LINEAGESAE: VERSIONED FEATURE VOCABULARIES ACROSS SAE RETRAINING AND MODEL UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

A feature index in a sparse autoencoder (SAE) is useful only if it keeps denoting the same feature after the SAE is retrained or the language model changes. Existing methods either encourage reproducibility or align independently learned dictionaries after training. Neither decides, for a saved parent and a changed view, which of its addresses to retain and which to free for new features. We introduce LineageSAE, a parent–child SAE that builds a versioned feature vocabulary from a saved parent and current activations without replaying the parent's data. A corrected marginal-utility test marks each parent address as inherited or released, and a parent-guided decoder handoff transfers reconstruction to the child. A temporary ranking loss uses the parent's pre-activation scores to align released addresses across independently trained children, including when ReLU outputs are zero.

On SynthSAEBench and toy superposition with feature deletion, addition, and frequency changes, LineageSAE reaches child–child support Jaccard, the overlap of firing samples at the same address, of 0.779 and 0.799 without post-training assignment. The strongest matched baselines reach 0.607 and 0.483. Reconstruction is close to L_2-regularized TopK on SynthSAEBench, and on Toy the reconstruction error is comparable to the strongest sparse coding baselines. A trained child keeps its index assignments and can parent the next update.

1 INTRODUCTION

Sparse autoencoders (SAEs) express language-model activations as sparse combinations of learned features, providing units for interpretation and intervention (Cunningham et al., 2024; Gao et al., 2025; Makelov et al., 2025). Each feature has a decoder direction and an activation that depends on the input. Once a feature is used in a circuit analysis, an intervention, or a monitor, its index links that analysis to the SAE. Retraining can break this link: independent runs on the same model and data can recover different features and assign related features to different indices (Paulo & Belrose, 2025; Leask et al., 2025).

The consequences extend beyond renumbering. When a monitored feature changes, we need to distinguish a change in the language model from a change in its SAE decomposition. Without a reliable correspondence, the features underlying earlier annotations and interventions must be located and checked again before they can support comparisons across updates. Model and data changes introduce a second difficulty: some old features remain useful, whereas others need to give way to new structure. Keeping every assignment fixed can restrict adaptation; retraining freely loses continuity. A vocabulary that supports cumulative analysis must accommodate both.

Existing approaches address parts of this problem. Weight regularization improves reproducibility across runs (Jedryszek & Crook, 2026), while post-hoc matching recovers correspondences between learned dictionaries (Cao et al., 2026; Lan et al., 2025). Joint training establishes shared feature spaces across available models or datasets (Thasarathan et al., 2025; Váradi & Kmetty, 2026); SharedSAE also adapts new models to a frozen shared dictionary (Ognev et al., 2026). These approaches improve consistency or recover correspondences, but do not by themselves specify which assignments in an existing vocabulary should be retained and how vacated indices should be reused. We study this sequential setting with access to a saved SAE and current activations, without requiring the SAE's original training data.

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.

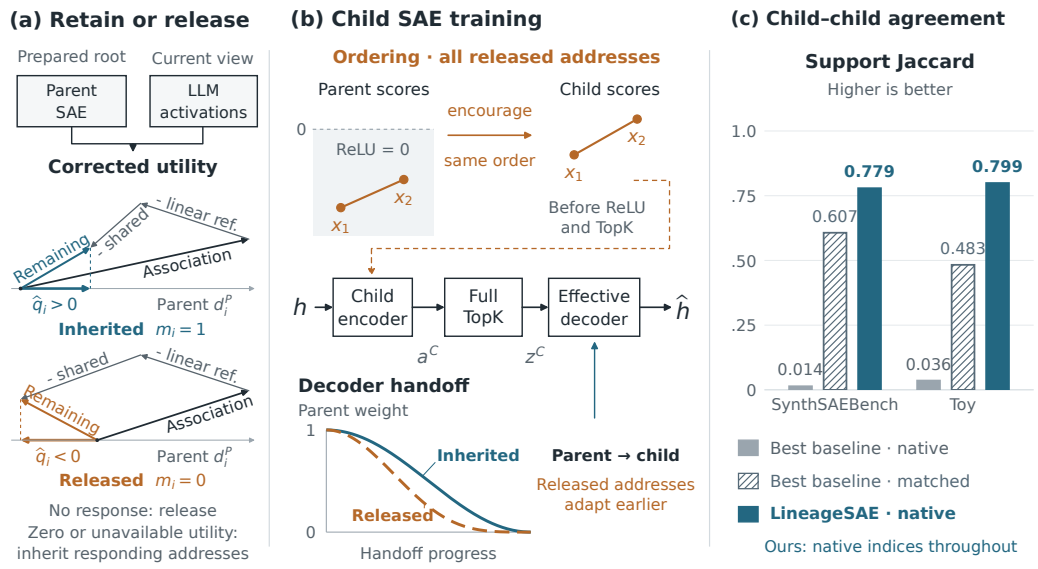

Figure 1: **LineageSAE overview.** (a) Corrected utility determines which addresses to inherit or release. (b) Parent decoder weights decay as children learn; response ordering guides released addresses even with zero ReLU output (shaded region). (c) Child–child support Jaccard (overlap of firing samples) from Table 2. LineageSAE retains learned indices; baseline maxima at native and matched indices are selected independently across methods for each benchmark. Panels (a,b) are schematic; reconstruction and sparsity comparisons are in the table.

We introduce LineageSAE, which organizes vocabulary updates through parent–child relationships. A feature's native index is its *address*; a *view* specifies the model checkpoint, activation extraction site, and input distribution. Before training, each address is declared *inherited*, retaining its parent feature as a reference, or *released*, keeping its capacity for a new role. Children train independently from the same parent and can become parents of subsequent updates.

The decision to inherit rests on what a parent direction still contributes. Figure 1(a) illustrates the test: from the input–response association, subtract the association expected from the average linear response and the contribution explained by another address, then project the remainder onto the parent direction. Release requires either no observed response or negative corrected utility; a rare responding feature can still be retained. Initially, child activations reconstruct through the parent decoder. This guidance fades as the child decoder takes over, with released addresses adapting earlier (Figure 1(b), lower curves).

Release creates a further problem: reconstruction alone does not determine which new feature occupies which address. Our key observation is that an inactive parent feature can still supply an input preference. In the shaded region of Figure 1(b), both inputs have zero output after ReLU, which clips negative scores to zero, yet their original scores remain ordered. A temporary ranking loss transfers this ordering to every released address. It gives independent children a common reference for learning new features while allowing their response magnitudes and decoder directions to change. The constraint ends before training finishes; deployment uses the child's learned indices and requires no parent SAE.

In controlled tests of feature deletion, addition, and frequency changes, LineageSAE achieves child–child support Jaccard (overlap of firing samples at the same address) of 0.779 on SynthSAEBench and 0.799 on toy superposition, without post-training assignment. The best matched baselines reach 0.607 and 0.483, respectively (Figure 1(c)). Reconstruction error is similar to L_2-regularized TopK on SynthSAEBench and higher on Toy (Table 2). Further experiments examine domain transfer, language-model updates, and successive vocabulary generations (Section 3); per-experiment objectives and schedules appear in Appendix A.5.

2 METHOD

2.1 VERSIONED FEATURE VOCABULARIES

A sparse autoencoder (SAE) represents language-model activations as combinations of a small number of features. Analyses built on these features often refer to specific feature indices. When the model or data changes, we want features that remain supported to retain their indices while making room for new features. We therefore call a decoder atom's native index i an *address* and organize vocabulary updates through parent–child relationships.

A *view* is determined by a language-model checkpoint, an activation extraction site, and an input distribution. Given a parent SAE and current-view activations, LineageSAE trains a child without replaying the parent's data. Each update assigns an address a fixed state $m_i \in \{0, 1\}$: 1 denotes *Inherited* and 0 *Released*. Inherited addresses use their parent features as references; released addresses retain their capacity but may take on new roles. Release withdraws the requirement to retain the old feature, without determining whether its concept still exists. The first SAE prepared as described below defines the *root vocabulary*; each child can become the parent of the next update.

2.2 PREPARING AN INHERITABLE ROOT VOCABULARY

Before deciding which features to inherit, we prepare their comparison space. Decoder atoms may share input-varying components that matter for reconstruction but rarely affect which addresses activate, making distinct features appear similar in decoder space. Separating these components into a *shared reconstruction subspace* lets the SAE dictionary express feature differences more clearly.

Let $x_t \in \mathbb{R}^d$ be input activation t, D the pretrained SAE's decoder matrix, and z_t its sparse coefficients. TopK selects the k highest encoding scores. Let Δ_t be the gap between the weakest selected and strongest excluded scores; $g_t = \nabla_{x_t} \Delta_t$ describes this gap's sensitivity to input perturbations. To find directions used heavily for reconstruction but weakly affecting the selection gap, we compare reconstruction variation with selection sensitivity through the generalized eigenproblem

$$\frac{\mathrm{Cov}_t(Dz_t)}{\mathrm{tr}\,\mathrm{Cov}_t(Dz_t)} v_j = \lambda_j \left(\frac{\mathbb{E}_t[g_t g_t^\top]}{\mathrm{tr}\,\mathbb{E}_t[g_t g_t^\top]} + \frac{\gamma}{d} I \right) v_j. \tag{1}$$

Here Cov denotes covariance and tr the trace; each side is normalized to remove its overall scale. The regularizer $\gamma > 0$ stabilizes the problem, and I is the identity matrix. Larger λ_j identify directions with greater reconstruction variation relative to selection sensitivity. Orthonormalizing the top r eigenvectors gives U, with complementary projection $P = I - UU^\top$.

These shared components still contain information needed for reconstruction. An identity bypass preserves them while the SAE focuses on the remainder:

$$h_t = P(x_t - \bar{x}), \qquad \widehat{x}_t = \bar{x} + UU^\top (x_t - \bar{x}) + \widehat{h}_t. \tag{2}$$

Here $\bar{x}$ is the input mean and $\widehat{h}_t$ the SAE reconstruction of h_t. Restricting decoder atoms and bias to the orthogonal complement of U makes residual and original-space reconstruction errors equal. We refine the projected, normalized dictionary with AuxK, an auxiliary reconstruction loss that supplies learning signals to underused atoms, to restore feature coverage. We fix U within a lineage segment; substantial view changes trigger renewed subspace estimation and root preparation.

2.3 ADDRESS RELEASE VIA CORRECTED MARGINAL UTILITY

After a view change, an address may still be selected even when its parent decoder direction contributes little. Figure 1(a) illustrates our test: subtract a linear response reference and the contribution explained by another address, then assess the remainder along the parent direction.

Consider a TopK SAE, which selects the k highest encoding scores. For projected current-view input h_t, let $a^P_{t,i}$ be the parent's score before ReLU and TopK, and j_t the strongest excluded address. We measure response above the selection boundary as $r_{t,i} = [a^P_{t,i} - \max\{a^P_{t,j_t}, 0\}]_+$, where $[a]_+ = \max(a, 0)$. With parent encoder vector e^P_i and indicator $\mathbf{1}\{\cdot\}$, away from nondifferentiable boundaries,

$$\nabla_h r_{t,i} = \mathbf{1}\{r_{t,i} > 0\} \left(e^P_i - \mathbf{1}\{a^P_{t,j_t} > 0\} e^P_{j_t} \right).$$

The two terms track changes in the address's score and, when positive, the competing threshold.

We use $\mathrm{Cov}(h, r_i)$ to measure how inputs and responses vary together across samples. Such association exists even for a response that follows a fixed linear rule. We therefore use a linear response as a reference, setting its slope to the gradient averaged over all samples, $\mathbb{E}[\nabla_h r_i]$. With input covariance $\Sigma = \mathrm{Cov}(h)$, this reference produces covariance $\Sigma\mathbb{E}[\nabla_h r_i]$. Subtracting it measures association beyond this average linear response (Appendix A.2).

To test whether the parent direction is needed, we first let another address explain the target's remaining association while keeping the target response fixed. Let $C_i = \mathrm{Cov}(h, r_i) - \Sigma\mathbb{E}[\nabla_h r_i]$ denote this association. Among other addresses with nonzero C_j and positive response variance, we choose the direction $u_i = C_j/\|C_j\|_2$ best aligned with C_i. For positive target variance, we fit $b_i = u_i[u_i^\top C_i]_+ / \mathrm{Var}(r_i)$. For unit parent decoder atom d_i^P, projecting the remaining association onto that direction gives the marginal utility

$$q_i = (d_i^P)^\top \Big[\underbrace{\mathrm{Cov}(h, r_i)}_{\text{observed association}} - \underbrace{\Sigma\mathbb{E}[\nabla_h r_i]}_{\text{linear response reference}} - \underbrace{b_i \,\mathrm{Var}(r_i)}_{\text{shared contribution}} \Big]. \tag{3}$$

The gray arrows in Figure 1(a) show the two subtractions; the colored remainder projects onto the parent direction. Positive q_i means that adding a small contribution along this direction improves the corrected reconstruction objective, supporting inheritance.

We fit the background on one data fold and evaluate it on the other, then exchange their roles. Let $\widehat{q}_i$ sum the two utilities and $\widehat{p}_i$ denote the observed positive-response fraction. For the decision below, undefined estimates receive the neutral value $\widehat{q}_i = 0$, so insufficient statistics alone do not trigger release. We set

$$m_i = \begin{cases} 0, & \widehat{p}_i = 0 \text{ or } \widehat{q}_i < 0, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

Nonresponding samples give zero gradients; the full-sample covariance, mean gradient, and background fit all reflect response frequency. Holding other population quantities fixed, a positive frequency factor scales utility without changing its sign (Appendix A.1). An observed rare address can thus remain inherited when its utility is nonnegative. States remain fixed during child training.

2.4 Parent-Guided Decoder Handoff

Retaining a feature concerns both its direction and the inputs that activate it. Initially, child coefficients reconstruct through parent atoms, letting reconstruction gradients use established feature directions to guide address usage. Reconstruction then gradually passes to the child's own decoder, allowing features to adapt to the current view.

Inherited encoder rows and decoder atoms are initialized from the parent. All addresses compete in the same TopK selection. The effective decoder in Figure 1(b) reconstructs their sparse coefficients through the blended atoms

$$d_{i,s}^{\mathrm{eff}} = \mathrm{Norm}\big[\beta_s^{2-m_i} d_i^P + (1 - \beta_s^{2-m_i}) d_{i,s}^C\big], \tag{5}$$

where $d_{i,s}^C$ is the child atom at step s, Norm normalizes to unit length, and β_s decreases from one to zero. Inherited addresses use parent weight β_s and released addresses β_s^2. The lower curves in Figure 1(b) show the consequence: released addresses lose parent influence sooner and can change direction earlier.

While the parent still carries most of the reconstruction, the child decoder must also prepare to take over. An auxiliary reconstruction loss therefore trains it on child coefficients with gradients stopped. AuxK continues to support underused atoms. A further decoder-direction constraint encourages each child atom to have higher cosine similarity to its own parent atom than to other parent atoms. This constraint remains on inherited addresses and weakens with the handoff on released addresses. Losses and schedules are specified in Appendix A.

2.5 Guiding Released-Address Reuse through Response Ordering

Release provides capacity for new features, but reconstruction alone leaves their indices unspecified: independent children may learn similar content at different addresses. Common input preferences during training give children similar starting points for exploring new features at these addresses.

The shaded region in Figure 1(b) shows why an inactive parent address can still guide reuse: both inputs have zero ReLU output, yet one has a higher linear score. Such preferences remain accessible before ReLU and TopK. We retain their relative ordering while allowing response magnitudes, sparse selection, and decoder directions to adapt to the current view.

All released addresses $\mathcal{R} = \{i : m_i = 0\}$ begin reuse with their parent encoder rows and unit decoder atoms. Let $a^C_{t,i}$ be the child's linear score before ReLU and TopK. For fixed training pairs (t, t'), we retain comparisons with $|a^P_{t,i} - a^P_{t',i}| > \delta\sigma^P_i$: σ^P_i is the parent-score standard deviation over training samples, and $\delta > 0$ sets a minimum gap to avoid treating nearly equal scores as a reliable ordering.

For minibatch B, let $\mathcal{P}_B$ contain the retained triples (t, t', i) with $t \in B$ and $i \in \mathcal{R}$. The response-ordering loss is

$$\mathcal{L}_{\text{rank}} = \frac{w_{s,B}}{\max\{|\mathcal{P}_B|, 1\}} \sum_{(t,t',i)\in\mathcal{P}_B} \Big[\mu - \underbrace{\operatorname{sgn}(a^P_{t,i} - a^P_{t',i})}_{\text{parent preference}} \underbrace{\frac{a^C_{t,i} - a^C_{t',i}}{\operatorname{sg}(\sigma^C_{i,B})}}_{\text{standardized child-score gap}} \Big]^2_+. \tag{6}$$

Here sgn takes the sign, sg stops gradients, and $\sigma^C_{i,B}$ is the minibatch child-score standard deviation; both standard deviations have positive lower bounds. The margin is $\mu > 0$; $w_{s,B}$ combines input scale and schedule. Following the parent's order by this margin incurs no penalty. Comparing scores before sparse selection provides learning signals even to addresses that have yet to activate.

Reconstruction drives current-view learning, decoder handoff supplies a fading parent reference, and ordering coordinates input preferences across all released addresses. These objectives train jointly. As indicated by the dashed arrow in Figure 1(b), ordering gradients reach only the released encoder rows. The ordering weight reaches zero before training ends, leaving a final adaptation phase. Inference needs only the child SAE, fixed projection, and identity bypass.

3 Experiments

Data and training. Pythia-1.4B-deduped layer-12 experiments transfer from WikiText-103 to Stack Smol XS Python code, or across 300 language-model updates on Wiki, with dictionary width $N = 4096$ and sparsity budget $k = 16$. SynthSAEBench embeds 512 features in 128 dimensions; six conditions delete and add 51 features, with or without reduced survivor frequencies ($N = 1024$, $k = 16$). Toy superposition embeds 128 features in 32 dimensions, deletes and adds eight, and makes eight survivors rarer ($N = 256$, $k = 8$). Children train for 15,000 updates using the frozen parent and current-view activations. Training, calibration, and test splits are disjoint; Appendices B.1 and A.5 provide sample counts, seeds, and optimization settings.

Baselines. Comparisons cover alternative architectures, L_2-regularized TopK, Ordered SAE (OSAE), Identifiable SAE (iSAE), archetypal constraints, joint training (SharedSAE), and feature pooling (Stable Pool). We retain each method's sparsity mechanism and report measured operating points. Section 4 describes the methods; Appendix B.1 specifies implementations and budgets.

Metrics. Parent–child (P–C) and child–child (C–C) comparisons measure continuity across versions and agreement among descendants. Decoder cosine compares directions; support Jaccard J compares firing samples; Spearman correlation ρ compares activation ranks, including zeros. Native scores use deployed indices; matched scores use a calibration-fitted permutation. Normalized mean squared error (NMSE) divides reconstruction error by input variance; L_0 counts active features per input. Cross-entropy (CE) recovery measures how much SAE reconstruction reverses the loss increase from zeroing an activation. Bold/underline indicate best/second-best comparable values. Residual-input NMSE (†) normalizes by residual energy and full-activation NMSE by activation energy; both are reported in the appendix.

3.1 Address Continuity under SAE Retraining

Adapting to a new domain. Wiki-to-Code tests agreement during domain adaptation, evaluating both encoders on the same Code inputs. LineageSAE reaches C–C decoder cosine 0.6436, versus Stable Pool's 0.4321, with 95.31% CE recovery (Table 1). Lower P–C cosine (0.3024) and P–C/C–C $J = 0.1176/0.3494$ indicate greater agreement among Code descendants than with the Wiki parent. Relaxed Archetypal SAE (RA-SAE) reaches C–C cosine 0.4859, matching its random-index control of 0.4876, so decoder cosine there tracks shared geometry rather than address correspondence (Appendix Table 33).

Table 1: Wiki-to-Code response and decoder continuity, with reconstruction and sparsity.

	P–C response		C–C response		Decoder cosine↑		Reconstruction		
Method	$J\uparrow$	$\rho\uparrow$	$J\uparrow$	$\rho\uparrow$	P–C	C–C	NMSE↓	L_0	CE rec.↑
L2-TopK	0.0025	0.0023	0.0012	0.0004	0.0284	0.0249	<u>0.0140</u>	16.00	95.3%
OSAE	0.0016	0.0008	0.0019	0.0007	0.0089	0.0004	0.0152	15.88	94.2%
RA-SAE	0.0040	0.0072	0.0001	0.0003	<u>0.4718</u>	0.4859	0.0632	1.07	30.5%
BatchTopK	0.0012	0.0006	0.0011	0.0003	0.0026	0.0056	**0.0139**	16.19	**95.59%**
Gated SAE	0.0138	0.0395	0.0038	0.0063	0.2563	0.0043	0.0167	12.99	92.70%
SharedSAE	0.0007	0.0002	0.0002	0.0000	0.0190	0.0296	0.0395	16.00	77.25%
Stable Pool	<u>0.0818</u>	<u>0.1378</u>	<u>0.2167</u>	<u>0.2878</u>	0.2330	0.4321	0.0142	16.00	95.25%
MP-SAE	0.0026	0.0018	0.0032	0.0049	0.0004	0.0087	0.0210	11.43	89.30%
iSAE	0.0028	0.0028	0.0009	0.0001	0.0105	−0.0001	0.0149	16.00	94.75%
AbsTopK	0.0041	0.0045	0.0028	0.000004	0.0073	0.0003	0.1774†	16.00	<u>95.42%</u>
Archetypal	0.0042	0.0075	0.00003	0.0001	**0.4743**	<u>0.4863</u>	0.0577	1.18	31.08%
Matryoshka	0.0009	0.0004	0.0044	0.0039	0.0007	0.0056	0.0176	19.10	90.16%
LineageSAE	**0.1176**	**0.1968**	**0.3494**	**0.4430**	0.3024	**0.6436**	0.0141	15.97	95.31%

Adapting to known feature changes. Table 2 separates native correspondence from post-hoc alignment. LineageSAE's native C–C J is $0.779/0.799$ on SynthSAEBench/Toy, above the strongest matched baseline on either benchmark. Matching does not improve its scores: correspondence is already present at the learned indices. Matching-pursuit SAE (MP-SAE) achieves the lowest NMSE on both benchmarks; LineageSAE is close to L_2-TopK on SynthSAEBench (0.113 versus 0.112), and its Toy error is comparable to the best sparse coding baselines.

Table 2: Response continuity under feature deletion and addition. Native and matched scores compare deployed and calibration-aligned addresses, respectively.

	SynthSAEBench				Toy superposition			
Method	NMSE↓	$J_{\text{nat}}\uparrow$	$J_{\text{match}}\uparrow$	$\rho_{\text{match}}\uparrow$	NMSE↓	$J_{\text{nat}}\uparrow$	$J_{\text{match}}\uparrow$	$\rho_{\text{match}}\uparrow$
L2-TopK	<u>0.112</u>	0.006	0.453	0.499	0.094	0.013	<u>0.483</u>	<u>0.548</u>
OSAE	0.169	0.008	0.314	0.391	0.154	0.018	0.217	0.309
RA-SAE	0.119	0.007	0.446	0.482	<u>0.093</u>	0.016	0.448	0.501
SharedSAE	0.155	0.006	0.426	0.473	0.360	0.016	0.091	0.139
Stable Pool	0.1142	<u>0.5566</u>	0.5645	0.6561	0.0985	<u>0.4229</u>	0.4288	0.4638
BatchTopK	0.143	0.008	0.448	0.478	0.123	0.020	0.432	0.445
Gated SAE	0.118	0.008	0.436	0.481	0.126	0.015	0.272	0.350
MP-SAE	**0.006**	0.008	<u>0.607</u>	<u>0.729</u>	**0.015**	0.021	0.186	0.272
iSAE	0.503	0.001	0.050	0.073	0.311	0.004	0.065	0.101
AbsTopK	0.185	0.007	0.401	0.442	0.170	0.002	0.036	0.052
Archetypal	0.297	0.007	0.349	0.392	0.097	0.019	0.428	0.483
Matryoshka	0.157	0.014	0.425	0.462	0.144	0.036	0.411	0.443
LineageSAE	0.113	**0.779**	**0.772**	**0.838**	0.106	**0.799**	**0.794**	**0.854**

3.2 Model Updates and Successive Vocabulary Versions

Updating the underlying model. After 300 Pythia updates, LineageSAE retains P–C/C–C decoder cosine $0.9706/0.9814$ and CE recovery 95.69% (Table 3). Response agreement is also high:

P–C/C–C $J = 0.6163/0.6959$, versus $0.3902/0.6746$ for Stable Pool. The larger P–C gain shows that the benefit extends beyond agreement among descendants to continuity with the existing vocabulary, at competitive CE recovery.

Table 3: Response and decoder continuity across a Pythia model update.

	P–C response		C–C response		Decoder cosine↑		Reconstruction		
Method	$J\uparrow$	$\rho\uparrow$	$J\uparrow$	$\rho\uparrow$	P–C	C–C	NMSE↓	L_0	CE rec.↑
L2-TopK	0.0434	0.0580	0.0015	0.0005	0.0973	0.0244	0.0150	16.00	95.5%
OSAE	0.0317	0.0417	0.0016	0.0001	0.0593	0.0006	0.0163	15.85	94.6%
RA-SAE	0.0072	0.0079	0.0002	0.0002	0.5385	0.5441	0.0551	1.01	35.4%
BatchTopK	0.0405	0.0496	0.0028	0.0020	0.0705	0.0040	0.0154	16.02	95.30%
Gated SAE	0.1011	0.1412	0.0014	0.0009	0.2026	0.0073	0.1946†	15.11	**95.71%**
SharedSAE	0.3401	0.3934	0.0012	−0.0001	0.4984	0.0202	0.0239	16.00	94.25%
Stable Pool	0.3902	0.5387	0.6746	0.7952	0.7565	0.9539	0.0147	16.00	95.57%
MP-SAE	0.0126	0.0199	0.0124	0.0195	0.0018	−0.0003	0.0210	24.32	90.46%
iSAE	0.0645	0.0903	0.0013	−0.0004	0.1254	−0.0001	0.1973†	16.00	95.61%
AbsTopK	0.0546	0.0785	0.0027	0.0006	0.1237	−0.0005	0.2037†	16.00	95.66%
Archetypal	0.0068	0.0074	0.0000	0.0000	0.5384	0.5424	0.0512	1.02	36.65%
Matryoshka	0.0048	0.0043	0.0055	0.0052	0.0050	0.0068	0.0189	16.75	90.12%
LineageSAE	**0.6163**	**0.7527**	**0.6959**	**0.8109**	**0.9706**	**0.9814**	**0.0145**	15.951	95.69%

Passing a vocabulary through generations. Three chains train four generations on fixed Wiki data, with each child using the preceding child as parent. Figure 2 shows within-chain parent–child J rising from 0.6773 to 0.7673, while root-relative J falls from 0.6773 to 0.5573; ρ and decoder cosine show the same separation. The former compares successive updates, whereas the latter always compares with G_0, so small local changes can accumulate into long-term drift. Cross-chain J declines from 0.7879 to 0.7160 as independent chains diverge, while CE recovery stays near 95.8%. Independent retraining has near-zero native agreement, whereas freezing the parent decoder fixes agreement but cannot adapt released addresses (Appendix Table 25).

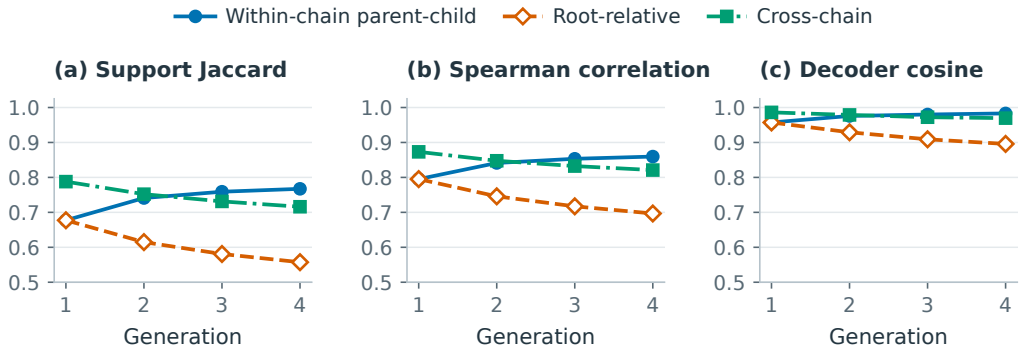

Figure 2: Continuity across four generations. Within-chain parent–child compares successive generations; root-relative compares with the initial SAE; cross-chain compares independent descendants at the same generation. Full results appear in Appendix Table 22.

3.3 Inheritance, Release, and Reuse

Continuity and feature coverage. Known features test whether consistent addresses represent the intended content. Recovery requires activation correlation and decoder cosine of at least 0.8, with benchmark-specific correspondence rules in Appendix D.1. Figure 3 shows complete recovery of new and ordinary retained SynthSAEBench features, but 52.7% of rare survivors. Toy recovery is $68.1\%/75.0\%/88.4\%$ for rare/new/other retained features. Stable addresses and feature coverage are complementary objectives.

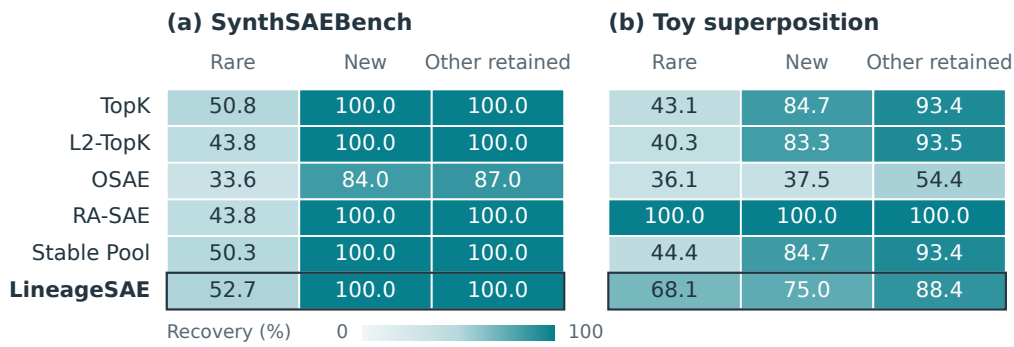

Figure 3: Known-feature recovery (%). Rare features have reduced frequency; new features are added during the update. The RA-SAE row uses three separately trained, 10,000-step models per comparison (one old, two new) with independently sampled candidate pools; recovery is measured on the two new models using the frozen held-out test sets and 0.8 correlation and decoder-cosine thresholds. Darker cells indicate higher recovery.

Releasing obsolete roles and coordinating reuse. Figure 4(a) shows that utility releases 103–104 of 105 ordinary deleted-feature addresses and all five difficult deletions, while releasing no ordinary survivor and at most $3/153$ rare survivors. In panel (b), $68/70$ (97.1%) new features recovered in released addresses occupy the same address across children, with complete agreement in five conditions. The next comparison therefore isolates release, decoder handoff, and response ordering under matched conditions.

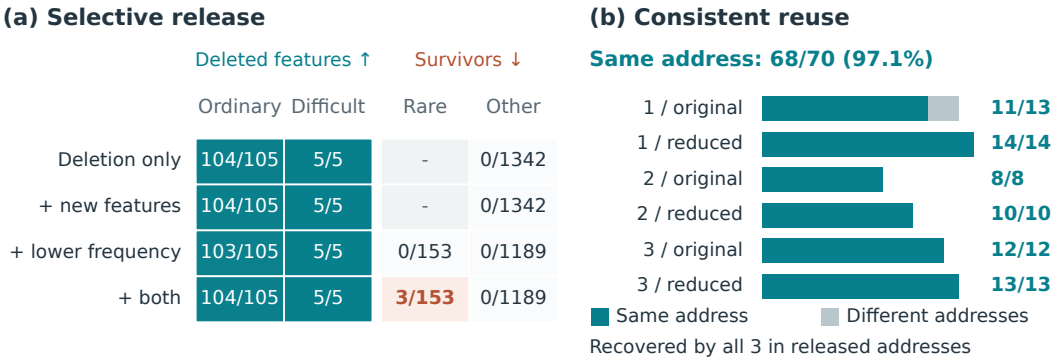

Figure 4: Release and reuse. (a) Released/evaluated addresses; survivor releases are errors. (b) Each bar counts new features recovered in released addresses by all three children; teal denotes identical addresses across children. Fractions give same-address/total counts within this group. Rows identify the task and original or reduced survivor frequencies.

Component ablations. We isolate release decisions, decoder handoff, and response ordering across three matched conditions with one parent and two independently trained children per condition (Table 4). Corrected utility releases $105/105$ ordinary and $5/5$ similar deletions, versus $37/105$ and $0/5$, while releasing only $3/102$ rare survivors. The full method recovers $40.67/51$ new features at the same native address; removing response ordering or decoder handoff reduces this to $31.00/51$ or $24.67/51$. All variants recover the 51 new features, so the gap measures coordination. Uncorrected utility scores higher on agreement because it releases far fewer addresses (39 versus 113 overall), preserving inherited structure rather than identifying deleted features.

Table 4: Release decisions and new-feature address agreement under three matched feature-change conditions.

(a) Release decisions

	Deleted: released ↑		Surviving: released ↓	
Variant	Ordinary	Similar	Rare	Other
LineageSAE	**105/105**	**5/5**	3/102	**0/1240**
Uncorrected utility	37/105	0/5	**2/102**	**0/1240**

(b) Address coordination

Variant	New at same address ↑	C–C J ↑	C–C cosine↑
LineageSAE	40.67/51	0.7757	0.9790
No decoder handoff	24.67/51	0.7281	0.9299
No response ordering	31.00/51	0.4953	0.7729
Uncorrected utility	**48.00/51**	**0.7882**	**0.9883**

Scope of transfer. Layer, width, and Qwen replications retain 91.39%–95.42% CE recovery. Appendix E.2 reports these results and additional baseline comparisons.

4 Related Work

Sparse feature vocabularies and reproducibility. Independent SAE runs can learn different dictionaries (Paulo & Belrose, 2025; Leask et al., 2025); reconstruction alone does not establish feature identity (Makelov et al., 2025). Weight regularization, Ordered SAE (OSAE), and Identifiable SAE (iSAE) improve reproducibility (Jedryszek & Crook, 2026; Wang et al., 2025; Nelson et al., 2026). Stable Feature Pool initializes children from pooled source dictionaries (Gerasimov et al., 2026). Archetypal SAEs constrain decoder atoms to convex combinations of training data, optionally allowing bounded departures (Fel et al., 2025). LineageSAE instead maintains identities relative to a supplied parent while making room for changed features.

Sparse coding architectures. TopK and BatchTopK impose sparsity budgets (Gao et al., 2025; Bussmann et al., 2025a); Gated SAE separates selection from magnitude estimation (Rajamanoharan et al., 2024a), and JumpReLU learns thresholds (Rajamanoharan et al., 2024b). AbsTopK selects signed coefficients by magnitude (Zhu et al., 2025); Matryoshka SAE trains nested dictionaries (Bussmann et al., 2025b); MP-SAE selects atoms successively from the remaining residual (Costa et al., 2025). These architectures provide feature-recovery comparisons at their measured sparsity.

Correspondence and shared feature spaces. Semantic optimal transport matches activation-weighted feature distributions (Cao et al., 2026), while universality studies recover related features across models (Lan et al., 2025). Universal and Procrustes-conditioned SAEs learn shared features jointly across views (Thasarathan et al., 2025; Váradi & Kmetty, 2026). SharedSAE combines model-specific encoders and decoders with shared identities and adapts new models to a frozen dictionary (Ognev et al., 2026). LineageSAE studies sequential continuity from a saved parent and current activations, including which existing assignments to release and how to reuse their indices.

Model adaptation and teacher transfer. SAE-FT and SAE-FD constrain language-model adaptation or forgetting through sparse features (Morelli et al., 2026; Zhang et al., 2026). We maintain the SAE vocabulary itself. The frozen parent supplies temporary decoder and response guidance, serving a teacher-like role (Hinton et al., 2015); deployment uses the child alone.

5 Conclusion

LineageSAE maintains a versioned vocabulary through parent-guided inheritance, selective release, and coordinated reuse. The experiments show address continuity alongside adaptation to changed features, domains, and model versions. Four successive generations retain substantial agreement with their root while accumulating a measurable, gradual drift, and the released addresses acquire recurring new roles at the same indices across independent runs. A single fixed release rule covers domain transfer, model updates, controlled feature changes, and repeated retraining without per-transition tuning. These results separate address continuity from feature recovery: inheritance preserves coordinates when parent directions remain supported, whereas release allows adaptation and ranking coordinates new roles without post-training assignment. The remaining root-relative drift, lower recovery for rare survivors, and Toy reconstruction tradeoff motivate better release tests, adaptive calibration, and causal evaluations of stable addresses. Scope, residual failure modes, and open directions are recorded in the supplementary appendix.

Reproducibility Statement

Appendix A provides mathematical derivations and training details for LineageSAE. Appendix B documents the datasets, data splits, optimization settings, random seeds, baseline implementations, and evaluation metrics. Appendix B.5 describes the software environment and commands for data preparation, training, and evaluation. Source code is available at `https://github.com/AnonymousSCI/LineageSAE`.

AI Use Statement

Generative AI tools were used primarily to implement experimental code, generate result plots and tables, and polish portions of the language. The authors developed the ideas and methodology, designed the experiments, wrote the manuscript, and checked the experimental results and final document. The authors reviewed the AI-assisted code, figures, tables, and text and take responsibility for the final contents of this paper.

References

Gilles Blanchard, Motoaki Kawanabe, Masashi Sugiyama, Vladimir Spokoiny, and Klaus-Robert Müller. In search of non-Gaussian components of a high-dimensional distribution. *Journal of Machine Learning Research*, 7(9):247–282, 2006. URL `https://jmlr.org/papers/v7/blanchard06a.html`.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. In *International Conference on Learning Representations*, 2025a. URL `https://proceedings.iclr.cc/paper_files/paper/2025/file/84ca3f2d9d9bfca13f69b48ea63eb4a5-Paper-Conference.pdf`.

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

Tue M. Cao, Nguyen Do, and My T. Thai. Semantic optimal transport for sparse autoencoder feature matching and circuit compression. *arXiv preprint arXiv:2605.28567*, 2026.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba E. Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=Ll5miDx8KB`.

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*, 2024.

Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E. Ba, and Talia Konkle. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 16543–16572, 2025.

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*, 2025.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

Piotr Jedryszek and Oliver M. Crook. Stable and steerable sparse autoencoders with weight regularization. *arXiv preprint arXiv:2603.04198*, 2026.

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*, 2025.

Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. In *International Conference on Learning Representations*, 2025.

Aleksandar Makelov, Georg Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. In *International Conference on Learning Representations*, 2025.

Fabian Morelli, Arnas Uselis, Ankit Sonthalia, and Seong Joon Oh. Sparse autoencoders enable robust and interpretable fine-tuning of CLIP models. *arXiv preprint arXiv:2605.15961*, 2026.

Walter Nelson, Theofanis Karaletsos, and Francesco Locatello. Toward identifiable sparse autoencoders. *arXiv preprint arXiv:2605.31245*, 2026. URL https://arxiv.org/abs/2605.31245.

Daniil Ognev, Célian Vasson, Lijie Hu, Kentaro Inui, and Benjamin Heinzerling. SharedSAE: One feature dictionary across language models. *arXiv preprint arXiv:2609.04344*, 2026. URL https://arxiv.org/abs/2609.04344.

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

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/01772a8b0420baec00c4d59fe2fbace6-Abstract-Conference.html.

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*, 2024b.

Hiroaki Sasaki, Gang Niu, and Masashi Sugiyama. Non-Gaussian component analysis with log-density gradient estimation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 1177–1185. PMLR, 2016. URL https://proceedings.mlr.press/v51/sasaki16.html.

Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981. doi: 10.1214/aos/1176345632.

Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos G. Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59304–59325, 2025.

Bendegúz Váradi and Zoltán Kmetty. Cross-seed explainability using procrustes-conditioned joint end-to-end TopK sparse autoencoders. *arXiv preprint arXiv:2607.08499*, 2026.

Sophie L. Wang, Alex Quach, Nithin Parsan, and John J. Yang. Enforcing orderedness to improve feature consistency. *arXiv preprint arXiv:2512.02194*, 2025.

Mingxu Zhang, Yuhan Li, Lujundong Li, Dazhong Shen, Hui Xiong, and Ying Sun. SAE-FD: Sparse autoencoder feature distillation for continual learning of large language models. *arXiv preprint arXiv:2605.25525*, 2026.

Xudong Zhu, Mohammad Mahdi Khalili, and Zhihui Zhu. AbsTopK: Rethinking sparse autoencoders for bidirectional features. *arXiv preprint arXiv:2510.00404*, 2025. URL https://arxiv.org/abs/2510.00404.

Appendix of LineageSAE: Versioned Feature Vocabularies across SAE Retraining and Model Updates

A Method Details

A.1 The Corrected Objective and Its Derivative

The boundary response in Section 2.3 can equivalently be written

$$r_{t,i} = \mathbf{1}\{i \in \mathrm{TopK}(t)\}[a^P_{t,i} - \max\{a^P_{t,(k+1)}, 0\}]_+. \tag{7}$$

The indicator is redundant because every excluded score lies below or at the boundary. The gradient includes changes in the target score and in the strongest excluded score when that boundary is positive. For each fold, estimate the input mean, response means and variances, input covariance Σ, and corrected associations

$$C_i = \mathrm{Cov}(h, r_i) - \Sigma\mathbb{E}[\nabla_h r_i]. \tag{8}$$

For target i on fitting fold A, the other addresses with positive response variance and nonzero corrected associations provide candidate unit directions. Let $u_{i,A}$ be the candidate with the largest positive projection onto $C_{i,A}$. The background coefficient is $b_{i,A} = u_{i,A}[u_{i,A}^\top C_{i,A}]_+ / \mathrm{Var}_A(r_i)$. If either fitting fold lacks positive target variance or a usable background, no aggregate utility is estimated. For the decision in Eq. (4), we represent this case by $\widehat{q}_i = 0$. This is a neutral decision value, not a measured zero utility.

Define $X^A_t = h_t - \bar{h}_A$ and $H^A_{t,i} = r_{t,i} - \bar{r}_{i,A}$ using fold-A means. On evaluation fold B, let $G_{i,B} = \Sigma_B\mathbb{E}_B[\nabla_h r_i]$, where Σ_B is centered using fold B's own input mean. For the fitted background $b_{i,A}$, the corrected objective is

$$J_{i,A\to B}(a) = \mathbb{E}_B\left[\|X^A_t - (b_{i,A} + ad^P_i)H^A_{t,i}\|_2^2\right] + 2(b_{i,A} + ad^P_i)^\top G_{i,B}. \tag{9}$$

The cross-fitted marginal utility is

$$\begin{aligned} q_{i,A\to B} &= -\frac{1}{2}\left.\frac{\partial J_{i,A\to B}(a)}{\partial a}\right|_{a=0} \\ &= (d^P_i)^\top\left(\mathbb{E}_B[X^A_t H^A_{t,i}] - b_{i,A}\mathbb{E}_B[(H^A_{t,i})^2] - G_{i,B}\right). \end{aligned} \tag{10}$$

When both directional estimates are defined, we exchange the folds and use $\widehat{q}_i = q_{i,A\to B} + q_{i,B\to A}$ in Eq. (4). These evaluation moments retain fold-A centering; simply substituting fold-B covariances and variances into the population expression in Eq. (3) would omit mean-shift terms. A small positive coefficient along the original parent direction is beneficial when $q > 0$. Release requires strictly negative aggregate utility, or a response that is zero on all scoring samples. The latter condition is $\widehat{p}_i = 0$, where $\widehat{p}_i$ is the positive-response fraction across both folds. The neutral decision value retains responding addresses when statistics are insufficient. An observed-to-reference ratio below one is equivalent to negative utility only when its aggregate reference is positive.

Frequency dependence. Let $p_i = \Pr(r_i > 0) > 0$ and $\bar{h} = \mathbb{E}[h]$. Away from nondifferentiable selection boundaries, r_i and its gradient are zero outside its positive response set. Consequently,

$$C_i = p_i c_i, \qquad c_i = \mathbb{E}[(h - \bar{h})r_i - \Sigma\nabla_h r_i \mid r_i > 0]. \tag{11}$$

For population fitting and evaluation under the same distribution, a selected unit background direction u_i gives

$$q_i = p_i(d^P_i)^\top\left(c_i - u_i[u_i^\top c_i]_+\right). \tag{12}$$

Thus a smaller positive frequency factor alone does not reverse the utility sign. This is not invariance to a view change: the conditional association, input covariance and selected background can also change. Nor does it guarantee retention under finite sampling. A rare response may go unobserved, and noisy cross-fitted utilities may have the wrong sign.

A.2 Response Sensitivity and Its Association Reference

This section derives the reference in Eq. (3) from the response's selection margin, then gives its linear and Gaussian interpretations. The reference is estimated from current-view samples.

Differentiating the selection margin. Let e_i^P be the parent encoder weight vector for address i, so that $a_{t,i}^P = (e_i^P)^\top (h_t - b^P)$, with parent decoder bias b^P. Let j_t index the strongest excluded score, and let $\mathbf{1}\{\cdot\}$ denote an indicator. Away from score ties and zero boundaries,

$$\nabla_h r_{t,i} = \mathbf{1}\{r_{t,i} > 0\} \left(e_i^P - \mathbf{1}\{a_{t,j_t}^P > 0\} e_{j_t}^P\right). \tag{13}$$

The second term tracks movement of the competing boundary whenever it is positive; otherwise the threshold is zero and only the target score varies. The positive part makes the response continuous at its activation boundary, and the full response is a continuous, piecewise affine function of the input. Its almost-everywhere gradient therefore incorporates the competition while assigning zero gradient to zero responses away from boundaries.

For a fold of n samples, define $\bar{h} = n^{-1} \sum_t h_t$ and $\widehat{\Sigma} = n^{-1} \sum_t (h_t - \bar{h})(h_t - \bar{h})^\top$. The implemented reference is

$$\widehat{G}_i = \widehat{\Sigma} \left(\frac{1}{n} \sum_t \nabla_h r_{t,i} \right) = \widehat{\Sigma} \left[\widehat{p}_i e_i^P - \frac{1}{n} \sum_{t : r_{t,i} > 0} \mathbf{1}\{a_{t,j_t}^P > 0\} e_{j_t}^P \right], \tag{14}$$

where here $\widehat{p}_i = n^{-1} \sum_t \mathbf{1}\{r_{t,i} > 0\}$ is the fold's positive-response fraction. Averaging over all samples incorporates response frequency as well as the distribution of competing addresses. $\widehat{G}_i$ is an address-specific association vector, not a scalar measure of sensitivity or a gradient-norm penalty.

An exact linear-reference interpretation. Let $\mathbb{E}$ and Cov refer to the current population, $\mu_h = \mathbb{E}[h]$, $\Sigma = \mathrm{Cov}(h)$, and $\bar{g}_i = \mathbb{E}[\nabla_h r_i(h)]$. Define a linear reference response with the same mean and average gradient:

$$r_i^{\mathrm{lin}}(h) = \mathbb{E}[r_i] + \bar{g}_i^\top (h - \mu_h). \tag{15}$$

Its input association is $\mathrm{Cov}(h, r_i^{\mathrm{lin}}) = \Sigma \bar{g}_i$. Consequently, the corrected association obeys

$$C_i = \mathrm{Cov}(h, r_i) - \Sigma \mathbb{E}[\nabla_h r_i] = \mathrm{Cov}\left(h, r_i - r_i^{\mathrm{lin}}\right). \tag{16}$$

This algebraic identity holds without assuming Gaussian inputs. The reference slope is the mean response gradient, not a least-squares coefficient; the remaining response need not be linearly uncorrelated with the input. The identity re-expresses the statistic without replacing the SAE's responses.

Gaussian interpretation through Stein's identity. For $h \sim \mathcal{N}(\mu_h, \Sigma)$, Gaussian integration by parts gives

$$\mathrm{Cov}(h, r_i(h)) = \Sigma \mathbb{E}[\nabla_h r_i(h)], \tag{17}$$

under the usual integrability conditions, using the weak gradient of the continuous piecewise affine response (Stein, 1981). Thus $C_i = 0$ at the Gaussian population level: the reference accounts for the association induced by applying the response function to Gaussian inputs. For example, $x \sim \mathcal{N}(0, 1)$ and $r(x) = [x]_+$ give $\mathrm{Cov}(x, r) = 1/2 = \mathbb{E}[r'(x)]$.

In the implementation, both covariance and mean gradient are estimated on the actual current-view data. For a non-Gaussian distribution D and a Gaussian distribution G with the same mean and covariance, $\Sigma \mathbb{E}_D[\nabla r_i]$ need not equal $\mathrm{Cov}_G(h, r_i) = \Sigma \mathbb{E}_G[\nabla r_i]$. The method neither draws Gaussian samples nor estimates a noise covariance.

Connection to non-Gaussian component analysis. When Σ is positive definite, whiten the input as $y = \Sigma^{-1/2}(h - \mu_h)$, so that $\mathbb{E}[y] = 0$ and $\mathrm{Cov}(y) = I$, and write the response in these coordinates as $f_i(y) = r_i(\mu_h + \Sigma^{1/2} y)$. The chain rule gives

$$\Sigma^{-1/2} C_i = \mathbb{E}[y f_i(y) - \nabla_y f_i(y)]. \tag{18}$$

For singular covariance after projection, the same identity holds in orthonormal coordinates on the covariance support, where it is invertible. The right-hand side has the form of the operator used in non-Gaussian component analysis (NGCA) (Blanchard et al., 2006; Sasaki et al., 2016). Under the density-model and regularity assumptions of NGCA, this operator identifies the low-dimensional non-Gaussian subspace containing the response. Here it supplies a response-specific association statistic, which the release rule evaluates against the corrected background.

A.3 Parent-Relative Decoder Constraint

Let $\hat{d}_i^C$ and d_j^P be unit decoder columns. The cosine margin and atomwise weight are

$$\mu_i = \langle \hat{d}_i^C, d_i^P \rangle - \max_{j \neq i} \langle \hat{d}_i^C, d_j^P \rangle, \qquad \omega_{i,s} = m_i + (1 - m_i)\beta_s. \tag{19}$$

The region penalty applies a weighted squared hinge $\omega_{i,s}[\delta - \mu_i]_+^2$ to child decoder atoms. The frozen parent receives no gradient. Unlike the temporary decoder interpolation, this training regularizer remains active on inherited addresses; parent parameters are unnecessary for the deployed child's forward computation.

A.4 Root Projection and Reconstruction

The covariance of Dz_t weights decoder directions by their actual use; the unweighted statistic $DD^\top$ does not. Orthonormalization of generalized eigenvectors is required before forming $P = I - UU^\top$. The exact bypass restores token-varying components without assigning them feature addresses. Provided $\widehat{h}_t \in U^\perp$, the original-space and residual-space squared errors are equal. The two normalized errors measure different quantities: full-activation NMSE divides by activation energy and residual-input NMSE by residual energy (Appendix F.4), and both are reported. Projected-root refinement starts from the ordered projected parameters and retains AuxK; it does not initialize a new vocabulary. The random-initialization control is evaluated in Table 14. Each original decoder atom d_i is replaced by $d_i^{\mathrm{root}} = Pd_i / \|Pd_i\|_2$ when its projection is nonzero.

A.5 Training and Deployment

The main method applies Eq. (6) to all released addresses. The following specification gives its optimization and deployment schedule, followed by the residual-learning, inherited-response, and assignment objectives used in the component studies. The child reconstruction uses its own sparse coefficients with the effective decoder in Eq. (5). The auxiliary native-decoder loss is

$$\mathcal{L}_{\mathrm{take}} = \frac{1}{d}\mathbb{E}_t \left\| \mathrm{sg}(h_t - b^C) - \sum_i \mathrm{sg}(z_{t,i}^C) d_i^C \right\|_2^2. \tag{20}$$

Only child decoder atoms receive its gradients. The main-text total objective is $\mathcal{L}_{\mathrm{rec}} + \beta_s \mathcal{L}_{\mathrm{take}} + \eta_{\mathrm{region}} \mathcal{L}_{\mathrm{region}} + \eta_{\mathrm{aux}} \mathcal{L}_{\mathrm{AuxK}} + \mathcal{L}_{\mathrm{rank}}$. The residual-learning and address-cost controls additionally use $\mathcal{L}_{\mathrm{reuse}}$ and, where enabled, $\tau u_s \mathcal{A}_B / d$, respectively. Here $\mathcal{L}_{\mathrm{rec}}$ is mean squared reconstruction error, $\mathcal{L}_{\mathrm{region}}$ is the sum of weighted decoder hinges above divided by $\max\{\sum_i \omega_{i,s}, 1\}$, and AuxK exposes inactive addresses to reconstruction gradients. The synthetic protocol uses 15,000 updates, batch size 512, learning rate 3×10^{-4}, no weight decay, gradient clipping at one, $\delta = 0.01$, region weight one, dead threshold 200,000 token presentations, auxiliary sparsity 512, and AuxK weight $1/32$.

For exact schedules, define

$$c(s; a, b) = \begin{cases} 1, & s \leq a, \\ \frac{1}{2}[1 + \cos(\pi(s-a)/(b-a))], & a < s < b, \\ 0, & s \geq b. \end{cases} \tag{21}$$

We use $\beta_s = c(s; 6000, 9000)$, $u_s = \mathbf{1}\{s \geq 3000\} c(s; 6000, 14000)$, and $\lambda_s = 1 + 23c(s; 3000, 11250)$. The dictionary prior reaches a final strength of 1. At child creation, inherited encoder and unit decoder vectors are copied from the parent; released rows initially use random parameters, and the decoder bias is initialized to the current training-input mean. At step 3,000, released encoder and decoder rows are reset to their parent values and their Adam moments cleared. The mask is never recomputed.

Residual-learning objective. Let $\mathcal{I} = \{i : m_i = 1\}$ and $\mathcal{R} = \{i : m_i = 0\}$. The frozen parent residual is $\xi_t = h_t - b^P - \sum_{i \in \mathcal{I}} z_{t,i}^P d_i^P$, where b^P is the parent decoder bias and $z_{t,i}^P$ its full-vocabulary TopK coefficient. For n samples, Y and Z contain centered residuals and released child coefficients as rows. The residual-learning comparison uses

$$\mathcal{L}_{\mathrm{reuse}} = \frac{u_s}{nd} \min_V \left\{ \|Y - ZV^\top\|_F^2 + \lambda_s \|V - D_{\mathcal{R}}^P\|_F^2 \right\}, \tag{22}$$

where V is an auxiliary decoder, $D^P_{\mathcal{R}}$ contains the parent released atoms, and $\|\cdot\|_F$ is the Frobenius norm. The weight u_s and reference strength λ_s follow the schedules above; this loss is disabled in the main-text method. For Eq. (22), the minimizer satisfies $V_*^\top = (Z^\top Z + \lambda_s I)^{-1}(Z^\top Y + \lambda_s (D^P_{\mathcal{R}})^\top)$. We solve this system in double precision, detach the optimum when evaluating the minimized objective, and propagate gradients only to released encoder rows. Full-vocabulary TopK still includes inherited rows in the competition. The address-cost method retains native indices after all 15,000 updates. The residual-learning configuration omits $\mathcal{A}_B$ and uses training-data-only parent-relative assignment instead (Appendix A.8). The final 1,000 updates have no parent decoding, response supervision, residual-reuse loss or address cost; the inherited decoder-region regularizer remains. The 12,000-step language-model component experiments use their own schedules in Appendix B.

A.6 Shared Response Ordering

The main-text method applies ordering to all released addresses $\mathcal{R}$. The residual-learning comparison restricts ordering to the zero-boundary-response set $\mathcal{Z} = \{i : \widehat{p}_i = 0\}$, denoted G_1 in the diagnostics. For M training samples indexed from zero, pair sample t with $t' = M - 1 - t$ in every child. Let $\Delta^P_{tt'i} = a^P_{t,i} - a^P_{t',i}$ and let σ^P_i be the parent score standard deviation over the training set, lower-bounded by 10^{-5}. We retain triples satisfying $|\Delta^P_{tt'i}| > 0.1\sigma^P_i$. The parent score is $a^P_{t,i} = (e^P_i)^\top (h_t - b^P)$. The child score is $a^C_{t,i} = (e^C_i)^\top (h_t - \mathrm{sg}(b^C))$, where e^P_i, e^C_i are encoder rows and b^C is the child decoder bias. Both scores precede ReLU and TopK.

The child scale is $\sigma^C_{i,B} = \max\{\mathrm{Std}_{t\in B}(a^C_{t,i}), 10^{-5}\}$, using the sample standard deviation. It is detached, as is $v_B = (|B|d)^{-1} \sum_{t\in B} \|h_t - b^P\|_2^2$. The main-text weight is $w_{s,B} = \eta u_s v_B$. We set $\eta = 0.1$ and $\mu = 0.5$ in Eq. (6). An empty set $\mathcal{P}_B$ contributes zero loss. The frozen parent, child bias and normalization scales receive no gradients from this term. Only child encoder rows in $\mathcal{R}$ are updated in the main-text method ($\mathcal{Z}$ in the residual-learning comparison).

Inherited-response regression. The inherited-response control replaces $\mathcal{L}_{\mathrm{rank}}$ with

$$\mathcal{L}_{\mathrm{response}} = \alpha\beta_s v_B \frac{\sum_{t\in B}\sum_{i\in\mathcal{I}} \mathbf{1}\{z^P_{t,i} > 0\}(a^C_{t,i} - \mathrm{sg}(z^P_{t,i}))^2}{\mathrm{sg}\big(\max\{\sum_{t\in B}\sum_{i\in\mathcal{I}}(z^P_{t,i})^2, \epsilon\}\big)}, \tag{23}$$

with $\alpha = 0.5$ and $\epsilon = 10^{-8}$. This term updates inherited encoder rows only. This term is disabled in the main method, the shared-ordering comparison, and the address-cost control. Inherited-response regression uses no $\mathcal{A}_B$.

A.7 Training with the Address Cost

This control adds a training-time cost for the residual direction associated with each released address. Regressing the fixed residual ξ on released child activation z^C_i with an intercept gives slope $\psi_i = \mathrm{Var}_B(z^C_i)^+ \mathrm{Cov}_B(\xi, z^C_i)$, where $+$ denotes the pseudoinverse and zero variance gives zero slope. With $\omega_i = \mathbb{E}_B[(z^C_i)^2]$, the address cost is

$$\mathcal{A}_B = \sum_{i\in\mathcal{R}} \mathbb{E}_B \|z^C_i \psi_i - z^C_i d^P_i\|_2^2 = \sum_{i\in\mathcal{R}} \omega_i \|\psi_i - d^P_i\|_2^2. \tag{24}$$

It is weighted by $\tau u_s/d$, where $\tau \geq 0$ controls its strength. For a batch B of n samples, compute centered covariance and variance with divisor n for each released child activation and the fixed parent residual. Use uncentered energy $\omega_i = n^{-1}\sum_{t\in B}(z^C_{t,i})^2$. Neither ψ_i nor ω_i is normalized. All moments are accumulated in double precision. With $q = |\mathcal{R}|$ and double-precision machine epsilon ϵ_{64}, the numerical pseudoinverse sets $\psi_i = 0$ when $\mathrm{Var}_B(z^C_i) \leq q\epsilon_{64} \max_j \mathrm{Var}_B(z^C_j)$; otherwise it divides the covariance vector by the variance. No ridge or moving-average estimator is added to this univariate regression.

Differentiate Eq. (24) through covariance, variance and energy. The detached-solution envelope derivative used for the joint ridge objective does not apply to this different cost. On the address-cost branch, detach inherited encoder rows, all decoder parameters and the input; retain full-vocabulary TopK competition and update released encoder rows only. The initial synthetic pilot uses $\tau = 0.1$ and the same u_s as residual reuse. Minibatch ratios differ from the full-training-data regression used in parent-relative assignment and are not unbiased estimates of that statistic.

A.8 PARENT-RELATIVE ASSIGNMENT

The residual-learning and inherited-response comparisons apply the following assignment after training. Table 1 also retains this assignment, whereas Table 2 uses the learned native addresses without assignment. The address-cost variant does not use it either. The residual ξ is fixed by the frozen parent's inherited reconstruction, as defined above. For each released child feature j, use current-view training samples to estimate the univariate least-squares coefficient and activation energy $\psi_j = \mathrm{Var}(z_j^C)^+ \mathrm{Cov}(\xi, z_j^C)$ and $\omega_j = \mathbb{E}[(z_j^C)^2]$, respectively. Here $+$ denotes the Moore–Penrose pseudoinverse; ψ_j is not normalized. The assignment is

$$\pi^* = \arg\min_{\pi:\mathcal{R}\xrightarrow{\mathrm{bij}}\mathcal{R}} \sum_{j\in\mathcal{R}} \omega_j \|\psi_j - d^P_{\pi(j)}\|_2^2, \tag{25}$$

where bij denotes a one-to-one assignment. Jointly permuting released encoder rows and decoder atoms preserves the reconstruction function. Inherited addresses remain fixed, and no other child or evaluation sample is used to determine the permutation. This procedure minimizes the full-training-data address cost over permutations of a fixed learned vocabulary. Directly optimizing Eq. (24) during training instead changes activations and learned parameters; the two optimization procedures are not equivalent. The joint ridge-regression map V in Eq. (22) is distinct from the univariate regression vectors ψ_j used here.

B EXPERIMENTAL SETUP AND REPRODUCIBILITY

The appendix first specifies data, metrics, implementations, and the reproduction workflow. It then examines inheritance and root construction, controlled release and reuse, transfer and successive generations, and complete baseline measurements. This organization separates the evidence for each mechanism while retaining all reported operating points.

B.1 DATASETS, VIEWS, AND DATA SPLITS

A *view* is the distribution of activations on which an SAE is trained; it can change through the input corpus or the underlying language model. An address is a fixed feature index in an SAE dictionary. We evaluate parent–child (P–C) continuity across views and child–child (C–C) agreement between independently trained descendants.

Language-model activations. Wiki uses WikiText-103-derived text, and Code uses the Python portion of the Stack Smol XS. Text preparation normalizes whitespace for duplicate checking, shuffles documents with a fixed seed, and partitions them into 80% training, 10% calibration, and 10% validation documents. The tokenizer appends an end-of-sequence token between documents and forms blocks of 128 tokens. Activation extraction records token identities, sequence identities, and positions so that model versions can be evaluated on paired inputs. Splits remain disjoint during correspondence fitting and scoring. The fixed-view Wiki experiment contains 199,936 training and 29,184 held-out activation vectors.

The primary language model is Pythia-1.4B-deduped, using layer 12 with activation dimension $d = 2048$, dictionary width $N = 4096$, and TopK budget $k = 16$. The update experiment further trains this model for 300 updates on Wiki with learning rate 5×10^{-6}, weight decay 0.01, batch size four, and 128-token sequences, for 153,600 token presentations. Children train on updated-model activations with the frozen parent SAE; paired old/new activations are used for evaluation. Wiki-to-Code P–C evaluation instead applies both encoders to the same held-out Code inputs. Additional experiments vary Pythia's layer or dictionary size and use Qwen2.5-0.5B.

Controlled feature changes. SynthSAEBench represents 512 generating features in $d = 128$ dimensions, with $(N, k) = (1024, 16)$, and replaces 51 features in each update. Six conditions cross three task groups with unchanged survivor frequencies (R0) or reduced frequencies for 51 survivors (R1). Each condition contains 65,536 training, 32,768 calibration, and 32,768 test samples. R0 has 461 ordinary survivors; R1 has 410 ordinary and 51 rare survivors.

Toy superposition represents 128 features in $d = 32$ dimensions, with $(N, k) = (256, 8)$. Each update replaces eight features and reduces eight survivor frequencies from 0.05 to 0.005; the re-

maining 104 survivors keep their frequencies. The splits contain 131,072 training, 32,768 annotation/calibration, and 65,536 test samples. Parent seeds 101–103 share one generating representation. The controlled comparison uses parent-specific rank-two preprocessing for every method and includes the shared component when evaluating original-space reconstruction.

B.2 Optimization, Initialization, and Replication

LineageSAE children use 15,000 updates, batch size 512, learning rate 3×10^{-4}, zero weight decay, and gradient clipping at one. The encoder has no independent bias; the decoder has a learned bias. The region margin is 0.01 and its loss weight is one. AuxK, the auxiliary reconstruction loss for inactive features, uses weight $1/32$, auxiliary sparsity 512, and a dead-feature threshold of 200,000 token presentations. The initialization, reset at step 3,000, decoder handoff, and response-ranking schedules are specified in Appendix A.5.

The selected language-task comparisons use one parent and two children, giving two P–C pairs and one C–C pair. Controlled LineageSAE comparisons average the child pairs within each condition and then give equal weight to conditions. The additional three-child studies use seeds 1, 2, and 3, giving three unordered C–C pairs. Where independent baselines also have three parent seeds, their P–C score averages nine parent/child pairs. Seed pairs sharing a model are dependent; conditions and training runs define the replication units. Means with error bars use the aggregation specified for that experiment, and the CE audit reports mean $\pm$ sample standard deviation over three children.

Baseline training retains each architecture's optimization and sparsity mechanism. The expanded grid uses recorded completed checkpoints and caps new runs at 10,000 updates. In the selected Wiki-to-Code comparison, Gated SAE uses full activations and 10,000 updates for all three models. For SynthSAEBench, BatchTopK uses a 15,000-update parent and two 10,000-update children; Gated SAE uses 15,000 updates for each model. Both are evaluated on the six paired conditions. AbsTopK on Toy uses full activations and 10,000 updates for each model. MP-SAE selects the training-development minimum-NMSE setting on SynthSAEBench and configuration 0 on Toy. All reported L_0 values are measured operating points, including methods with learned rather than fixed sparsity.

B.3 Metrics and Aggregation

For address i, let A_i and B_i be its firing-sample sets in two SAEs. Support Jaccard is $J_i = |A_i \cap B_i|/|A_i \cup B_i|$; an empty union contributes zero. Spearman correlation ρ_i ranks all activation magnitudes, including zeros, with average ranks for ties; an undefined correlation contributes zero. We average over the fixed address count, then over model pairs, then equally over conditions. An active address fires at least once on the evaluation split, and L_0 is the mean number of active features per input. Coverage is $|A_i \cap B_i|/|A_i|$ for a parent firing set A_i; precision uses the child firing set B_i in the denominator. Zero denominators contribute zero.

Decoder and response correspondence. Native decoder cosine compares unit decoder vectors at deployed addresses. The main language-task scores average addresses within each pair and then pairs. A reported pairwise median first takes the median over addresses within a pair. Response-profile cosine instead compares centered mean representations of the tokens on which an address fires, as defined in Appendix C.1. Calibration matching fits a one-to-one Hungarian assignment using Pearson activation correlation on the calibration split and fixes this assignment before test evaluation. It is a separate diagnostic from training-data assignment of released addresses. Wiki-to-Code LineageSAE uses the latter assignment with inherited addresses fixed; the controlled main comparison uses learned indices directly. Under domain shift, decoder and response metrics measure complementary changes in feature directions and usage.

Reconstruction and language-model behavior. Normalized mean squared error (NMSE) divides summed reconstruction error by the centered energy of the evaluated inputs. For a projected SAE, full reconstruction restores the frozen shared component before computing the original-activation denominator (Appendix F.4). The expanded grid additionally reports residual-input NMSE, which uses residual energy as its denominator and is defined in Appendix F.4. AbsTopK on Wiki-to-Code and Gated SAE, iSAE, and AbsTopK on the Pythia update report residual-input NMSE in the selected comparisons. The Pythia-update BatchTopK run uses its own recorded operating point.

Let ℓ_{clean}, ℓ_{zero}, and ℓ_{SAE} denote next-token cross-entropy (CE) with the clean activation, a zeroed activation site, and the SAE reconstruction at that site. CE recovery is $100(\ell_{\text{zero}} - \ell_{\text{SAE}})/(\ell_{\text{zero}} - \ell_{\text{clean}})$. Identity replacement checks the intervention by reinserting the captured activation; cache relative root mean squared error (RMSE) compares stored and live activations. Raw CE is compared on identical models, inputs, and targets. Missing measurements are denoted by dashes.

Presentation. Light shading identifies LineageSAE rows. For comparable methods and metrics within a task, bold and underline indicate the best and second-best distinct values, with ties marked equally. Ranking applies to comparable methods and metrics within a task; counts, L_0, and cross-task summaries are reported as measured values.

B.4 Baseline Implementations and Model Access

Table 5 groups methods by the experimental question they address. L2-TopK denotes the L_2-regularized TopK SAE; RA-SAE denotes Relaxed Archetypal SAE. All methods are evaluated on the same task splits within a comparison. Their native objectives, initialization procedures, sparsity mechanisms, and required model access define each baseline.

Table 5: Baseline families and their roles in the evaluation.

Family	Methods	Evaluation role
Sparse-coding architectures	BatchTopK, AbsTopK, Gated SAE, Matryoshka SAE, MP-SAE	Measured on all four tasks; additional settings in Tables 20 and 21
Stability and identifiability	iSAE, L2-TopK, OSAE, Archetypal SAE, RA-SAE	Measured on all four tasks; candidate-pool diagnostics in Table 33
Feature pooling	Stable Pool	Three source SAEs and two children; source cost in Table 7
Post-hoc correspondence	Decoder-Hungarian, Activation-Hungarian, Semantic OT	Correspondence of fixed trained dictionaries
Shared representations	SharedSAE	Joint old/new-view training on all four tasks
Mechanism controls	Independent retraining; parent-copy continuation; frozen parent decoder	Successive generations in Table 25

Independent and shared dictionaries. TopK, L2-TopK, OSAE, and Stable Pool use the project implementations of their published objectives; RA-SAE uses its upstream implementation with the benchmark data adapter. The iSAE implementation follows the paper; its stochastic-regularizer estimator is recorded with the run configuration. SharedSAE jointly trains old- and new-view branches. Its language-task scores average two old/new branch pairs and one pair of new branches from different runs; the Pythia update uses three joint checkpoints. Toy samples the parent and updated distributions independently for its two branches. The independently sampled Archetypal candidate pools supply the native decoder cosines; the shared-candidate checkpoint set supplies response, reconstruction, CE, and calibration-matched measurements. Appendix F.5 evaluates the effect of this candidate construction directly.

Stable Feature Pool constructs one pool from three old-view source SAEs and initializes two new-view children from it. An old-view-only map puts each source dictionary in pool order, giving six P–C pairs; C–C compares the two children at their deployed pool addresses. CE recovery averages the two children. Source training is counted separately from child training. The controlled independent-pool experiment in Appendix D.1 instead gives each child its own source-seed group. Semantic Optimal Transport (Semantic OT) maps fixed L2-TopK dictionaries after training. Decoder-Hungarian and activation-Hungarian matching are evaluation procedures applied to trained SAEs.

BatchTopK. The upstream version is commit `b9aab1c6156381ae7ae2997e3490e7b99e195dde`, with `transformer-lens` 1.17.0, `datasets` 2.18.0, `einops` 0.7.0, and `wandb` 0.30.0 in its frozen environment. Training retains the batch top-k rule, initialization, optimizer, schedules, dead-feature loss, and checkpoint selection. Finite Wiki/Code activation streams are cycled to the specified budget. The 15,000-update comparison uses batches of 512; its corpora contain

approximately 3.4k and 1.9k 128-token sequences. Selected comparisons use a held-out-calibrated scalar threshold at inference. The additional architecture study in Table 20 records the batchwise operating point.

The paper-setting validation uses GPT-2 Small residual-stream layer 8, $(N, k) = (12288, 32)$, OpenWebText, learning rate 3×10^{-4}, batch size 4096, and 10^9 training tokens. The post-loop checkpoint is step 244139. The threshold is the mean smallest positive selected activation over 20 calibration batches of 1024 tokens. On the subsequent 65,536 tokens, $\theta = 0.7176$, mean $L_0 = 31.799$, and 12,286 of 12,288 addresses are active. Original-space NMSE is 0.010680 and CE recovery is 98.38% over 65,024 next-token targets. Capture and intervention both use `blocks.8.hook_resid_pre`, with zero identity-replacement logit difference. These measurements use a restarted training stream and assess in-distribution reconstruction. Checkpoint hashes and evaluator versions identify the corresponding run.

Gated SAE. The implementation is `saprmarks/dictionary_learning` commit `60ec6bf5264944d64a4ca271f45a29ebfb9d4946`. `GatedAutoEncoder` separates gating and magnitude networks and applies the ℓ_1 penalty to the gate. `GatedAnnealTrainer` uses p-annealing, `ConstrainedAdam` with betas $(0.0, 0.999)$, learning-rate and sparsity warmups, and dead-neuron resampling. Precomputed activation arrays replace the streaming data interface, and inference uses the native gate. A training-development penalty sweep selects the operating point closest to the comparison's sparsity. In the GPT-2 extension, $\alpha = 1.8$ gives child mean $L_0 = 78.8$, compared with BatchTopK's 94.2 and the fixed-k rows. CE recovery is 98.29% at this setting and 83.16% at matched $k = 16$.

Implementation records. The source inventory is summarized in Table 6. OSAE's additional upstream replication requires a released implementation; the measured ordered-SAE control uses the project implementation. The reference model/data settings include license-controlled Gemma-2-2B, public Pythia-70M and Pile, and the Dolma crawl manifest. TopK, iSAE-ME, and Crosscoder records are supplementary to the selected four-task inventory.

Table 6: Implementation sources for the additional architecture comparisons.

Method	Implementation and data interface	Evaluation record
BatchTopK	Pinned upstream implementation; OpenWebText ActivationStore	Measured task extensions and paper-setting validation
Gated SAE	Authors' gated objective and model setup	Measured task extensions
OSAE	Project implementation of nested dropout; Pile/Dolma reference setting	Project control measured; upstream replication pending

B.5 Reproduction Workflow and Computational Cost

Environment and inputs. The project specifies Python ≥ 3.10, PyTorch $\geq 2.2, < 3$, NumPy $\geq 1.26, < 3$, SciPy $\geq 1.11, < 2$, Transformers $\geq 4.46, < 5$, PyArrow $\geq 15, < 22$, PyYAML $\geq 6, < 7$, and Safetensors $\geq 0.4, < 1$. Per-run source snapshots and environment records identify the versions used for each result; baseline-specific environments are described above. The language-model stage produces reusable activation arrays, after which SAE training reads the arrays without keeping the language model in memory. Each run requires the parent checkpoint, projection basis and center, training/calibration/test arrays, configuration, and random seeds.

Execution order. From the repository root, install the project with `pip install -e .`. The `prepare_text.py` and `extract_activations.py` scripts in `scripts/` accept `--config`; the extraction script also accepts `--view`. Configurations specify local data and model paths, the hook layer, sequence length, and output directories. Resolve these paths before extraction and retain the emitted sample identifiers. For a trained parent and current-view residual array, release estimation is:

```
python scripts/compute_current_view_release.py \
  --parent PARENT.pt --training-input TRAIN.npy \
```

```
  --output RELEASE_DIR --view wiki_to_code \
  --sample-count 32768 --sample-seed 20260921 --device cuda:0
```

Uppercase arguments denote local artifact paths. This command writes the release mask, fold statistics, input hashes, and a manifest. Use a fresh output directory. The scoring sample is split equally for cross-fitting; the resulting mask is fixed for the children of that parent.

The supplementary residual-learning runs use `scripts/lineagesae_b_cross_view.py`. Its preparation stage reads the release manifest and the training configuration from a reference run; its training stage checks sample alignment and writes native and assigned checkpoints. The following commands reproduce one child of that study:

```
python scripts/lineagesae_b_cross_view.py prepare \
  --release-dir RELEASE_DIR --reference-manifest REFERENCE.json \
  --repo-root . --output PREPARED_DIR --device cuda:0
python scripts/lineagesae_b_cross_view.py train \
  --prepared PREPARED_DIR --seed 1 --output CHILD_DIR --device cuda:0
```

Repeat the training command for the recorded seeds in separate output directories. The full-address response-ranking objective uses `src/anchorsae/shared_response_ordering.py` with the selected released address IDs; its mathematical definition, initialization, and disabled auxiliary terms are specified in Appendix A.5. This is a separate training configuration from the residual-learning command above.

Evaluation and result records. `scripts/evaluate_c69_cross_view.py` accepts the parent, repeated `--child SEED=CHECKPOINT` arguments, separate parent/child calibration arrays and validation arrays, a task name, device, and output path. Correspondence fitting uses calibration arrays; scores use validation arrays. `scripts/evaluate_b_cross_view.py` additionally selects `--coordinates native` or `--coordinates assigned` and reconstructs the original activation space. The recursive entry point is `scripts/run_b_recursive.py`; each generation consumes the preceding deployed checkpoint and recomputes its release mask. The expanded tables derive from `baseline_grid_results.json` under `results/baseline_grid_2026-09-24/`; full-activation NMSE is recorded in its `lineage_selected_full_nmse_v1/` subdirectory.

Retain checkpoint SHA-256 hashes, source and configuration hashes, completed update counts, split hashes, release masks, and pair-level metrics with each aggregate. Reproduction can proceed in three stages: recompute metrics from fixed checkpoints; retrain children from a fixed parent; then repeat activation extraction and parent construction. These stages isolate evaluation, optimization, and upstream data variation.

Feasibility and budget. A 15,000-update child at batch size 512 uses 7.68 million token presentations; a 12,000-update child uses 6.144 million. These counts include repeated passes over cached activations. Each chain is sequential, while different seeds or chains can run independently. The largest standard SAE has encoder and decoder matrices of size 4096×2048; cached arrays can be read in minibatches. Model extraction and CE evaluation additionally require the corresponding frozen language model. The profiled ridge solver chooses the smaller equivalent feature-space or sample-space linear system and solves it in double precision.

Table 7 counts source training and three children at a common 12k child horizon. Stable Pool uses 360k source updates versus 12k for the LineageSAE parent, a source-update ratio of 30; total token presentations differ by a factor of 8.25. Pool deduplication and ranking are additional construction work. These are training-budget counts; hardware-dependent runtime and peak memory require measurement on the reproduction system.

Table 7: Source and child training costs for three reusable dictionaries.

Method	Sources	Source updates	Source tokens	Total tokens	Construction
Independent TopK	0	0	0	18.43M	none
Stable Pool	3	360k	184.32M	202.75M	Deduplication / ranking
LineageSAE	1	12k	6.14M	24.58M	none

C Inheritance and Root Construction

The central question is whether a parent can preserve feature addresses across independently optimized children. We first isolate decoder guidance and geometric constraints, then test vocabulary construction, training duration, and the dependence of addresses on the chosen root.

C.1 Decoder Handoff and Takeover Ablations

The Wiki split contains 199,936 training and 29,184 held-out activation vectors. An ordinary root and the residual-space parent each train for 12,000 updates. Only the rank-four subspace estimated from root seed 0 is used by the children. All SAEs disable the independent encoder bias and retain a decoder bias. Child seeds 1, 2 and 3 use independently initialized weights and independently shuffled batches. Each 12k run presents 6.144 million tokens to the optimizer; these are repeated presentations of stored activations. The 12k schedule uses batch size 512, learning rate 3×10^{-4}, zero weight decay, region margin 0.01 and weight one, and AuxK weight $1/32$. The parent decoder coefficient decays from one to zero between steps 3,000 and 9,000; the takeover control adds a native-decoder loss with weight one. The TopK+AuxK control shares optimization and the frozen residual space but removes the parent decoder and region penalty. The model is Pythia-1.4B-deduped at layer 12, with input dimension 2,048, dictionary width 4,096, and sparsity 16. Response profiles are centered mean representations of firing tokens after the frozen model's output normalization and unembedding (the linear map to vocabulary logits). Correspondence scores average the per-pair median over three child pairs.

This fixed-view ablation tests decoder handoff and the parent-relative decoder constraint with independently initialized children. "Full" in Table 8 includes both components; "Handoff only" removes the region penalty and "Region only" removes decoder handoff. An active atom fires on at least one evaluation input. Identity retention is the fraction of pairs in a one-to-one Hungarian matching that keep their feature index; an atom is outside its parent region when another parent atom has greater decoder cosine than its own parent atom.

The parent decoder yields same-index response-profile cosine 0.9796 without the region penalty, compared with 0.0060 for the region penalty alone. Combining them reduces the fraction outside the parent region from 8.78% to 2.11%. These measurements distinguish guidance of address usage from geometric constraints on decoder directions.

Table 8: Decoder handoff establishes response agreement; the region constraint improves geometric address retention.

Metric	Full	Handoff only	Region only	Control
NMSE ↓	<u>0.01568</u>	**0.01566**	0.01770	0.01579
Active atoms	99.54%	99.59%	**99.75%**	<u>99.72%</u>
Same-index profile cosine	**0.9813**	<u>0.9796</u>	0.0060	−0.0009
Matched profile cosine	**0.9803**	<u>0.9784</u>	0.8479	0.8904
Identity kept after matching	**91.82%**	<u>90.57%</u>	0.060%	0.043%
Same-index decoder cosine	**0.9781**	<u>0.9752</u>	0.0926	0.0010
Outside parent region ↓	**2.11%**	<u>8.78%</u>	88.77%	–

Table 9 varies projection and decoder takeover separately. Takeover means training the native child decoder on detached child coefficients while reconstruction still uses interpolated parent and child atoms. It increases same-index profile cosine in both spaces, from 0.9813 to 0.9841 after projection

and from 0.9677 to 0.9752 in the full space, with a small increase in reconstruction error. Projection and native-decoder preparation thus have distinct measurable effects.

Table 9: Effects of shared-component projection and native-decoder takeover.

Space	Decoder training	NMSE ↓	Profile cos. ↑	Kept ↑	Decoder cos. ↑
Residual	interpolation	0.01568	0.9813	91.82%	0.9781
Residual	+ takeover	0.01572	**0.9841**	92.95%	**0.9821**
Full	interpolation	0.01611	0.9677	89.60%	0.9522
Full	+ takeover	0.01614	0.9752	**92.96%**	0.9661

C.2 Functional-Asymmetry and Bias Controls

Mean squared canonical cosine between independently extracted rank-four subspaces is 0.848 for seeds 0 and 1 and 0.833 for seeds 0 and 2. Removing these directions from calibration encoder inputs preserves 97.1% of TopK assignments. Decoder reconstruction explains 98.0%–99.99% of the energy along those directions.

The largest-coordinate fraction is the maximum fraction of active decoder columns assigned to one largest-magnitude signed coordinate. Let π_j be the fraction assigned to signed coordinate j, among the $2d$ possible coordinate/sign combinations. Coordinate entropy is $-\sum_j \pi_j \log \pi_j / \log(2d)$, with $0 \log 0 = 0$.

Table 10: Reconstruction and decoder utilization before and after shared-component removal.

Metric	Original	Residual
Normalized MSE	0.01748	0.01596
Active atoms	58.70%	**90.91%**
Largest-coordinate fraction	25.91%	**7.36%**
Coordinate entropy	0.640	**0.752**

These distributional diagnostics report reconstruction, decoder utilization, and coordinate entropy; address identity is measured by the profile and matching metrics reported above. Refitting the optimal constant decoder bias with coefficients and columns fixed leaves 99.02%–99.98% of the extracted coefficient energy after mean subtraction. Learned and refitted bias norms are 57.85 and 57.62; their difference has norm 0.93, with only 0.39 inside the extracted subspace. The signal is token varying.

C.3 Address Agreement after Extended Training

This fixed-view comparison uses decoder handoff and the region constraint with independently initialized children. The parent is trained from random initialization in the projected residual space. TopK+AuxK, L2-regularized TopK, OSAE and LineageSAE train for 120k updates with the same data, sparsity and optimization. Stable Feature Pool uses three 120k source dictionaries and is evaluated after its 2.048M-token tuning protocol and after 120k updates. Its children share their initialization; LineageSAE children have independent random weights. The LineageSAE parent also completes a 120k run; its checkpoint is selected on validation rather than child consistency. The selected 24k checkpoint and final checkpoint have same-address similarity 0.9794 and identity retention 96.66%.

The decoder-cosine columns average the three pairwise medians. "Ind." denotes independent initialization, and "Pool" denotes initialization from the shared feature pool. The source count is the number of prerequisite long-trained dictionaries.

The L2 comparison includes the published 10^{-4} penalty and the largest penalty in $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$ retaining at least 99% active coverage, selected without a cross-seed criterion. The strong penalty's high matched score accompanies only 5.83% active atoms. Semantic

Table 11: Address agreement and reconstruction after extended fixed-view training.

Address continuity

Method	Same-index cos.	Matched cos.	Identity kept
TopK+AuxK	0.0044	0.8769	0.027%
L2-TopK, 10^{-4}	0.0116	0.9590	0.000%
L2-TopK, 10^{-6}	0.0095	0.9224	0.050%
OSAE	−0.0007	0.7582	0.122%
Stable Pool, 4k	0.9842	0.9841	**98.64%**
Stable Pool, 120k	0.9870	0.9870	98.11%
LineageSAE	**0.9887**	**0.9886**	97.49%

Reconstruction and initialization

Method	Initialization	Sources	NMSE	Active
TopK+AuxK	Ind.	0	0.01605	99.27%
L2-TopK, 10^{-4}	Ind.	0	0.02607	5.83%
L2-TopK, 10^{-6}	Ind.	0	**0.01562**	**99.94%**
OSAE	Ind.	0	0.01731	97.97%
Stable Pool, 4k	Pool	3	0.01582	99.89%
Stable Pool, 120k	Pool	3	0.01575	99.72%
LineageSAE	Ind.	1	0.01568	99.46%

optimal transport is evaluated as a post-hoc matcher: matched decoder cosine is 0.826–0.830 on long TopK runs, with no native identity matches.

Extended training alone leaves independently trained baselines with low same-index decoder agreement, even when post-hoc matching finds similar directions. Address agreement and index retention hold together across source counts: parent-guided children reach cosine 0.9887 using one source SAE while retaining 97.49% of indices, and Stable Pool reaches 0.9870 using three sources.

C.4 Root Relativity and Within-View Robustness

The following experiments use the fixed-view handoff setup in Appendix C.1. Each root defines its own address indices. Children of the same root have same-index profile cosine near 0.981, whereas children of different roots have cosine near zero before matching (Table 12). This supports a root-relative vocabulary: indices are shared within a lineage, not across arbitrary roots.

Table 12: Address agreement within and across independently constructed roots.

Comparison	NMSE	Profile cos.	Matched cos.	Kept
Children of root A	0.01568	0.9813	0.9803	91.82%
Children of root B	0.01568	0.9810	0.9798	91.24%
Across roots A and B	–	−0.0005	0.9080	0.0029%

Each within-root result averages three child pairs; the cross-root result averages nine pairs. Layer-8 and within-Code runs recalibrate both subspace and root in their own setting. The width comparison changes (N, k) together to retain the sparsity ratio. Handoff schedules and loss weights are fixed.

Table 13 tests the same mechanism across layers, sparsity budgets, dictionary widths, and within-domain data. Same-index profile cosine remains between 0.9436 and 0.9862 across these settings. Index retention stays above 90% on the Wiki runs and at layer 8, and is lower on Code (61.51%) and at the $k = 32$ budget (77.63%), where the evaluation view changes most.

Table 13: Fixed-view inheritance across layers, sparsity budgets, dictionary widths, and domains.

Data	Layer	N	k	NMSE	Active	Profile cos.	Kept
Wiki	12	4096	8	0.01759	98.41%	0.9858	93.72%
Wiki	12	4096	16	0.01568	99.54%	0.9813	91.82%
Wiki	12	4096	32	0.01405	99.88%	0.9546	77.63%
Wiki	12	2048	8	0.01954	99.67%	0.9862	86.47%
Code	12	4096	16	0.01950	93.80%	0.9436	61.51%
Wiki	8	4096	16	0.01304	99.76%	0.9782	92.19%

C.5 Root Construction and Token-Lookup Controls

This experiment compares projection followed by AuxK refinement with training a root from random initialization. An optional token lookup adds a separate Tokenized-SAE decoder table. It tests root construction with a frozen rank-four subspace and 15,000-step Code children. All root-construction conditions share the same binary address certification without background correction and the same parent-encoder routing.

Semantic-profile cosine compares the response profiles defined in Appendix C.1; it is distinct from decoder cosine. "Inherited" and "Released" restrict that comparison to the corresponding address subset. "Active" is the fraction of addresses firing on held-out inputs, and "Kept" is the fraction retaining their index after matching.

Table 14: Root initialization and token lookup in Wiki-to-Code retraining.

Response-profile agreement

Root	Lookup	All	Inherited	Released
Projected + AuxK	–	**0.8797**	**0.8627**	**0.8897**
Projected + AuxK	✓	0.8024	0.7352	0.8269
Random initialization	✓	0.8145	0.7505	0.8425
Random initialization	–	<u>0.8762</u>	<u>0.8596</u>	<u>0.8876</u>

Address retention and reconstruction

Root	Lookup	Kept	NMSE	Active
Projected + AuxK	–	**0.4567**	**0.0195**	<u>0.9332</u>
Projected + AuxK	✓	0.3444	0.0206	0.8215
Random initialization	✓	0.3512	<u>0.0204</u>	0.8215
Random initialization	–	<u>0.4485</u>	**0.0195**	**0.9372**

Projected refinement preserves 66.5% of the initial root's TopK assignments, compared with 7.7% after retraining from random initialization. Without the token lookup, projected and randomly retrained roots yield similar sibling profile cosine (0.8797 versus 0.8762) while retaining 45.67% and 44.85% of indices respectively. Projection preserves the vocabulary measured in Appendix F.4. Removing the lookup in the projected-root condition improves active coverage from 0.8215 to 0.9332 and reduces NMSE from 0.0206 to 0.0195. These are separate controls for vocabulary preservation and the additional token table.

D Release and Reuse under Controlled Feature Changes

D.1 Controlled Comparison and Semantic Recovery

Feature deletion, addition, and frequency reduction distinguish three goals: retaining useful parent features, releasing obsolete addresses, and assigning newly learned features consistently. The sample counts and feature groups are defined in Appendix B.1. We evaluate both the response at a known address and recovery of the underlying generating feature, since either measurement alone leaves part of this question open.

Controlled training settings. The controlled comparison uses 15,000 updates and batch size 512. L2-TopK selects weight decay on a training-only development split with seed 901 from $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$, giving 10^{-6} on SynthSAEBench and 10^{-7} on Toy. RA-SAE selects relaxation radius $\delta \in \{0.1, 1\}$ and learning rate $\{3 \times 10^{-4}, 3 \times 10^{-3}\}$, using $\delta = 1$ and 3×10^{-4} on both tasks. Final training uses the full training split. The LineageSAE residual-learning comparison combines decoder handoff, profiled ridge reuse, response ordering on zero-response addresses, and training-data parent-relative assignment. The corresponding objectives are defined in Appendix A.5. The resulting response and reconstruction measurements are the ones reported in the main comparison tables.

For the independent-pool control, the three child pipelines construct pools from disjoint source-seed groups $\{1, 2, 3\}$, $\{4, 5, 6\}$, and $\{7, 8, 9\}$. Each source also trains for 15,000 updates. This gives nine distinct sources across three independent pools, compared with three sources when a single pool is shared. LineageSAE uses one parent within each condition. The controlled evaluation comprises 162 final models: six methods, nine conditions, and three children. Source and development models are counted separately.

Response preservation. Both parent and child encode the same updated held-out inputs. The pretraining release mask fixes the inherited subset $\mathcal{I}$ and released subset $\mathcal{R}$. Scores use each subset's full address count as denominator, then average children within conditions and conditions equally. C–C evaluation uses the three unordered child pairs; P–C evaluation uses the three parent/child pairs. Response is therefore reported separately within the inherited and released subsets, after training-data assignment of released addresses. The inherited subset carries substantially more parent response than the released subset, which is consistent with preserving one part of the vocabulary while allowing another part to change.

Known-feature recovery. SynthSAEBench selects the atom with maximum calibration activation correlation for each generating feature, then requires calibration and test Pearson correlations and decoder cosine to be at least 0.8. This rule permits an atom to represent more than one generating feature. Toy first requires annotation Pearson correlation and residual-space decoder cosine of at least 0.8, then uses a one-to-one Hungarian assignment with dummy unmatched slots. Test Pearson correlation must also be at least 0.8. All methods use the same correspondence rule within a benchmark. Rare-feature recovery averages conditions with reduced-frequency survivors. Deleted-feature probes use a separate diagnostic view in which those features are present and are excluded from survivor/new-feature recovery. SynthSAEBench admits many-to-one recovery, whereas Toy assigns one atom per feature by Hungarian matching; each benchmark therefore reports its own recovery rule.

D.2 Controlled Removal: Mechanism and Failure Cases

The 24 synthetic views share coefficient samples within each task group. Repeated difficult-deletion observations involve three distinct parent addresses. This paired design isolates changes caused by deletion and frequency reduction while holding the generating representation fixed. The selected background's response correlates 0.7031–0.8076 with a surviving similar feature's coefficient, which is its largest absolute covariance source in each fold. Some backgrounds elsewhere in the vocabulary are most strongly associated with the target itself, so the selected background's composition is reported explicitly for each fold.

At a common population and on the positive-projection branch of the fit, let v be the unit background direction. The utility takes the form

$$\left[d_i^P - ((d_i^P)^\top v)v\right]^\top C_i. \tag{26}$$

If $C_i = ad_i^P + bv$ with $a > 0$, this is $a[1 - ((d_i^P)^\top v)^2] > 0$ for non-collinear directions. Pure background gives zero utility, so a negative cross-fitted score arises when the parent direction falls below the fitted reference. The within-fold target contribution is positive in all 22,352 retained-feature observations, but other contributions can still cause false releases. Of the 440 paired deletion observations, 415/420 ordinary and 20/20 difficult scores become negative. The difficult cases show why correction for competing responses matters; the ordinary deletion failures and rare-feature false releases identify the finite-sample limits of the rule.

D.3 New-Feature Reuse: Protocol and Assignment Results

This subsection evaluates address assignment with fixed residual-learning checkpoints (Appendix D.1). It isolates the effect of renumbering learned features without changing their reconstruction. The six reuse conditions use the fixed parent ($d = 128$, $N = 1024$, $k = 16$) and 65,536 current training inputs per condition. All children use native full-vocabulary TopK throughout training. At update 3,000, released encoder rows are initialized from their parent rows. The residual auxiliary weight is one through update 6,000 and cosine-decays to zero at update 9,000; decoder handoff follows the same decay interval. Training ends at update 15,000 with batch size 512, learning rate 3×10^{-4}, AuxK weight $1/32$, and the unchanged decoder takeover and region objectives. The comparison baseline replaces this residual auxiliary objective with the finite parent-encoder-routing stage while retaining the same release mask and other objectives. Initializations and minibatch orders are paired within seed.

After training, scalar residual projections and activation energies are fitted over the complete current training inputs. Each child independently solves the parent-relative assignment; generating directions, semantic coefficients, and other children do not enter this fit. Evaluation uses a separate set of 65,536 inputs: the first half chooses each semantic feature's atom by maximum activation correlation over the full child vocabulary, and the second half measures its correlation with the generating coefficient. Recovery requires this held-out correlation and cosine between the actual child decoder atom and generating direction to meet the stated threshold. For old-address retention, the corresponding parent address is fixed before child evaluation. The same evaluation protocol is used for the routing baseline.

R0 retains survivor frequencies and R1 reduces them. Each of the three task groups adds 51 generating features. A common address requires recovery by all three children at the same index.

Table 15: New features recovered at one common address across three children, out of 51 per condition.

Assignment	Task 1, R0	Task 1, R1	Task 2, R0	Task 2, R1	Task 3, R0	Task 3, R1
Without assignment	24	26	24	27	26	22
With assignment	**48**	**45**	**48**	**46**	**49**	**49**

The assignment comparison raises common-address recovery from 22–27 to 45–49 of the 51 new features while keeping trained weights and the reconstruction function fixed. It isolates the naming part of reuse: learning a feature and assigning it the same address across children are separate requirements. The full-address response-ranking objective in the main method addresses this coordination during training.

E Transfer and Successive Model Updates

E.1 Release, Response, and Reconstruction across Views

The additional transfer experiments use one frozen parent and three children per condition. They combine decoder handoff, profiled ridge reuse, and shared response ordering with coefficient 0.1 and margin 0.5; the inherited-response regression weight is zero. Release uses 32,768 current-view training rows sampled with seed 20260921 and split equally for cross-fitting. Zero-response and finite negative-utility addresses form disjoint release groups. The mask is fixed before 15,000-step child training.

Table 16: Addresses released by zero response or negative marginal utility.

Transition	(N, k)	Zero response	Negative utility	Total
Wiki → Code	$(4096, 16)$	164	447	611
Pythia update	$(4096, 16)$	4	44	48
Wiki → Code	$(2048, 8)$	423	157	580

Wiki-to-Code uses the frozen Wiki parent and Code training activations; the model-update experiment uses the frozen parent SAE and updated-model activations. Training-data assignment fixes inherited addresses and permutes only released addresses. Calibration matching is fitted independently of held-out scoring. All support and rank scores retain the full dictionary denominator.

Release counts vary with both the transition and dictionary capacity: 611 addresses are released for the standard Wiki-to-Code setting and 48 for the Pythia update. The smaller Wiki-to-Code dictionary releases 580 of 2048 addresses. Release therefore adapts the available capacity to evidence in the current view rather than allocating a fixed fraction of the vocabulary.

Response and decoder measurements. A corpus change alters feature firing frequencies and activation ranks. Table 17 reports those changes alongside the decoder-direction measurements in Table 19. The Pythia rows without a baseline name refer to LineageSAE.

Table 17: Response continuity across domains at the deployed addresses.

Condition	P–C J	C–C J	P–C ρ	C–C ρ
Pythia L12, $(4096, 16)$	0.1096	0.3178	0.1842	0.4027
BatchTopK, Pythia L12	0.0169	0.0107	0.0069	0.0043
Gated SAE, Pythia L12	0.0061	0.0015	0.0164	0.0011
Pythia L8, $(4096, 16)$	0.0957	0.2759	–	–
Pythia L12, $(2048, 8)$	0.0984	0.3157	–	–
Qwen2.5-0.5B, $(1792, 16)$	0.1220	0.3485	–	–

We also compute pairwise maximum cosine correlation (PW-MCC): rectangular Hungarian matching maximizes the absolute decoder-cosine matrix, independently of activation-based matching. P–C/C–C means are $(0.3161, 0.7477)$ for Wiki-to-Code, $(0.9420, 0.9858)$ for the Pythia update, $(0.3686, 0.7735)$ for Qwen2.5-0.5B, and $(0.2819, 0.8065)$ for the $(N, k) = (2048, 8)$ dictionary. The smaller dictionary has 1929.3 active addresses, averaged over three children.

Removing the inheritance module reduces C–C Jaccard from 0.3178 to 0.0211 and decoder cosine from 0.6088 to 0.0999, connecting parent guidance to address continuity during domain transfer. The controlled feature-change experiments measure deletion recognition, decoder handoff, and coordination of newly introduced features under matched conditions.

Language-model intervention. Table 18 reports the sample counts, cross-entropy denominators, cache consistency, and identity-replacement checks for every additional transfer condition. CE recovery is a mean and sample standard deviation over three children. Pythia layer 8 uses its own Code validation split.

E.2 Additional Transfer and Architecture Comparisons

Wiki-to-Code retraining is repeated at Pythia layer 8, with half the dictionary width and sparsity budget, and on Qwen2.5-0.5B. Each setting uses its own Wiki parent and three Code children. Schedules and loss weights are unchanged. Qwen's 896-dimensional residual stream uses $N = 1792$ under the same twofold expansion rule. Table 19 reports the resulting transfer measurements.

The additional baseline comparisons use the recorded model, layer, and sparsity settings in Appendix E.1. The GPT-2 rows evaluate their own pretrained model, layer, and sparsity setting, as recorded in Appendix B.4. These BatchTopK measurements use batchwise inference; the main comparisons use calibrated fixed thresholds. Response metrics in Table 20 use native addresses.

The six-condition architecture comparison in Table 21 aggregates six SynthSAEBench conditions with the full-address evaluator under the residual-learning configuration and reports the corresponding checkpoints. The two MP-SAE operating points have measured L_0 below the target; the selection column identifies reconstruction and sparsity settings. LineageSAE has the highest native agreement and gains only 0.0120 in J from matching, indicating that most correspondence already resides at the deployed addresses.

Table 18: Language-model reconstruction and activation-intervention checks.

Evaluation samples and language-model loss

Condition	Sequences	Targets	Clean CE	Zero-site CE
Pythia L12, $N = 4096$	164	20,828	2.2831	11.3664
Pythia update	228	28,956	3.0939	12.4020
Pythia L12, $N = 2048$	164	20,828	2.2831	11.3664
Pythia L8	162	20,574	1.5875	13.0558
Qwen2.5-0.5B	129	16,383	1.9019	14.2926

Intervention checks and reconstruction

Condition	Cache RMSE	Maximum logit difference	CE recovery
Pythia L12, $N = 4096$	0.005683	0	$95.12 \pm 0.05\%$
Pythia update	0.004948	0	$95.55 \pm 0.01\%$
Pythia L12, $N = 2048$	0.005683	0	$91.39 \pm 0.10\%$
Pythia L8	0.005145	0	$95.42 \pm 0.13\%$
Qwen2.5-0.5B	0.004646	0	$93.68 \pm 0.09\%$

Table 19: Wiki-to-Code continuity across layers, dictionary widths, and model families.

Model / layer	(N, k)	P–C cos.	C–C cos.	NMSE	CE rec.
Pythia-1.4B / 12	$(4096, 16)$	0.3087	0.6088	0.0143	95.12%
Pythia-1.4B / 8	$(4096, 16)$	0.2793	0.5559	0.0155	95.42%
Pythia-1.4B / 12	$(2048, 8)$	0.2676	0.6113	0.0171	91.39%
Qwen2.5-0.5B	$(1792, 16)$	0.1720	0.6792	0.0035	93.68%

Table 20: Response continuity and reconstruction on additional language-model transitions.

Address continuity

Transition	Method	P–C cos.	C–C cos.	P–C J	C–C J
Pythia	LineageSAE	0.3087	0.6088	0.1096	0.3178
Pythia	BatchTopK	–	–	0.0169	0.0107
Pythia	Gated SAE	–	–	0.0061	0.0015
GPT-2	BatchTopK	–	–	0.0054	0.0030
GPT-2	Gated SAE	–	–	0.0031	0.0023

Reconstruction

Transition	Method	NMSE	L_0	CE rec.
Pythia	LineageSAE	0.0143	15.960	95.12%
Pythia	BatchTopK	0.1303	109.4	–
Pythia	Gated SAE	0.0146	21.7	95.92%
GPT-2	BatchTopK	0.1925	94.2	100.6%
GPT-2	Gated SAE	0.0099	78.8	98.3%

Table 21: Native and matched response agreement across sparse-coding architectures.

Response continuity

Method	Selection	Native J	Native ρ	Matched J	Matched ρ
BatchTopK	–	0.0084	0.0072	0.4511	0.4828
AbsTopK	–	0.0070	0.0002	0.4075	0.4462
Gated SAE	–	0.0083	0.0060	0.4360	0.4810
Matryoshka SAE	–	0.0149	0.0100	0.4244	0.4617
Archetypal SAE	–	0.0066	0.0012	0.3532	0.3972
RA-SAE	–	0.0072	0.0013	0.4460	0.4817
MP-SAE	Minimum NMSE	0.0076	0.0056	**0.6099**	**0.7314**
MP-SAE	Target sparsity	0.0103	0.0085	0.4481	0.5867
LineageSAE	–	**0.4621**	**0.4910**	0.4741	0.5108

Reconstruction and sparsity

Method	Selection	NMSE	L_0
BatchTopK	–	0.14330	16.00
AbsTopK	–	0.18528	16.00
Gated SAE	–	0.11820	17.30
Matryoshka SAE	–	0.15668	16.24
Archetypal SAE	–	0.29380	15.99
RA-SAE	–	0.11898	16.00
MP-SAE	Minimum NMSE	**0.00651**	11.56
MP-SAE	Target sparsity	0.00813	13.63
LineageSAE	–	0.11330	16.00

E.3 Continuity over Successive Generations

Three chains use one Pythia-1.4B layer-12 Wiki root for four generations. Generation $g + 1$ reads only the assigned child from generation g in the same chain, recomputes its release mask, and runs the complete residual-learning protocol. The evaluation contains 13 checkpoints and 36 pair comparisons. Adjacent comparisons use a child and its immediate parent; root-relative comparisons use the initial root; cross-chain comparisons use descendants at the same generation. Calibration is diagnostic, and validation uses the fixed Wiki split. Table 24 additionally retains release-count ranges and its recorded decoder aggregates.

Table 22 gives the complete results underlying Figure 2, including reconstruction, CE recovery, and released-address counts.

Original-space NMSE is 0.0157049 for the root and 0.0156171–0.0156459 for children. The control runs exclude the root from this average. No root reset, sibling checkpoint, or multi-parent fit is used.

The CE evaluation uses 228 Wiki sequences and 28,956 next-token targets. Clean, zero-site, and bypass-only CE are 3.5897, 12.4523, and 8.6902. Identity replacement changes no logit; cached and live hook activations have relative RMSE 0.0051.

The native control scores average 12 pairs per relation; NMSE averages 12 child checkpoints. The controls use the same parent, current-view split, release-mask procedure, 15,000-step budget, and assignment evaluator. Independent retraining has near zero native address agreement; copying the parent preserves agreement, and freezing its decoder preserves it most strongly while giving the highest NMSE.

Further protocol variants. A four-generation protocol with three chains passes each child to the next generation under its recorded decoder-handoff schedule. In the model-update protocol, Pythia-1.4B is further trained for 300 updates, and layer-12 activations are extracted on the same Wiki splits. The subspace and root are re-estimated on the updated model; children then train for 15,000 updates. Evaluation compares new-model children trained from different seeds on the updated model.

Table 22: Complete generation-level measurements underlying Figure 2.

Address continuity

Relation	Generation	J	ρ	Decoder cos.
Adjacent	1	0.6773	0.7951	0.9572
Adjacent	2	0.7410	0.8413	0.9758
Adjacent	3	0.7590	0.8533	0.9800
Adjacent	4	0.7673	0.8596	0.9831
Root-relative	1	0.6773	0.7951	0.9572
Root-relative	2	0.6148	0.7458	0.9291
Root-relative	3	0.5808	0.7172	0.9090
Root-relative	4	0.5573	0.6967	0.8958
Cross-chain	1	0.7879	0.8731	0.9889
Cross-chain	2	0.7520	0.8476	0.9788
Cross-chain	3	0.7313	0.8324	0.9722
Cross-chain	4	0.7160	0.8210	0.9696

Reconstruction and capacity

Generation	NMSE	CE recovery	Released
1	0.015618	95.79%	38.0
2	0.015625	95.79%	35.0
3	0.015639	95.79%	36.0
4	0.015643	95.82%	33.7

Table 23: Native and calibration-matched continuity across vocabulary generations.

Relation	Pairs	Native J	Native ρ	Decoder cos.	Matched J	Matched ρ
Adjacent generation	12	0.7361	0.8373	0.9740	0.7304	0.8305
Root-relative	12	0.6075	0.7387	0.9228	0.6017	0.7310
Cross-chain	12	0.7468	0.8435	0.9767	0.7415	0.8372

Table 24: Generation-level continuity, reconstruction, and released-address ranges.

Address continuity

Relation	Generation	J	ρ	Decoder cos.
Adjacent	1	0.6773	0.7951	0.9572
Adjacent	2	0.7410	0.8413	0.9758
Adjacent	3	0.7590	0.8533	0.9800
Adjacent	4	0.7673	0.8596	0.9831
Root-relative	1	0.6773	0.7951	0.9572
Root-relative	2	0.6148	0.7458	0.9291
Root-relative	3	0.5808	0.7172	0.9090
Root-relative	4	0.5573	0.6967	0.8958
Cross-chain	1	0.7879	0.8731	0.9889
Cross-chain	2	0.7520	0.8476	0.9788
Cross-chain	3	0.7313	0.8324	0.9722
Cross-chain	4	0.7160	0.8210	0.9696

Reconstruction and capacity

Generation	NMSE	CE recovery	Released
1	0.015618	95.79%	38.0 [38,38]
2	0.015625	95.79%	35.0 [31,39]
3	0.015639	95.79%	36.0 [32,39]
4	0.015643	95.82%	33.7 [28,38]

Table 25: Successive retraining with independent, copied, frozen, and inherited dictionaries.

Adjacent generations

Method	J	ρ	Decoder cos.
Independent retraining	0.0015	0.0005	0.0021
Parent copy + continue	0.7510	0.8480	0.9876
Frozen parent decoder	**0.8166**	**0.8919**	**1.0000**
LineageSAE	0.7361	0.8373	0.9844

Root-relative

Method	J	ρ	Decoder cos.
Independent retraining	0.0013	0.0001	0.0010
Parent copy + continue	0.5947	0.7296	0.9632
Frozen parent decoder	**0.7427**	**0.8435**	**1.0000**
LineageSAE	0.6075	0.7387	0.9561

Cross-chain

Method	J	ρ	Decoder cos.
Independent retraining	0.0013	0.0000	0.0020
Parent copy + continue	0.7551	0.8516	0.9884
Frozen parent decoder	**0.8138**	**0.8899**	**1.0000**
LineageSAE	0.7468	0.8435	0.9864

Reconstruction

Method	NMSE	CE recovery
Independent retraining	**0.01548**	**95.96%**
Parent copy + continue	0.01562	95.83%
Frozen parent decoder	0.01580	95.69%
LineageSAE	0.01563	95.80%

Across generations, adjacent Jaccard rises from 0.6773 to 0.7673, so each generation agrees more closely with its immediate parent than the previous generation did. CE recovery remains near 95.8% throughout. The frozen-decoder control retains the most agreement but has the highest NMSE; independent retraining has the lowest NMSE but near-zero native agreement; LineageSAE retains 0.7361 adjacent J at 0.01563 NMSE. It permits decoder adaptation while preserving useful continuity with a single preceding vocabulary.

F Complete Comparisons and Correspondence Diagnostics

F.1 Language-Task Response Measurements

The complete response columns for the selected language-task models are given in Table 26. The evaluation protocol and model access are defined in Appendix B.1.

Table 26: Parent–child and child–child response continuity on the two language tasks.

	Wiki-to-Code				Pythia model update			
Method	P–C J	P–C ρ	C–C J	C–C ρ	P–C J	P–C ρ	C–C J	C–C ρ
L2-TopK	0.0025	0.0023	0.0012	0.0004	0.0434	0.0580	0.0015	0.0005
OSAE	0.0016	0.0008	0.0019	0.0007	0.0317	0.0417	0.0016	0.0001
RA-SAE	0.0040	0.0072	0.0001	0.0003	0.0072	0.0079	0.0002	0.0002
BatchTopK	0.0012	0.0006	0.0011	0.0003	0.0405	0.0496	0.0028	0.0020
Gated SAE	0.0138	0.0395	0.0038	0.0063	0.1011	0.1412	0.0014	0.0009
SharedSAE	0.0007	0.0002	0.0002	0.0000	0.3401	0.3934	0.0012	−0.0001
Stable Pool	0.0818	0.1378	0.2167	0.2878	0.3902	0.5387	0.6746	0.7952
MP-SAE	0.0026	0.0018	0.0032	0.0049	0.0126	0.0199	0.0124	0.0195
iSAE	0.0028	0.0028	0.0009	0.0001	0.0645	0.0903	0.0013	−0.0004
AbsTopK	0.0041	0.0045	0.0028	0.000004	0.0546	0.0785	0.0027	0.0006
Archetypal SAE	0.0042	0.0075	0.00003	0.0001	0.0068	0.0074	0.0000	0.0000
Matryoshka SAE	0.0009	0.0004	0.0044	0.0039	0.0048	0.0043	0.0055	0.0052
LineageSAE	**0.1176**	**0.1968**	**0.3494**	**0.4430**	**0.6163**	**0.7527**	**0.6959**	**0.8109**

F.2 Complete Four-Task Measurements

Each task is presented in two panels: native address continuity, followed by calibration-matched correspondence and reconstruction. The selected pairs consist of one old-view and two new-view SAEs, with Stable Pool's three-source construction defined in Appendix B.4. SynthSAEBench gives its six task/rarity conditions equal weight. The controlled grid uses the residual-learning checkpoints and selected-pair aggregation, complementing the full-address ranking comparison in Table 2.

Native decoder cosine is omitted for the shared-candidate Archetypal checkpoints; the independent-candidate measurement appears in Appendix F.5. Language-task NMSE uses the recorded input-space denominator, with residual-input NMSE for LineageSAE and its full-activation counterpart in Table 32. MP-SAE is evaluated at its own selected operating point on each benchmark. In the panels below, NMSE† normalizes by residual energy, whereas the full-activation values in Table 32 normalize by activation energy; the two are reported separately.

The Wiki-to-Code and model-update panels report LineageSAE's native correspondence alongside the strongest response baselines, and the controlled grid reports the remaining architectures on the same measurements under the residual-learning configuration. Reconstruction and address continuity are separate axes and are reported separately throughout.

Table 27: Wiki-to-Code: complete selected-pair measurements.

Native address continuity

Method	P–C cos.	C–C cos.	P–C J	C–C J	P–C ρ	C–C ρ
iSAE	0.0105	-0.0001	0.0028	0.0009	0.0028	0.0001
AbsTopK	0.0073	0.0003	0.0041	0.0028	0.0045	0.0000
MP-SAE	0.0004	0.0087	0.0026	0.0032	0.0018	0.0049
Archetypal SAE	–	–	0.0042	0.0000	0.0075	0.0001
RA-SAE	–	–	0.0040	0.0001	0.0072	0.0003
Matryoshka SAE	0.0007	0.0056	0.0009	0.0044	0.0004	0.0039
BatchTopK	0.0026	0.0056	0.0012	0.0011	0.0006	0.0003
Gated SAE	<u>0.2563</u>	0.0043	0.0138	0.0038	0.0395	0.0063
SharedSAE	0.0190	0.0296	0.0007	0.0002	0.0002	0.0000
Stable Pool	0.2330	<u>0.4321</u>	<u>0.0818</u>	<u>0.2167</u>	<u>0.1378</u>	<u>0.2878</u>
L2-TopK	0.0284	0.0249	0.0025	0.0012	0.0023	0.0004
OSAE	0.0089	0.0004	0.0016	0.0019	0.0008	0.0007
LineageSAE	**0.3024**	**0.6436**	**0.1176**	**0.3494**	**0.1968**	**0.4430**

Matched correspondence and reconstruction

Method	P–C cos.	C–C cos.	NMSE†	L_0	CE rec.
iSAE	0.1289	0.4657	0.0149	16.00	94.7%
AbsTopK	0.1382	0.5628	0.1774	16.00	<u>95.4%</u>
MP-SAE	0.1062	0.4506	0.0210	11.43	89.3%
Archetypal SAE	<u>0.4742</u>	0.4913	0.0577	1.18	31.1%
RA-SAE	**0.4773**	0.5075	0.0632	1.07	30.5%
Matryoshka SAE	0.0694	0.2554	0.0176	19.10	90.2%
BatchTopK	0.1657	<u>0.6149</u>	0.0139	16.19	**95.6%**
Gated SAE	0.1308	0.2927	0.0167	12.99	92.7%
SharedSAE	0.0432	0.1368	0.0395	16.00	77.2%
Stable Pool	–	0.5923	0.0142	16.00	95.2%
L2-TopK	0.1701	0.5961	0.0140	16.00	95.3%
OSAE	0.1094	0.4624	0.0152	15.88	94.2%
LineageSAE	0.2062	**0.6469**	0.1701	15.97	95.3%

Table 28: SynthSAEBench: complete selected-pair measurements.

Native address continuity

Method	P–C cos.	C–C cos.	P–C J	C–C J	P–C ρ	C–C ρ
iSAE	0.4370	0.0024	0.0165	0.0010	0.0281	-0.0000
AbsTopK	0.1746	0.0038	0.0614	0.0072	0.0749	-0.0001
MP-SAE	0.0542	0.0011	0.0223	0.0077	0.0238	0.0056
Archetypal SAE	–	–	0.0342	0.0068	0.0396	0.0014
RA-SAE	–	–	0.0729	0.0070	0.0872	0.0013
Matryoshka SAE	0.0027	0.0082	0.0114	0.0145	0.0067	0.0097
BatchTopK	0.2058	0.0155	0.0779	0.0076	0.1003	0.0064
Gated SAE	0.2105	-0.0010	0.0716	0.0081	0.0949	0.0056
SharedSAE	0.3047	0.0037	0.1287	0.0055	0.1681	0.0003
Stable Pool	**0.7228**	**0.9421**	**0.3060**	**0.5566**	**0.3918**	**0.6504**
L2-TopK	0.2734	-0.0018	0.0998	0.0065	0.1226	0.0012
OSAE	0.1957	0.0010	0.0684	0.0080	0.0806	0.0008
LineageSAE	0.6331	0.5961	0.2557	0.4622	0.3268	0.4910

Matched correspondence and reconstruction

Method	P–C cos.	C–C cos.	NMSE†	L_0	CE rec.
iSAE	0.1705	0.1613	0.5026	16.00	–
AbsTopK	0.4793	0.5187	0.1853	16.00	–
MP-SAE	**0.8422**	**0.9435**	**0.0065**	11.57	–
Archetypal SAE	0.4189	0.5465	0.2968	15.99	–
RA-SAE	0.5699	0.6058	0.1189	16.00	–
Matryoshka SAE	0.4787	0.5216	0.1567	16.18	–
BatchTopK	0.5324	0.6056	0.1427	15.92	–
Gated SAE	0.5624	0.5782	0.1179	17.21	–
SharedSAE	0.5290	0.6129	0.1556	16.00	–
Stable Pool	–	0.8351	0.1142	15.89	–
L2-TopK	0.5775	0.6354	0.1114	16.00	–
OSAE	0.4522	0.4723	0.1704	16.00	–
LineageSAE	0.4759	0.6339	0.1132	16.00	–

Table 29: Toy superposition: complete selected-pair measurements.

Native address continuity

Method	P–C cos.	C–C cos.	P–C J	C–C J	P–C ρ	C–C ρ
iSAE	0.4884	0.0095	0.0257	0.0036	0.0392	0.0004
AbsTopK	0.0103	-0.0033	0.0022	0.0022	0.0000	-0.0003
MP-SAE	0.0052	-0.0060	0.0224	0.0209	0.0093	0.0103
Archetypal SAE	–	–	0.0605	0.0186	0.0741	0.0115
RA-SAE	–	–	0.0842	0.0170	0.0981	0.0052
Matryoshka SAE	0.0131	0.0241	0.0321	0.0362	0.0293	0.0313
BatchTopK	0.3447	-0.0013	0.0098	0.0201	0.0000	0.0170
Gated SAE	0.3356	-0.0101	0.0134	0.0152	-0.0002	0.0117
SharedSAE	0.0038	0.0003	0.0159	0.0162	0.0002	0.0003
Stable Pool	**0.7873**	**0.9900**	**0.2306**	0.4229	**0.3032**	0.4567
L2-TopK	0.3382	0.0110	0.1098	0.0109	0.1420	0.0021
OSAE	0.2408	-0.0055	0.0883	0.0177	0.1026	0.0024
LineageSAE	0.7484	0.7183	0.0140	**0.5260**	-0.0005	**0.5872**

Matched correspondence and reconstruction

Method	P–C cos.	C–C cos.	NMSE†	L_0	CE rec.
iSAE	0.1812	0.1868	0.3106	8.00	–
AbsTopK	-0.0033	0.0633	0.1701	8.00	–
MP-SAE	0.6108	0.6329	**0.0151**	11.25	–
Archetypal SAE	0.5667	0.5928	0.0966	8.00	–
RA-SAE	**0.6314**	0.6457	0.0904	8.00	–
Matryoshka SAE	0.4600	0.4814	0.1441	7.98	–
BatchTopK	-0.0055	0.4484	0.1230	8.00	–
Gated SAE	-0.0064	0.4452	0.1261	9.33	–
SharedSAE	-0.0027	0.4690	0.3615	8.00	–
Stable Pool	–	0.6067	0.0985	7.89	–
L2-TopK	0.5706	0.6437	0.1050	8.00	–
OSAE	0.4434	0.4259	0.1560	8.00	–
LineageSAE	-0.0069	**0.7298**	0.1136	8.00	–

Table 30: Pythia model update: complete selected-pair measurements.

Native address continuity

Method	P–C cos.	C–C cos.	P–C J	C–C J	P–C ρ	C–C ρ
iSAE	0.1254	-0.0001	0.0645	0.0013	0.0903	-0.0004
AbsTopK	0.1237	-0.0005	0.0546	0.0027	0.0785	0.0006
MP-SAE	0.0018	-0.0003	0.0126	0.0124	0.0199	0.0195
Archetypal SAE	–	–	0.0068	0.0000	0.0074	-0.0000
RA-SAE	–	–	0.0072	0.0002	0.0079	0.0002
Matryoshka SAE	0.0050	0.0068	0.0048	0.0055	0.0043	0.0052
BatchTopK	0.0705	0.0040	0.0405	0.0028	0.0496	0.0020
Gated SAE	0.2026	0.0073	0.1011	0.0014	0.1412	0.0009
SharedSAE	0.4984	0.0202	0.3401	0.0012	0.3934	-0.0001
Stable Pool	0.7565	0.9539	0.3902	0.6746	0.5387	0.7952
L2-TopK	0.0973	0.0244	0.0434	0.0015	0.0580	0.0005
OSAE	0.0593	0.0006	0.0317	0.0016	0.0417	0.0001
LineageSAE	**0.9706**	**0.9814**	**0.6163**	**0.6959**	**0.7527**	**0.8109**

Matched correspondence and reconstruction

Method	P–C cos.	C–C cos.	NMSE†	L_0	CE rec.
iSAE	0.6432	0.6306	0.1973	16.00	95.6%
AbsTopK	0.6266	0.6182	0.2037	16.00	**95.7%**
MP-SAE	0.4391	0.4021	0.0210	24.32	90.5%
Archetypal SAE	0.5434	0.5433	0.0512	1.02	36.6%
RA-SAE	0.5430	0.5401	0.0551	1.01	35.4%
Matryoshka SAE	0.3298	0.3300	0.0189	16.75	90.1%
BatchTopK	0.6314	0.6166	0.0154	16.02	95.3%
Gated SAE	0.6378	0.6185	0.1946	15.11	**95.7%**
SharedSAE	0.7671	0.6024	0.0239	16.00	94.2%
Stable Pool	–	0.9478	0.0147	16.00	95.6%
L2-TopK	0.6611	0.6590	0.0150	16.00	95.5%
OSAE	0.5250	0.5348	0.0163	15.85	94.6%
LineageSAE	**0.9589**	**0.9708**	0.1990	15.95	**95.7%**

F.3 Feature Recovery with a Shared Pool

The two Stable Pool children share the old-view feature pool and use different training seeds. Individual recovery applies the benchmark's activation and decoder-direction criteria. Common-address recovery also requires the two children to use the same index. The denominator is the number of added generating features: SynthSAEBench aggregates six conditions with 51 new features each.

Table 31: New-feature recovery and common-address recovery for Stable Pool.

Task	Child 1 recovered	Child 2 recovered	Same-address recovery
SynthSAEBench	306/306	306/306	214/306 (69.93%)
Toy superposition	7/8	7/8	6/8 (75.00%)

Both children recover all 306 new SynthSAEBench features individually, but only 214 share their address, giving 69.93% common-address recovery. On Toy, individual recovery is 7/8 for each child and common-address recovery is 6/8. Shared initialization preserves useful geometry: both children recover all 306 new features, and 214 of them occupy the same address (69.93%).

F.4 Full-Activation Reconstruction

For the two language tasks, the selected LineageSAE checkpoints encode a residual after removing a fixed four-dimensional shared subspace. Let h be the original activation, c the fixed center, and U the shared basis. The SAE input is $r = (I - UU^\top)(h - c)$, and the full reconstruction used in Tables 1 and 3 is $\hat{h} = h - r + D(E(r))$, where E and D are the selected SAE encoder and decoder. Full-activation normalized mean squared error (NMSE) divides $\sum \|h - \hat{h}\|^2$ by $\sum \|h - \bar{h}\|^2$ on the held-out view. The residual-input NMSE in the experiment grid uses the same reconstruction error but divides by $\sum \|r - \bar{r}\|^2$. The two denominators serve the two measurements introduced above, and both are reported for the selected checkpoints.

The old/new1/new2 checkpoint SHA-256 hashes and held-out input hashes were verified against the selected three-model evaluation records before reconstruction. Re-evaluating those six checkpoints reproduced the earlier residual-input NMSE within 2×10^{-5}. An independent pass over the held-out activations checked both centered-energy denominators and the resulting full-activation NMSE. The two new-model means, each evaluated on its held-out activations, are:

Table 32: Residual-input and full-activation reconstruction error for the selected language-task checkpoints.

Task	Residual-input NMSE	Full-activation NMSE
Wiki-to-Code	0.170139	0.014117
Pythia model update	0.199005	0.014493

F.5 Independent Candidate-Pool Diagnostic for Relaxed Archetypal SAE

Table 33 reports a random-index control for the independent-candidate Relaxed Archetypal SAE checkpoints. The Wiki-to-Code and Pythia model-update native decoder cosines also appear in Tables 1 and 3; this diagnostic adds the random-index controls and the SynthSAEBench and Toy cosines, which are not columns in the main comparison tables.

The response and reconstruction measurements use the shared-candidate checkpoint set described in Appendix B.4. Each independent-candidate run uses $\delta = 1$, learning rate 3×10^{-4}, and 10,000 updates. P–C averages old/new1 and old/new2; C–C compares new1/new2. SynthSAEBench averages its six conditions and reuses a task's identical old input across rarity conditions. Wiki-to-Code and model update reuse the old SAE with the same input hash. The random control permutes one dictionary before same-index scoring.

Table 33: Native and randomly permuted decoder correspondence with independently sampled candidate pools.

Task	P–C native	C–C native	P–C random	C–C random
Wiki-to-Code	0.4718	0.4859	0.4722	0.4876
SynthSAEBench	0.0060	0.0075	0.0071	0.0080
Toy superposition	0.0011	0.0013	−0.0196	0.0182
Pythia model update	0.5385	0.5441	0.5397	0.5441

The independent-candidate language-task cosines sit at their random-index controls: Wiki-to-Code C–C is 0.4859 versus 0.4876, and model-update C–C is 0.5441 in both cases. A high raw cosine therefore reflects shared decoder geometry, and the response metrics identify address-specific correspondence. The random-index comparison quantifies this effect directly.

F.6 Semantic Optimal Transport as Post-Hoc Feature Matching

Semantic Optimal Transport (Semantic OT) estimates a correspondence between features of independently trained sparse autoencoders (SAEs) after training; it does not change the underlying L_2-regularized SAE checkpoints. Table 34 therefore compares native feature addresses with the mapped addresses on the same held-out examples. A valid target feature activated at least 16 times in the calibration split, which was separate from the held-out split. Each gain subtracts the native value computed on that same valid-target set from its mapped value. P–C averages the two old/new comparisons, and C–C compares the two new SAEs. The six SynthSAEBench conditions are averaged within each comparison type. Decoder cosine compares feature directions, J is the Jaccard similarity of binary activation supports, and ρ is Spearman correlation of activation magnitudes. Multiple target features may select one source feature. The mapped scores give the correspondence that post-hoc matching recovers at evaluation time; the underlying L_2 SAE checkpoints, their reconstruction, sparsity, cross-entropy recovery, and ground-truth feature recovery are unchanged by the map.

Semantic OT substantially improves correspondence for fixed L2-TopK dictionaries. For example, Wiki-to-Code C–C Jaccard rises from 0.0012 to 0.3160 on the valid-target set, and Toy C–C cosine rises from 0.0116 to 0.8547. These improvements quantify information recoverable by matching after training. They complement the native-address objective, which asks whether the correspondence is already available at deployment.

Table 34: Changes in feature correspondence after Semantic Optimal Transport.

Decoder cosine

Task	Pair	Native	Mapped	Gain
Wiki-to-Code	P–C	0.0288	0.0707	0.0420
	C–C	0.0261	0.6014	0.5753
SynthSAEBench	P–C	0.2745	0.4711	0.1966
	C–C	−0.0021	0.5387	0.5408
Toy superposition	P–C	0.3401	0.6598	0.3197
	C–C	0.0116	0.8547	0.8431
Pythia model update	P–C	0.1000	0.6665	0.5666
	C–C	0.0252	0.6624	0.6371

Support Jaccard

Task	Pair	Native	Mapped	Gain
Wiki-to-Code	P–C	0.0027	0.0200	0.0173
	C–C	0.0012	0.3160	0.3148
SynthSAEBench	P–C	0.1077	0.3280	0.2203
	C–C	0.0068	0.4620	0.4552
Toy superposition	P–C	0.1413	0.2271	0.0857
	C–C	0.0145	0.6176	0.6031
Pythia model update	P–C	0.0455	0.3545	0.3090
	C–C	0.0016	0.3546	0.3529

Activation Spearman

Task	Pair	Native	Mapped	Gain
Wiki-to-Code	P–C	0.0025	0.0324	0.0299
	C–C	0.0004	0.4422	0.4419
SynthSAEBench	P–C	0.1319	0.3904	0.2586
	C–C	0.0012	0.4969	0.4957
Toy superposition	P–C	0.1808	0.3493	0.1684
	C–C	0.0029	0.7115	0.7085
Pythia model update	P–C	0.0607	0.4932	0.4325
	C–C	0.0006	0.4944	0.4939

F.7 Calibration-Matched Decoder and Parent–Child Diagnostics

Tables 35 and 36 supplement the selected four-task main results. P–C averages the two comparisons of the old-view sparse autoencoder (SAE) with each new-view SAE; C–C compares the two new-view SAEs. Stable Feature Pool instead averages six old-parent/new-child comparisons after mapping the old parents into pool order; its two children share pool initialization. Decoder cosine compares their decoder directions. Native scores use the deployed feature indices, whereas matched scores use feature correspondence fitted on a separate calibration split. Matching is fitted on the calibration split and applied only for evaluation. J denotes the Jaccard similarity of binary activation supports, ρ is Spearman correlation of activation magnitudes, and L_0 is the mean number of active features per sample. Paired cells in Table 36 report P–C/C–C in that order. The calibration-matched Archetypal measurements use the shared-candidate checkpoints; independent-candidate native measurements are analyzed in Appendix F.5. Stable Pool has no separate old-parent/new-child pair to match, since its two children share the pool initialization.

Table 35: Decoder correspondence after calibration-based feature matching.

Wiki-to-Code

Method	P–C cosine	C–C cosine
iSAE	0.1289	0.4657
AbsTopK	0.1382	0.5628
MP-SAE	0.1062	0.4506
Archetypal	<u>0.4742</u>	0.4913
RA-SAE	**0.4773**	0.5075
Matryoshka SAE	0.0694	0.2554
BatchTopK	0.1657	<u>0.6149</u>
Gated SAE	0.1308	0.2927
SharedSAE	0.0432	0.1368
L2-TopK	0.1701	0.5961
Stable Pool	–	0.5923
OSAE	0.1094	0.4624
LineageSAE	0.2062	**0.6469**

Pythia model update

Method	P–C cosine	C–C cosine
iSAE	0.6432	0.6306
AbsTopK	0.6266	0.6182
MP-SAE	0.4391	0.4021
Archetypal	0.5434	0.5433
RA-SAE	0.5430	0.5401
Matryoshka SAE	0.3298	0.3300
BatchTopK	0.6314	0.6166
Gated SAE	0.6378	0.6185
SharedSAE	<u>0.7671</u>	0.6024
L2-TopK	0.6611	0.6590
Stable Pool	–	<u>0.9478</u>
OSAE	0.5250	0.5348
LineageSAE	**0.9589**	**0.9708**

Table 36: Additional parent–child and child–child diagnostics under controlled feature changes.

SynthSAEBench

Method	Native cos.	P–C J	Native ρ	Matched cos.	L_0
iSAE	0.4370/0.0024	0.0165	0.0281/−0.0000	0.1705/0.1613	16.00
AbsTopK	0.1746/0.0038	0.0614	0.0749/−0.0001	0.4793/0.5187	16.00
MP-SAE	0.0542/0.0011	0.0223	0.0238/0.0056	**0.8422/0.9435**	11.57
Archetypal SAE	–	0.0342	0.0396/0.0014	0.4189/0.5465	15.99
RA-SAE	–	0.0729	0.0872/0.0013	0.5699/0.6058	16.00
Matryoshka SAE	0.0027/0.0082	0.0114	0.0067/0.0097	0.4787/0.5216	16.18
BatchTopK	0.2058/0.0155	0.0779	0.1003/0.0064	0.5324/0.6056	15.92
Gated SAE	0.2105/−0.0010	0.0716	0.0949/0.0056	0.5624/0.5782	17.21
SharedSAE	0.3047/0.0037	0.1287	0.1681/0.0003	0.5290/0.6129	16.00
Stable Pool	**0.7228/0.9421**	**0.3060**	**0.3918/0.6504**	–/0.8351	15.89
L2-TopK	0.2734/−0.0018	0.0998	0.1226/0.0012	0.5775/0.6354	16.00
OSAE	0.1957/0.0010	0.0684	0.0806/0.0008	0.4522/0.4723	16.00
LineageSAE	0.6331/0.5961	0.2557	0.3268/0.4910	0.4759/0.6339	16.00

Toy superposition

Method	Native cos.	P–C J	Native ρ	Matched cos.	L_0
iSAE	0.4884/0.0095	0.0257	0.0392/0.0004	0.1812/0.1868	8.00
AbsTopK	0.0103/−0.0033	0.0022	0.0000/−0.0003	−0.0033/0.0633	8.00
MP-SAE	0.0052/−0.0060	0.0224	0.0093/0.0103	0.6108/0.6329	11.25
Archetypal SAE	–	0.0605	0.0741/0.0115	0.5667/0.5928	8.00
RA-SAE	–	0.0842	0.0981/0.0052	**0.6314**/0.6457	8.00
Matryoshka SAE	0.0131/0.0241	0.0321	0.0293/0.0313	0.4600/0.4814	7.98
BatchTopK	0.3447/−0.0013	0.0098	0.0000/0.0170	−0.0055/0.4484	8.00
Gated SAE	0.3356/−0.0101	0.0134	−0.0002/0.0117	−0.0064/0.4452	9.33
SharedSAE	0.0038/0.0003	0.0159	0.0002/0.0003	−0.0027/0.4690	8.00
Stable Pool	**0.7873/0.9900**	**0.2306**	**0.3032**/0.4567	–/0.6067	7.89
L2-TopK	0.3382/0.0110	0.1098	0.1420/0.0021	0.5706/0.6437	8.00
OSAE	0.2408/−0.0055	0.0883	0.1026/0.0024	0.4434/0.4259	8.00
LineageSAE	0.7484/0.7183	0.0140	−0.0005/**0.5872**	−0.0069/**0.7298**	8.00

G SCOPE, FAILURE MODES, AND OPEN DIRECTIONS

This section records what the reported measurements cover, where the method still fails, and which questions remain open. The corresponding protocols and per-setting numbers are in Appendix B.

Identity is relative to a lineage root. Indices are shared within a lineage, not across independently constructed roots: children of the same root reach same-index profile cosine near 0.981, whereas children of different roots are near zero before matching (Table 12). A descendant therefore inherits the index system of its root rather than a vocabulary that is canonical across the literature. Recovering a common system from independently trained roots, which post-hoc matching approximates at evaluation time, remains an open problem and a reason release is defined relative to a parent.

Accumulated drift and the cost of adaptation. Adjacent-generation J rises from 0.6773 to 0.7673 over four generations: each version is more consistent with its immediate predecessor than the previous version was with its own. Root-relative agreement follows the accumulated displacement from the original root, so a lineage that adapts repeatedly becomes progressively more specific to the current view. The mechanism controls in Table 25 characterize the two ends of this trade-off. Freezing the parent decoder gives the highest agreement (0.8166 adjacent J, decoder cosine 1.0000) at the highest NMSE (0.01580); independent retraining gives the lowest NMSE (0.01548) with near-zero native agreement (0.0015). LineageSAE occupies the intermediate point (0.7361 adjacent J, 0.01563 NMSE, 95.80% CE recovery): it carries the previous vocabulary forward while leaving released addresses free to adapt. Controlled removal and common-address recovery are the measurements that credit this behavior (Figure 4).

Known-feature recovery and its threshold. Recovery is measured against a fixed correlation and decoder-cosine threshold. A feature that becomes rare enough to fall below the finite-sample resolution of the response statistics can lose its address even though no semantic change occurred. Sub-0.8 correlation thresholds, longer calibration windows, and lower sparsity budgets each move this boundary, and none is a substitute for recovery measurement.

Transfer scope. Section 3.3 covers one model family at two layers, one reduced dictionary width, and one additional model family, with the per-setting measurements in Appendix E.2. Correspondence at scale has therefore been tested on a set of architectures and domains that can be extended further. The synthetic studies share one generating representation per task group and reuse coefficient samples across views, so their variation reflects training rather than generator changes. Causal-effect preservation under intervention, non-TopK sparsity mechanisms, vocabularies larger than $N = 4096$, and lineages over more than four generations remain open.

Thresholds inside the release rule. Release depends on background correction, a fixed cross-fitting split, and finite-sample utility estimates. In the controlled study the target contribution is positive in all $22{,}352$ retained-feature observations, $415/420$ ordinary and $20/20$ difficult deletions turn negative, and the remaining failures are exactly the ordinary deletions whose competing similar features were not fully corrected. Where the selected background's largest covariance source is a surviving similar feature (0.7031–0.8076 response correlation), the correction is effective; the residual failures indicate the boundary of what a linear background adjustment can separate, and they are the failures a nonlinear or iterative correction would need to address.