0.743	0.702	0.201	0.013	0.797	0.633	0.041	793.7
		FPS	0.985	0.621	**0.737**	0.157	0.009	0.803	0.619	0.019	665.0
	50M	Random	**0.992**	0.716	0.728	0.191	0.013	**0.813**	0.645	0.013	825.4
		FPS	0.988	0.607	0.730	0.157	0.011	0.803	0.605	**0.122**	669.6
	500M	Baseline	**0.992**	0.700	0.723	0.180	**0.014**	0.812	**0.646**	0.036	803.6
9	5M	Random	0.985	**0.707**	0.751	0.185	**0.013**	0.797	0.591	0.041	304.1
		FPS	0.964	0.668	0.760	0.121	0.006	0.782	0.576	**0.225**	**239.0**
	25M	Random	**0.988**	0.667	0.746	**0.189**	0.010	0.800	0.597	0.028	355.6
		FPS	0.977	0.554	0.766	0.138	0.006	0.758	0.582	0.118	298.7
	50M	Random	**0.988**	0.594	0.755	0.186	0.010	0.815	0.600	0.186	375.2
		FPS	0.980	0.549	**0.782**	0.153	0.007	0.762	**0.615**	0.201	306.6
	500M	Baseline	**0.988**	0.564	0.766	0.184	0.012	**0.819**	0.594	0.167	377.8
10	5M	Random	0.979	**0.637**	0.792	**0.185**	0.010	0.806	0.573	0.135	181.3
		FPS	0.957	0.586	0.801	0.129	0.006	0.781	0.558	0.086	**145.6**
	25M	Random	0.984	0.578	0.792	0.178	0.008	**0.809**	0.590	0.184	205.0
		FPS	0.969	0.515	**0.810**	0.136	0.007	0.767	0.571	0.156	176.0
	50M	Random	0.985	0.547	0.805	0.169	0.009	0.804	**0.595**	**0.206**	216.3
		FPS	0.975	0.494	0.799	0.142	0.006	0.771	0.573	0.128	179.1
	500M	Baseline	**0.987**	0.497	0.798	0.182	**0.010**	0.808	0.580	0.188	219.9
11	5M	Random	0.969	0.588	0.830	0.144	0.006	0.792	0.534	0.154	100.8
		FPS	0.939	**0.613**	0.841	0.097	0.004	0.755	0.522	0.137	**82.6**
	25M	Random	0.974	0.510	0.837	**0.168**	0.006	0.781	0.553	**0.169**	103.4
		FPS	0.957	0.502	0.838	0.108	0.005	0.766	0.550	0.092	89.9
	50M	Random	0.975	0.502	**0.846**	0.156	0.007	**0.794**	0.567	0.137	106.7
		FPS	0.962	0.520	0.844	0.110	0.005	0.751	0.547	0.088	89.6
	500M	Baseline	**0.979**	0.482	0.844	0.146	**0.007**	0.776	**0.573**	0.030	107.2

Table 6: **Per-trainer SAE evaluation for Gemma-2-2B, BatchTopK (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
0	5M	Random	0.979	0.947	0.815	0.352	0.047	0.732	0.555	0.041	79.8
		FPS	0.951	0.928	0.819	0.243	0.025	0.733	0.517	0.063	**50.9**
	25M	Random	0.985	0.928	0.833	**0.366**	0.043	**0.787**	0.540	**0.101**	80.6
		FPS	0.974	0.891	0.845	0.268	0.025	0.724	0.535	0.024	55.2
	50M	Random	0.985	**0.950**	0.835	0.356	**0.050**	0.729	**0.578**	0.024	79.7
		FPS	0.979	0.844	**0.853**	0.285	0.032	0.739	0.573	0.045	57.6
	500M	Baseline	**0.987**	0.907	0.821	0.354	0.039	0.765	0.573	0.065	80.2
1	5M	Random	0.985	0.930	0.758	**0.345**	0.096	0.735	0.587	0.032	158.7
		FPS	0.964	0.929	0.771	0.279	0.080	0.714	0.554	0.028	**102.2**
	25M	Random	0.990	**0.986**	0.779	0.308	0.129	0.757	0.603	0.006	160.8
		FPS	0.982	0.953	**0.815**	0.293	0.090	**0.765**	0.588	**0.067**	109.8
	50M	Random	0.990	0.981	0.778	0.344	0.087	0.760	0.600	0.019	160.0
		FPS	0.985	0.947	0.811	0.180	**0.165**	0.701	0.601	0.019	115.7
	500M	Baseline	**0.992**	0.975	0.800	0.328	0.087	0.704	**0.619**	0.058	159.9

Table 7: **Per-trainer SAE evaluation for Gemma-2-2B, BatchTopK (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
2	5M	Random	0.980	0.899	0.785	0.330	**0.052**	**0.793**	0.556	0.048	79.6
		FPS	0.954	**0.900**	0.848	0.255	0.022	0.769	0.536	0.095	**53.4**
	25M	Random	0.987	0.896	0.826	0.327	0.038	0.789	**0.589**	0.056	80.7
		FPS	0.977	0.666	**0.863**	0.231	0.028	0.721	0.585	0.102	57.4
	50M	Random	0.987	0.883	0.838	**0.339**	0.028	0.739	0.566	0.102	80.6
		FPS	0.984	0.660	0.857	0.242	0.019	0.738	0.576	0.091	59.9
	500M	Baseline	**0.990**	0.750	0.831	0.261	0.019	0.731	0.584	**0.168**	80.8
3	5M	Random	0.987	0.905	0.757	**0.337**	0.104	0.747	0.599	0.015	160.1
		FPS	0.969	0.886	0.780	0.229	0.084	0.739	0.573	0.037	**104.9**
	25M	Random	0.990	0.933	0.774	0.294	**0.153**	0.733	**0.633**	0.037	161.0
		FPS	0.985	0.802	0.798	0.236	0.091	**0.753**	0.598	0.082	114.2
	50M	Random	**0.992**	**0.963**	0.781	0.243	0.113	0.717	0.616	0.041	160.5
		FPS	0.988	0.821	**0.811**	0.239	0.058	0.701	0.595	**0.088**	119.5
	500M	Baseline	**0.992**	0.923	0.792	0.202	0.060	0.721	0.619	0.084	161.8

Table 8: **Per-trainer SAE evaluation for Gemma-2-2B, JumpReLU (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
0	5M	Random	0.977	0.910	0.825	0.365	0.047	0.736	0.523	0.015	83.4
		FPS	0.949	0.910	0.828	0.232	0.034	0.695	0.511	**0.094**	**54.3**
	25M	Random	0.982	0.895	0.811	**0.383**	0.043	0.715	**0.577**	0.043	83.0
		FPS	0.972	0.842	0.844	0.290	0.029	0.691	0.542	0.079	57.3
	50M	Random	0.985	**0.914**	0.813	0.374	**0.053**	0.733	0.573	0.030	82.7
		FPS	0.979	0.802	**0.849**	0.277	0.031	0.729	0.567	0.026	60.3
	500M	Baseline	**0.987**	0.865	0.822	0.325	0.045	**0.748**	0.572	0.090	83.0
1	5M	Random	0.982	0.923	0.769	0.344	**0.124**	**0.766**	0.570	0.034	167.7
		FPS	0.962	0.918	0.771	0.242	0.054	0.743	0.534	0.009	**107.6**
	25M	Random	0.987	0.952	0.766	**0.389**	0.099	0.736	**0.622**	0.011	166.9
		FPS	0.982	0.949	0.783	0.298	0.097	0.733	0.561	0.015	114.5
	50M	Random	0.990	**0.958**	0.761	0.342	0.083	0.734	0.590	0.019	166.6
		FPS	0.985	0.901	**0.789**	0.305	0.046	0.744	0.579	**0.069**	120.4
	500M	Baseline	**0.992**	0.952	0.775	0.325	0.108	0.729	0.604	0.019	166.7

Table 9: **Per-trainer SAE evaluation for Gemma-2-2B, JumpReLU (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
2	5M	Random	0.977	**0.873**	0.809	**0.369**	**0.044**	0.733	0.561	0.069	83.5
		FPS	0.956	0.757	0.835	0.196	0.014	0.694	0.539	0.071	**55.5**
	25M	Random	0.987	0.827	0.835	0.328	0.029	**0.765**	0.567	0.081	82.9
		FPS	0.975	0.717	**0.867**	0.246	0.027	0.725	0.581	0.092	59.0
	50M	Random	0.987	0.834	0.825	0.308	0.039	0.737	0.572	0.084	82.8
		FPS	0.984	0.663	0.852	0.264	0.020	0.741	0.582	**0.221**	62.0
	500M	Baseline	**0.990**	0.729	0.813	0.255	0.022	0.718	**0.582**	0.133	83.0
3	5M	Random	0.985	0.895	0.765	**0.379**	0.054	0.734	0.573	0.043	167.5
		FPS	0.965	0.898	0.764	0.256	0.047	0.703	0.581	0.056	**110.8**
	25M	Random	0.990	0.919	0.764	0.327	0.053	**0.784**	0.617	0.047	166.8
		FPS	0.985	0.787	0.780	0.298	0.035	0.706	0.605	0.109	117.2
	50M	Random	**0.992**	**0.937**	0.776	0.343	**0.096**	0.728	0.609	0.099	166.7
		FPS	0.988	0.805	**0.799**	0.272	0.057	0.699	0.590	0.103	123.2
	500M	Baseline	**0.992**	0.872	0.755	0.231	0.053	0.709	**0.630**	**0.308**	167.4

Table 10: **Per-trainer SAE evaluation for Gemma-2-2B, Matryoshka (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
0	5M	Random	0.979	0.948	0.845	0.404	0.107	0.755	0.561	0.028	79.5
		FPS	0.941	0.967	0.863	0.295	0.053	0.724	0.526	0.015	**49.1**
	25M	Random	0.984	0.966	0.844	0.455	0.104	0.743	0.549	0.006	80.0
		FPS	0.969	0.953	**0.884**	0.340	0.092	0.760	0.545	**0.090**	54.4
	50M	Random	0.984	0.982	0.851	**0.460**	**0.130**	0.776	0.562	0.041	79.5
		FPS	0.975	0.975	0.863	0.373	0.106	0.734	**0.569**	0.056	57.4
	500M	Baseline	**0.985**	**0.993**	0.849	0.459	0.130	**0.806**	0.560	0.034	80.1
1	5M	Random	0.985	0.963	0.830	0.335	0.177	0.776	0.581	0.028	159.3
		FPS	0.956	0.992	0.838	0.259	0.116	0.719	0.554	0.039	**97.1**
	25M	Random	0.988	0.988	0.840	0.422	0.187	0.770	**0.616**	0.017	160.1
		FPS	0.980	**0.998**	**0.862**	0.342	0.142	0.732	0.594	0.039	106.8
	50M	Random	0.988	0.976	0.831	**0.459**	**0.206**	0.757	0.594	0.024	159.7
		FPS	0.985	0.998	0.855	0.370	0.147	0.723	0.570	**0.075**	112.1
	500M	Baseline	**0.990**	0.978	0.827	0.442	0.204	**0.785**	0.610	0.011	159.3

Table 11: **Per-trainer SAE evaluation for Gemma-2-2B, Matryoshka (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Unlearning ↑	Sparsity (L0) ↓
2	5M	Random	0.975	0.925	0.772	0.399	0.086	0.756	0.543	0.026	79.5
		FPS	0.949	0.967	0.873	0.266	0.074	0.760	0.540	**0.086**	**50.6**
	25M	Random	0.985	**0.992**	0.832	0.393	**0.106**	**0.783**	0.602	0.017	80.1
		FPS	0.975	0.958	**0.885**	0.339	0.048	0.756	**0.612**	0.038	55.1
	50M	Random	**0.987**	0.974	0.829	0.428	0.101	0.767	0.610	0.043	79.9
		FPS	0.980	0.961	0.881	0.325	0.086	0.772	0.581	0.079	57.9
	500M	Baseline	**0.987**	0.980	0.820	**0.433**	0.091	0.782	0.607	0.062	79.6
3	5M	Random	0.984	0.949	0.785	0.368	0.130	**0.781**	0.591	0.047	159.1
		FPS	0.967	**0.987**	0.843	0.268	0.119	0.744	0.554	0.039	**99.7**
	25M	Random	0.990	0.946	0.813	0.361	**0.167**	0.775	0.602	0.058	160.7
		FPS	0.984	0.956	**0.863**	**0.373**	0.106	0.743	0.607	0.049	109.7
	50M	Random	0.990	0.975	0.823	0.355	0.161	0.750	0.619	0.038	159.7
		FPS	0.987	0.973	0.856	0.311	0.112	0.774	0.586	**0.092**	116.7
	500M	Baseline	**0.992**	0.986	0.817	0.352	0.151	0.753	**0.624**	0.051	160.2

Table 12: **Per-trainer SAE evaluation for Pythia-160M, ReLU (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
0	5M	Random	0.975	**0.944**	0.676	0.123	0.035	0.725	0.499	1208.2
		FPS	0.955	0.942	0.680	0.105	0.021	**0.735**	0.499	**1000.0**
	25M	Random	**0.980**	0.907	0.688	0.142	0.033	0.712	0.500	1200.3
		FPS	0.960	0.942	0.689	0.120	0.029	0.727	0.501	1074.3
	50M	Random	0.978	0.913	0.697	**0.146**	0.034	0.719	0.501	1180.7
		FPS	0.962	0.918	0.702	0.127	0.030	0.724	**0.502**	1117.9
	500M	Baseline	0.969	0.915	**0.713**	0.143	**0.037**	0.716	0.499	1144.5
1	5M	Random	0.967	0.916	0.677	0.120	0.031	0.724	0.497	949.5
		FPS	0.949	**0.916**	0.700	0.110	0.020	0.733	0.497	**767.8**
	25M	Random	**0.975**	0.834	0.687	0.142	**0.032**	0.732	0.498	936.1
		FPS	0.954	0.892	0.707	0.116	0.026	**0.740**	**0.501**	825.0
	50M	Random	0.972	0.904	0.701	**0.144**	0.029	0.718	0.498	918.9
		FPS	0.958	0.893	0.706	0.129	0.026	0.730	0.498	858.3
	500M	Baseline	0.966	0.864	**0.715**	0.135	0.032	0.710	0.498	879.2
2	5M	Random	0.960	**0.900**	0.679	0.130	0.031	0.729	0.496	668.3
		FPS	0.933	0.882	0.702	0.110	0.018	0.734	0.495	**525.0**
	25M	Random	0.965	0.866	0.698	**0.143**	0.031	0.740	0.497	652.6
		FPS	0.953	0.842	0.723	0.113	0.024	**0.746**	0.498	565.3
	50M	Random	**0.968**	0.828	0.711	0.138	0.030	0.704	0.496	637.9
		FPS	0.951	0.802	0.721	0.128	0.024	0.727	0.498	587.7
	500M	Baseline	0.960	0.782	**0.724**	0.134	**0.031**	0.715	**0.499**	602.9
3	5M	Random	0.949	0.806	0.690	0.124	0.032	0.718	0.496	387.4
		FPS	0.906	**0.811**	0.704	0.103	0.018	0.723	0.491	**301.6**
	25M	Random	0.951	0.794	0.709	0.122	0.033	0.723	**0.496**	376.7
		FPS	0.929	0.682	0.724	0.111	0.023	**0.745**	0.494	319.9
	50M	Random	0.953	0.758	0.713	0.122	**0.034**	0.724	0.494	367.6
		FPS	0.942	0.642	**0.738**	0.114	0.025	0.724	0.492	332.0
	500M	Baseline	**0.953**	0.674	0.732	**0.126**	0.031	0.723	0.494	345.9
4	5M	Random	0.933	**0.793**	0.708	0.106	**0.040**	0.717	**0.494**	268.9
		FPS	0.893	0.726	0.711	0.086	0.020	0.717	0.489	**213.1**
	25M	Random	**0.937**	0.715	0.716	0.119	0.036	0.720	0.491	260.1
		FPS	0.914	0.601	0.725	0.096	0.023	0.731	0.490	221.5
	50M	Random	0.936	0.697	0.722	**0.121**	0.040	0.717	0.492	255.5
		FPS	0.925	0.560	0.735	0.096	0.024	0.718	0.491	227.8
	500M	Baseline	0.935	0.615	**0.741**	0.106	0.035	**0.738**	0.494	241.1
5	5M	Random	0.908	0.700	0.703	0.081	0.046	0.742	0.491	179.3
		FPS	0.853	**0.703**	0.696	0.075	0.022	0.721	0.491	**148.8**
	25M	Random	**0.913**	0.634	0.702	0.083	**0.052**	0.726	**0.492**	172.6
		FPS	0.887	0.550	0.720	0.083	0.035	0.727	0.490	151.2
	50M	Random	0.910	0.629	0.715	0.083	0.050	**0.750**	0.488	169.7
		FPS	0.893	0.520	0.714	0.080	0.036	0.714	0.489	154.4
	500M	Baseline	0.910	0.560	**0.727**	**0.102**	0.046	0.727	0.489	162.8

Table 13: **Per-trainer SAE evaluation for Pythia-160M, ReLU (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
6	5M	Random	0.972	**0.892**	0.659	0.099	0.012	0.719	0.499	1861.4
		FPS	0.944	0.878	0.666	0.076	0.008	0.719	0.495	**1371.2**
	25M	Random	0.976	0.800	0.671	0.088	0.014	0.714	0.500	2002.8
		FPS	0.964	0.784	0.679	0.090	0.010	0.706	0.499	1646.3
	50M	Random	**0.977**	0.809	0.671	0.093	0.014	0.730	0.499	2012.6
		FPS	0.967	0.728	0.674	0.083	0.012	0.718	0.498	1758.5
	500M	Baseline	0.968	0.659	**0.683**	**0.100**	**0.016**	**0.735**	**0.500**	1997.5
7	5M	Random	0.972	**0.866**	0.663	0.091	0.014	0.708	0.497	1472.9
		FPS	0.939	0.798	0.661	0.078	0.009	0.696	0.495	**1076.9**
	25M	Random	0.974	0.748	0.675	**0.094**	0.012	0.709	0.497	1569.1
		FPS	0.960	0.698	**0.695**	0.084	0.010	**0.735**	**0.498**	1274.0
	50M	Random	**0.976**	0.731	0.673	0.094	0.014	0.721	0.497	1570.6
		FPS	0.965	0.645	0.686	0.088	0.011	0.711	0.498	1359.9
	500M	Baseline	0.967	0.629	0.672	0.090	**0.016**	0.730	0.497	1550.1
8	5M	Random	0.964	**0.810**	0.660	0.085	0.014	0.734	0.497	1079.1
		FPS	0.936	0.740	0.669	0.079	0.009	0.711	0.494	**794.2**
	25M	Random	0.969	0.662	0.679	0.087	0.012	0.708	0.496	1126.1
		FPS	0.954	0.587	**0.700**	0.083	0.012	**0.749**	**0.497**	908.2
	50M	Random	**0.970**	0.632	0.676	**0.090**	0.014	0.712	0.497	1123.6
		FPS	0.960	0.506	0.693	0.076	0.012	0.747	0.494	962.8
	500M	Baseline	0.965	0.530	0.686	0.088	**0.017**	0.745	0.496	1095.8
9	5M	Random	0.951	**0.676**	0.668	0.079	0.013	0.730	**0.496**	710.5
		FPS	0.916	0.599	0.675	0.066	0.011	0.714	0.493	**551.5**
	25M	Random	0.957	0.555	0.691	**0.085**	0.017	0.729	0.494	720.4
		FPS	0.939	0.390	0.691	0.078	0.013	0.738	0.493	590.7
	50M	Random	**0.957**	0.510	0.684	0.083	0.017	0.741	0.494	713.1
		FPS	0.947	0.381	0.689	0.069	0.015	0.752	0.493	613.3
	500M	Baseline	0.953	0.408	**0.699**	0.075	**0.020**	**0.761**	0.494	685.2
10	5M	Random	0.940	**0.585**	0.680	0.069	0.019	0.740	0.493	559.0
		FPS	0.900	0.522	0.681	0.061	0.013	0.724	0.492	**456.1**
	25M	Random	0.946	0.432	0.678	**0.076**	0.019	0.725	**0.494**	557.4
		FPS	0.924	0.340	0.686	0.065	0.015	0.744	0.493	475.4
	50M	Random	**0.946**	0.402	0.699	0.075	0.019	0.742	0.492	551.4
		FPS	0.937	0.302	0.688	0.061	0.018	0.742	0.492	485.9
	500M	Baseline	0.941	0.355	**0.705**	0.065	**0.024**	**0.747**	0.491	524.9
11	5M	Random	0.921	**0.517**	0.703	0.052	0.024	0.732	**0.492**	448.5
		FPS	0.868	0.443	0.692	0.045	0.015	0.700	0.491	**382.8**
	25M	Random	0.928	0.376	0.693	**0.058**	0.028	**0.739**	0.489	437.4
		FPS	0.904	0.325	0.686	0.048	0.021	0.735	0.490	392.9
	50M	Random	**0.930**	0.349	0.701	0.048	0.027	0.726	0.489	429.8
		FPS	0.916	0.289	0.700	0.050	0.029	0.725	0.491	396.2
	500M	Baseline	0.919	0.288	**0.717**	0.048	**0.032**	0.731	0.491	407.8

Table 14: **Per-trainer SAE evaluation for Pythia-160M, BatchTopK (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
0	5M	Random	0.946	0.902	0.733	0.189	**0.245**	0.707	0.492	142.6
		FPS	0.916	0.845	0.744	**0.216**	0.163	**0.707**	0.490	**106.3**
	25M	Random	0.954	0.911	0.755	0.183	0.222	0.704	0.492	137.5
		FPS	0.940	0.852	0.776	0.196	0.163	0.697	0.489	108.2
	50M	Random	0.953	**0.914**	0.774	0.027	0.224	0.697	0.491	135.3
		FPS	0.949	0.840	0.768	0.175	0.183	0.698	0.488	118.1
	500M	Baseline	**0.961**	0.875	**0.802**	-0.184	0.202	0.695	**0.494**	125.0
1	5M	Random	0.950	0.860	0.732	-0.186	0.193	**0.728**	**0.492**	216.7
		FPS	0.930	0.820	0.750	**-0.071**	**0.219**	0.711	0.489	167.8
	25M	Random	0.958	**0.948**	0.751	-0.216	0.194	0.720	0.492	211.7
		FPS	0.950	0.915	0.759	-0.154	0.185	0.725	0.491	**167.1**
	50M	Random	0.966	0.943	0.769	-0.153	0.181	0.718	0.492	210.3
		FPS	0.961	0.912	0.773	-0.409	0.199	0.725	0.490	174.5
	500M	Baseline	**0.970**	0.911	**0.807**	-0.308	0.168	0.722	0.490	204.5

Table 15: **Per-trainer SAE evaluation for Pythia-160M, BatchTopK (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
2	5M	Random	0.945	**0.886**	0.692	**0.229**	0.248	0.708	0.492	352.4
		FPS	0.916	0.836	0.703	0.189	0.210	0.704	0.488	294.7
	25M	Random	0.961	0.862	0.717	0.200	**0.260**	0.697	**0.492**	330.6
		FPS	0.946	0.769	0.763	0.196	0.215	**0.710**	0.483	276.4
	50M	Random	0.961	0.857	0.736	-0.084	0.235	0.709	0.490	315.7
		FPS	0.952	0.635	0.746	0.180	0.145	0.691	0.488	282.9
	500M	Baseline	**0.961**	0.782	**0.783**	-0.061	0.224	0.684	0.490	**267.7**
3	5M	Random	0.952	0.852	0.695	**0.040**	**0.213**	0.740	**0.494**	424.9
		FPS	0.934	0.877	0.705	0.038	0.180	0.731	0.491	346.4
	25M	Random	0.966	0.898	0.721	-0.174	0.180	0.730	0.487	393.6
		FPS	0.950	0.811	0.760	-0.077	0.203	0.738	0.487	**321.6**
	50M	Random	0.965	**0.903**	0.751	-0.155	0.139	0.726	0.494	373.7
		FPS	0.958	0.752	0.767	-0.243	0.192	**0.747**	0.481	331.3
	500M	Baseline	**0.970**	0.894	**0.778**	-0.271	0.156	0.721	0.488	362.0

Table 16: **Per-trainer SAE evaluation for Pythia-160M, JumpReLU (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
0	5M	Random	0.946	0.857	0.732	0.240	0.229	0.696	0.491	139.5
		FPS	0.909	0.809	0.747	0.131	0.210	0.691	0.491	**106.7**
	25M	Random	0.955	0.907	0.742	**0.246**	0.213	0.686	0.486	141.6
		FPS	0.942	0.832	0.763	0.161	0.175	0.686	0.490	111.9
	50M	Random	**0.957**	**0.908**	0.753	0.208	**0.246**	0.695	0.492	141.5
		FPS	0.948	0.828	**0.766**	0.214	0.186	0.680	0.488	115.6
	500M	Baseline	0.956	0.889	0.762	0.204	0.199	**0.711**	**0.493**	138.0
1	5M	Random	0.958	0.870	0.711	0.097	0.240	0.713	0.491	222.7
		FPS	0.931	0.832	0.713	0.156	0.264	0.716	0.490	**170.1**
	25M	Random	0.961	0.903	0.725	**0.253**	0.245	0.703	0.494	224.4
		FPS	0.956	0.847	0.746	0.116	0.254	0.702	0.492	177.0
	50M	Random	**0.963**	0.941	0.723	0.101	**0.266**	0.701	**0.498**	224.9
		FPS	0.956	0.887	0.751	-0.056	0.237	0.711	0.493	182.7
	500M	Baseline	0.960	**0.967**	**0.768**	-0.119	0.249	**0.729**	0.497	216.5

Table 17: **Per-trainer SAE evaluation for Pythia-160M, JumpReLU (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
2	5M	Random	0.950	**0.833**	0.703	0.147	0.211	0.698	0.494	297.4
		FPS	0.909	0.739	0.714	0.125	0.165	0.665	0.490	**255.9**
	25M	Random	0.958	0.818	0.727	**0.244**	**0.238**	0.698	0.494	309.8
		FPS	0.944	0.654	0.733	0.163	0.168	**0.722**	0.487	266.2
	50M	Random	0.961	0.780	0.725	0.218	0.205	0.676	0.492	315.0
		FPS	0.957	0.635	0.747	0.207	0.184	0.690	**0.494**	272.5
	500M	Baseline	**0.963**	0.737	**0.752**	0.162	0.191	0.700	0.487	322.2
3	5M	Random	0.959	0.819	0.698	0.087	0.232	0.712	0.494	378.0
		FPS	0.931	0.800	0.691	**0.182**	0.260	0.709	0.490	**322.4**
	25M	Random	0.961	0.834	0.706	-0.085	0.213	0.726	0.489	397.8
		FPS	0.958	0.832	0.723	-0.066	0.265	**0.730**	**0.494**	344.5
	50M	Random	0.965	0.864	0.714	0.042	**0.276**	0.724	0.489	403.6
		FPS	0.961	0.815	0.725	-8.46e-04	0.238	0.717	0.494	355.7
	500M	Baseline	**0.968**	**0.882**	**0.739**	0.046	0.184	0.719	0.487	409.5

Table 18: **Per-trainer SAE evaluation for Pythia-160M, Matryoshka (16K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
0	5M	Random	0.934	0.902	0.758	0.136	0.244	**0.727**	0.486	132.5
		FPS	0.914	0.892	0.755	**0.182**	0.223	0.693	0.490	107.5
	25M	Random	**0.952**	0.907	0.791	-0.040	**0.277**	0.720	**0.493**	126.6
		FPS	0.931	0.938	0.795	-0.086	0.238	0.699	0.493	**101.8**
	50M	Random	0.949	0.890	0.803	-0.302	0.213	0.697	0.489	120.6
		FPS	0.941	**0.952**	0.800	0.037	0.232	0.714	0.490	107.5
	500M	Baseline	0.951	0.942	**0.819**	-0.187	0.232	0.702	0.487	117.6
1	5M	Random	0.942	0.701	0.740	-0.072	0.210	0.719	0.491	208.6
		FPS	0.934	0.812	0.759	-0.221	0.179	**0.729**	0.489	161.3
	25M	Random	0.961	0.792	0.789	-0.185	0.227	0.689	0.491	201.8
		FPS	0.946	**0.923**	0.790	-0.135	0.166	0.727	0.493	161.3
	50M	Random	0.961	0.862	0.788	-0.402	0.192	0.708	0.489	196.0
		FPS	0.953	0.889	0.796	**-0.067**	**0.234**	0.719	0.493	**158.9**
	500M	Baseline	**0.966**	0.904	**0.802**	-0.167	0.185	0.707	**0.493**	194.1

Table 19: **Per-trainer SAE evaluation for Pythia-160M, Matryoshka (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	RAVEL Disent. ↑	Sparsity (L0) ↓
2	5M	Random	0.948	0.891	0.687	0.120	0.214	**0.717**	0.490	340.3
		FPS	0.923	0.877	0.690	**0.209**	0.214	0.717	0.490	314.7
	25M	Random	0.954	0.939	0.712	-0.259	**0.251**	0.714	0.492	330.7
		FPS	0.945	0.941	0.734	-0.035	0.192	0.704	0.489	283.3
	50M	Random	0.954	**0.967**	0.744	-0.085	0.219	0.694	0.489	300.2
		FPS	0.951	0.942	0.747	0.011	0.173	0.710	0.489	**270.7**
	500M	Baseline	**0.958**	0.950	**0.763**	-0.154	0.228	0.706	**0.493**	279.1
3	5M	Random	0.953	0.843	0.694	**0.046**	0.175	0.725	**0.496**	404.8
		FPS	0.938	0.838	0.712	-0.338	0.186	0.728	0.492	343.2
	25M	Random	0.965	0.881	0.720	-0.179	0.190	**0.736**	0.488	389.4
		FPS	0.954	**0.960**	0.738	-0.026	0.193	0.724	0.494	**325.7**
	50M	Random	0.963	0.859	0.731	-0.478	0.181	0.723	0.493	378.3
		FPS	0.959	0.950	**0.740**	-0.107	**0.193**	0.729	0.490	334.1
	500M	Baseline	**0.966**	0.924	0.740	-0.386	0.180	0.722	0.490	379.4

Table 20: **Per-trainer SAE evaluation for Qwen3-8B, ReLU (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	Sparsity (L0) ↓
0	5M	Random	0.993	0.959	0.823	**0.178**	**0.155**	0.732	1337.0
		FPS	0.984	**0.970**	0.808	0.116	0.057	0.732	**1199.0**
	25M	Random	0.993	0.894	**0.833**	0.147	0.101	0.726	1835.8
		FPS	0.991	0.860	0.812	0.137	0.065	0.732	1638.7
	50M	Random	**0.995**	0.815	0.815	0.151	0.090	**0.780**	1944.7
		FPS	0.991	0.755	0.793	0.125	0.054	0.737	1703.2
	500M	Baseline	**0.995**	0.734	0.777	0.146	0.066	0.666	2003.9
1	5M	Random	0.987	**0.949**	0.817	**0.188**	**0.148**	0.700	878.7
		FPS	0.978	0.940	0.809	0.115	0.065	0.739	**778.5**
	25M	Random	0.991	0.859	**0.832**	0.149	0.133	0.716	1225.5
		FPS	0.987	0.737	0.813	0.127	0.073	**0.774**	1065.5
	50M	Random	**0.993**	0.778	0.830	0.157	0.129	0.769	1306.0
		FPS	0.989	0.693	0.809	0.132	0.084	0.759	1122.0
	500M	Baseline	**0.993**	0.699	0.797	0.155	0.101	0.673	1355.0
2	5M	Random	0.984	**0.892**	0.818	**0.164**	**0.183**	0.698	502.3
		FPS	0.973	0.869	0.812	0.128	0.089	0.733	**428.8**
	25M	Random	0.987	0.767	**0.839**	0.143	0.168	0.720	693.9
		FPS	0.984	0.628	0.825	0.132	0.096	0.728	587.6
	50M	Random	0.987	0.704	0.829	0.155	0.147	**0.768**	746.8
		FPS	0.984	0.561	0.833	0.142	0.106	0.742	628.2
	500M	Baseline	**0.989**	0.591	0.819	0.142	0.109	0.687	788.2
3	5M	Random	0.980	**0.767**	0.833	**0.161**	0.171	0.695	243.9
		FPS	0.962	0.677	0.836	0.121	0.088	0.745	**198.0**
	25M	Random	0.982	0.645	**0.843**	0.147	0.191	0.687	314.6
		FPS	0.977	0.453	0.843	0.113	0.111	**0.749**	260.0
	50M	Random	0.984	0.575	0.838	0.155	**0.211**	0.703	338.1
		FPS	0.978	0.433	0.837	0.124	0.125	0.742	282.3
	500M	Baseline	**0.986**	0.494	0.837	0.138	0.178	0.706	360.5
4	5M	Random	0.977	**0.668**	0.850	**0.151**	0.168	0.708	160.4
		FPS	0.953	0.564	0.846	0.112	0.083	0.730	**127.2**
	25M	Random	0.980	0.549	0.854	0.140	0.169	0.699	192.4
		FPS	0.973	0.393	0.854	0.121	0.131	0.706	160.3
	50M	Random	0.980	0.484	0.847	0.149	**0.197**	**0.749**	202.9
		FPS	0.975	0.392	**0.866**	0.122	0.146	0.733	172.1
	500M	Baseline	**0.984**	0.422	0.854	0.134	0.197	0.736	216.0
5	5M	Random	0.969	**0.599**	0.867	0.124	0.135	–	97.1
		FPS	0.937	0.519	0.869	0.108	0.079	0.715	**74.0**
	25M	Random	0.975	0.438	**0.872**	**0.147**	0.167	0.707	105.3
		FPS	0.962	0.373	0.869	0.093	0.155	0.709	87.4
	50M	Random	0.975	0.410	0.868	0.100	**0.216**	**0.720**	108.3
		FPS	0.968	0.346	0.871	0.082	0.189	0.714	91.7
	500M	Baseline	**0.977**	0.394	0.870	0.086	0.211	0.711	111.5

Table 21: **Per-trainer SAE evaluation for Qwen3-8B, BatchTopK (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	Sparsity (L0) ↓
2	5M	Random	0.984	0.974	0.852	**0.282**	0.303	**0.703**	85.6
		FPS	0.968	0.962	0.878	0.189	0.258	0.679	**63.4**
	25M	Random	0.986	0.951	0.867	0.264	0.303	0.692	86.1
		FPS	0.977	**0.983**	**0.882**	0.202	0.296	0.671	66.9
	50M	Random	0.986	0.944	0.867	0.258	0.289	0.692	85.8
		FPS	0.982	0.943	0.877	0.236	**0.313**	0.698	69.6
	500M	Baseline	**0.989**	0.880	0.866	0.180	0.247	0.669	87.2
3	5M	Random	0.987	0.947	0.854	**0.273**	**0.341**	**0.738**	170.0
		FPS	0.977	**0.999**	0.840	0.195	0.278	0.700	**127.7**
	25M	Random	0.989	0.957	**0.869**	0.210	0.315	0.692	172.1
		FPS	0.986	0.973	0.860	0.202	0.262	0.673	132.2
	50M	Random	**0.991**	0.979	0.854	0.212	0.290	0.683	171.5
		FPS	0.987	0.986	0.859	0.204	0.277	0.663	138.8
	500M	Baseline	**0.991**	0.871	0.838	-0.012	0.204	0.646	173.8

Table 22: **Per-trainer SAE evaluation for Qwen3-8B, JumpReLU (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	Sparsity (L0) ↓
2	5M	Random	0.980	**0.990**	0.857	**0.336**	0.226	0.652	87.4
		FPS	0.968	0.943	0.860	0.192	0.106	**0.700**	68.5
	25M	Random	0.984	0.932	0.869	0.270	**0.229**	0.664	87.3
		FPS	0.982	0.965	**0.884**	0.212	0.098	0.677	**67.7**
	50M	Random	0.986	0.962	0.876	0.257	0.198	0.671	87.4
		FPS	0.982	0.909	0.868	0.242	0.151	0.662	70.1
	500M	Baseline	**0.987**	0.833	0.867	0.206	0.132	0.672	88.3
3	5M	Random	0.986	0.971	0.846	**0.301**	**0.301**	**0.723**	174.7
		FPS	0.980	**0.999**	0.835	0.224	0.200	0.707	**134.1**
	25M	Random	0.989	0.946	0.849	0.279	0.224	0.691	176.0
		FPS	0.986	0.982	0.846	0.250	0.192	0.709	135.4
	50M	Random	0.989	0.962	0.842	0.231	0.289	0.671	177.3
		FPS	0.987	0.974	**0.854**	0.245	0.175	–	140.7
	500M	Baseline	**0.991**	0.941	0.823	0.158	0.116	0.624	179.2

Table 23: **Per-trainer SAE evaluation for Qwen3-8B, Matryoshka (64K).**

Trainer	Budget	Selection	Loss Recovered ↑	1−Abs. Fraction ↑	AutoInterp ↑	SCR Top-20 ↑	TPP Top-20 ↑	Sparse Probe Top-1 ↑	Sparsity (L0) ↓
2	5M	Random	0.980	0.982	0.848	0.290	**0.342**	0.662	85.3
		FPS	0.962	**0.998**	0.892	0.238	0.286	**0.726**	**58.3**
	25M	Random	0.984	0.990	0.863	0.268	0.325	0.708	85.5
		FPS	0.978	0.986	**0.894**	0.296	0.298	0.672	62.6
	50M	Random	**0.986**	0.986	0.857	0.365	0.311	0.691	85.9
		FPS	0.980	0.978	0.894	0.312	0.299	0.704	65.7
	500M	Baseline	0.984	0.974	0.850	**0.372**	0.314	0.705	86.1
3	5M	Random	0.986	0.905	0.835	0.334	0.333	0.716	169.6
		FPS	0.975	0.970	0.876	0.236	0.307	0.699	**111.2**
	25M	Random	**0.989**	0.964	0.854	0.316	0.329	0.694	170.9
		FPS	0.984	**0.996**	**0.885**	0.227	0.310	0.667	122.3
	50M	Random	**0.989**	0.957	0.842	**0.345**	**0.352**	0.717	171.0
		FPS	0.986	0.978	0.868	0.344	0.327	**0.748**	128.2
	500M	Baseline	**0.989**	0.979	0.822	0.323	0.318	0.689	171.4

B PROOFS: WHEN SHOULD FEATURE MATCHINGS AGREE?

B.1 SEPARATE ENCODER AND DECODER FEATURE ALIGNMENT 1

Proof. Define

$$a_{r,N} = \sup_{\theta} \left| \widehat{R}_{r,N}(\theta) - R(\theta) \right|.$$

Statistical consistency (assumption (ii)) implies $a_{1,N} \xrightarrow{p} 0$ and $a_{q,N} \xrightarrow{p} 0$.

Let $\theta^\star$ be a population minimizer. Approximate empirical optimality gives

$$R(\widehat{\theta}_{r,N}) \le \widehat{R}_{r,N}(\widehat{\theta}_{r,N}) + a_{r,N} \tag{9}$$

$$\le \widehat{R}_{r,N}(\theta^\star) + \eta_{r,N} + a_{r,N} \tag{10}$$

$$\le R^\star + 2a_{r,N} + \eta_{r,N}. \tag{11}$$

Here $\eta_{r,N} \ge 0$ denotes the optimization error, i.e., $\widehat{R}_{r,N}(\widehat{\theta}_{r,N}) \le \inf_\theta \widehat{R}_{r,N}(\theta) + \eta_{r,N}$, and $\eta_{r,N} \xrightarrow{p} 0$ by optimization consistency (assumption (iv)).

Hence

$$R(\widehat{\theta}_{r,N}) - R^\star \xrightarrow{p} 0.$$

By encoder identifiability and separation, any sequence whose encoder features remain at least ϵ away from the population encoder orbit must incur population excess risk at least $\Delta_E(\epsilon) > 0$. Therefore

$$\Pr\left[d_E(\widehat{\theta}_{r,N}, \theta^\star) \ge \epsilon \right] \le \Pr\left[2a_{r,N} + \eta_{r,N} \ge \Delta_E(\epsilon) \right] \to 0.$$

The same argument using decoder separation gives $d_D(\widehat{\theta}_{r,N}, \theta^\star) \xrightarrow{p} 0$. Applying the triangle inequality separately in encoder and decoder feature spaces gives

$$d_E(\widehat{\theta}_{q,N}, \widehat{\theta}_{1,N}) \xrightarrow{p} 0$$

and

$$d_D(\widehat{\theta}_{q,N}, \widehat{\theta}_{1,N}) \xrightarrow{p} 0.$$

□

B.2 PROOF OF AGREEMENT OF ENCODER AND DECODER HUNGARIAN MATCHINGS) 2

Proof. By Theorem 1, both normalized encoder and decoder feature systems converge to their respective population feature systems after one-to-one alignment.

By assumption (iii), each run's encoder and decoder features approach the population features under a single labeling shared by encoder and decoder. Let $P_N^\star$ denote the permutation that aligns the two runs' labelings. Because distinct population features are separated by a positive margin, small perturbations cannot change the optimal Hungarian assignment, so with probability tending to one $P_E^{(N)} = P_N^\star$, and likewise $P_D^{(N)} = P_N^\star$.

Hence,

$$\Pr\left(P_E^{(N)} = P_D^{(N)} \right) \to 1.$$

□

Table 24: **FPS distribution shift for Qwen3-8B and Pythia-160M.*** Metrics are defined as in Table 1. The Gemma-2-2B pattern holds across model families: random selection is distribution-neutral at every budget, while FPS shifts the selected subset toward higher-norm activations, with divergence growing as the subset shrinks.

			Selected vs. 500M stream		
Model	**Selection**	**Coreset**	**Δ mean norm (%)**	**KL (bits)**	**Mean-dir. cosine**
Qwen3-8B	FPS	5M	$+33.8 \pm 1.3$	4.02	0.889 ± 0.007
		25M	$+21.4 \pm 0.4$	2.08	0.952 ± 0.002
		50M	$+16.4 \pm 0.3$	1.35	0.970 ± 0.001
	Random	5M	$+0.1 \pm 0.2$	0.000	1.000 ± 0.000
		25M	0.0 ± 0.1	0.000	1.000 ± 0.000
		50M	0.0 ± 0.1	0.000	1.000 ± 0.000
Pythia-160M	FPS	5M	$+23.7 \pm 0.5$	1.49	0.900 ± 0.004
		25M	$+15.5 \pm 0.3$	0.78	0.953 ± 0.002
		50M	$+11.9 \pm 0.3$	0.51	0.973 ± 0.001
	Random	5M	0.0 ± 0.3	0.000	1.000 ± 0.000
		25M	0.0 ± 0.1	0.000	1.000 ± 0.000
		50M	0.0 ± 0.1	0.000	1.000 ± 0.000

*Same protocol as Table 1: all selectors are applied to identical activation pools; entries are mean ± SD across pool refreshes (50 for norm statistics, 15 for direction statistics). Entries shown as ± 0.000 have SD below 10^{-4}.

Table 25: **Feature correspondence under increasingly strict alignment criteria.** *Encoder-decoder agreement* is the fraction of features for which independently computed encoder and decoder Hungarian matchings select the same reference latent. *Shared features* additionally require both encoder and decoder cosine similarities to exceed the specified threshold τ. Results are reported for Qwen3-8B, and Pythia-160M.

			Counterpart (%)	Shared features (%)		
Model	**SAE**	**Budget**	**Enc.–dec. agree**	$\tau \geq 0.3$	$\tau \geq 0.5$	$\tau \geq 0.7$
Qwen3-8B	ReLU	5M	69.85	33.07	22.90	18.58
		25M	79.47	51.16	36.23	25.19
		50M	**82.49**	**61.70**	**41.41**	**29.51**
	BatchTopK	5M	66.06	34.16	7.34	1.14
		25M	81.05	65.56	35.63	8.83
		50M	**84.69**	**72.78**	**49.49**	**18.24**
Pythia-160M	ReLU	5M	73.90	47.81	29.79	23.54
		25M	78.11	61.07	37.96	29.09
		50M	**79.85**	**65.89**	**42.46**	**32.16**
	BatchTopK	5M	47.19	31.81	7.80	1.72
		25M	64.92	55.84	28.81	8.37
		50M	**69.35**	**61.61**	**38.13**	**14.86**

Table 26: **Paired statistical tests for Gemma-2-2B ReLU and BatchTopK.** Panel A compares the 500M reference with Random subsets ($\Delta = 500\text{M} - \text{Random}$); Panel B compares FPS with Random at matched budgets ($\Delta = \text{FPS} - \text{Random}$). Each entry reports Δ/p from a paired two-sided t-test across matched SAE configurations.

	ReLU			BatchTopK		
Metric	**5M**	**25M**	**50M**	**5M**	**25M**	**50M**
A. 500M vs. Random						
Loss Rec.	$+.005/ < .001^{***}$	$+.002/.002^{**}$	$+.002/.014^{*}$	$+.007/.006^{**}$	$+.002/.015^{*}$	$+.002/.092$
Sparse Probe	$+.018/.008^{**}$	$+.016/.005^{**}$	$+.004/.363$	$-.022/.355$	$-.036/.050$	$-.006/.769$
1-Abs.	$-.098/ < .001^{***}$	$-.053/ < .001^{***}$	$-.030/ < .001^{***}$	$-.032/.516$	$-.047/.252$	$-.055/.135$
AutoInterp	$+.020/ < .001^{***}$	$+.014/ < .001^{***}$	$+.006/.085$	$+.032/.039^{*}$	$+.008/.365$	$+.003/.734$
SCR	$+.007/.287$	$+.002/.765$	$+.003/.454$	$-.055/.173$	$-.037/.239$	$-.034/.133$
TPP	$+.001/.016^{*}$	$+.001/.031^{*}$	$+.000/.616$	$-.023/.080$	$-.040/.136$	$-.018/.225$
RAVEL	$+.016/.123$	$+.006/.260$	$+.005/.555$	$+.024/.006^{**}$	$+.007/.539$	$+.009/.231$
Unlearning	$+.009/.602$	$+.002/.902$	$-.008/.418$	$+.060/.076$	$+.044/.243$	$+.047/.005^{**}$
B. FPS vs. Random						
Loss Rec.	$-.020/ < .001^{***}$	$-.010/ < .001^{***}$	$-.007/.001^{**}$	$-.023/.002^{**}$	$-.009/.009^{**}$	$-.005/.010^{*}$
Sparse Probe	$-.016/.027^{*}$	$-.005/.425$	$-.019/.007^{**}$	$-.013/.108$	$-.026/.346$	$-.017/.346$
1-Abs.	$-.011/.242$	$-.063/.001^{**}$	$-.054/ < .001^{***}$	$-.010/.169$	$-.108/.104$	$-.126/.049^{*}$
AutoInterp	$+.006/.014^{*}$	$+.014/ < .001^{***}$	$+.010/.023^{*}$	$+.026/.143$	$+.027/.019^{*}$	$+.025/.007^{**}$
SCR	$-.064/ < .001^{***}$	$-.050/ < .001^{***}$	$-.037/ < .001^{***}$	$-.090/.004^{**}$	$-.067/.041^{*}$	$-.084/.084$
TPP	$-.006/ < .001^{***}$	$-.004/ < .001^{***}$	$-.004/ < .001^{***}$	$-.022/.006^{**}$	$-.032/.070$	$-.001/.982$
RAVEL	$-.012/.053$	$-.011/.016^{*}$	$-.010/.215$	$-.029/.006^{**}$	$-.015/.138$	$-.003/.642$
Unlearning	$+.016/.366$	$+.006/.673$	$+.015/.380$	$+.022/.123$	$+.019/.593$	$+.014/.351$

Observation. Reduced-data SAEs become increasingly comparable to the 500M reference as the budget grows, with several metrics already showing parity at small budgets. Selection effects are metric-dependent: Random generally favors reconstruction and SCR, whereas FPS frequently improves AutoInterp. $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$.