BEYOND COORDINATEWISE INFERENCE IN SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse autoencoders (SAEs) are often used as if each learned coefficient represented an independent, editable concept. We show this assumption is structurally false. Standard SAE encoders, including affine, ReLU, TopK, and Gated SAEs, compute each coefficient from a single affine score of the activation. The selection step in TopK compares coordinates, but each score remains a projection of the input alone. We call these coordinatewise-score encoders. Because SAE dictionaries are overcomplete, such scores cannot each respond to their own feature's decoder direction while ignoring all others, so coefficient errors persist even when the encoder is given the dictionary that generated the data. The errors have two consequences: (1) after a single-coordinate edit is decoded into activation space and the resulting activation is re-encoded, the inferred code change can involve unedited coordinates or even move the target coordinate in the wrong direction, creating **attribution ambiguity**. (2) During training, the same limitation lets the decoder adapt to compensate for the coefficient error instead of the encoder correcting it, which we call **decoder absorption**, a mechanism of which feature absorption—a well-known SAE phenomenon—is a special case. We show that a cross-coordinate encoder can overcome this failure, and introduce a local consistency loss that further improves edit consistency and reconstruction quality over standard affine and scalar-nonlinear baselines in the large majority of configurations tested.

1 INTRODUCTION

Sparse autoencoders (SAEs) are widely used to decompose model activations ($x \in \mathbb{R}^d$) into sparse, interpretable coordinates (Bricken et al., 2023; Cunningham et al., 2023; Gao et al., 2025). After training, an SAE provides an encoder (Enc) and a model decoder ($\widetilde{D}$), which researchers can use to edit features and steer model behavior (Templeton et al., 2024; Marks et al., 2025). For example, we apply a requested edit δ, positive or negative, to encoded coordinate i, producing the edited activation ($x' = x + \delta \widetilde{d}_i$). We would then pass x' through the rest of the language model to observe how its behavior changes.

The interpretation of such an edit rests on the SAE's coordinates, the semantic system through which activations are read: the encoded coefficient ($\widehat{z}_i(x)$) is read as the amount of feature i present in x, and the edit changes feature i on that same basis. We would therefore expect the encoder to read x' as x with coordinate i increased by δ and no other coordinate changed.

This expectation can fail even when the SAE is given the dictionary that generated the data. In such a controlled dictionary (Appendix F.4), we perform a standard ablation, setting feature 0 to zero with the affine encoder's reading, $\delta = -\widehat{z}_0(x)$, so that $x' = x - \widehat{z}_0(x)\, d_0$. However, we find that the reference coefficient of feature 0 in the edited activation, $z_0^\star(x')$, remains nonzero and varies with the coactive feature (Figure 1).

In trained SAEs, the same question arises without reference coefficients: the encoder is the only map that assigns semantic coordinates to an activation, and the edit is made on the basis of the interpretation it provides. But if we re-encode the edited activation x' with the same SAE encoder Enc, the resulting encoded code ($\widehat{z}(x') = \mathrm{Enc}(x')$) can disagree with the requested change (Figure 2): the target can change by the wrong amount, while unedited encoded coefficients ($\widehat{z}_k(x')$) also change. TopK selection can worsen this mismatch: an unedited encoded coefficient ($\widehat{z}_k(x')$) can suddenly become

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.

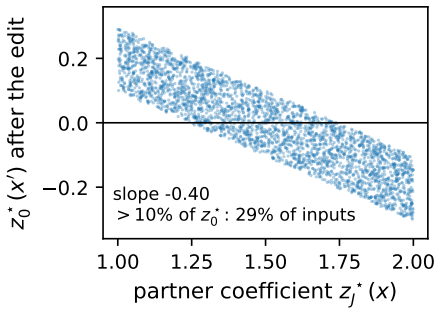


Figure 1: Setting feature 0 to zero with the encoder's own reading, $\delta = -\widehat{z}_0(x)$, in the controlled dictionary of Section 6 with an affine encoder and the reference decoder. Each point is one input; the vertical axis is the reference coefficient $z_0^\star(x')$ after the edit, whose requested value is zero. Appendix F.4 gives the protocol and the Scalar and Cross encoders.

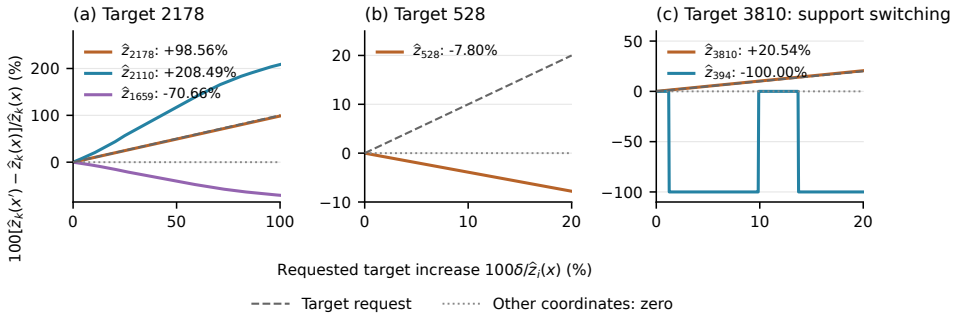


Figure 2: Re-encoding after single-coordinate edits. Panel (a) shows changes in the target and unedited coordinates; (b) shows a target reversal; (c) shows TopK support switching. All three panels use TopK with $K = 20$. The horizontal axis is the requested target increase; each solid curve reports changes relative to that coordinate's original encoded coefficient $(\widehat{z}_k(x))$. Both axes use percentages. Appendix F gives protocols and sources; Appendix F.3 shows all twenty originally active coordinates for (a–b) and for a third case.

zero and even become positive again later as the requested increase grows (Figure 2(c)). The returned edit $(\Delta_i^{\mathrm{ret}}(x, \delta) \equiv \mathrm{Enc}(x') - \mathrm{Enc}(x))$ can differ from the requested change of δ in coordinate i. By the SAE's own account, the intervention then changed feature i by a different amount than requested and changed other features as well, so a downstream change cannot be attributed to feature i alone.

Section 3 proves that standard affine encoders cannot recover the reference coefficients $(z_i^\star(x))$ in an overcomplete dictionary, even when provided with the correct decoder directions. **Section 4** shows that the resulting coefficient errors compromise the reliability of SAE steering, causing the attribution ambiguities described above. **Section 5** demonstrates that during training, these errors drive decoder absorption, as the decoder adapts to compensate for the encoder's deficit instead of the encoder correcting it. This motivates cross-coordinate encoders, whose candidate values, and not only their support selection, are computed across coordinates, and local consistency regularization to improve inference and encourage the intended edit response (**Section 6**). We evaluate their effects on editing and reconstruction in trained language-model SAEs (**Section 7**).

Related work. The closest prior work concerns the gap between SAE encoders and exact sparse inference. O'Neill et al. (2025) show that trained encoders under-recover sparse codes that inference-time optimization recovers; Sun et al. (2026) relate amortized inference to absorption, splitting, and dead or dense latents; Cui et al. (2026) derive conditions under which reconstruction-trained SAEs fail to recover features. These results characterize what trained encoders or reconstruction-optimal SAEs recover; we instead show that the failure is structural: no coordinatewise-score encoder can exactly recover the coefficients of an overcomplete dictionary, even when the true decoder is given

(Section 3). Chanin et al. (2025) attribute feature absorption to sparsity pressure on hierarchical features; we give a complementary account in which absorption arises when the decoder compensates for the encoder's coefficient error (Section 5). Appendix A gives an **extended discussion**.

2 Setup, Terminology, and Notation

Let d denote the activation dimension and m the number of dictionary features. In a standard SAE, an encoder ($\mathrm{Enc} : \mathbb{R}^d \to \mathbb{R}^m_{\geq 0}$) maps an activation ($x$) to a sparse encoded code ($\widehat{z}(x)$). It computes candidate values ($v_i(x)$) before nonnegativity and support selection (Figure 3). In the affine and scalar nonlinear encoders of Section 6, each candidate value is a scalar function of one affine scalar score ($s_i(x) = w_i^\top x + b_i$) computed by that coordinate's own parameters; we call these **coordinatewise-score encoders**. The ReLU selector zeros negative values; TopK keeps the K largest candidates and applies ReLU. Gated uses a separate binary gate to select positive magnitude values (Appendix H.1). An affine encoder with a TopK-like selector is cross-coordinate in selection, not in score computation, and is therefore still a coordinatewise-score encoder. The model decoder ($\widetilde{D}$) and decoder bias (b_{dec}) give the reconstruction ($\widehat{x}(x)$) and reconstruction residual ($r(x)$):

$$\widehat{z}(x) = \mathrm{Enc}(x), \qquad \widehat{x}(x) = \widetilde{D}\widehat{z}(x) + b_{\mathrm{dec}}, \qquad r(x) = \widehat{x}(x) - x. \tag{1}$$

Training minimizes the reconstruction MSE (L_{rec}), defined as

$$L_{\mathrm{rec}} = \mathbb{E}\big[\|r(x)\|_2^2\big]. \tag{2}$$

The objective often also includes a sparsity penalty on the encoded code ($\widehat{z}(x)$).

Table 1: Notation for Sections 2–5. Additional notation is in Appendix B.

Term	Notation	Meaning / domain
Activation	$x \in \mathbb{R}^d$	Model state that the SAE aims to decompose.
Reference code	$z^\star(x) \in \mathbb{R}^m_{\geq 0}$	Generates controlled data $x = Dz^\star(x)$; entry i: reference coefficient ($z_i^\star(x)$).
Reference decoder	$D \in \mathbb{R}^{d\times m}$	Generates controlled data; column i: reference direction ($d_i \in \mathbb{R}^d$).
Model decoder	$\widetilde{D} \in \mathbb{R}^{d\times m}$	Reconstructs and edits activations; column i: model direction ($\widetilde{d}_i \in \mathbb{R}^d$).
Reconstruction	$\widehat{x}(x) \in \mathbb{R}^d$	$\widetilde{D}\widehat{z}(x) + b_{\mathrm{dec}}$, with $b_{\mathrm{dec}} \in \mathbb{R}^d$.
Reconstruction residual	$r(x) = \widehat{x}(x) - x$	Difference between the reconstructed and original activations, in $\mathbb{R}^d$.
Coefficient error	$\Delta z(x) = \widehat{z}(x) - z^\star(x)$	Difference between the encoded and reference coefficients, in $\mathbb{R}^m$.
Scalar score	$s_i(x) = w_i^\top x + b_i$	Affine function with row $w_i \in \mathbb{R}^d$ and bias b_i; Figure 3.
Candidate value	$v_i(x) \in \mathbb{R}$	Value before nonnegativity and selection; vector $v(x) \in \mathbb{R}^m$.
Encoded code	$\widehat{z}(x) = \mathrm{Enc}(x)$	Output after selection; its entries are encoded coefficients ($\widehat{z}_i(x)$), equal to $\mathrm{ReLU}(v_i(x))$ if i is selected and 0 otherwise.
Active support	$S(x) = \mathrm{supp}(\widehat{z}(x))$	Indices with positive output.
Requested edit	$\delta e_i \in \mathbb{R}^m$	Desired change; e_i is a coordinate basis vector.
Edited activation	$x' = x + \delta\widetilde{d}_i$	Decoded change added to the original input.
Returned edit	$\Delta_i^{\mathrm{ret}}(x, \delta)$	Code change inferred by the same SAE encoder after re-encoding, $\mathrm{Enc}(x') - \mathrm{Enc}(x)$.
Edit-return error	$\epsilon_i(x, \delta)$	$\|\Delta_i^{\mathrm{ret}}(x, \delta) - \delta e_i\|_2/\|\delta\|$, $\delta \neq 0$.
Reconstruction MSE	L_{rec}	Squared residual norm averaged across examples.
Partner loading	c	Fitted slope ratio β/α (App. C.3).

Term	Notation	Meaning / domain
Decoder absorption	ρ	$\widetilde{d}_j = d_j - \rho d_i$; fits: App. C.3.

3 Why Affine Encoders Cannot Recover Sparse Coefficients

Assume a known reference decoder ($D = [d_1, \ldots, d_m]$) and nonnegative sparse reference codes ($z^\star(x)$) that generate a family of activations (x):

$$x = Dz^\star(x) = \sum_{k=1}^{m} z_k^\star(x) d_k. \tag{3}$$

Assume each reference code ($z^\star(x)$) is unique within the specified sparse family. We ask whether the SAE can recover it, $\widehat{z}(x) = z^\star(x)$. For the affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$), let the fixed parameters be w_i, b_i. Every positive encoded coefficient ($\widehat{z}_i(x)$) satisfies

$$\widehat{z}_i(x) = w_i^\top x + b_i = z_i^\star(x) w_i^\top d_i + \sum_{j \neq i} z_j^\star(x) w_i^\top d_j + b_i. \tag{4}$$

If every pair of reference coordinates can co-occur with freely varying positive coefficients, matching coefficients for exact recovery requires $w_i^\top d_i = 1$ and $w_i^\top d_j = 0$ for every $j \neq i$. Applying these requirements to every row gives $WD = I_m$, where $W = [w_1, \ldots, w_m]^\top$; exact recovery also forces every $b_i = 0$. For an overcomplete reference decoder ($D \in \mathbb{R}^{d \times m}$) with $m > d$, this is impossible:

$$\mathrm{rank}(WD) \leq d < m = \mathrm{rank}(I_m). \tag{5}$$

Thus the affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$) cannot recover the reference code ($z^\star(x)$) across this family even with the correct reference decoder (D) and a unique exact sparse solution. Appendix E measures the corresponding off-diagonal contribution inside trained affine SAEs, using each model's own decoder in place of a reference one: summed over the other active coordinates, the off-diagonal contribution to $\widehat{z}_i(x)$ is at least half of coordinate i's own contribution for half of all active coordinates. A controlled illustration of recovery failure appears in Table 3 (setup in Appendix C.1). Sections 4 and 5 examine the consequences of this limitation for editing and reconstruction training.

4 Attribution Ambiguity: Editing One Coordinate Changes Others

A requested edit (δe_i) is intended to change only coordinate i, so that any resulting change in the language model's behavior can be attributed to that coordinate. It produces the edited activation ($x' = x + \delta \widetilde{d}_i$), which we can pass to the remaining language model to observe its behavior. But re-encoding x' with the same encoder Enc can give a returned edit ($\Delta_i^{\mathrm{ret}}(x, \delta) \equiv \mathrm{Enc}(x') - \mathrm{Enc}(x)$) that differs from the request. We call this phenomenon **attribution ambiguity**.

For an affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$), consider an encoded coefficient ($\widehat{z}_k(x)$) that remains positive before and after the edit. Substituting $x' = x + \delta \widetilde{d}_i$ into $\widehat{z}_k(x) = w_k^\top x + b_k$ and using linearity gives

$$\begin{aligned} \widehat{z}_k(x') &= w_k^\top x' + b_k = w_k^\top (x + \delta \widetilde{d}_i) + b_k \\ &= \widehat{z}_k(x) + \delta w_k^\top \widetilde{d}_i, \\ [\Delta_i^{\mathrm{ret}}(x, \delta)]_k &= \widehat{z}_k(x') - \widehat{z}_k(x) = \delta w_k^\top \widetilde{d}_i. \end{aligned} \tag{6}$$

For the returned edit ($\Delta_i^{\mathrm{ret}}(x, \delta)$) to match the requested edit δe_i on a fixed active support $S(x)$, the target must satisfy $w_i^\top \widetilde{d}_i = 1$, while every other active coordinate $k \neq i$ must satisfy $w_k^\top \widetilde{d}_i = 0$. As shown in Section 3, an affine encoder cannot generally satisfy these conditions simultaneously for an overcomplete dictionary. The SAE can thus describe the edited activation as changes in several coordinates, confounding attribution to the intended encoded coefficient ($\widehat{z}_i(x)$).

Figure 2(a–b) shows these effects in trained SAEs. In panel (a), the target nearly doubles as requested while another coordinate increases by 208.49%; in panel (b), a requested 20% increase produces a

7.80% decrease in the target encoded coefficient ($\widehat{z}_{528}(x')$). In a third case (Appendix F.3), 11 of the 19 eligible unedited coordinates change by at least 10% of their original values. The edit also changes the candidate values ($v_k(x')$) that TopK ranks, so the active support ($S(x')$) can change: in Figure 2(c), the unedited encoded coefficient ($\widehat{z}_{394}(x')$) drops to zero, becomes positive again, and returns to zero as the requested edit (δe_{3810}) increases.

5 How Coefficient Error Drives Decoder Absorption

When the affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$) cannot recover the reference code ($z^\star(x)$), reconstruction training can adapt even a correct model decoder ($\widetilde{D} = D$) to the inaccurate outputs. For an input with two coactive reference coordinates, $x = z_i^\star(x)d_i + z_j^\star(x)d_j$, define the affine encoder's partner loading (c), coefficient error, and the compensating decoder direction by

$$\begin{aligned} \widehat{z}_i(x) &= z_i^\star(x) + c z_j^\star(x), & \Delta z_i(x) &\equiv \widehat{z}_i(x) - z_i^\star(x) = c z_j^\star(x), \quad c \equiv w_i^\top d_j, \\ \widetilde{d}_j &\equiv d_j - c d_i, & \widehat{z}_i(x) d_i + z_j^\star(x)\widetilde{d}_j &= z_i^\star(x) d_i + z_j^\star(x) d_j = x. \end{aligned} \tag{7}$$

Here $\widehat{z}_j(x) = z_j^\star(x) > 0$, and all other encoded coordinates are zero. We call $\Delta z_i(x)$ the **coefficient error** and the compensating decoder change **decoder absorption**. We denote the amount of d_i absorbed into the compensating change of $\widetilde{d}_j$ by ρ, so $\widetilde{d}_j = d_j - \rho d_i$. In this construction, $\rho = c$. For the $c < 0$ case, $\widehat{z}_i(x)$ is too small, so $\widetilde{d}_j$ adds $|c|d_i$ to supply the missing contribution. When coordinate j occurs only on inputs where i is active, this systematic compensation is feature absorption: i is the parent and j is the child (Chanin et al., 2025). For $c > 0$, $\widehat{z}_i(x)$ is too large, so the same column subtracts cd_i to cancel the excess. The $c < 0$ branch also aligns with the sparsity incentive behind that account, and should therefore be favored during training. Appendix C.3 gives the reconstruction loss and unit-column normalization. For example, with $c = -\frac{1}{2}$ and reference coefficients ($z_i^\star(x) = 1,\ z_j^\star(x) = 2$), the parent encoded coefficient ($\widehat{z}_i(x)$) is zero. The child alone reconstructs $x = 2(d_j + \frac{1}{2}d_i)$, while parent-only inputs still give $\widehat{z}_i(x) = z_i^\star(x)$.

Table 4 reports this behavior. With the decoder fixed at $\widetilde{D} = D$, the affine encoder's coefficient error leaves $L_{\mathrm{rec}} = 1.25 \times 10^{-2}$, whereas $z^\star(x)$ reconstructs exactly. Allowing the partner decoder directions to train lets the coordinatewise-score encoders absorb their coefficient error; for the affine encoder, L_{rec} falls to 9.51×10^{-8}. As predicted by Equation 7, the learned columns subtract the excess contribution from feature i: decoder absorption ($\rho = 0.500000$) matches the fitted $c = 0.499999$. Appendix G traces this mechanism in a normally trained GPT-2 SAE: the target coefficient decreases, the child decoder contribution grows along the parent direction, and the corresponding gradient, Adam updates, and logit effects are measured. This gives a concrete training trace of the decoder-absorption mechanism and shows how the feature-absorption pattern studied by Chanin et al. (2025) appears as a special case.

6 Why Cross-Coordinate Encoders Can Overcome the Recovery Limitation

A coordinatewise-score encoder with a scalar nonlinearity faces the same limitation as an affine encoder. The limitation in Section 3 comes from each candidate value being computed from a single score, not from the encoder's affine form or from limited expressivity. Recovering coordinate i requires knowing how much each other active coordinate contributes to x, and a coordinatewise-score encoder discards that information before any nonlinearity is applied (Figure 3). Granting an arbitrarily expressive scalar nonlinearity therefore does not remove the limitation (Proposition 1, Appendix D). The affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$) and scalar nonlinear encoder ($\mathrm{Enc}_{\mathrm{sc}}$) each compute one scalar score ($s_i(x) = w_i^\top x + b_i$) for coordinate i. $\mathrm{Enc}_{\mathrm{aff}}$ uses $v_i(x) = s_i(x)$; $\mathrm{Enc}_{\mathrm{sc}}$ applies a function $g_i : \mathbb{R} \to \mathbb{R}$ to obtain the candidate value ($v_i(x) = g_i(s_i(x))$). Because $x = Dz^\star(x)$ (Equation 3), the score

$$s_i(x) = w_i^\top x + b_i = \sum_j (w_i^\top d_j)\, z_j^\star(x) + b_i \tag{8}$$

can mix several reference coefficients ($z_j^\star(x)$). However, g_i observes only the resulting scalar $s_i(x)$, rather than those coefficients separately: no matter how expressive g_i is, if $s_i(x_a) = s_i(x_b)$ then

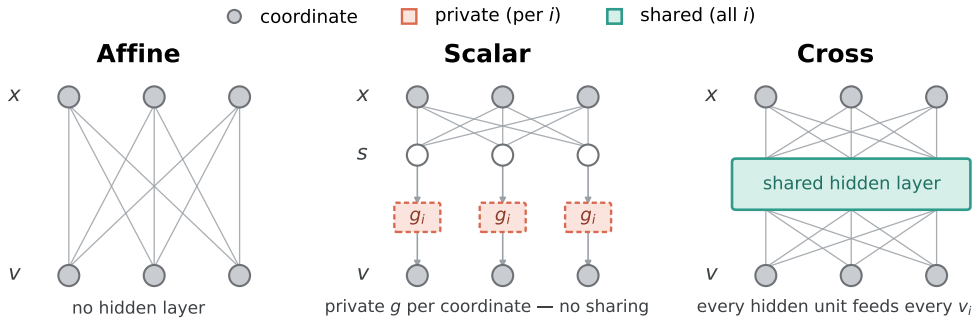


Figure 3: Affine, Scalar, and Cross encoders. For coordinate i, Affine uses the scalar score $s_i(x) = w_i^\top x + b_i$ directly as $v_i(x)$; Scalar applies a private nonlinear map $v_i(x) = g_i(s_i(x))$; Cross computes $v(x)$ through hidden units shared across all outputs. Coral components are private to one coordinate; teal components are shared across coordinates. All three produce candidate values $v(x)$; the selector (ReLU, TopK, BatchTopK, or Gated) is applied afterward and is the same for each, so the distinction drawn here concerns score computation, not support selection.

$v_i(x_a) = g_i(s_i(x_a)) = g_i(s_i(x_b)) = v_i(x_b)$. For example, if $s_i(x) = z_i^\star(x) + z_j^\star(x)$, the coefficient pairs $(z_i^\star(x), z_j^\star(x)) = (1, 2)$ and $(2, 1)$ both produce $s_i(x) = 3$, although the required target coefficient $z_i^\star(x)$ is different. Again, avoiding this ambiguity requires $w_i^\top d_i \neq 0$ and $w_i^\top d_j = 0$ for every coactive partner j, which cannot hold for all rows (Section 3). $\mathrm{Enc}_{\mathrm{aff}}$ and $\mathrm{Enc}_{\mathrm{sc}}$ are therefore coordinatewise-score encoders (Section 2): each candidate value $(v_i(x))$ is a scalar function of one affine score computed by that coordinate's own parameters, namely w_i and b_i, together with the hidden units of g_i in $\mathrm{Enc}_{\mathrm{sc}}$.

A cross-coordinate encoder combines information across coordinates. The cross-coordinate encoder $(\mathrm{Enc}_{\mathrm{cross}})$ uses a multivariate candidate-value map $(F : \mathbb{R}^d \to \mathbb{R}^m)$ to compute $v(x) = F(x)$ jointly from the full activation. Unlike $\mathrm{Enc}_{\mathrm{aff}}$ and $\mathrm{Enc}_{\mathrm{sc}}$, its hidden units are shared across outputs: each hidden unit reads x and can contribute to every candidate value $v_i(x)$ (Figure 3). Consequently, inputs that produce the same affine scalar score $s_i(x)$ can still activate different shared hidden units, allowing the encoder to assign them different candidate values for coordinate i.

Local consistency regularization. Reconstruction alone does not enforce the intended edit response, so we add a local consistency loss (L_{LC}). It imposes local response constraints on the encoder: on the current active support $(S(x))$, each candidate value $(v_k(x))$ should change by δ along its own model direction and remain unchanged along the other model directions indexed by that support. We penalize violations with finite differences. With unit-norm model directions $(\widetilde{d}_i)$, sample a target i from the active support $(S(x))$ and a small signed edit δ. Holding this support fixed, penalize

$$L_{\mathrm{LC}} = \mathbb{E}_{x,i,\delta}\left[\frac{\sum_{k\in S(x)} \left(v_k(x + \delta\widetilde{d}_i) - v_k(x) - \delta \mathbf{1}\{k = i\}\right)^2}{\max\{\delta^2, \varepsilon\}}\right], \tag{9}$$

where $\varepsilon = 10^{-8}$ is a numerical floor in this loss, and the candidate value $(v_k(x))$ is computed before the final ReLU or support selection. The desired change in v_i is δ; other candidate values (v_k) on $S(x)$ should stay unchanged. The controlled experiments instead sample and sum over the pre-truncation support $(S_{\mathrm{pre}}(x) = \mathrm{supp}(\mathrm{ReLU}(v(x))))$ before the supplied-pair mask (Appendix C.2). Gradients pass through both encoder evaluations, but not through the edit direction or sampled quantities in this loss. We call this training variant the locally regularized cross-coordinate encoder $(\mathrm{Enc}_{\mathrm{cross+LC}})$ (abbreviated Cross+LC); its architecture and parameter count are unchanged. Weighting and sampling are specified in Appendices C.2 and H.1 for the controlled and trained-SAE experiments.

Synthetic experiments: comparing the three encoders. We use a target reference direction (d_0) and four partner reference directions $(d_1, \ldots, d_4)$ in $\mathbb{R}^4$. The partners are the standard basis vectors

Table 2: Notation for the encoder variants introduced in this section.

Short label	Notation	Meaning
Cross	$\mathrm{Enc}_{\mathrm{cross}}$	Cross-coordinate encoder with jointly computed candidate values.
Cross+LC	$\mathrm{Enc}_{\mathrm{cross+LC}}$	Same encoder trained with Local Consistency Loss L_{LC} at weight λ_{LC}.

and $d_0 = \frac{1}{2}(1,1,1,1)^\top$, so the target direction overlaps every partner equally, $d_0^\top d_j = \frac{1}{2}$. No single affine row can then respond to d_0 while ignoring all four partners, which is exactly the condition Section 3 requires. We generate each input as $x = z_0^\star(x)d_0 + z_J^\star(x)d_J$, sampling the partner J uniformly from $\{1,2,3,4\}$ and the two reference coefficients $(z_0^\star(x), z_J^\star(x))$ independently from $\mathrm{Uniform}[1,2]$ (Appendix C.1).

We measure target coefficient recovery by $L_{\mathrm{coef}} = \mathbb{E}[(\widehat{z}_0(x) - z_0^\star(x))^2]$. With the model decoder $(\widetilde{D} = D)$ fixed, Table 3 shows that the cross-coordinate encoder $(\mathrm{Enc}_{\mathrm{cross}})$ achieves much lower coefficient MSE (L_{coef}) than the affine encoder $(\mathrm{Enc}_{\mathrm{aff}})$ or scalar nonlinear encoder $(\mathrm{Enc}_{\mathrm{sc}})$; Appendix D gives a cross-coordinate encoder that recovers $z^\star(x)$ exactly on this family. The advantage of cross-coordinate inference extends to decoder absorption (Table 4). Just as in the affine case discussed in Section 5, the scalar nonlinear encoder retains a large partner loading $(c = 0.499961)$, which is almost exactly absorbed into the decoder $(\rho = 0.500002)$. By contrast, the cross-coordinate encoder has $c = 0.001496$ and $\rho = 0.001524$; all tested Cross+LC variants likewise keep both quantities near zero (Appendix C.5).

Table 3: Recovery with the model decoder $(\widetilde{D} = D)$ fixed (five-seed means), under the supplied-pair protocol: reconstruction uses the support selector $(\mathcal{G}_J)$, which keeps only the two reference-active coordinates. Scalar and Cross have 995 parameters. Training details: Appendix C.2.

Model	coefficient MSE (L_{coef})	reconstruction MSE (L_{rec})
Affine $(\mathrm{Enc}_{\mathrm{aff}})$	1.6718×10^{-2}	1.2539×10^{-2}
Scalar $(\mathrm{Enc}_{\mathrm{sc}})$	1.5888×10^{-2}	1.2025×10^{-2}
Cross $(\mathrm{Enc}_{\mathrm{cross}})$	$\mathbf{3.3377 \times 10^{-8}}$	$\mathbf{7.5608 \times 10^{-8}}$

Table 4: Decoder absorption in the controlled two-coordinate family. Affine, Scalar, and Cross are five-seed means. "Before" uses the model decoder fixed to the reference decoder (D), so $\widetilde{D} = D$; "after" follows 4,000 joint-training updates with fixed $\widetilde{d}_0 = d_0$ and trainable unit-norm partner columns. Protocol and fits: Appendix C.3.

Encoder	L_{rec}, before	L_{rec}, after	c	ρ
Affine $(\mathrm{Enc}_{\mathrm{aff}})$	1.2539×10^{-2}	9.5145×10^{-8}	0.499999	0.500000
Scalar $(\mathrm{Enc}_{\mathrm{sc}})$	1.2025×10^{-2}	3.7825×10^{-7}	0.499961	0.500002
Cross $(\mathrm{Enc}_{\mathrm{cross}})$	7.5608×10^{-8}	1.9408×10^{-7}	0.001496	0.001524

7 Evaluation on Trained Language-Model SAEs

We test whether the encoder comparisons in Section 6 translate into more consistent edits and better reconstruction. Unlike the controlled experiments, these activations have no supplied reference code $(z^\star(x))$: we measure each trained SAE's behavior, not recovery of known semantics.

7.1 Experimental setup

We train SAEs with 4,096 features on 768-dimensional MLP activations from GPT-2 and OPT-125M, using TopK ($K \in \{50, 100\}$), Gated, and ReLU ($\lambda_{\text{sp}} \in \{2, 4\}$) selectors: twelve matched settings. Each setting is trained with Affine, Scalar, Cross, and Cross+LC at $\lambda_{\text{LC}} \in \{0.005, 0.02, 0.05, 0.1\}$, at seeds 42 and 43, giving 168 SAEs and twenty-four matched setting–seed pairs per comparison. Models are trained on OpenWebText for 30,000 steps and evaluated on held-out WikiText-103 development text. Appendix H gives configurations and metric definitions.

7.2 Edit consistency under steering

For an originally active coordinate i, we apply the requested edit (δe_i) with $\delta = \alpha \widehat{z}_i(x)$, where α is the relative edit level:

$$\alpha \in \{\pm 0.01, \pm 0.05, \pm 0.1, \pm 0.2, \pm 0.5, \pm 0.75, \pm 0.9, \pm 1\}.$$

For responses on the originally active coordinates, let $a(x) = \text{ReLU}(v(x))$ denote the nonnegative candidate values before the final selector (the magnitude path for Gated), and define the root-mean-square (RMS) response error

$$R_{\text{RMS}}(x, i, \alpha) = \frac{\left\| \left[a(x + \delta \widetilde{d}_i) - a(x) - \delta e_i \right]_{S(x)} \right\|_2}{|\delta| \sqrt{|S(x)|}}. \tag{10}$$

This metric measures whether the target changes by δ while the other initially active values remain fixed. The factor $\sqrt{|S(x)|}$ converts the error norm into a per-coordinate RMS. We first average over the sampled edits for each input, then average equally over inputs and the sixteen edit levels.

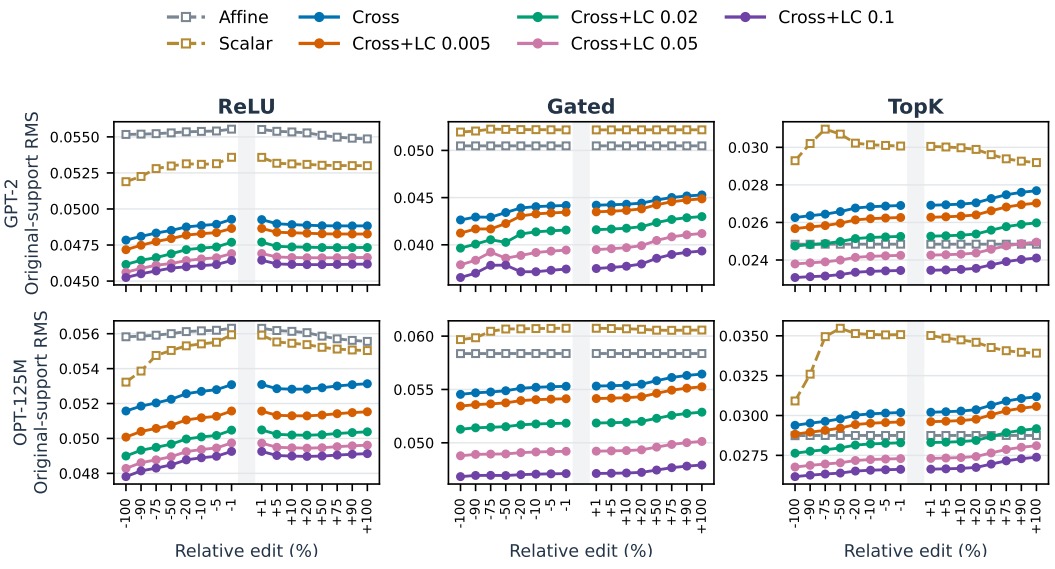


Figure 4: Original-support RMS at all sixteen signed edit levels, across six language-model–selector groups; lower values are better. Each curve averages the four setting–seed pairs in its panel (two sparsity settings at each of two seeds). Horizontal positions are categorical; the gap separates negative and positive edits. Each panel uses its own vertical range.

Comparing each setting–seed pair against its own two baselines, the unregularized cross-coordinate encoder has lower sixteen-level mean RMS than both baselines in all sixteen Gated and ReLU setting–seed pairs; in the eight TopK pairs it improves over Scalar but not Affine. With local consistency regularization, Cross+LC at $\lambda_{\text{LC}} = 0.05$ is below both baselines in twenty-two of twenty-four pairs, and at 0.1 it is below both in all twenty-four. Each of these counts splits evenly between the two seeds, so the replication holds setting by setting rather than only on average. Figure 4 shows that the same ordering holds at every edit level, not only in the sixteen-level mean: increasing the local

consistency weight shifts the curves downward throughout both edit ranges, and Cross+LC at 0.05 and 0.1 remains below both baseline curves at every level. Appendix H.2 reports the same comparison under the remaining combinations of measurement scope and readout stage, where the ordering of the methods is preserved.

Appendix H.5 takes the same edit one step further downstream. In each trained SAE we edit the coordinate whose model direction is closest to an external sentiment direction; instead of re-encoding the edited activation, we let the language model continue from the edited activation and score the continuations with a sentiment classifier independent of the SAE, so the measured quantity is the output text rather than the SAE code. On average across the twelve matched settings, at both seeds, the Cross+LC variants move the classifier score further in the requested direction than Affine and Scalar, at a comparable perplexity cost on unrelated held-out text.

7.3 Reconstruction quality and joint edit consistency

We report explained variance

$$\mathrm{EV} = 1 - \frac{\sum_x \|\widehat{x}(x) - x\|_2^2}{\sum_x \|x - \bar{x}\|_2^2}$$

on unedited inputs, where $\bar{x}$ is the mean activation over the evaluation set. This is a standard measure of SAE reconstruction quality (Gao et al., 2025; Rajamanoharan et al., 2024b).

For each matched setting and seed, Figure 5 counts a joint improvement when a cross-coordinate model achieves both lower RMS and higher EV than Affine and Scalar. The unregularized Cross encoder satisfies both conditions in sixteen of twenty-four setting–seed pairs, while Cross+LC at $\lambda_{\mathrm{LC}} = 0.05$ does so in twenty-two. At 0.1, RMS improves in all twenty-four pairs and EV in twenty-one, yielding twenty-one joint improvements. Figure 16(b) in Appendix H.3 compares the full EV distributions across methods and settings: Cross and the locally regularized variants through 0.05 have higher EV than both baselines in all twenty-four pairs. Thus, when tuned appropriately, local consistency improves edit consistency without sacrificing reconstruction quality.

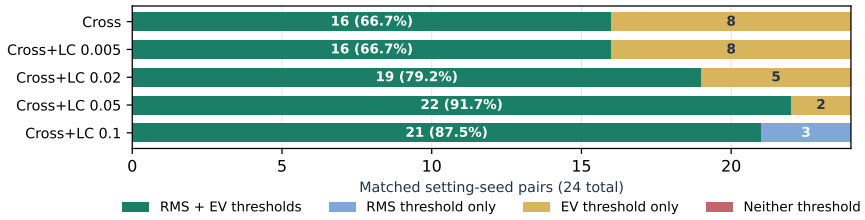


Figure 5: Joint RMS and EV outcomes over the twenty-four matched setting–seed pairs; each threshold is a strict improvement over both Affine and Scalar (lower sixteen-level mean RMS, higher EV on unedited inputs).

Additional metrics. Appendix H.4 reports fixed-code deletion false positives, active coefficients whose deletion lowers reconstruction error with the decoder and other coefficients fixed. Unregularized Cross lowers this rate in all twenty-four pairs relative to both baselines, by 8.2%–96.1% relative to the lower-rate baseline (88.4%–96.1% across the eight ReLU pairs); mean L_0 and the number of features used are comparable across all 168 SAEs, so the lower rate does not come from sparser codes or fewer used features.

8 Conclusion

Coordinatewise-score encoders cannot recover the coefficients of an overcomplete dictionary, a strict limitation on SAEs as instruments of interpretability. Cross-coordinate encoders with local consistency regularization reliably reduce this error but do not remove it. We therefore urge researchers to recognize this limitation when using SAEs and to report the resulting attribution ambiguity alongside steering results. We see closing the remaining gap as an important direction for future work.

AI USE STATEMENT

We used generative AI tools to aid and polish writing (language revision, notation and terminology consistency checks, and redundancy checks); for retrieval and discovery (finding related work and checking references against their primary sources); and for other supporting tasks (codebase organization, plotting code, preparation of figures, tables, and LaTeX, and checking the manuscript against the code). The authors reviewed all AI-assisted material and take responsibility for the final content.

ETHICS STATEMENT

Our work aims to improve the reliability of mechanistic interpretability methods, with the long-term goal of supporting safer and more reliable use of large language models. The study uses existing models and text datasets and does not involve human-participant experiments.

REPRODUCIBILITY STATEMENT

Appendix D provides the theoretical assumptions and proofs. Appendices C and H describe the data, model configurations, training settings, random seeds, and evaluation protocols for the controlled and trained-SAE experiments, respectively.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 107286–107325, 2024. doi: 10.52202/079017-3408.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

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

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with Matryoshka sparse autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 6077–6101. PMLR, 2025. URL https://proceedings.mlr.press/v267/bussmann25a.html.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/764ff7477b8e24dbe01531f6791e8bdf-Abstract-Conference.html.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL

https://proceedings.neurips.cc/paper_files/paper/2025/hash/311bd32bcd952f93cf820bacd6955f0a-Abstract-Conference.html.

Jingyi Cui, Qi Zhang, Yifei Wang, and Yisen Wang. On the limits of sparse autoencoders: A theoretical framework and reweighted remedy. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fc6bd0eef19459655d5b097af783661d-Abstract-Conference.html.

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

David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proceedings of the National Academy of Sciences*, 100(5): 2197–2202, 2003. doi: 10.1073/pnas.0437847100. URL https://www.pnas.org/doi/10.1073/pnas.0437847100.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24375–24410, 2024. doi: 10.52202/079017-0768.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

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. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 399–406, 2010. URL https://icml.cc/2010/papers/449.pdf.

Rémi Gribonval and Karin Schnass. Dictionary identification—sparse matrix-factorization via ℓ_1-minimization. *IEEE Transactions on Information Theory*, 56(7):3523–3539, 2010. doi: 10.1109/TIT.2010.2048466. URL https://doi.org/10.1109/TIT.2010.2048466.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. Transformer Circuits Thread, 2024. URL https://transformer-circuits.pub/2024/crosscoders/. Research update, October 25, 2024.

Alireza Makhzani and Brendan Frey. k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013. URL https://arxiv.org/abs/1312.5663.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Charles O'Neill, Alim Gumran, and David Klindt. Compute optimal inference and provable amortisation gap in sparse autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46877–46896. PMLR, 2025. URL https://proceedings.mlr.press/v267/o-neill25a.html.

Gonçalo Paulo, Stepan Shabalin, and Nora Belrose. Transcoders beat sparse autoencoders for interpretability. *arXiv preprint arXiv:2501.18823*, 2025.

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

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.

Wenjie Sun, Di Wang, and Lijie Hu. The price of amortized inference in sparse autoencoders. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/76358e88c3ebd38ba58af9d0d84b71e4-Abstract-Conference.html.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

A Extended Related Work

SAE training objectives and architectures. SAEs learn sparse representations of model activations to identify interpretable features (Bricken et al., 2023; Cunningham et al., 2023), motivated by the superposition of features in shared activation dimensions (Elhage et al., 2022). Research has developed through changes to both training objectives and architectures. Standard SAEs reconstruct activations at a single LLM layer. Single-layer transcoders instead approximate the MLP input–output mapping within that layer (Dunefsky et al., 2024; Paulo et al., 2025), while cross-layer transcoders use features from one layer to predict outputs at that layer and subsequent layers (Ameisen et al., 2025). Crosscoders jointly reconstruct activations across layers or models using shared sparse features (Lindsey et al., 2024). End-to-end SAE training optimizes the preservation of language-model outputs (Braun et al., 2024). Architectural developments include TopK and BatchTopK for explicit sparsity control (Makhzani & Frey, 2013; Gao et al., 2025; Bussmann et al., 2024), Gated SAEs for separating feature detection from magnitude estimation (Rajamanoharan et al., 2024a), and JumpReLU SAEs with learned activation thresholds (Rajamanoharan et al., 2024b). Matryoshka SAEs use nested dictionaries to learn features at multiple levels of abstraction (Bussmann et al., 2025).

Amortized sparse inference and its limits. A learned encoder approximates sparse inference with a fixed computation, an approach developed in learned sparse coding (Gregor & LeCun, 2010). O'Neill et al. (2025) show that standard SAE encoders can fail to recover sparse coefficients even when recovery is possible. Sun et al. (2026) connect amortized inference to trade-offs among absorption, splitting, and dead or dense latents. Cui et al. (2026) derive conditions under which reconstruction-trained SAEs fail to recover features in a theoretical setting and propose a reweighted reconstruction objective. MP-SAE uses sequential, residual-based encoding (Costa et al., 2025). Uniqueness of sparse representations and identifiability of learned dictionaries are longstanding questions in sparse coding (Donoho & Elad, 2003; Gribonval & Schnass, 2010). Our work traces how limitations of coordinatewise-score inference lead to attribution ambiguity in feature editing (Section 4) and decoder absorption during training (Section 5), and how these mechanisms can cause failures in downstream interpretation and steering with trained SAEs.

Feature absorption. Feature absorption occurs when a general feature fails to activate and a more specific feature represents both concepts. Chanin et al. (2025) show that sparsity pressure can encourage this behavior for hierarchical features. Section 5 extends this account by showing how coefficient errors can drive decoder absorption even when the active support is correct.

B Additional Notation

Table 5: Additional notation for experiments and proofs.

Term	Notation	Meaning / definition location
Support selector	$\mathcal{G}; \mathcal{G}_J, \mathcal{G}_+$	Preserves selected positive values. Rules: Appendices C.2, D.
Reference support	$S^\star(x) = \mathrm{supp}(z^\star(x))$	Positive reference coordinates; Appendix D.
Allowed support family	$\mathcal{S}; \Sigma \in \mathcal{S}$	Permitted supports and their column subsets D_Σ; Appendix D.
Coefficient regression	α, β, γ	Target/partner slopes and intercept; Eq. 15, Appendix C.3.
Decoder fit	λ_j, μ_j	Reference-direction loadings, with $\rho_j = -\mu_j/\lambda_j$; Eq. 17, Appendix C.3.
Pre-truncation support	$S_{\mathrm{pre}}(x) = \mathrm{supp}(\mathrm{ReLU}(v(x)))$	Indices of positive candidates; used for the controlled L_{LC} before the supplied-pair mask (Appendix C.2).
Local consistency loss	L_{LC}	Pre-selection response mismatch; Eq. 9, Section 6.
Local consistency weighting	$\lambda_{\mathrm{LC}}, \omega_t$	Auxiliary weight and warmup/ramp; Eq. 14, Appendix C.2.
Sparsity weighting	λ_{sp}	Weight on the ReLU or Gated sparsity penalty; Appendix H.1.
Coefficient MSE	L_{coef}	Target-coordinate recovery error; Eq. 11, Appendix C.1.
Nonnegative candidate value	$a_i(x) = \mathrm{ReLU}(v_i(x))$	Coordinate i before the final selector; $a(x)$ is the vector used in Section 7.2.
Edit-consistency metrics	$R_{S,+}, R_{m,+}$; $R_{S,\mathrm{final}}, R_{\mathrm{full}} \equiv \epsilon_i$	Candidate and final-code errors on original support S or all m coordinates; Eq. 33.
Original-support RMS	R_{RMS}; $R_{\mathrm{RMS,final}}$	$R_{S,+}/\sqrt{\lvert S\rvert}$ and $R_{S,\mathrm{final}}/\sqrt{\lvert S\rvert}$; Appendix H.2.
Edit scales	α; τ	Relative edit level, $\delta = \alpha\widehat{z}_i(x)$ (Section 7.2); downstream steering multiplier, edit $\tau\bar{n}\,\widetilde{d}_i$ (Appendix H.5). This α is distinct from the regression slope above.
Explained variance	EV	Reconstruction quality relative to evaluation-set activation variance; Section 7.3.
Mean active count	$L_0 = \mathbb{E}_x\lVert\widehat{z}(x)\rVert_0$	Mean number of positive encoded coefficients; Appendix H.
Reconstruction correction	$q_i(x)$; $q_i^+(x)$	Optimal fixed-decoder coefficient adjustment and its positive part; Eq. 26, Appendix G.
Gradient component	G_j	Negative-gradient alignment component; Eq. 27, Appendix G.
Decoder cosine	$\kappa_j^{(t)}$	Cosine of the current partner direction with the fixed 3k target direction; Appendix G.
Adam-step alignment	$\Delta_{\mathrm{Adam}}\kappa_j^{(t)}$	Change in decoder cosine across one realized Adam update; Appendix G.

C Controlled Experiments: Setup and Editing Diagnostics

This appendix gives the setup for the controlled comparison of the affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$), the scalar nonlinear encoder ($\mathrm{Enc}_{\mathrm{sc}}$), and the cross-coordinate encoder ($\mathrm{Enc}_{\mathrm{cross}}$). Table 3 reports recovery with the fixed decoder, and Table 4 reports joint training. Table 6 adds the fixed-decoder editing comparison. Tables 7 and 8 separately compare local consistency weights for the locally regularized cross-coordinate encoder ($\mathrm{Enc}_{\mathrm{cross+LC}}$).

C.1 Reference construction and evaluation scale

We use the data distribution in Section 6, whose reference directions $(d_0, \dots, d_4)$ in $\mathbb{R}^4$ are

$$d_0 = \tfrac{1}{2}(1,1,1,1)^\top, \quad d_1 = (1,0,0,0)^\top, \quad d_2 = (0,1,0,0)^\top,$$
$$d_3 = (0,0,1,0)^\top, \quad d_4 = (0,0,0,1)^\top.$$

The reference support $(S^\star(x))$ is $\{0, J\}$, with target $i = 0$ and sampled partner $j = J$. The remaining reference coefficients $(z_k^\star(x))$, $k \notin \{0, J\}$, are zero. The pair $\{0, J\}$ is identifiable on this construction, meaning that within the specified nonnegative two-column support family, x uniquely determines $z^\star(x)$: three components of the activation (x) equal $z_0^\star(x)/2$, and the remaining component has the strictly positive increment $z_J^\star(x)$. The representation is unique among the nonnegative representations using at most two columns of the reference decoder (D); allowing additional columns removes that restriction. Appendix D gives the exact inverse. The reference coefficients $(z_0^\star(x), z_J^\star(x))$ are used for evaluation and are never supplied as training labels.

The three unregularized models minimize only the reconstruction MSE (L_{rec}) from Equation 2; the locally regularized cross-coordinate encoder $(\text{Enc}_{\text{cross+LC}})$ adds the objective in Appendix C.2. We evaluate target recovery using the coefficient MSE (L_{coef}):

$$L_{\text{coef}} = \mathbb{E}_x\left[\left(\widehat{z}_0(x) - z_0^\star(x)\right)^2\right]. \tag{11}$$

Thus the reconstruction MSE (L_{rec}) averages over examples and sums over the four components; it is neither divided by four nor normalized by the norm or variance of the activation (x). The model decoder $(\widetilde{D})$ initially equals the reference decoder (D) and remains fixed except in the joint-training stage (Appendix C.3). Every table entry is averaged over seeds $0, \dots, 4$. Reconstruction and coefficient MSEs use 65,536 held-out examples; the editing subset is specified in Appendix C.4. Fitted quantities are computed within each seed before averaging.

C.2 Models, initialization, and selection protocols

This subsection specifies the controlled synthetic experiments. Complete configurations for the trained language-model SAEs are collected in Appendix H.1.

The affine encoder $(\text{Enc}_{\text{aff}})$ uses the candidate value $(v_k(x) = s_k(x))$, where the scalar score $(s_k(x) = w_k^\top x + b_k)$. The scalar nonlinear encoder (Enc_{sc}) instead uses the candidate value $(v_k(x) = g_k(s_k(x)))$, with

$$g_k(s) = a_k^{\text{skip}} s + b_k^{\text{skip}} + \sum_{h=1}^{64} v_{kh}^{\text{out}} \,\text{ReLU}(a_{kh}^{\text{in}} s + b_{kh}^{\text{in}}). \tag{12}$$

All parameters, including w_k, b_k, are trainable. The superscripts in Equation 12 distinguish internal network parameters from the candidate value $(v_k(x))$ and from the regression parameters introduced below. The cross-coordinate encoder $(\text{Enc}_{\text{cross}})$ computes the candidate values $(v(x))$ jointly from x with 99 shared hidden units; its input and output weight matrices have shapes $U \in \mathbb{R}^{99\times 4}$ and $V \in \mathbb{R}^{5\times 99}$, with no linear skip connection. Both nonlinear models have 995 trainable parameters; the affine encoder $(\text{Enc}_{\text{aff}})$ has 25.

In the supplied-pair protocol, the support selector $(\mathcal{G}_J)$ applies ReLU and sets all coordinates except $0, J$ to zero:

$$[\mathcal{G}_J(v)]_k = \mathbf{1}\{k \in \{0, J\}\}\,\text{ReLU}(v_k), \qquad \widehat{z}(x) = \mathcal{G}_J(v(x)). \tag{13}$$

The mask enforces $\|\widehat{z}(x)\|_0 \le 2$ and supplies the reference pair, making it stronger than a sparsity constraint alone. The partner index J is not an input to any scalar score $(s_k(x))$, g_k, or F. This control tests coefficient values with the correct support supplied; no sparsity penalty is used. The resulting active support $(S(x))$ can be a subset of $\{0, J\}$ if the ReLU returns zero. In the additional-coordinate protocol, the support selector $(\mathcal{G}_+)$ retains every positive candidate value $(v_k(x))$, so $\widehat{z}(x) = \mathcal{G}_+(v(x)) = \text{ReLU}(v(x))$. This protocol allows additional coordinates in the reconstruction and is not matched in sparsity to the supplied-pair protocol.

For each seed and each of these two selection protocols, an affine encoder $(\text{Enc}_{\text{aff}})$ starts with weight matrix $D^\top$ and zero bias and receives 2,000 training steps. The three models then start from this

pretrained affine function. In Equation 12, initialization uses $a_k^{\text{skip}} = 1$, $b_k^{\text{skip}} = 0$, and $v_{kh}^{\text{out}} = 0$. The cross-coordinate encoder ($\text{Enc}_{\text{cross}}$) represents its linear part with paired hidden units, using $x_q = \text{ReLU}(x_q) - \text{ReLU}(-x_q)$, and copies the pretrained bias. Its remaining hidden units initially have zero output weights. Each model then receives 8,000 steps with the same minibatch stream. All training uses Adam, learning rate 10^{-3}, and batch size 1,024.

The two selection protocols use separately pretrained affine functions: training with $\mathcal{G}_+$ does not start from a supplied-pair checkpoint. Joint training continues the supplied-pair checkpoints as described in Appendix C.3. Regression and editing diagnostics keep all model parameters fixed.

Local consistency regularization. The locally regularized cross-coordinate encoder ($\text{Enc}_{\text{cross+LC}}$) copies the initial cross-coordinate encoder ($\text{Enc}_{\text{cross}}$) before the 8,000-step stage: the same $4 \to 99 \to 5$ network with 995 parameters, and the same training minibatches. Its pre-truncation support ($S_{\text{pre}}(x)$) is $\text{supp}(\text{ReLU}(v(x)))$, without the supplied-pair mask. Neither the reference pair $\{0, J\}$ nor its coefficients are used to construct the auxiliary target; the reconstruction control still uses $\mathcal{G}_J$ from Equation 13. For each of the first 128 examples in a minibatch, we sample i uniformly from $S_{\text{pre}}(x)$ and set δ to a random sign times $\text{Uniform}[0.1, 0.3]\,\text{ReLU}(v_i(x))$. We cap $|\delta|$ at $0.02\|x\|_2$. An empty support contributes zero. In Equation 9, both target sampling and the support sum use $S_{\text{pre}}(x)$ in place of $S(x)$. The sum is not divided by support size, and no outside-support penalty is added.

At update t, hats denoting minibatch estimates, the objective is

$$\begin{aligned} \widehat{L}_{\text{train}} &= \widehat{L}_{\text{rec}} + \lambda_{\text{LC}}\omega_t \ \text{sg}(\widehat{L}_{\text{rec}})\widehat{L}_{\text{LC}}, \\ \omega_t &= \min\left\{1, \max\left\{0, \frac{t-250}{250}\right\}\right\}. \end{aligned} \tag{14}$$

The supplied-pair comparison in Appendix C.5 uses $\lambda_{\text{LC}} \in \{0.005, 0.02, 0.05\}$, changing only this weight across the same five seeds and training streams. Here sg stops gradients. Both evaluations of v in Equation 9 remain differentiable; the input, decoder, support, sampled index and edit magnitude are detached in this auxiliary loss. Reconstruction still updates all trainable model parameters, including the partner decoder columns during joint training. Decoder columns in these controls have unit norm. For non-unit columns, the same computation uses unit directions and multiplies candidate values and positive coefficients by the detached column norms; the decoder bias and these norms are also detached in the auxiliary loss. There is no additional sparsity or dead-feature loss in this experiment. Joint training continues from each model's fixed-decoder checkpoint for 4,000 steps, restarts Adam as for the baselines, and continues the local-consistency step counter and random stream without a second warmup. Tables 7 and 8 evaluate the original supplied-pair outputs. The detached reconstruction scale makes the effective regularizer weight proportional to the current L_{rec}. The penalty targets candidate-value changes under the current decoder and support; reference coefficients and support-changing full-code behavior are outside its training target.

C.3 Partner dependence and decoder absorption

Fitted dependence. For each frozen model, ordinary least squares fits the encoded coefficient ($\widehat{z}_0(x)$) against an intercept and the two reference coefficients ($z_0^\star(x), z_J^\star(x)$) on all 65,536 held-out examples, pooling the four values of J:

$$\widehat{z}_0(x) \approx \gamma + \alpha z_0^\star(x) + \beta z_J^\star(x), \qquad c = \frac{\beta}{\alpha} \quad (\alpha \neq 0). \tag{15}$$

Here γ is the intercept, α, β are target and partner slopes, and the partner loading (c) is their partner-to-target ratio. For the affine example with $w_i^\top d_i = 1$ in Section 5, $c = w_i^\top d_j$; for nonlinear models it summarizes fitted dependence.

Reconstruction loss along the absorption path. For fixed partner j, take the nonnegative encoded coefficients $\widehat{z}_0(x) = z_0^\star(x) + cz_j^\star(x)$ and $\widehat{z}_j(x) = z_j^\star(x) > 0$, with all other outputs zero, from Section 5. Fixing these outputs and $\widetilde{d}_0 = d_0$, vary the partner model direction ($\widetilde{d}_j = d_j - \rho d_0$). Then

$$r(x) = (c-\rho)z_j^\star(x)d_0, \qquad L_{\text{rec}} = (c-\rho)^2\mathbb{E}[(z_j^\star(x))^2]\|d_0\|_2^2. \tag{16}$$

For finite positive $\mathbb{E}[(z_j^\star(x))^2]$ and nonzero d_0, the minimum along this fixed-output path is $\rho = c$.

Joint training and column normalization. Joint training continues each fixed-decoder, supplied-pair checkpoint for 4,000 updates with the same optimizer settings. The target model direction $(\widetilde{d}_0)$ remains d_0; each other model direction $(\widetilde{d}_j)$ is its trainable raw vector divided by its ℓ_2 norm. The data distribution and supplied pair remain unchanged. We fit c by Equation 15 and each partner column by least squares:

$$\widetilde{d}_j \approx \lambda_j d_j + \mu_j d_0, \qquad \rho_j = -\frac{\mu_j}{\lambda_j}, \qquad \rho = \frac{1}{4}\sum_{j=1}^{4} \rho_j, \tag{17}$$

For nonzero fitted λ_j, the decoder absorption (ρ_j) measures the subtracted d_0 component relative to the d_j component, accounting for column scale; ρ averages the four partners.

Column normalization does not prevent exact reconstruction. For the two-coordinate example, let $\sigma_j = \|d_j - cd_0\|_2 > 0$. Then

$$\widetilde{d}_j = \frac{d_j - cd_0}{\sigma_j}, \qquad \widehat{z}_0(x) = z_0^\star(x) + cz_j^\star(x), \qquad \widehat{z}_j(x) = \sigma_j z_j^\star(x) \tag{18}$$

give

$$\widehat{z}_0(x)d_0 + \widehat{z}_j(x)\widetilde{d}_j = z_0^\star(x)d_0 + z_j^\star(x)d_j = x. \tag{19}$$

Here $\lambda_j = 1/\sigma_j$, $\mu_j = -c/\sigma_j$, and hence the decoder absorption (ρ_j) equals the partner loading (c). This identity requires coefficient one on $z_0^\star(x)$ and no intercept in $\widehat{z}_0(x)$; a generic fitted intercept or scale need not obey it. Table 4 compares the trained encoders with this prediction.

C.4 Editing with additional coordinates and a fixed decoder

With the model decoder $(\widetilde{D} = D)$ fixed, the reconstruction residual $(r(x) = D\Delta z(x))$ can be small even when the coefficient error $(\Delta z(x))$ is large: errors in different coordinates can cancel after decoding. We test editing at the trained checkpoints with $\mathcal{G}_+$, retaining every positive candidate value $(v_k(x))$ during encoding.

For a frozen encoder Enc, let $\delta \neq 0$ and $\widehat{z}_i(x) + \delta \geq 0$. The requested edit (δe_i) produces the edited activation $(x' = x + \delta\widetilde{d}_i)$. We define the returned edit $(\Delta_i^{\mathrm{ret}}(x,\delta))$ and edit-return error $(\epsilon_i(x,\delta))$ by

$$\Delta_i^{\mathrm{ret}}(x,\delta) = \mathrm{Enc}(x') - \mathrm{Enc}(x), \qquad \epsilon_i(x,\delta) = \frac{\|\Delta_i^{\mathrm{ret}}(x,\delta) - \delta e_i\|_2}{|\delta|}. \tag{20}$$

With $\widetilde{D} = D$, if both inputs belong to the reference family and $z^\star(x') = z^\star(x) + \delta e_i$, then

$$\Delta_i^{\mathrm{ret}}(x,\delta) - \delta e_i = \Delta z(x') - \Delta z(x). \tag{21}$$

Exact recovery at both inputs therefore gives $\Delta_i^{\mathrm{ret}}(x,\delta) = \delta e_i$. If the full coefficient errors $(\Delta z(x), \Delta z(x'))$ each have norm at most η, the triangle inequality gives $\epsilon_i(x,\delta) \leq 2\eta/|\delta|$. Nonzero errors can also cancel in the difference. Low target-only average coefficient MSE (L_{coef}) on the original inputs does not establish either per-input bound.

For the comparison in Table 6, we set $i = 0$ and $\delta = 0.25$, editing the original activation as $x' = x + 0.25d_0$. The norm in Equation 20 covers all five entries. We average $\epsilon_0(x, 0.25)$ over the same 32,645 held-out examples satisfying $z_0^\star(x) \leq 1.5$ for every model. This restriction keeps the edited target coefficient inside $[1, 2]$; the partner coefficient is unchanged. All three architectures have small reconstruction MSE (L_{rec}) but substantial edit-return error $(\epsilon_0(x, 0.25))$.

C.5 Local-consistency weight comparison

Tables 7 and 8 report the fixed-decoder and joint-training checkpoints for the locally regularized cross-coordinate encoder $(\mathrm{Enc}_{\mathrm{cross+LC}})$, using the protocol in Appendix C.2. The unregularized cross-coordinate encoder $(\mathrm{Enc}_{\mathrm{cross}})$ repeats the fixed-decoder result from Table 3 and the joint-stage continuation from Table 4. All entries use the prescribed final update, not a selected best checkpoint.

Table 6: Reconstruction and editing with the fixed model decoder ($\widetilde{D} = D$) under the additional-coordinate protocol, whose support selector ($\mathcal{G}_+$) retains every positive candidate value (five-seed means). These models are trained under that protocol, whereas Table 3 uses the supplied-pair protocol (Appendix C.2). Editing uses $\delta = 0.25$ on the held-out subset with $z_0^\star(x) \le 1.5$ (Appendix C.4).

Model	Reconstruction MSE L_{rec}	Edit-return error $\epsilon_0(x, 0.25)$
Affine ($\mathrm{Enc}_{\mathrm{aff}}$)	6.768×10^{-7}	0.679447
Scalar ($\mathrm{Enc}_{\mathrm{sc}}$)	3.221×10^{-6}	0.600906
Cross ($\mathrm{Enc}_{\mathrm{cross}}$)	6.001×10^{-7}	0.581391

Table 7: Local consistency weight comparison with the model decoder ($\widetilde{D} = D$) fixed (five-seed means). The cross-coordinate encoder ($\mathrm{Enc}_{\mathrm{cross}}$) and locally regularized cross-coordinate encoder ($\mathrm{Enc}_{\mathrm{cross+LC}}$) share the same network, initialization and minibatches; λ_{LC} is the only varied training setting (Appendix C.2).

Model	λ_{LC}	L_{coef}	L_{rec}
Cross ($\mathrm{Enc}_{\mathrm{cross}}$)	0	3.3377×10^{-8}	7.5608×10^{-8}
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.005	2.0452×10^{-8}	3.7676×10^{-8}
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.02	2.6988×10^{-6}	5.6522×10^{-6}
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.05	2.1214×10^{-6}	4.2635×10^{-6}

Table 8: Local consistency weight comparison under joint training (five-seed means). Each Table 7 checkpoint receives 4,000 additional updates with fixed $\widetilde{d}_0 = d_0$ and trainable unit-norm partner columns, retaining its λ_{LC} (Appendix C.2). The partner loading (c) and decoder absorption (ρ) are signed fits.

Model	λ_{LC}	L_{rec}, before	L_{rec}, after	c	ρ
Cross ($\mathrm{Enc}_{\mathrm{cross}}$)	0	7.5608×10^{-8}	1.9408×10^{-7}	0.001496	0.001524
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.005	3.7676×10^{-8}	5.5515×10^{-6}	0.002029	0.001729
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.02	5.6522×10^{-6}	1.5310×10^{-6}	0.001695	0.001855
Cross+LC ($\mathrm{Enc}_{\mathrm{cross+LC}}$)	0.05	4.2635×10^{-6}	1.2121×10^{-5}	0.001586	0.001752

D Architecture Proofs and the Scope of Recovery

The final support selector ($\mathcal{G}$) preserves selected positive candidate values ($v_k(x)$) and sets the other entries to zero.

Fix the reference decoder ($D \in \mathbb{R}^{d \times m}$) and a finite allowed support family ($\mathcal{S}$) of nonempty supports. For each $\Sigma \in \mathcal{S}$, let D_Σ contain the corresponding decoder columns and let $\Omega_\Sigma \subset (0, \infty)^{|\Sigma|}$ be a nonempty open set of reference coefficients. The admissible inputs on that support are $x = D_\Sigma \xi$ for $\xi \in \Omega_\Sigma$. Assume each D_Σ has full column rank and that these representations are unique across the allowed family, so the reference code ($z^\star(x)$) and reference support ($S^\star(x)$) are well defined. Openness permits the local independent coefficient changes used in the proof below. Proposition 1 characterizes the constraints imposed on one fixed scalar score across all these support-specific domains.

D.1 When a scalar nonlinear function can recover a coefficient

An arbitrary function g_i of a fixed scalar score $(s_i(x) = w_i^\top x + b_i)$ cannot recover different target coefficients from equal scores. No regularity of g_i is assumed.

Proposition 1. *Fix a target i that belongs to at least one allowed support. For every $\Sigma \in \mathcal{S}$ containing i, let $e_{i,\Sigma} \in \mathbb{R}^{|\Sigma|}$ denote its local basis vector, and write $\xi_{i,\Sigma} = e_{i,\Sigma}^\top \xi$ for the target entry of $\xi \in \Omega_\Sigma$. If an arbitrary scalar nonlinear function g_i satisfies*

$$g_i(w_i^\top D_\Sigma \xi + b_i) = \xi_{i,\Sigma} \quad \textit{for every } \xi \in \Omega_\Sigma \textit{ and every } \Sigma \ni i,$$

then, with the same nonzero $\alpha_i = w_i^\top d_i$ for every such support,

$$w_i^\top D_\Sigma = \alpha_i e_{i,\Sigma}^\top, \qquad \alpha_i \neq 0. \tag{22}$$

Equivalently, dividing the row by its target response gives $\ell_i = w_i/\alpha_i$, which must satisfy $\ell_i^\top D_\Sigma = e_{i,\Sigma}^\top$ on every support containing i. Conversely, a row satisfying those normalized constraints already recovers the active coefficient by the linear rule $\ell_i^\top x$.

Proof. Fix $\Sigma \ni i$. If $w_i^\top D_\Sigma = 0$, the score is constant while $\xi_{i,\Sigma}$ varies on the open set Ω_Σ, so recovery is impossible. For any $h \in \ker(w_i^\top D_\Sigma)$, choose an interior point ξ and a sufficiently small nonzero t so $\xi + th \in \Omega_\Sigma$. Both coefficient vectors give the same score. Exact recovery therefore requires $e_{i,\Sigma}^\top h = 0$, yielding

$$\ker(w_i^\top D_\Sigma) \subseteq \ker e_{i,\Sigma}^\top.$$

Since both linear functionals are nonzero and their kernels have the same dimension, $w_i^\top D_\Sigma = \alpha_i e_{i,\Sigma}^\top$ for some $\alpha_i \neq 0$. Its target entry is $w_i^\top d_i$, which does not depend on Σ. Dividing by this shared value proves the normalized constraints. The converse follows directly from $\ell_i^\top D_\Sigma \xi = \xi_{i,\Sigma}$. □

The result concerns a positive reference coefficient $(z_i^\star(x))$. The support selector $(\mathcal{G})$ cannot recover it by setting the output to zero. An outer ReLU can be included in g_i. Using the normalized rows ℓ_i and masking to the supplied reference support $(S^\star(x))$ suffices for exact recovery of the reference code $(z^\star(x))$; without it, coefficients outside $S^\star(x)$ must also be set to zero. In particular, this argument does not require the encoder row of every inactive coordinate to satisfy zero-output constraints on every support.

If the shared normalized constraints have no solution for some i, neither an affine function nor a scalar nonlinear function of one fixed scalar score $(s_i(x))$ can recover that coefficient throughout the allowed family. If every ordered pair of coordinates occurs together in a support with an open coefficient domain, applying the constraints to all rows gives $LD = I_m$, for $L = [\ell_1, \ldots, \ell_m]^\top$. When $m > d$, $\operatorname{rank}(LD) \leq d < m$ makes this impossible, as in the case with every ordered pair in Section 3. Overcompleteness alone, without the relevant support constraints, is not the obstruction.

D.2 The controlled family and an exact inverse in the tested architecture

For the data distribution in Section 6, take $\mathcal{S} = \{\{0, J\} : J = 1, \ldots, 4\}$ and $\Omega_{\{0,J\}} = (1, 2)^2$ in the preceding framework. The scalar-score constraint applied to the interior $(1, 2)^2$ of each support $\{0, J\}$ requires $w_0^\top d_J = 0$ for every $J = 1, \ldots, 4$. These four directions are the standard basis, so $w_0 = 0$, contradicting $w_0^\top d_0 \neq 0$. Thus no function of a single scalar score recovers the target throughout this family, even with the support selector $(\mathcal{G}_J)$.

For an activation (x) from this distribution, define

$$H_{\text{sum}}(x) = \sum_{q=1}^{4} x_q, \qquad H_{\text{diff}}(x) = \sum_{1 \leq p < q \leq 4} |x_p - x_q|.$$

Three components equal $z_0^\star(x)/2$; the fourth exceeds them by $z_J^\star(x)$. Hence

$$H_{\text{sum}}(x) = 2z_0^\star(x) + z_J^\star(x), \qquad H_{\text{diff}}(x) = 3z_J^\star(x).$$

It follows that the five outputs

$$F_0(x) = \tfrac{1}{2}H_{\text{sum}}(x) - \tfrac{1}{6}H_{\text{diff}}(x), \qquad F_k(x) = x_k - \tfrac{1}{4}H_{\text{sum}}(x) + \tfrac{1}{12}H_{\text{diff}}(x), \quad k = 1, \ldots, 4, \tag{23}$$

equal the complete reference code $(z^\star(x))$ on this family.

To implement Equation 23 in one hidden layer, use four units $h_q(x) = \text{ReLU}(x_q)$ and twelve units $h_{pq}(x) = \text{ReLU}(x_p - x_q)$, one for each ordered pair $p \neq q$. Since all $x_q > 0$ here,

$$H_{\text{sum}}(x) = \sum_q h_q(x), \qquad H_{\text{diff}}(x) = \sum_{p \neq q} h_{pq}(x).$$

Set the output weight of every h_q to $1/2$ in row zero, and that of every h_{pq} to $-1/6$. In row k, set the weight of h_q to $\mathbf{1}\{q = k\} - 1/4$, and that of every h_{pq} to $1/12$. Set all hidden and output biases to zero. This gives Equation 23 exactly. All five outputs are nonnegative, so the outer ReLU preserves them. No supplied reference support is needed.

The construction uses 16 of the 99 hidden units in the cross-coordinate encoder $(\text{Enc}_{\text{cross}})$ of Table 3; setting the remaining 83 to zero embeds it in that architecture.

E Off-Diagonal Contribution in Trained SAEs

Equation 4 writes an encoded coefficient $(\widehat{z}_i(x))$ as a diagonal term plus a sum of off-diagonal terms over the other active coordinates, and Equation 5 shows that the conditions removing that sum cannot all hold. This appendix measures the size of the analogous sum inside a trained SAE, formed from the model's own encoded coefficients and decoder directions rather than from a reference code.

A trained SAE has no reference decoder (D), so its own reconstruction supplies the decomposition, $x \approx \sum_k \widehat{z}_k(x)\widetilde{d}_k + b_{\text{dec}}$. Applying the encoder row to it gives

$$\widehat{z}_i(x) = \underbrace{\widehat{z}_i(x)\, w_i^\top \widetilde{d}_i}_{\text{diagonal}} + \underbrace{\sum_{j \neq i,\, j \in S(x)} \widehat{z}_j(x)\, w_i^\top \widetilde{d}_j}_{\text{off-diagonal}} + w_i^\top b_{\text{dec}} + b_i - w_i^\top r(x), \tag{24}$$

and we compare the two dictionary-attributable terms through the off-diagonal ratio $(C_i(x))$:

$$C_i(x) = \frac{\left|\sum_{j \neq i,\, j \in S(x)} \widehat{z}_j(x)\, w_i^\top \widetilde{d}_j\right|}{\left|\widehat{z}_i(x)\, w_i^\top \widetilde{d}_i\right|}. \tag{25}$$

The reconstruction residual $(r(x))$ belongs to neither term: it is the part of the activation (x) that the dictionary does not explain, and is therefore attributable to no coordinate. The sum is taken before the absolute value, so cancellation between partners is retained.

We evaluate the twelve Affine SAEs of Section 7 at 30,000 steps on the 8,192-row held-out dev pool, censusing every active pair (x, i) rather than sampling: 4,122,590 pairs in all. The off-diagonal sum reaches a quarter of the coordinate's own contribution for 67.5% of them, half of it for 50.1%, all of it for 28.9%, and twice it for 13.6%.

F Protocols for the Editing Examples

All four panels of Figure 2 use the edit protocol $x' = x + \delta\widetilde{d}_i$. The change is added to the original activation (x). The horizontal axis reports $100\delta/\widehat{z}_i(x)$; each vertical trace reports $100[\widehat{z}_k(x') - \widehat{z}_k(x)]/\widehat{z}_k(x)$, using that coordinate's own positive original encoded coefficient $(\widehat{z}_k(x))$.

F.1 TopK switching in Figure 2(c)

Figure 2(c) uses the WikiText-103 test activation at the token `sides` (token-stream position 12,881) and a GPT-2 MLP-layer-6 SAE checkpoint with 4,096 coordinates, TopK $K = 20$, seed 0, and 30,000 training steps.

Starting from this activation (x), we apply positive edits to coordinate 3810 up to 20% of its initial encoded coefficient $(\widehat{z}_{3810}(x))$ and re-encode after each edit. Coordinate 394 crosses the TopK boundary at relative edit sizes of approximately 1.20%, 9.89%, and 13.73%, producing the active–inactive–active–inactive sequence in the figure. The sweep uses 446 edit values, with denser sampling near the crossings; the crossing locations were separately verified with the float32 encoder.

F.2 Protocols for Figure 2(a–b) and the Target 1324 case

Panel (a) and the Target 1324 case (not shown in Figure 2; its full record is Figure 6) show the target and the two unedited coordinates with the largest absolute changes at the largest saved requested edit; panel (b) shows only the target. The models, inputs, and evaluations are archived; the figure is redrawn from their saved values.

Target 1324. The Target 1324 case uses a saved Qwen2-1.5B layer-14 SAE with input dimension 1,536, 4,096 coordinates, and TopK $K = 20$. The input is WikiText-103 test token-stream row 11,573. With the original encoded coefficient $(\widehat{z}_{1324}(x) = 4.641705)$, the requested edit (δe_{1324}) uses $\delta = 0.9\widehat{z}_{1324}(x) = 4.177535$. The nineteen unedited coordinates all start above half the median positive encoded coefficient $(\widehat{z}_k(x))$.

The saved interval check verifies positive coefficients and strict separation from unselected values before and after the edit, ensuring unchanged active support $(S(x'))$ throughout the interval. The saved float32 checks give minimum selection gaps of 0.027476 before the edit and 0.004965 after the largest edit on the analytic path. The 257 plotted points follow this path; direct float32 re-encoding after that edit differs by at most 1.05×10^{-5} in any encoded coefficient $(\widehat{z}_k(x'))$. The encoder's full edit-return error $(\epsilon_{1324}(x, \delta))$ is 0.170609.

Target 2178. Panel (a) uses a saved GPT-2 SAE with 4,096 coordinates and TopK $K = 20$, trained for 30,000 steps. The input is WikiText-103 test token-stream row 29,751. Here $\delta = \widehat{z}_{2178}(x) = 5.656280$. The original twenty coordinates remain active at all 257 saved encoder evaluations. The full edit-return error $(\epsilon_{2178}(x, \delta))$ is 1.232018.

Target 528. Panel (b) of Figure 2 uses the same archived GPT-2 SAE as panel (c), at WikiText-103 test token-stream row 24,155. Its original encoded coefficient $(\widehat{z}_{528}(x))$ is 0.931453. The requested edit (δe_{528}) uses $\delta = 0.2\widehat{z}_{528}(x) = 0.186291$.

The target decreases at every step of the saved 257-point encoder sweep. The original twenty coordinates remain in the active support $(S(x'))$ at every sampled edit. The full edit-return error $(\epsilon_{528}(x, \delta))$ is 1.461623.

F.3 All twenty originally active coordinates

For each example, the twenty coordinates in Figures 6–8 are exactly the original active support selected by its TopK $K = 20$ encoder on x. The same coordinate identities are retained for display after re-encoding; in all three saved sweeps, they remain active at every sampled edit. "Before" and "after" denote the encoded coefficients before the edit and after re-encoding. The bars show the actual percentage change for every coordinate; diamonds mark the requested change, which is zero for every non-target coordinate.

F.4 Ablation read against the reference dictionary

Figure 1 uses the controlled construction of Section 6 and Appendix C.1: inputs $x = z_0^\star(x)d_0 + z_J^\star(x)d_J$ with $z_0^\star(x), z_J^\star(x) \in [1, 2]$, the fixed reference decoder (D), and the fixed-decoder checkpoints of Table 3 (seeds $0, \ldots, 4$). For each of 65,536 held-out inputs we apply the ablation $x' = x - \widehat{z}_0(x)\, d_0$, where encoded coefficient $(\widehat{z}_0(x))$ is the encoder's own reading of coordinate 0 (ReLU of the candidate value). The edit moves x along d_0 only, so x' stays in the span of d_0 and d_J; these two directions are linearly independent, so x' has a unique decomposition $x' = a\, d_0 + b\, d_J$ on them: three components of x' equal $a/2$ and the remaining component equals $a/2 + b$. We verify $b = z_J^\star(x)$ to within 10^{-4}, so the edit leaves the partner unchanged, and report $a = z_0^\star(x')$, the true feature-0 coefficient after the edit. No SAE encoder is used in this readout.

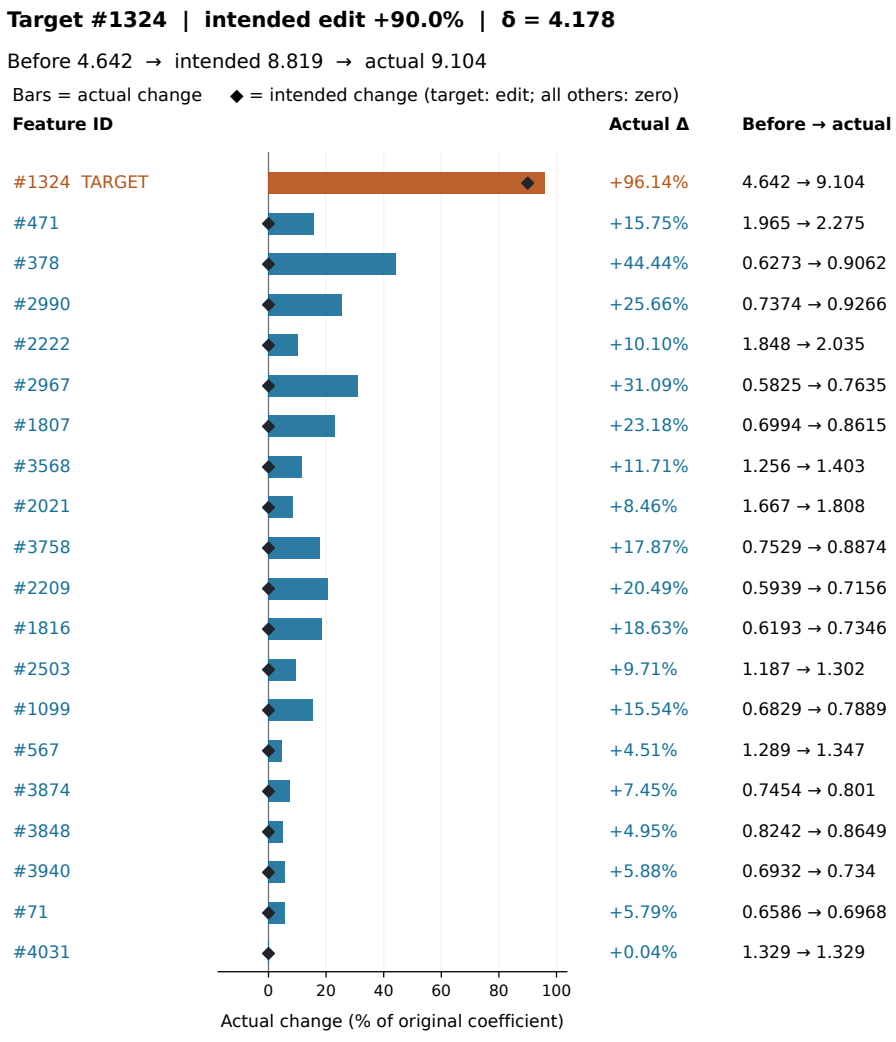


Figure 6: Complete original-support response for the Target 1324 case, using TopK with $K = 20$.

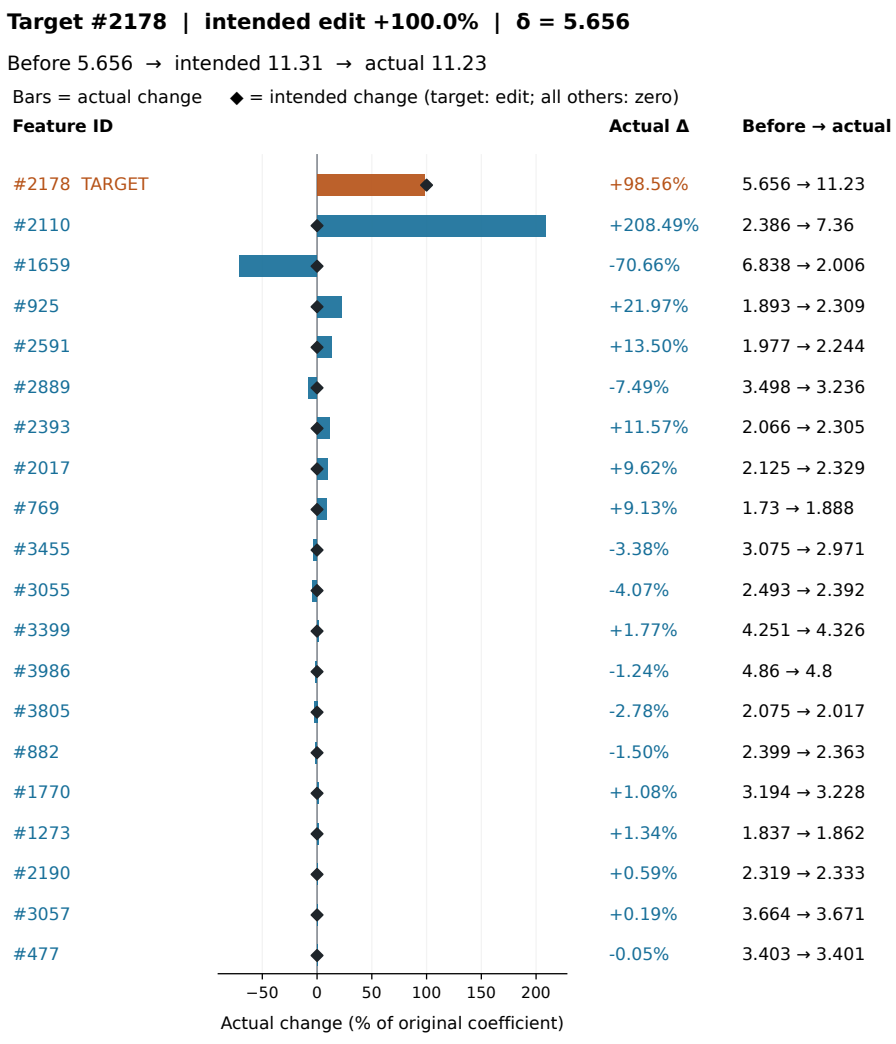


Figure 7: Complete original-support response for Figure 2(a), using TopK with $K = 20$.

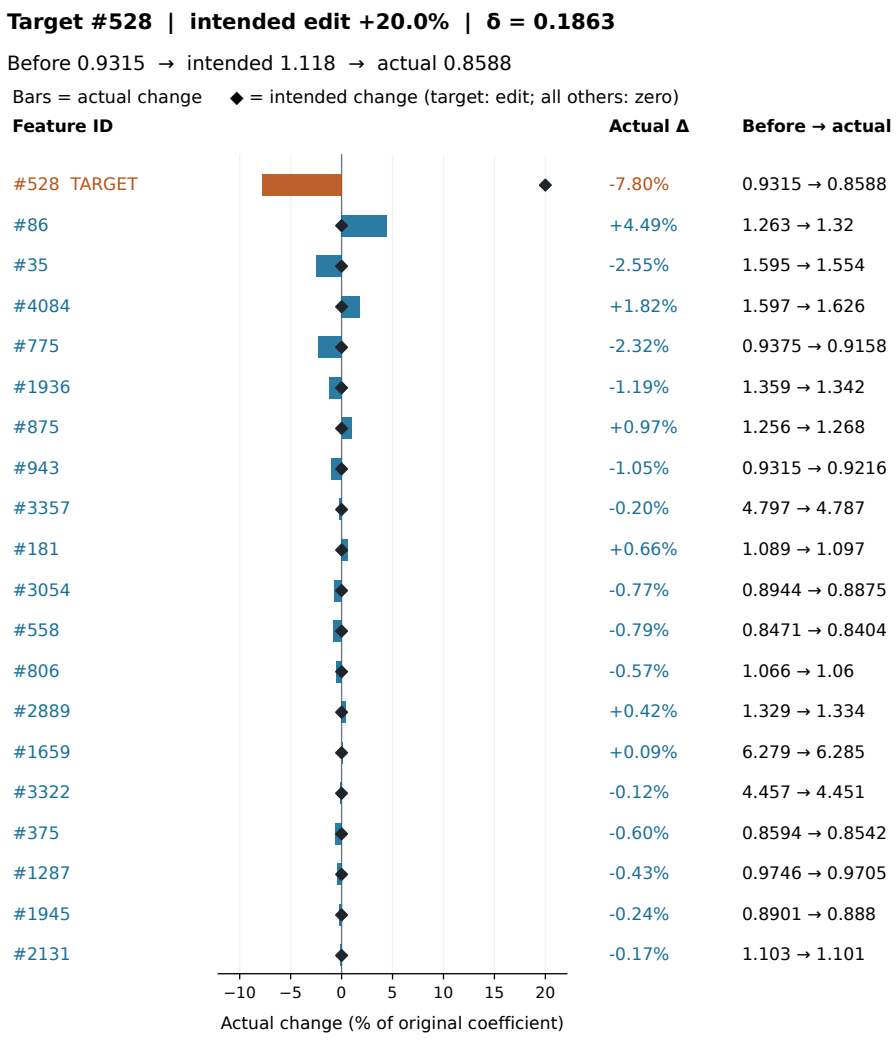


Figure 8: Complete original-support response for Figure 2(b), using TopK with $K = 20$.

Regressing $\widehat{z}_0(x)$ on $z_0^\star(x)$ and $z_J^\star(x)$ gives the fits in Table 9; the ratio of the partner slope to the target slope reproduces the partner loading (c) of Table 4. The remaining coefficient after the edit is $z_0^\star(x') = z_0^\star(x) - \widehat{z}_0(x)$. Its mean over inputs is close to zero for every encoder, because each encoder is calibrated on average, but for the coordinatewise-score encoders it ranges over about ± 0.3 on inputs whose original coefficient lies in $[1, 2]$: negative when the partner coefficient is large and positive when it is small. The cross-coordinate encoder removes the feature exactly (Figure 9). The direction subtracted is the reference direction in every case, so the discrepancy comes only from the encoder's reading.

Table 9: Ablation $x' = x - \widehat{z}_0(x)\, d_0$ read against the reference dictionary; means over seeds $0, \ldots, 4$. The last column counts inputs whose remaining true coefficient $|z_0^\star(x')|$ exceeds 10% of $z_0^\star(x)$.

Encoder	Fit of $\widehat{z}_0(x)$	$z_0^\star(x')$	Slope vs. $z_J^\star$	$> 10\%$
Affine	$-0.299 + 0.800\, z_0^\star + 0.400\, z_J^\star$	$0.299 + 0.200\, z_0^\star - 0.400\, z_J^\star$	-0.400	29.1%
Scalar	$-0.284 + 0.806\, z_0^\star + 0.383\, z_J^\star$	$0.284 + 0.194\, z_0^\star - 0.383\, z_J^\star$	-0.384	28.8%
Cross	$1.000\, z_0^\star$	0	0.000	0.0%

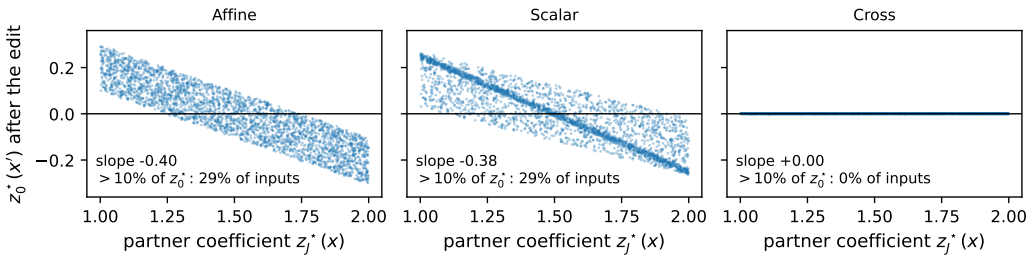


Figure 9: True feature-0 coefficient after the ablation $x' = x - \widehat{z}_0(x)\, d_0$, against the partner coefficient, for the three encoders (seed 0; 3,000 of the 65,536 inputs shown). The left panel is Figure 1.

G Decoder Absorption during Normal SAE Training

Figure 10 traces a case consistent with Section 5 during normal SAE training: coefficient $\widehat{z}_i$ decreases, and feature j's decoder learns to supply part of its decoded contribution, with similar effects on GPT-2 logits. We follow features $i = 2754$ and $j = 831$ in a GPT-2 SAE with an affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$) and TopK50. Superscripts (0) and (t) denote the learned 3k model and a later checkpoint. The reconstruction correction ($q_i(x)$) and gradient component (G_j) are defined below.

Training and observations. We train a 4,096-feature SAE with an affine encoder ($\mathrm{Enc}_{\mathrm{aff}}$) and TopK50 on frozen GPT-2 activations at `blocks.6.hook_mlp_out`. Training uses OpenWebText, batch size 4,096, Adam with initial learning rate 5×10^{-4}, decay over the final 6,000 of 30,000 steps, and auxiliary reconstruction loss for dead features: those with no positive output for more than 100 batches. The auxiliary loss reconstructs the residual using up to 384 dead features, with weight 0.1 reduced proportionally when fewer are available. Encoder, decoder, and biases are trained jointly. We save weights every 25 steps and evaluate the same 44 WikiText development documents every 100 steps. Repeating seed 42 reproduces bitwise-identical parameters at 3k, 10k and 30k: the same trajectory, not an independent-seed replication.

The training loss uses activations multiplied by 1.3213387940365042. Reported coefficients, scalar scores, decoder bias, residuals, and reconstruction MSE (L_{rec}) use the original GPT-2 activation units. Model directions ($\widetilde{d}_k$) have unit norm; w_i, b_i denote the affine row in this same coefficient convention. All 31 inputs activating both features at 3k, excluding token position zero, form the observation set; 30 have zero encoded coefficient ($\widehat{z}_i^{(t)}(x)$) at 10k.

A–B: Coefficient decrease and scalar-score contributions. Each checkpoint is evaluated on the same *making* activation in "*profit-making enterprise*". The encoded coefficient ($\widehat{z}_i(x)$) falls from 2.624 at 3k to zero at 3.9k, while $\widehat{z}_j(x)$ remains positive. At 4k the scalar score ($s_i(x) = 1.221$) is below the TopK cutoff (the 50th-largest score) 1.370; $\widehat{z}_j(x) = 4.349$.

Expand the same input using the learned decoder at 3k: $x = \widetilde{D}^{(0)}\widehat{z}^{(0)} + b_{\mathrm{dec}}^{(0)} - r^{(0)}$. Feature k then contributes $\widehat{z}_k^{(0)}(w_i^{(t)} - w_i^{(0)})^\top \widetilde{d}_k^{(0)}$ to the change in scalar score ($s_i(x)$); residual and bias terms complete the decomposition. At 4k the term from j is -0.851, versus -0.058 for the next most negative feature other than i; at 10k these values are -1.064 and -0.104.

C: Increasing coefficient $\widehat{z}_i$ can reduce squared reconstruction error. At each saved model, hold its decoder and other coefficients fixed. For the reconstruction residual ($r(x)$), define the signed reconstruction correction ($q_i(x)$) and positive reconstruction correction ($q_i^+(x)$) by

$$\begin{aligned} q_i(x) &= \operatorname*{arg\,min}_{q \geq -\widehat{z}_i(x)} \|r(x) + q\widetilde{d}_i\|_2^2 = \max\{-\widehat{z}_i(x), -r(x)^\top \widetilde{d}_i\}, \\ q_i^+(x) &= \max\{q_i(x), 0\}. \end{aligned} \tag{26}$$

Figure 10(C) plots $\widehat{z}_i(x) + q_i^+(x)$ alongside the actual outputs. This may add a 51st active coordinate. On *profit-making* at 4k, $q_i^+ = 1.671$ reduces squared reconstruction error by 2.792. Keeping $K = 50$ and feature j, the best single replacement instead sets coordinate 4042 to zero and increases i: squared reconstruction error falls by 3.450, comprising 0.452 from removal and 2.998 from subsequently increasing $\widehat{z}_i$. As a direction control, the optimal positive addition along fixed $\widetilde{d}_i^{(0)}$ falls from 1.436 at 4k to 0.146 at 5k; this adds a contribution along the 3k direction, rather than changing a current coefficient. These diagnostics quantify the reconstruction-optimal adjustment under the learned decoder.

D: Reconstruction correction correlates with the gradient component G_j. Figure 10(D) uses 80 actual training batches of 4,096 inputs, sampled every 25 steps from 3k to 4975. For each batch, $\mathbb{E}_{\mathrm{batch}}[\widehat{z}_j(x)q_i(x)]$ weights the reconstruction correction ($q_i(x)$) by coefficient $\widehat{z}_j(x)$. We compare it with the gradient component (G_j), which measures the training gradient's push on $\widetilde{d}_j$ toward $\widetilde{d}_i$:

$$G_j = -\left(\nabla_{\widetilde{d}_j^{\mathrm{raw}}} L_{\mathrm{total}}\right)^\top \frac{(I - \widetilde{d}_j\widetilde{d}_j^\top)\widetilde{d}_i}{\|\widetilde{d}_j^{\mathrm{raw}}\|_2}, \tag{27}$$

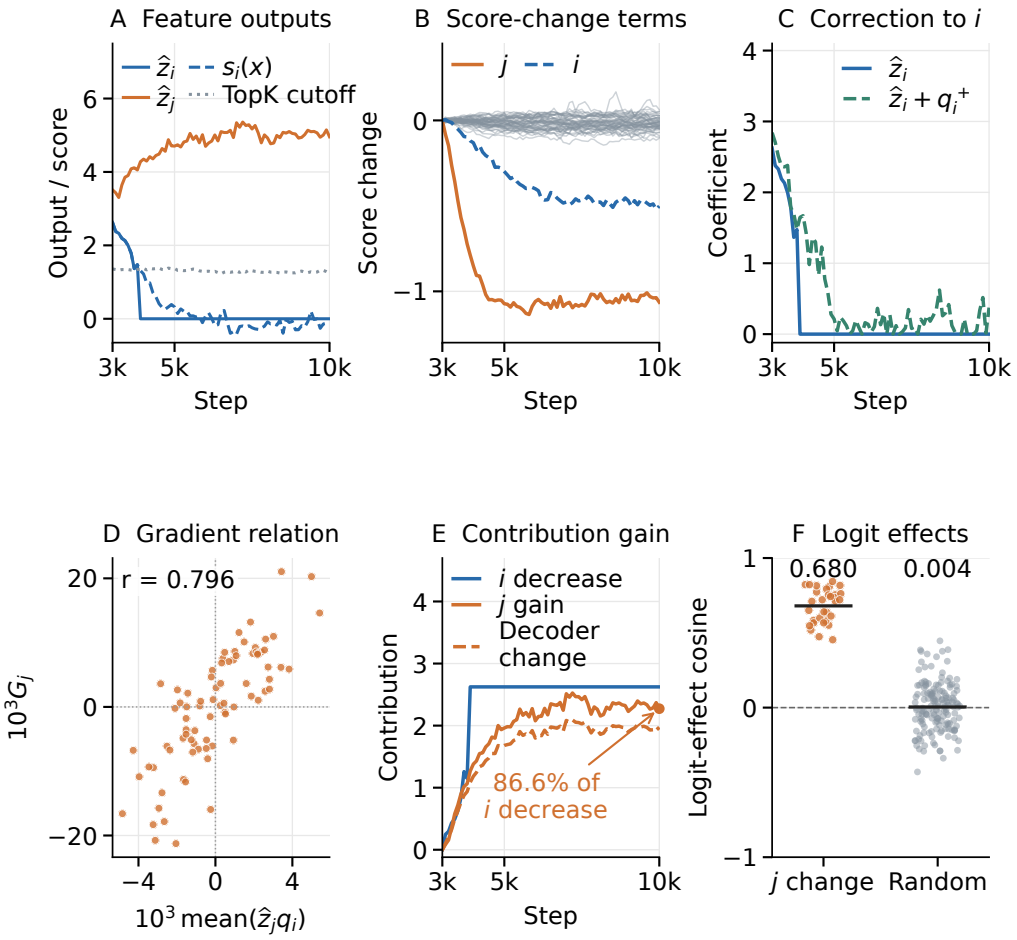


Figure 10: Absorption during normal SAE training. Panels A–C and E follow the same WikiText token *making* in "*profit-making enterprise*". Superscripts (0) and (t) denote 3k and a later checkpoint. **(A)** Encoded coefficients $(\hat{z}_i, \hat{z}_j)$, scalar score $(s_i(x))$, and TopK cutoff (the 50th-largest score). **(B)** Contributions to the change in $s_i(x)$, using the fixed 3k model decoder $(\widetilde{D}^{(0)})$; gray shows each of the 48 other initially active features separately. **(C)** Actual $\hat{z}_i$ versus $\hat{z}_i + q_i^+$, after increasing coefficient $\hat{z}_i$ to minimize squared reconstruction error; this may add one active coordinate. **(D)** Each point is one of 80 actual training batches. The horizontal axis is $\mathbb{E}_{\text{batch}}[\hat{z}_j q_i]$: the reconstruction correction (q_i) weighted by coefficient $\hat{z}_j$. The vertical axis is the gradient component (G_j), measuring the training gradient's push on $\widetilde{d}_j$ toward $\widetilde{d}_i$. Both axes have their 250-step window means removed; their Pearson correlation is 0.796. Zero on either axis denotes its window mean. **(E)** Coefficient decrease $\hat{z}_i^{(0)} - \hat{z}_i^{(t)}$ and feature j's contribution gain along $\widetilde{d}_i^{(0)}$; dashed orange isolates decoder change. At 10k, this gain is 86.6% of the coefficient decrease (orange solid / blue). **(F)** Full-vocabulary logit-effect cosines after separately removing feature j's decoder-change contribution at 10k and feature i's contribution at 3k. The same 31 inputs activated both features at 3k; five random directions per input match the decoder-change contribution's norm; black bars mark means. Detailed definitions and controls follow below.

where $\widetilde{d}_j^{\mathrm{raw}}$ is the unnormalized decoder parameter, $\widetilde{d}_j = \widetilde{d}_j^{\mathrm{raw}}/\|\widetilde{d}_j^{\mathrm{raw}}\|_2$, and L_{total} is the training loss: reconstruction error plus the auxiliary reconstruction loss for dead features, evaluated on the scaled training activations with the weighting given above. This measures alignment change toward the current direction $\widetilde{d}_i$ under the negative gradient, holding $\widetilde{d}_i$ fixed; the corresponding Adam updates are recorded separately below. Positive G_j means that the gradient pushes $\widetilde{d}_j$ toward $\widetilde{d}_i$. Before computing Pearson correlation, we subtract each variable's mean within eight 250-step windows, each containing ten samples. The two quantities share the reconstruction residual.

Controls are 64 uniformly sampled features active on at least 20 of 8,192 validation inputs at 3k, excluding i, j (seed 20260912). Repeating the same correlation with each control in place of j gives median, 75th percentile, 90th percentile, and maximum of 0.687, 0.731, 0.779, and 0.844; Feature 831's correlation exceeds 60 of these 64 values. Another 32 controls selected by nearest initial activation frequency and positive-coefficient mean give median 0.665.

E: Feature j's decoded contribution increases along $\widetilde{d}_i^{(0)}$. Along the fixed direction $\widetilde{d}_i^{(0)}$, feature j's total gain splits exactly into decoder and coefficient changes:

$$\begin{aligned}&(\widehat{z}_j^{(t)}\widetilde{d}_j^{(t)} - \widehat{z}_j^{(0)}\widetilde{d}_j^{(0)})^\top \widetilde{d}_i^{(0)} \\ &\quad = \widehat{z}_j^{(t)}(\widetilde{d}_j^{(t)} - \widetilde{d}_j^{(0)})^\top \widetilde{d}_i^{(0)} + (\widehat{z}_j^{(t)} - \widehat{z}_j^{(0)})(\widetilde{d}_j^{(0)})^\top \widetilde{d}_i^{(0)}.\end{aligned} \tag{28}$$

For *profit-making* at 10k, coefficient $\widehat{z}_i$ has decreased by 2.624; feature j's gain is 2.273 (86.6%), split into 1.933 from its decoder change (73.7%) and 0.340 from its coefficient change (13.0%). These percentages compare scalar projections along $\widetilde{d}_i^{(0)}$. Across all 31 inputs, the decoder-change term alone supplies 74.6% of the mean decrease in $\widehat{z}_i$:

$$\frac{\mathbb{E}\left[\widehat{z}_j^{(t)}(x)(\widetilde{d}_j^{(t)} - \widetilde{d}_j^{(0)})^\top \widetilde{d}_i^{(0)}\right]}{\mathbb{E}[\widehat{z}_i^{(0)}(x) - \widehat{z}_i^{(t)}(x)]} = \frac{1.415}{1.896} \simeq 0.746 \qquad (t = 10\mathrm{k}). \tag{29}$$

On these inputs, reconstruction MSE (L_{rec}) falls from 186.14 to 177.39; other parameter changes also contribute.

F: Feature j's decoder change produces a similar logit effect. From the same original GPT-2 activation (x), separately subtract

$$\widehat{z}_i^{(0)}(x)\widetilde{d}_i^{(0)} \qquad \text{or} \qquad \widehat{z}_j^{(t)}(x)(\widetilde{d}_j^{(t)} - \widetilde{d}_j^{(0)}), \qquad t = 10\mathrm{k}. \tag{30}$$

The second removal uses the complete vector from feature j's decoder change. Run the remaining frozen GPT-2 layers without SAE re-encoding, subtract modified logits from unmodified logits, and subtract the vocabulary mean from each logit-effect vector before computing cosine. Five isotropic random controls per input match the second removal's norm; feature removals keep their own magnitudes. Across the 31 inputs, logit-effect cosine averages 0.680 versus 0.004 for controls; the decoder-change and 3k contributions have mean root-mean-square logit effects of 0.026 and 0.031. For *profit-making*, both removals lower the logits of *company* and *enterprise* relative to the vocabulary mean.

The complete update record. Re-evaluating the 3k and current models on the same actual training activation (x) gives the exact residual decomposition

$$\begin{aligned}r^{(t)} = r^{(0)} &+ \widetilde{d}_i^{(0)}(\widehat{z}_i^{(t)} - \widehat{z}_i^{(0)}) + \sum_{k \neq i} \widetilde{d}_k^{(0)}(\widehat{z}_k^{(t)} - \widehat{z}_k^{(0)}) \\ &+ (\widetilde{D}^{(t)} - \widetilde{D}^{(0)})\widehat{z}^{(t)} + (b_{\mathrm{dec}}^{(t)} - b_{\mathrm{dec}}^{(0)}).\end{aligned} \tag{31}$$

Define the decoder cosine and its change under each realized Adam update by

$$\kappa_j^{(t)} = (\widetilde{d}_j^{(t)})^\top \widetilde{d}_i^{(0)}, \qquad \Delta_{\mathrm{Adam}}\kappa_j^{(t)} = \kappa_j^{(t+1)} - \kappa_j^{(t)}. \tag{32}$$

Because both model directions have unit norm, $\kappa_j^{(t)}$ is their cosine and $\Delta_{\mathrm{Adam}}\kappa_j^{(t)}$ is the projection of the realized normalized-decoder change onto the fixed 3k target direction. Each residual term gives

a corresponding reconstruction-gradient component. Figure 11(A) measures their negative-gradient alignment changes as in Equation 27, with $\widetilde{d}_i^{(0)}$ as the fixed direction and the training loss's input scale. Positive values increase κ_j; total gradients and actual Adam updates are recorded separately.

The complete 3k–5k replay matches every 25-step weight checkpoint and the optimizer, random state and data buffers at 4k and 5k. The gradient component from the change in $\widehat{z}_i$ is positive in 1,993 of 2,000 updates, the total negative gradient increases alignment in 1,257, and the actual Adam update has $\Delta_{\text{Adam}}\kappa_j^{(t)} > 0$ in 1,722, accumulating $+0.3573$ in κ_j. On the actual step-3500 inputs *working relationship* and *fishing queries*, $\widehat{z}_i$ falls from 1.658 and 1.099 at 3k to zero while $\widehat{z}_j > 0$. These OpenWebText training inputs differ from the WikiText figure examples.

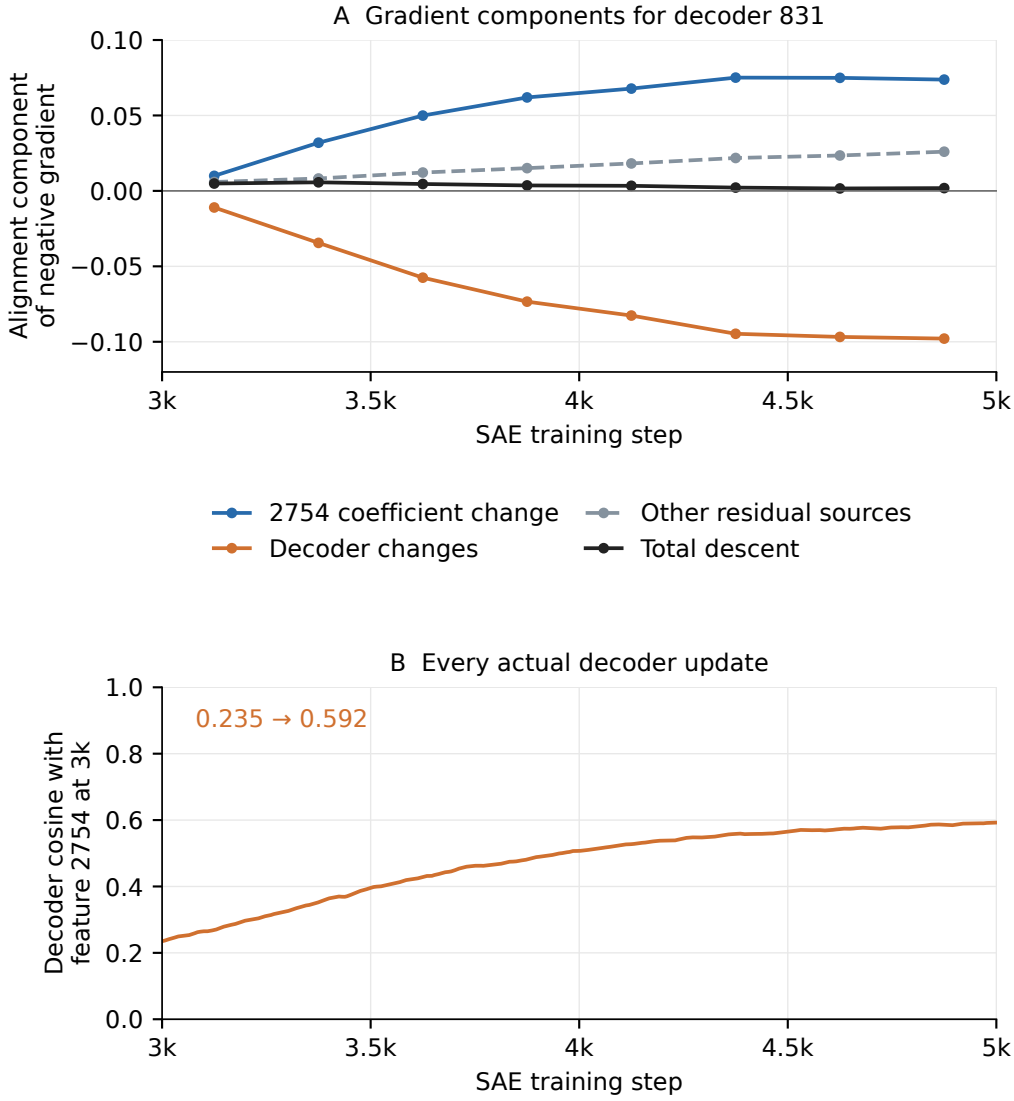


Figure 11: The complete 3k–5k formation window. **(A)** Each point averages 250 consecutive updates, covering all 2,000. Blue is the term from the change in $\widehat{z}_i$ in Equation 31; orange is the decoder-change term; gray combines the remaining components; black is the total training-loss negative gradient. The axis measures the alignment change under the negative gradient, toward $\widetilde{d}_i^{(0)}$. **(B)** The decoder cosine $\kappa_j^{(t)}$ after every Adam update, including all 278 updates with $\Delta_{\text{Adam}}\kappa_j^{(t)} < 0$.

Specificity, later training, and interpretation. The feature does not simply become inactive everywhere: $\widehat{z}_i$ remains positive on *was being built* throughout 3k–10k. The decoder cosine $\kappa_j^{(t)}$ increases from 0.235 at 3k to 0.625 at 10k. Figure 12 shows observations through 30k. The number of zero outputs $\widehat{z}_i$ among the 31 inputs ranges from 28–30 after 10k and ends at 29. At 30k, the input containing the phrase *two going each way* is one of the two reactivated cases, with $\widehat{z}_i = 0.896$. Coefficient $\widehat{z}_i$ stays zero on *profit-making* and positive on *being*. Feature j's decoder-change contribution has a similar logit effect to feature i's 3k contribution; the comparison is at the level of intervention effects.

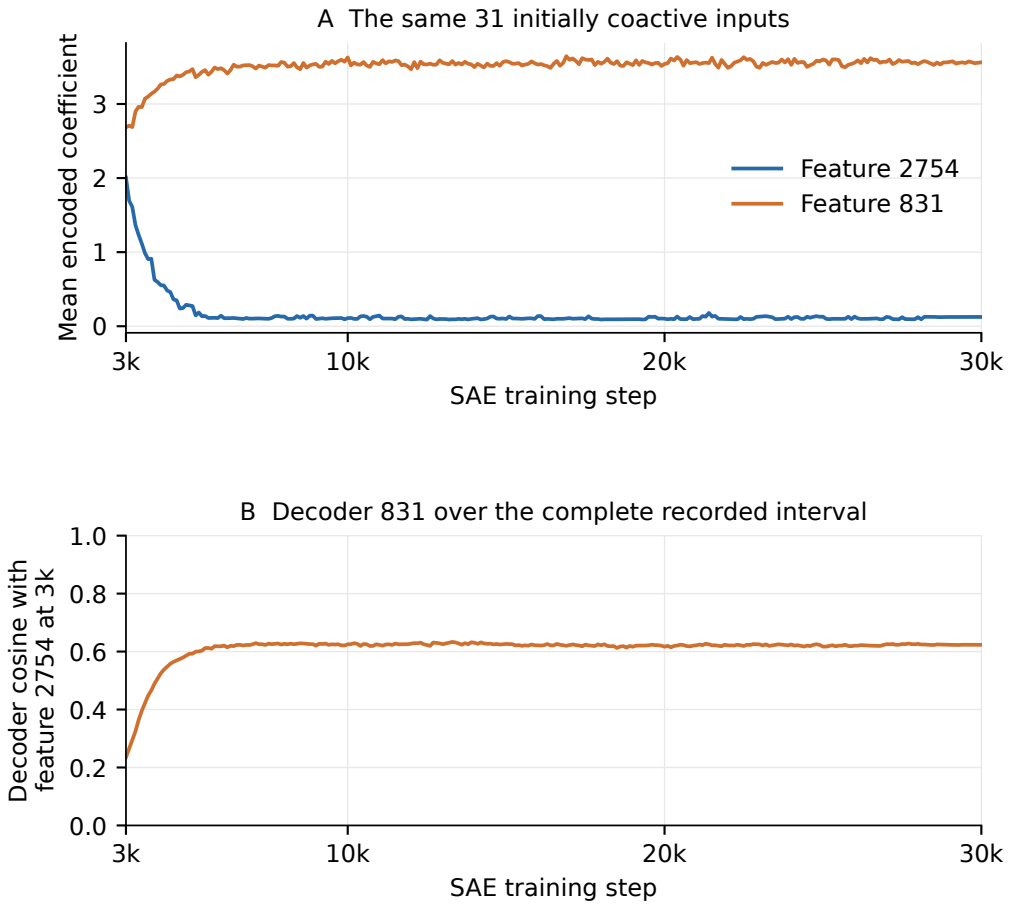


Figure 12: Continuation through step 30k. **(A)** Mean encoded coefficients $(\widehat{z}_i(x), \widehat{z}_j(x))$ over the same 31 initially coactive validation inputs. **(B)** Decoder cosine $\kappa_j^{(t)}$ with the fixed 3k target direction. All 100-step observations are retained, including later reactivation; the final cosine is 0.623.

H TRAINED-SAE PROTOCOLS AND COMPLETE COMPARISONS

H.1 TRAINING AND EVALUATION CONFIGURATION

This subsection records the common trained-SAE setup used in Section 7. We train independent SAEs on 768-dimensional MLP activations from GPT-2 and OPT-125M, each with 4,096 SAE features. TopK uses $K \in \{50, 100\}$; ReLU and Gated use sparsity-penalty weights $\lambda_{\text{sp}} \in \{2, 4\}$, giving twelve matched settings. The comparison contains Affine, Scalar, Cross, and Cross+LC at local-consistency weights $0.005, 0.02, 0.05, 0.1$. The grid is trained twice, at seeds 42 and 43, for 168 trained SAEs in total; the second seed repeats the recipe with the initialization and data-order seed changed and nothing else.

The real-model encoder architectures are summarized below. Cross+LC has exactly the same encoder architecture as Cross and differs only in its training objective. For ReLU and TopK, the listed map produces $v(x)$. For Gated, it produces the shared preactivation $p(x)$ used by the gate and magnitude heads below.

Encoder	Map before the selector-specific heads
Affine	One $768 \to 4096$ affine map.
Scalar	One affine score per feature, followed by an independent $1 \to 31 \to 30 \to 1$ ReLU MLP for each feature.
Cross / Cross+LC	A shared $768 \to 1536 \to 4096$ MLP, with ReLU and LayerNorm after the hidden layer.

ReLU outputs $\widehat{z} = \text{ReLU}(v)$. TopK keeps the K largest candidate values and then applies ReLU. Gated follows Rajamanoharan et al. (2024a): if $p_i(x)$ is the shared preactivation, its binary gate and magnitude path are

$$\chi_i(x) = \mathbf{1}\{p_i(x) + b_i^{\text{gate}} > 0\}, \qquad v_i(x) = e^{r_i^{\text{mag}}} p_i(x) + b_i^{\text{mag}}, \qquad \widehat{z}_i(x) = \chi_i(x)\,\text{ReLU}(v_i(x)).$$

Here $\chi_i(x)$ is the binary gate, r_i^{mag} is a learned log-scale, and $b_i^{\text{gate}}, b_i^{\text{mag}}$ are biases. The magnitude path produces the signed candidate value $(v_i(x))$ and its nonnegative candidate value $(a_i(x) = \text{ReLU}(v_i(x)))$ before the binary gate. For Affine, the magnitude-head scale and bias can be absorbed into the effective affine row w_i, b_i. The sparsity penalty, weighted by λ_{sp}, is computed in the stored parameterization before decoder-column normalization: ReLU sums encoded coefficients times their decoder-column norms; Gated instead weights the rectified gate preactivations by those norms. TopK models use an auxiliary reconstruction loss for features with no positive output in the preceding 100 batches; these are the dead features under the training-time criterion.

Training uses OpenWebText, batches of 4,096, Adam, and 30,000 optimizer steps. All reported evaluation samples come from held-out WikiText-103 development text unseen during training. The local-consistency objective is defined in Section 6. Cross+LC adds the reconstruction-scaled term and warmup in Equation 14 to the selector's training objective. On each of the first 128 minibatch inputs, it samples i uniformly from the original active support $(S(x))$ and uses a random sign times $\text{Uniform}[0.1, 0.3]\widehat{z}_i(x)$ for δ, capped at $0.02\|x\|_2$. The loss reads signed candidate values (v_k) on this fixed support; an empty support contributes zero. Decoder directions are normalized, with coefficients and candidate values rescaled by the corresponding column norms to preserve reconstruction. The remaining subsections define the reported metrics and comparisons.

H.2 EDIT-CONSISTENCY METRICS, SAMPLING, AND AGGREGATION

For each of the 2,048 fixed evaluation input vectors, all positive encoded coordinates are eligible edit targets. If more than 32 are active, 32 are sampled uniformly without replacement using a fixed generator; otherwise all are used. The same sampled events are reused across all sixteen levels. Different SAEs need not select the same feature IDs or support. This is a matched-input comparison of each SAE's own feature edits; feature identities remain model-specific.

Let $v' = v(x + \delta\widetilde{d}_i)$, $v = v(x)$, $a' = \text{ReLU}(v')$, $a = \text{ReLU}(v)$, and $S = S(x)$. For Gated, v is the magnitude-path candidate defined in Appendix H.1, so a is its nonnegative value before the binary gate. Let $[\cdot]_S$ retain only coordinates in the support before the edit. A readout is fixed by

two independent choices: which coordinates it measures (the support S before the edit, or all m), and at which stage it reads them (the nonnegative candidate values a before the final selector, or the actual encoded code after selection). Crossing the two gives four metrics, using the returned edit ($\Delta_i^{\text{ret}} = \text{Enc}(x + \delta\widetilde{d}_i) - \text{Enc}(x)$),

$$\begin{aligned} R_{S,+} &= \|[a' - a - \delta e_i]_S\|_2/|\delta|, \quad R_{m,+} = \|a' - a - \delta e_i\|_2/|\delta|, \\ R_{S,\text{final}} &= \|[\Delta_i^{\text{ret}} - \delta e_i]_S\|_2/|\delta|, \qquad R_{\text{full}} \equiv \epsilon_i(x,\delta) = \|\Delta_i^{\text{ret}} - \delta e_i\|_2/|\delta|. \end{aligned} \tag{33}$$

The two support-restricted metrics are reported per coordinate, as $R_{\text{RMS}} = R_{S,+}/\sqrt{|S|}$ and $R_{\text{RMS,final}} = R_{S,\text{final}}/\sqrt{|S|}$, including the edited target and dividing by the full original support; $|S|$ varies across inputs and across models, so this normalization is what makes those two comparable. The all-m metrics need no counterpart, since their coordinate count is the fixed dictionary width and dividing by it would rescale every model identically. R_{RMS} is the main-text metric. The restriction to S always uses the support before the edit, even if re-encoding changes which coordinates the final selector retains. The two final-code metrics are the only ones that reflect a changed support; the two candidate metrics read values before the selector, so an all-m candidate metric also includes coordinates whose values need not be trained to act as coefficients.

Every reported edit-consistency metric averages valid edits equally within an input, then inputs equally, then the sixteen levels equally. RMS normalization is applied before these averages, not to an already averaged error. Empty-support inputs have no edit event.

Relative-improvement views. The main text shows raw RMS with both baselines. Figures 13 and 14 express the same results relative to the stronger baseline, retaining all settings and all sixteen edit levels.

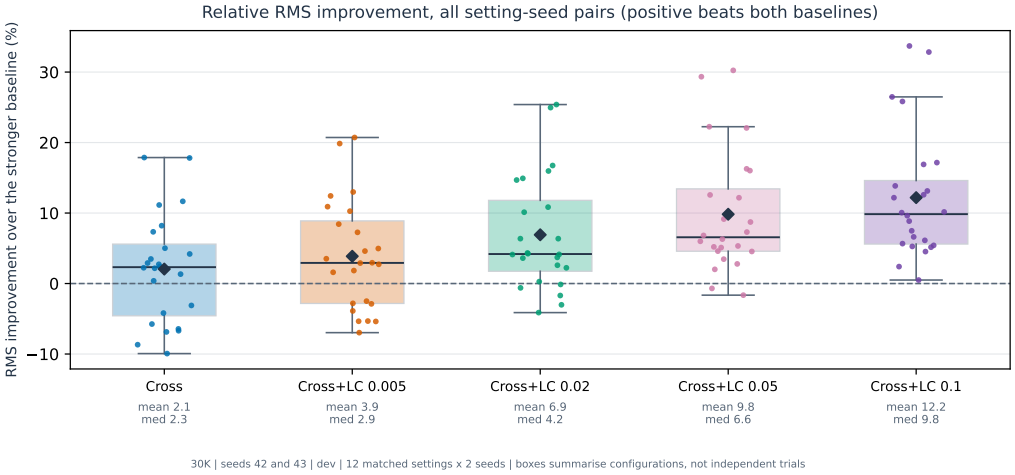


Figure 13: Relative improvement in original-support RMS across encoder variants. Each point is one of the twenty-four setting–seed pairs. The vertical axis is $100[1 - R_{\text{variant}}/\min(R_{\text{Affine}}, R_{\text{Scalar}})]$, where each R is a sixteen-level mean RMS and "variant" denotes the Cross or Cross+LC model shown; positive values beat both baselines. Boxes show quartiles, whiskers the full range, diamonds the mean, and horizontal box lines the median; the numbers below each box give mean and median.

All four scope–stage readouts. Table 10 reports mean errors for all four scope–stage readouts. The ordering of the methods is the same in all four columns, and Cross+LC at $\lambda_{\text{LC}} = 0.1$ attains the lowest error in each. For every setting–seed pair, Figure 15 plots the EV change relative to the higher-EV baseline against the percentage RMS improvement over the lower-RMS baseline.

H.3 Joint RMS and EV outcomes

Figure 16 reports, for the twenty-four matched setting–seed pairs of Section 7.3, the joint RMS and EV outcome counts and the complete EV distributions.

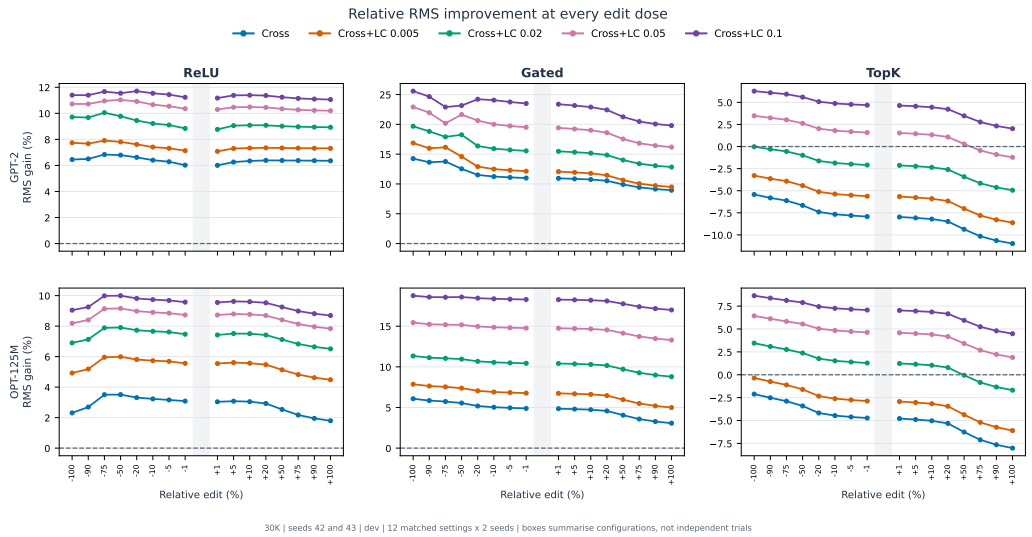


Figure 14: Relative RMS improvement across all sixteen edit levels. At each level we compute relative improvement over the lower-RMS baseline separately for each setting and seed, then average the four resulting values in that panel. Horizontal positions are categorical: adjacent labels need not be equally spaced in edit magnitude.

Table 10: Edit-consistency error under all four scope–stage readouts, averaged over the twelve matched settings after averaging the two seeds within each setting. Lower is better; the best value in each column is bold. The two support-restricted columns are divided by $\sqrt{|S(x)|}$ (Appendix H.2). All values use the sixteen-level mean.

Encoder	support S		all m	
	R_{RMS} (nonnegative a)	$R_{\mathrm{RMS,final}}$ (code $\widehat{z}$)	$R_{m,+}$ (nonnegative a)	R_{full} (code $\widehat{z}$)
Affine	0.0456	0.1417	0.585	1.642
Scalar	0.0475	0.1471	0.749	1.730
Cross	0.0430	0.1363	0.579	1.616
Cross+LC $\lambda_{\mathrm{LC}} = 0.005$	0.0421	0.1353	0.578	1.614
Cross+LC $\lambda_{\mathrm{LC}} = 0.02$	0.0407	0.1338	0.577	1.603
Cross+LC $\lambda_{\mathrm{LC}} = 0.05$	0.0394	0.1328	0.573	1.600
Cross+LC $\lambda_{\mathrm{LC}} = 0.1$	**0.0383**	**0.1312**	**0.566**	**1.580**

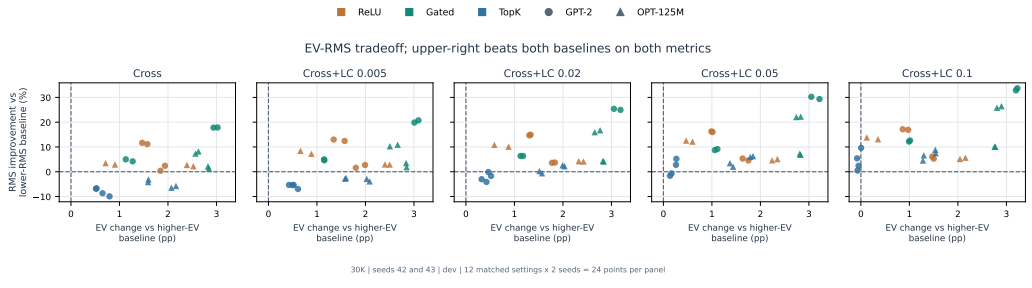


Figure 15: Complete EV–RMS tradeoffs for the five main cross-coordinate variants. Each panel retains all twenty-four setting–seed pairs. The horizontal reference is the higher-EV baseline and the vertical reference is the lower-RMS baseline; they can be different SAEs. The upper-right quadrant therefore strictly beats both baselines on both metrics.

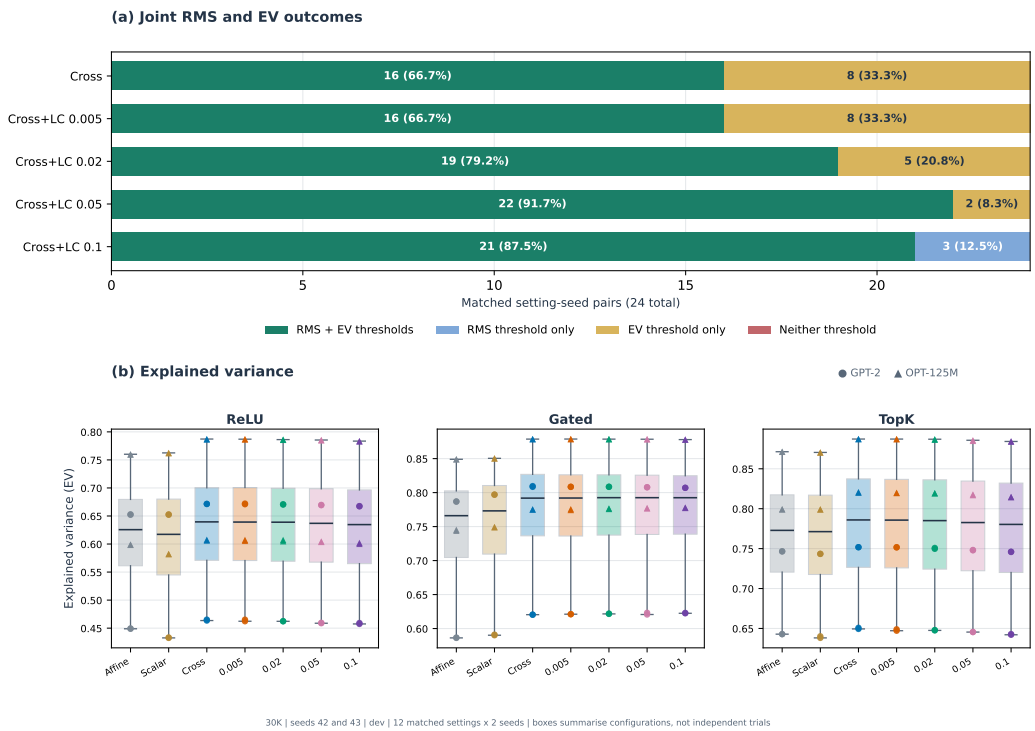


Figure 16: Joint RMS and reconstruction comparison over the twenty-four matched setting–seed pairs. **(a)** Each threshold requires a strict improvement over both Affine and Scalar: lower sixteen-level mean RMS or higher EV on unedited inputs. **(b)** Each point gives the EV of one trained SAE, with eight points per selector–method group (two language models $\times$ two sparsity settings $\times$ two seeds). Boxes show quartiles, horizontal lines the median, whiskers the full range.

H.4 Deletion false positives, sparsity, and feature use

For fixed encoded coefficients and decoder, deleting an active feature i replaces the residual by $r(x) - \widehat{z}_i(x)\widetilde{d}_i$. We count a deletion false positive if

$$\|r(x) - \widehat{z}_i(x)\widetilde{d}_i\|_2^2 < \|r(x)\|_2^2, \qquad \widehat{z}_i(x) > 0. \tag{34}$$

The FP rate divides the number of such events by *all* active (x, i) pairs on 4,096 evaluation inputs, with no target cap. There is no re-encoding or NNLS refit after deletion. This measures positive coefficients that increase current reconstruction error under the fixed code and decoder. Canonical rates use strict FP32 comparisons; a FP64 recomputation with identical encoded codes preserves the reported strict two-baseline win counts.

The unregularized cross-coordinate encoder improves FP over both baselines in all twenty-four setting–seed pairs (Figure 17), reducing the rate by 8.2%–96.1% relative to the lower-rate baseline. The eight ReLU reductions are 88.4%–96.1%. Cross+LC at weights 0.005 and 0.02 also improves over both baselines in all twenty-four pairs; at 0.05 and 0.1 it does so in twenty-two and twenty-one pairs, respectively.

For an input x, define $L_0(x) = \|\widehat{z}(x)\|_0 = |S(x)|$; reported L_0 is its mean over evaluation inputs. We also report the number of dictionary features positive at least once on those inputs. This evaluation-set feature-use count is distinct from the training-time dead-feature criterion in Appendix H.1. TopK has $L_0 = K$ up to rare numerical ties. Across Gated and ReLU settings, and across all 168 trained SAEs, the Cross-family L_0 and feature-use counts remain comparable to the baselines and show no systematic collapse (Table 11).

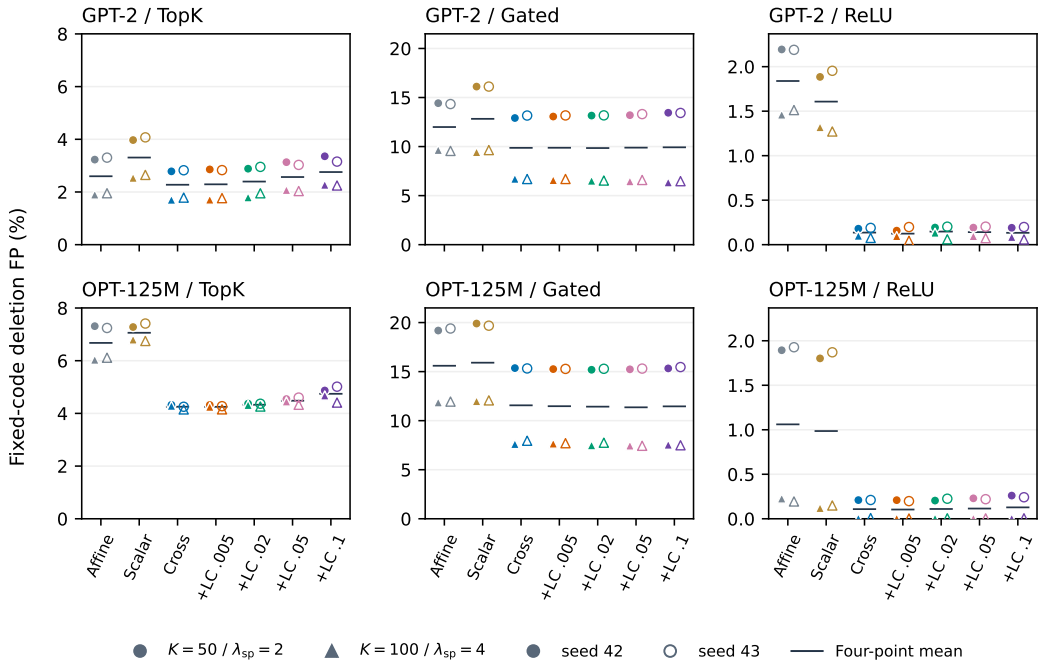


Figure 17: Fixed-code deletion FP rates. All 168 trained SAEs are shown: filled markers are seed 42 and open markers seed 43, with marker shape giving the sparsity setting. Short bars average the four setting–seed rates, rather than pooling their active-pair counts. +LC denotes Cross+LC, followed by its local consistency weight. The two language models share a vertical scale within each selector, but different selectors use different scales. All axes start at zero.

Table 11: Achieved sparsity and dictionary feature use across all 168 trained SAEs. Each entry averages the eight setting–seed pairs of its column (two language models $\times$ two sparsity settings $\times$ two seeds). L_0 is the mean number of positive encoded coefficients per evaluation input; feature use counts dictionary features positive at least once on those inputs, out of 4,096. TopK L_0 is omitted because the selector fixes it to K up to rare numerical ties.

	L_0		Features used (of 4,096)		
Model	Gated	ReLU	TopK	Gated	ReLU
Affine ($\mathrm{Enc}_{\mathrm{aff}}$)	89.5	92.2	3880	4035	3943
Scalar ($\mathrm{Enc}_{\mathrm{sc}}$)	97.9	96.6	3966	4032	3550
Cross ($\mathrm{Enc}_{\mathrm{cross}}$)	98.2	91.0	3884	4056	3872
Cross+LC 0.005	98.2	90.8	3882	4056	3874
Cross+LC 0.02	98.4	90.3	3887	4055	3874
Cross+LC 0.05	98.9	89.6	3900	4057	3867
Cross+LC 0.1	99.8	88.8	3904	4056	3858

H.5 Downstream steering check

The metrics above are computed inside the SAE's own representation: an edited activation is re-encoded and the resulting code is compared with the request. This subsection reports a more downstream measurement on the same checkpoints. The edited activation is not re-encoded; the language model continues from it, so the quantity being scored is the model's output text rather than its code.

At `blocks.6.hook_mlp_out`, every forward pass during generation replaces the activation with that checkpoint's own reconstruction and adds the requested edit,

$$x \longmapsto \widetilde{D}\widehat{z}(x) + b_{\text{dec}} + \tau\bar{n}\,\widetilde{d}_i,$$

where $\bar{n}$ is the mean activation norm over the twenty concept sentences below, excluding each sentence's first token, and τ is the dimensionless steering multiplier. Thus $\tau = 0$ leaves each checkpoint with its own zero-edit reconstruction baseline rather than a shared unedited model. The target coordinate i is chosen per checkpoint as the dictionary column with the highest cosine similarity to an external concept direction, obtained as a difference of means of `blocks.6.hook_mlp_out` activations over ten positive and ten negative sentiment sentences computed on the raw language model without any SAE. Across the 168 trained SAEs the selected similarity is 0.324 on average, ranging from 0.232 to 0.373. We use $\tau \in \{\pm 0.1, \pm 0.25, \pm 0.5, \pm 1, \pm 2\}$. This normalization differs from the α of Section 7.2, which scales the edit by the coordinate's own encoded coefficient; the two are not comparable magnitudes. Generation uses sixteen topic-neutral prompts, twenty-four new tokens, and greedy decoding. The continuations are scored by `distilbert-base-uncased-finetuned-sst-2-english`, a classifier independent of the SAE and of the language model's internals; perplexity is measured by teacher forcing on eight unrelated generic sentences under the same intervention.

Two quantities are reported, each averaged over the ten (magnitude, sign) combinations. The steering effect is the movement of the classifier's positive-class probability in the requested direction, so that both the positive and the negative request contribute positively when the edit works as asked. The perplexity cost is the increase in held-out perplexity. They are reported separately: a single combined score cannot distinguish an edit that fails to move the concept from one that moves it while damaging the model.

Table 12: Downstream steering check on the trained SAEs of Section 7 (protocol: Appendix H.5). Steering effect is the movement of an external sentiment classifier's positive-class probability in the requested direction; perplexity cost is the increase in held-out perplexity on unrelated text. Both average the five edit magnitudes in both signs, and each column averages the twelve matched settings at one seed. Bold marks Cross-family entries that beat both Affine and Scalar in the same column.

		Steering effect ↑		Perplexity cost ↓	
Model	λ_{LC}	seed 42	seed 43	seed 42	seed 43
Affine	—	0.083	0.081	8.02	7.86
Scalar	—	0.085	0.069	7.65	8.06
Cross ($\text{Enc}_{\text{cross}}$)	0	0.078	**0.095**	7.66	8.15
Cross+LC ($\text{Enc}_{\text{cross+LC}}$)	0.005	**0.122**	**0.126**	7.84	8.45
Cross+LC ($\text{Enc}_{\text{cross+LC}}$)	0.02	**0.129**	**0.119**	7.83	8.07
Cross+LC ($\text{Enc}_{\text{cross+LC}}$)	0.05	**0.093**	**0.108**	7.80	8.03
Cross+LC ($\text{Enc}_{\text{cross+LC}}$)	0.1	**0.109**	**0.123**	8.26	8.43

Table 12 gives both quantities for all 168 trained SAEs of Section 7, reported separately by seed. Every Cross+LC weight exceeds both baselines at both seeds, by 0.01–0.05 against a baseline effect near 0.08, while the perplexity cost is within 0.8 across all methods. The unregularized cross-coordinate encoder exceeds both baselines at seed 43 but not at seed 42.

This is a single-concept case study using one selected coordinate per trained SAE.