SPARSE-AUTOENCODER ATTRIBUTION GRAPHS ARE DECIDED AT THE MARGIN

Anonymous authors
Paper under double-blind review

ABSTRACT

Attribution graphs over sparse-autoencoder features predict an ablation's effect, and are compared across seeds. Under a TopK rule a feature is active where its pre-activation is positive and among the k largest, and its signed margin, positive on an active feature, measures the move that changes the membership. On five public TopK pairs of three Pythia models, zeroing one feature moves a downstream feature across the cutoff in 0.74 of cases or more, where the crossing features carry most of the response's energy. A write scales a feature's code by its dose. Over three doses the graph held on the active set has a median relative error of 0.247 or more on the code change. The margin graph adds the write's first-order change to every pre-activation and re-ranks, at the same one Jacobian-vector product per live feature, and errs by at most 0.033. At the median a crossing coordinate's code moves several times as far as its pre-activation, and the crossing features' decoded change has 1.24 times the model's change in norm or more. On one testbed of eight runs, sparse nonnegative maps writing one dictionary's atoms over another's explain 0.85 of the energy of the operator between two layers' codes on the source's block of live upstream and downstream features, and 0.47 on graphs masked by their margins' signs.

1 INTRODUCTION

Ablating one sparse-autoencoder feature often moves a feature at the next layer across its cutoff, so the attribution graph a reader consults to predict that ablation was built on a set the ablation changes. A sparse autoencoder under a TopK rule keeps, at each token, the features whose pre-activations are positive and among the k largest (Gao et al., 2025). An attribution graph over two such dictionaries at adjacent layers carries the derivative of each downstream feature's pre-activation with respect to each upstream feature's code, taken across the block between them (Ameisen et al., 2025; Marks et al., 2025; Farnik et al., 2025). On the public dictionaries of pythia-70m at layers 3 and 4, zeroing one feature moves a downstream feature across the cutoff in 0.84 of cases. At that ablation the graph held on the active set predicts the downstream code's change with a median relative error of 0.739, and a graph that keeps each feature's signed distance to the cutoff errs by 0.121.

The field's graph objects are built on the features the clean input makes active. Ameisen et al. (2025) write that "The nodes in the attribution graph represent active features", and Dunefsky et al. (2024, p. 17) derive the attribution between transcoder features "assuming that this feature is active". In the graph of Marks et al. (2025, p. 4) the "nodes correspond to feature activations or SAE errors", and each edge is linearized at the clean activations (Marks et al., 2025, p. 19). Attribution patching makes "a first-order Taylor expansion" at the clean activation (Kramár et al., 2024, p. 4), and its corrections recompute the attention softmax or average the gradient along a path (Kramár et al., 2024; Hanna et al., 2024). The first-order expansion of a TopK code at the clean input keeps the live support, since its Jacobian there projects onto it. Ameisen et al. (2025) name the features that expansion leaves out, the inactive ones "that are only 'one ablation away' from being active".

Across seeds, the comparisons reported for dictionaries score one feature at a time, by a cosine rule on encoder and decoder or by how often a feature reappears over many runs (Paulo & Belrose, 2026; Gerasimov et al., 2026). A score of that kind reads each feature's direction and never the operator between two layers, so it cannot say how much of a graph's operator survives a change of seed.

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.

A feature's margin is signed and positive when the feature is live. Its size is the distance its pre-activation can travel before the membership changes. This paper makes four contributions. First, it measures how often a write moves a downstream feature across the cutoff, against the write's dose, on five public TopK dictionary pairs and a testbed, and reads a ReLU dictionary beside them as a continuous comparison (Section 3). Second, it defines the margin graph, proves it exact where the block is affine and measures it against the graph held on the active set and two of the field's devices (Section 4). A Lipschitz constant along a write bounds each pre-activation's first-order error. A predicted membership whose margin clears the band those bounds set keeps its sign. The constant is read from Jacobians at three points along each write, so the band is an audit after the write, and a calibration locates the write norm at which the error reaches a declared level. Third, it reads the code's response against the model's own change and bounds from below the response a swap forces (Section 5). On the one layer triple read, a chain that carries a decoded middle code through the next block fails at the third layer, and the residual stream carries the write there. Fourth, on one testbed it carries a graph's operator across seeds through sparse nonnegative maps and places the masked graph's loss on the target's inactive atoms, which the signs of their margins select (Section 6).

2 SETUP

A sparse autoencoder reads the residual stream of a language model at one layer. Write d for the width of the stream and m for the width of the dictionary. The encoder $E \in \mathbb{R}^{m\times d}$ has rows $e_j^\top$, and the decoder has one unit column for each feature. At an activation x the pre-activations are $p = E(x - b_{\mathrm{pre}}) + b_{\mathrm{enc}}$, which is affine in x. The code z keeps some of the pre-activations and sets the rest to zero, and the features it keeps form the live support S. A TopK rule keeps the k largest positive pre-activations, with k less than m (Gao et al., 2025), and its cutoff θ_k is the k-th largest pre-activation. A JumpReLU rule keeps each pre-activation that clears the feature's own threshold (Rajamanoharan et al., 2024). A ReLU rule keeps every positive one (Bricken et al., 2023). The running example is the public TopK release on pythia-70m, whose dictionaries have width 32,768 and keep 16 features at a token (EleutherAI, 2024b), and Table 1 describes every pair read.

Two adjacent layers carry two dictionaries. Write D for the upstream dictionary's decoder, whose unit columns d_i are the upstream features' directions, and E and G for the downstream dictionary's encoder and decoder. The block f carries the upstream residual stream at the token's position to the downstream one, with every other position held fixed, and J is its Jacobian at the token. The operator $W = EJD$ carries upstream codes to downstream pre-activations. Its entry W_{ji} is the derivative of downstream pre-activation j with respect to upstream code i, which is the virtual weight of Ameisen et al. (2025) between the two features. Write Q and P for the diagonal projectors onto the upstream and the downstream live supports. The masked code Jacobian PWQ keeps the operator on the co-active block, and it is the Jacobian between two dictionaries' codes that Farnik et al. (2025) sparsify.

Under TopK the margin of feature j is $p_j - \max(\theta_{k+1}, 0)$ when j is live and $p_j - \max(\theta_k, 0)$ when it is not. Under JumpReLU the margin is p_j less the feature's threshold, and under ReLU it is p_j itself. Every statement below that reads a margin as a membership assumes a strict gap, that no two pre-activations it reads are equal, none is zero and none equals its feature's threshold. Under that assumption a feature is live exactly when its margin is positive, and the margin is how far its own pre-activation must move, with every other held fixed, before its membership changes.

A write on upstream feature i at dose α sets the feature's code u_i to αu_i. It adds $v = (\alpha - 1)u_i d_i$ to the upstream activation, so the upstream reconstruction residual stays in the edited input. A dose of zero is the zero ablation. To first order the downstream pre-activations change by $\widehat{\Delta p} = EJv$, which is one Jacobian-vector product and equals column i of W scaled by $(\alpha - 1)u_i$. The fixed-support graph predicts the downstream code $z + P\,\widehat{\Delta p}$, landing the whole change on the old live support. The margin graph predicts the code the rule assigns to the predicted pre-activations $p + \widehat{\Delta p}$, so it re-ranks them and names which features enter and which leave. Column i of W, one product per upstream feature at a token, enters both graphs, and the margin graph keeps every downstream row of that column where the fixed-support graph keeps the live rows. A prediction $\widehat{\Delta z}$ of the exact code change Δz has the relative error

$$\frac{\|\widehat{\Delta z} - \Delta z\|}{\|\Delta z\|}, \tag{1}$$

Table 1: The dictionaries read, one row per pair. The width counts a dictionary's features and k those TopK keeps. Cell F is the testbed trained here on pythia-70m, 8 runs, two sharing an initialization. Budgets are in billions of tokens, cell F's in activations. FVU is the fraction of variance unexplained up and downstream, lower being closer, on the 512 positions of the pinned Pile validation file the readings take, and for cell F on the block held out before training, as its runs' range.

source	model	layers	rule	width	k	budget	FVU up	FVU down
release	pythia-70m	3, 4	topk	32,768	16	8.2 B	0.125	0.148
release	pythia-70m	1, 2	topk	32,768	16	8.2 B	0.110	0.165
release	pythia-70m	4, 5	topk	32,768	16	8.2 B	0.148	0.103
release	pythia-160m	6, 7	topk	32,768	32	8.2 B	0.071	0.063
release	pythia-70m-deduped	3, 4	topk	32,768	16	none stated	0.126	0.164
release	pythia-70m-deduped	3, 4	relu	32,768	none	2 B	0.174	0.217
cell F	pythia-70m	3, 4	topk	16,384	32	199,999,488	0.1090 to 0.1094	0.155 to 0.156

with both changes taken over every coordinate of the downstream dictionary. A graph's error is the median of that relative error over the interventions, pooled over three predictor doses, the zero ablation and the doses 0.5 and 0.9, which were fixed before any predictor was read.

Five public TopK dictionary pairs and one ReLU pair are read at pinned revisions, each under the arithmetic of its own library (EleutherAI, 2026; Marks et al., 2024), and the pairs were fixed before any reading. The release on pythia-70m is read at layers 3 and 4. It is also read at the early block from 1 to 2 and at the last of its 6 blocks, from 4 to 5, and the release on pythia-160m at the middle of its 12 blocks, from 6 to 7 (EleutherAI, 2024a). The deduplicated TopK and ReLU releases share a model and a pair of layers at one width (EleutherAI, 2024c; saprmarks, 2024). A testbed pair, cell F, was trained for this paper on pythia-70m at layers 3 and 4 over 8 runs, two of which share an initialization and differ in data order, the retraining pair. Every dictionary is read in one gauge, each decoder column at unit norm in the residual stream's own units, which fixes the units of codes, pre-activations and cutoffs and in which the public dictionaries are stored. Cell F centres and scales each activation as $(x - \mu)/s$, where μ is the per-coordinate mean and s the mean distance of an activation from μ over a block held out before training. It stores unit columns in those units, so in the gauge its code is $z = s\, z^{\mathrm{st}}$ and its biases are $b_{\mathrm{pre}} = \mu + s\, b^{\mathrm{st}}_{\mathrm{pre}}$ and $b_{\mathrm{enc}} = s\, b^{\mathrm{st}}_{\mathrm{enc}}$, while its columns and rows are the stored ones read on raw activations. The readings take one position from each sequence of 2,048 tokens of the Pile's validation file, held out for the testbed, which trained on the training split. Every single-feature write on every live upstream feature is read at the first 512 positions, and the transport of Section 6 reads 1,024 sequences held out from the choice of its cap on a representation's atoms. Appendix B gives each pair's provenance and every count.

3 The crossing within a model

What does one write do to the downstream support? At rest the nearest margins are small. On the public TopK pairs the median over tokens of the nearest inactive margin runs from -0.018 to -0.008 in pre-activation units, and the cutoff stands at 0.06 to 0.15 of the largest pre-activation.

A single-feature ablation moves a downstream feature across the cutoff in 0.74 to 0.89 of zero ablations on the five public pairs, and in 0.92 on cell F, its 8 runs between 0.91 and 0.92. The rate follows the size of the write against the margins (Figure 1). It falls to 0.19 to 0.25 at a dose of 0.9 and to zero at one, which writes nothing. Past one it rises again, to 0.95 to 0.99 at a dose of 4.

The features that cross carry most of the code's response. Over the zero ablations that cross, the entering and leaving coordinates carry a mean of 0.73 to 0.84 of the response's energy on the five pairs and 0.79 on cell F, and Appendix B gives each formula. The entering coordinates alone carry 0.33 to 0.39 on the five pairs and 0.37 on cell F. At the median over a write's crossing coordinates and then over the crossing writes, a crossing coordinate's code changes 5.7 to 10.1 times as much as its pre-activation, an entrant jumping from zero to near the cutoff and a leaver falling to zero.

The ReLU pair crosses its zero nearly as often, and its crossing coordinates carry almost none of the response. A zero ablation moves one of its downstream features across zero in 0.69 of cases

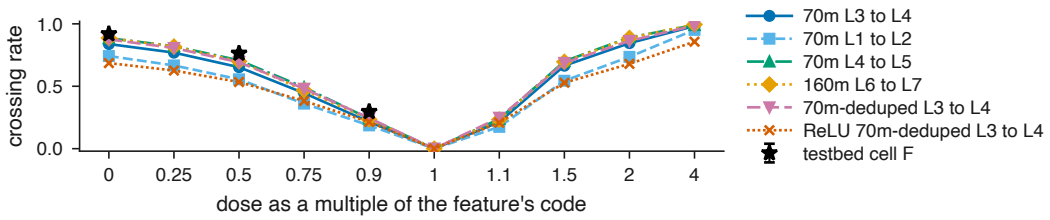

Figure 1: The crossing rate, the share of single-feature writes that move a downstream feature into or out of the live support, higher meaning more crossings, against the dose, over one write on every live upstream feature at each of the 512 positions of the pinned Pile validation file. Cell F, the testbed on pythia-70m, stands at the zero ablation and the doses 0.5 and 0.9 as the mean over its 8 runs, the bar their range. Pairs are labelled by Pythia model size and layers, and the doses stand at equal spacing.

and changes its live count in 0.59. Its crossing coordinates carry 0.05 of the response's energy, and their code changes 0.43 times as much as their pre-activation, since a ReLU code is continuous at its cutoff. The ReLU release differs from the deduplicated TopK release in its rule, its library, its budget and its context, so the contrast is one instance of the continuity Theorem 1 states.

4 THE MARGIN GRAPH

What predicts the code under a write at the product count of the graph held on the active set? Theorem 1 answers where the block is affine.

Theorem 1 (exactness on an affine segment, and what the fixed-support graph misses)**.** *The TopK code is piecewise linear in the pre-activations. It is linear on each cell of the partition cut by the hyperplanes $p_j = p_l$ and $p_j = 0$, continuous across $p_j = 0$, and discontinuous across a hyperplane $p_j = p_l$ that separates a member of the support from a non-member. Assume a strict gap and that the block is affine along the write, $f(x + tv) = f(x) + tJv$ for $t \in [0, 1]$. Then the margin graph's prediction is the exact code, entering and leaving coordinates included. The fixed-support graph and the margin graph differ only on the entering and the leaving coordinates, so on an affine segment the fixed-support graph's relative error on a write is at least the square root of the entering set's share of that write's response energy. Write θ_k for the cutoff before the write and r for the write's reach, the largest absolute first-order change of a downstream pre-activation. Where the token carries at least k positive pre-activations, an entering coordinate's code jumps from zero to at least $\theta_k - r$ and a leaving coordinate's falls from at least θ_k to zero, while each pre-activation moves by at most r.*

On the five public TopK pairs the margin graph's median error runs from 0.013 to 0.033 against 0.247 to 0.669 for the fixed-support graph, and on cell F it stands at 0.017 against 0.786 (Table 2). A bootstrap over the positions puts the difference of the two medians above zero on every pair read, and Appendix B gives each interval. Removing the re-ranking alone, so that membership is read against the old cutoff, raises the margin graph's error on cell F to 0.558.

The margin graph also names the features that enter and leave. Its precision on the entering set, micro-averaged over the writes, runs from 0.80 to 0.90 on the five pairs and its recall from 0.82 to 0.92. Ranking the features inactive before the write by their predicted pre-activations, at the true count of entrants, reaches a precision of 0.84 to 0.93, and ranking them by margin alone reaches 0.42 to 0.49, and Appendix B prints the confusion counts beside the exact oracle's. On the ReLU pair both graphs err little, at 0.008 and 0.017, since its code never jumps.

Two of the field's devices on the graph objects of Section 1 are read on cell F, on the fixed support and re-ranked. Integrated gradients average the first-order term at n points along the write, and the softmax device recomputes the attention weights at the written input with every other map at first order. On the fixed support, integrated gradients err by 0.777 at 5 products and by 0.777 at 10 (Figure 2). The recomputed attention softmax errs by 0.783 there, against 0.786 for the fixed-support graph. Re-ranked, the softmax device errs by 0.013, under the margin graph's 0.017 at the same one product on 8 of 8 runs, and integrated gradients err by 8.8×10^{-6} and 2.2×10^{-6} at 5 and 10 products. On the five public TopK pairs the re-ranked softmax device errs by 0.011 to 0.026, under the margin graph on 5 of them, and on the fixed support by 0.19 to 0.66, and Appendix B gives each device's

Table 2: The margin graph against the fixed-support graph. Each graph's median relative error on the downstream code's change and on that change decoded, lower being better, over one write on every live upstream feature at the 512 positions of the pinned Pile validation file, pooled over the zero ablation and the doses 0.5 and 0.9. Precision and recall, higher being better, are the margin graph's on the entering set, micro-averaged. Cell F, the testbed on pythia-70m, averages its 8 runs.

dictionaries	model	layers	fixed support code	margin graph code	fixed support decoded	margin graph decoded	precision	recall
TopK release	pythia-70m	3, 4	0.564	0.033	0.512	0.036	0.82	0.83
TopK release	pythia-70m	1, 2	0.247	0.023	0.287	0.029	0.86	0.86
TopK release	pythia-70m	4, 5	0.663	0.033	0.625	0.036	0.86	0.86
TopK release	pythia-160m	6, 7	0.617	0.013	0.577	0.014	0.90	0.92
TopK release	pythia-70m-deduped	3, 4	0.669	0.031	0.625	0.034	0.80	0.82
ReLU release	pythia-70m-deduped	3, 4	0.017	0.008	0.017	0.009	0.73	0.97
cell F	pythia-70m	3, 4	0.786	0.017	0.749	0.018	0.87	0.89

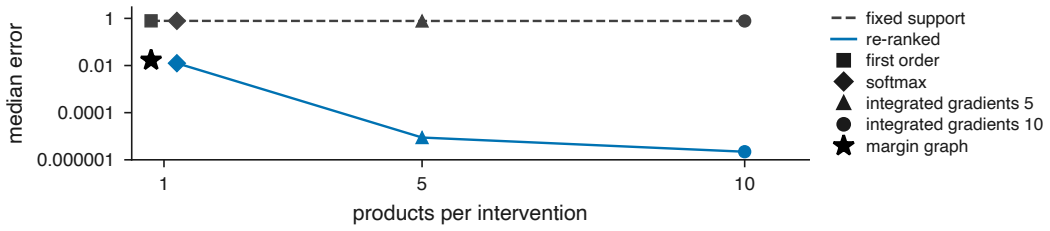

Figure 2: The field's devices against the margin graph on cell F, the testbed of 8 runs on pythia-70m, as the mean over runs of each run's median relative error on the downstream code, lower being better, against products per intervention, over one write on every live upstream feature at each of the 512 positions of the pinned Pile validation file at the zero ablation and the doses 0.5 and 0.9. Dashed lines join the devices on the fixed support and solid lines the devices re-ranked, markers at one product stand apart, and the star is the margin graph.

errors per pair beside its paths and its cost beyond the product. A re-ranked device's remaining error is its own error on the pre-activations, carried through the re-ranking, support mistakes included.

Off an affine segment the first-order term errs, and Theorem 2 bounds the error per write.

Theorem 2 (the first-order error and the band on the predicted margins)**.** *Assume a strict gap. Fix a write v and assume the block's Jacobian is Lipschitz along the segment, $\|J(x+tv) - J(x)\|_2 \le L_v\, t\, \|v\|$ for $t \in [0,1]$, with L_v the segment's own constant. Then each downstream pre-activation's first-order error is at most $\varepsilon_j(v) = \|e_j\|(L_v/2)\|v\|^2$. Read the margins on the predicted pre-activations, and let $\bar{\varepsilon}_{\text{out}}$ and $\bar{\varepsilon}_{\text{in}}$ be the largest $\varepsilon_l(v)$ outside and inside the predicted support, each zero where its set is empty. A predicted member whose predicted margin exceeds its own $\varepsilon_j(v)$ plus $\bar{\varepsilon}_{\text{out}}$ stands in the exact support, and a predicted non-member the size of whose negative predicted margin exceeds its own $\varepsilon_l(v)$ plus $\bar{\varepsilon}_{\text{in}}$ stands outside it, so such an entrant enters and such a leaver leaves. Where one constant L bounds L_v for every write, the same holds with $\varepsilon_j = \|e_j\|(L/2)\|v\|^2$.*

A constant along each write is read from three Jacobians, at the write's base, midpoint and endpoint, as the largest ratio of a Jacobian difference to the distance between its points. It is a lower bound on the segment's own, so the band is an empirical rule. Two of its points lie on the written segment, so the rule is an audit after the write. The rule accepts a predicted membership where the band holds under the write's own constant and the write's norm stands inside the regime below, set on half the 512 positions. On the other half it reads 4 sampled features per position at three doses over cell F's 8 runs and the five public TopK pairs. It accepts no wrong membership among 668,393,327, at 0.59 to 0.87 of a single-feature write's predicted memberships and 0.32 to 0.57 of its live ones. At the zero ablation it accepts 0.012 to 0.067 of a subset write's, again with none wrong (Appendix E).

The first-order error grows with the write. The regime reads the pre-activations' error, the norm of the first-order change of the downstream pre-activations less their exact change, over the exact change's norm, against the write's norm in units of the downstream cutoff's median over the positions, both in the gauge of Section 2. On single-feature writes the median error first reaches 0.3 at

a write of 5.55 on cell F, its 8 runs pooled, and 3.90 on the release at layers 3 and 4, and on subset writes at 7.54 and 4.48. The two populations pooled give boundaries of 7.26 and 4.36, within a ratio of 1.66, under the factor of 2.0 declared before the reading, and the median write stands at 1.68 and 3.57 cutoff units. Appendix E gives every run's and pair's boundary, set on half the positions and scored on the other, from 3.22 to 9.60 on the public pairs' single-feature writes.

The graph is built once per token at one product per live feature, and a write on any subset at any dose is then a sum of its members' columns and a sort. The subsets are random sets of two, four, eight and sixteen live upstream features, 16 per size at each position, ablated to zero. On the subsets of two the once-built graph errs by 0.287 on cell F, against 0.770 for the fixed-support graph. On the subsets of sixteen it errs by 0.458 against 0.551, and on the release at that size it errs above the fixed-support graph, at 0.670 against 0.631. The path predictor re-ranks integrated gradients, and at $n = 5$ it errs by at most 0.0043 on cell F and 0.124 on the release at every size (Appendix E).

5 THE READOUT AGAINST THE MODEL

Does the code's response say what the model did? A write changes the downstream activation by Δx, and the code reads that change through its decoder as $G\,\Delta z$. The part of $G\,\Delta z$ on features live before and after the write is its stable part. The part on features that entered or left is its entering-and-leaving part. The rest of the activation change, $\Delta x - G\,\Delta z$, is the change of the reconstruction residual, so the three parts sum to Δx, to within 1.2×10^{-14} on every intervention read.

Over the crossing interventions the entering-and-leaving part alone has a median norm of 1.24 to 1.72 times the activation change's on the five TopK pairs, and 1.56 on cell F (Figure 3a). Its projection on the activation change, over the change's squared norm, has a median of 0.21 to 0.29, and 0.24 on cell F. The whole decoded change has 1.41 to 1.82 times the activation change's norm, at a median cosine with that change of 0.27 to 0.50 and 0.38 on cell F.

Theorem 4 in Appendix A bounds that response from below at a swap on any block, by the cutoff times the separation of the swapping decoder columns less three reaches per swapped pair. On all 345,178 crossing single-feature writes at every position and predictor dose of the five public TopK pairs and cell F, the measured part stands at or above the bound. On the writes that swap one pair the

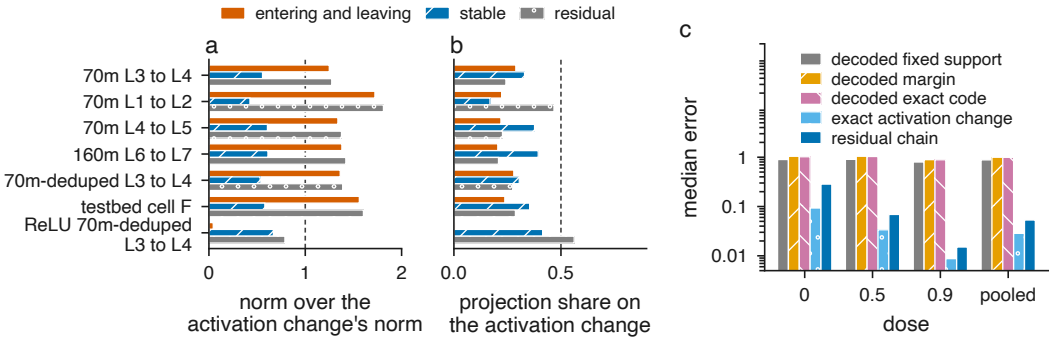

Figure 3: The decoded change against the activation's own change, and five chains across two blocks. The stable part decodes the change on features live throughout and the entering-and-leaving part the change on features that entered or left. The reconstruction residual's change completes the activation change. Panels a and b give each part's norm and projection share against the activation change, medians over the crossing writes at the zero ablation and the doses 0.5 and 0.9 on the 512 positions of the pinned Pile validation file, cell F, the testbed on pythia-70m, averaging its 8 runs. Dashed lines mark the change's norm and 0.5, the ceiling on the share declared before the reading. Panel c gives each chain's median error on the third layer's code, lower being better, on the pythia-70m release at layers 3, 4 and 5 over 24,576 writes. Through the second block the fixed-support, margin and exact chains carry their decoded middle code change, the exact activation change chain the exact change and the residual chain one product through both blocks. Each chain crosses the second block at first order at the clean middle activation, and all but the fixed-support chain re-rank at the third layer. Pairs are labelled by Pythia model size and layers.

bound is positive on 0.55 of the five public pairs' writes and 0.70 of cell F's. Where it is positive its median ratio to the activation change is 1.21 and 1.28, against a measured median of 2.61 and 2.53 on the same writes. The reconstruction residual's change cancels the excess.

Over every intervention the ReLU pair's decoded change has a median projection share of 0.38 on the activation change, a median norm ratio of 0.64 and a median cosine of 0.60. The rule R2 counts a decoded change as following the activation change where its share is at least 0.5 and its norm ratio between 0.50 and 1.25, and the ReLU pair's share falls under that floor, so R2 fired.

On the pythia-70m release at layers 3, 4 and 5, the chains that carry the fixed-support graph's, the margin graph's and the exact decoded middle code through the second block err at the third layer's code by 0.884, 0.999 and 0.996 (Figure 3c). The exact activation change chain errs by 0.029, and the residual chain, one product through both blocks in the residual stream, errs by 0.053 at one product per live feature. Every chain carries its middle change through the second block at first order, as one Jacobian-vector product at the clean middle activation. All but the fixed-support chain, which holds the third layer's support, then re-rank at the third layer. The rule H3 took as its reference write the zero ablation of the token's live upstream feature of largest code, whose first-order change of each middle pre-activation is that feature's reach. It split the margin graph's decoded middle code change on every two-hop write at the three doses into its parts on the middle features whose margins lie beyond and within that reach, and asked the first part's share of the exact middle change to exceed the second's by 0.2. The difference was 0.16, so H3 fired, and Appendix E gives the operands. On affine segments a decoded chain's third-layer pre-activations differ from the residual chain's by the second block's image of what the decoded code drops, which for the exact code is the reconstruction residual's change (Theorem 4), and the chain errors above are that difference under the third layer's rule.

6 TRANSPORT ACROSS SEEDS

How much of a graph's operator survives a change of seed? Take dictionaries A and B from two seeds of one model, each with an upstream decoder and a downstream encoder at one layer pair. Each atom of the source A is written as a sparse nonnegative combination of the target B's atoms. The decoder columns give a map M with $D_A \approx D_B M$, and the encoder rows a map N with $E_A \approx N E_B$. On cell F, Table 3 gives what transports and Figure 4 where the rest is lost.

Theorem 3 (the operator transports, and the mask term reads the signs of the margins)**.** *Let $W_A = E_A J D_A$ and $W_B = E_B J D_B$ be the operators of A and B at a token. If $D_A = D_B M$ and $E_A = N E_B$, then $W_A = N W_B M$ at every token and for any two widths, with no matching. Write Q_A, Q_B and P_A, P_B for the projectors onto the two dictionaries' upstream and downstream live supports. The masked identity $P_A W_A Q_A = P_A N (P_B W_B Q_B) M Q_A$ holds at a token where every row of N at a live downstream feature of A places no weight on an inactive downstream atom of B, and every column of M at a live upstream feature of A places no weight on an inactive upstream atom of B. The part the masks change, $P_A N (W_B - P_B W_B Q_B) M Q_A$, is the transport carried by the maps' weight on B's atoms whose margins are not positive at the token. It reads the sign of each margin and never its size. Without exact representation, write $\Delta D = D_A - D_B M$ and $\Delta E = E_A - N E_B$. Then $W_A - N W_B M = \Delta E\, J D_A + N E_B J\, \Delta D$ exactly, and its Frobenius norm is at most $\|J\|_2 (\|\Delta E\|_F \|D_A\|_2 + \|N E_B\|_2 \|\Delta D\|_F)$.*

So the masked graph of B, carried to A, drops the part of the operator that rides on B's inactive atoms, and that weight leaves or enters whole as a margin's sign flips.

The maps are fitted on cell F by nonnegative orthogonal matching pursuit on unit vectors (Nguyen et al., 2019), to a relative residual of 0.436, with at most 32 atoms in each representation. A vector that misses the tolerance keeps its closest fit, so the identity is read in its approximate form. The decoder columns have unit norm in the gauge of Section 2, and the fitted coefficients C of the encoder rows are carried back to that gauge's rows by their norms, $N_{jl} = C_{jl}\, \|e_j^A\| / \|e_l^B\|$. Maps are fitted over the 30 ordered pairs among the six independent seeds and both directions of the retraining pair, and Appendix C specifies the fit, the choice of its cap and every null to the count and the seed. A prediction $\widehat{W}$ of a reference operator W explains the fraction $1 - \sum_t \|W_t - \widehat{W}_t\|_F^2 / \sum_t \|W_t\|_F^2$ of the reference's energy, summed over the held-out positions t, on the whole operator unmasked and on the source's co-active block otherwise, averaged over a class's pairs.

Table 3: Transport across seeds on cell F, the testbed of 8 runs on pythia-70m, on the 1,024 held-out positions of the pinned Pile validation file. The first five columns are explained fractions, one less the residual energy over the reference's energy summed over the positions, higher being closer, on the whole operator unmasked and on the source's co-active block otherwise. The one-to-one matching is an optimal assignment on positive cosine at nonnegative projections. The masked reading carries the target's masked code Jacobian through the sparse maps and the unmasked-target reading its whole operator. The last four columns are energies over the block's. The gap is the unmasked-target reading less the masked one, larger meaning more lost to the masks. Before rounding it is the mask term's energy plus the cross term, twice the mask term's signed inner product with the representation residual. That residual's own energy is the last column, larger for a looser fit, and it cancels in the gap. Between seeds averages the 30 ordered pairs of runs of independent seeds and the retraining pair its two directions. The two rows under each are the 2.5 and 97.5 percentiles of 2,000 draws from seed 20260925, each leaving one run out and resampling the positions as Appendix C states, so they describe this testbed.

pair	matching unmasked	sparse unmasked	matching masked	sparse masked	sparse, target unmasked	gap energy	mask term energy	cross term energy	residual energy
between seeds	0.22	0.57	0.36	0.47	0.85	0.38	0.34	0.05	0.15
low end	0.220	0.568	0.352	0.457	0.847	0.379	0.330	0.044	0.143
high end	0.222	0.571	0.376	0.476	0.857	0.390	0.343	0.053	0.153
retraining pair	0.35	0.64	0.43	0.51	0.85	0.34	0.29	0.05	0.15
low end	0.349	0.638	0.426	0.501	0.846	0.334	0.286	0.047	0.149
high end	0.350	0.640	0.441	0.515	0.851	0.346	0.297	0.050	0.154

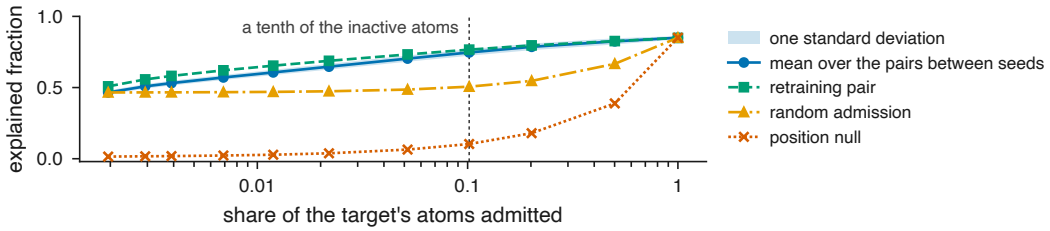

Figure 4: The admission curve on cell F, the testbed of 8 runs on pythia-70m, on 1,024 held-out positions of the pinned Pile validation file, as the explained fraction of Table 3 on the masked code Jacobian, higher being closer, against the share of the target's atoms admitted to its mask. Its live atoms stay, and its inactive atoms enter nearest the cutoff first. Between seeds averages the 30 pairs, one standard deviation shaded. Random admission draws as many inactive atoms uniformly at 3 draws a pair, and the position null permutes the target's positions. The dotted line marks a tenth of the inactive atoms, where the rule declared before the reading asked for half the gap recovered.

Two rules declared before the reading fired on the maps. F3 fired, since the maps represent 0.71 of the decoder columns at a median of 5 atoms, over its bound of 4, and 0.24 of the encoder rows, under its floor of 0.5. F4 fired, since a narrower dictionary of width 2,048 written over cell F's atoms takes two or more atoms for a mean share of 0.85 of its columns, under the largest between seeds, 0.89.

On the unmasked operator the sparse maps explain 0.57 of the energy between seeds against 0.22 for a one-to-one matching of atoms. That gain of 0.35 clears the bound of 0.19 declared before the reading. Random maps of matched sparsity put each vector on as many target atoms as its fitted map, drawn uniformly, with the coefficients refitted, and the permutation null relabels the target's atoms under the fitted maps. The maps beat every one of 20 random draws and every one of 200 null draws on 30 of 30 pairs, the random maps explain 0.003 on average, and the null explains -0.53, a fraction below zero. On the masked code Jacobian the same maps explain 0.47, and read with the whole target dictionary available, the target's unmasked operator carried through the same maps onto the same block, they explain 0.85. The gap between the two readings is 0.38 between seeds and 0.34 on the retraining pair. It splits exactly into the mask term's own energy at 0.34 and twice the mask term's inner product with the representation residual at 0.05, each over the reference's energy. The representation residual's own energy, 0.15, enters both readings and cancels in the gap.

Theorem 3 places that loss on the target's atoms of non-positive margin, and the admission curve tests whether the atoms nearest the cutoff carry most of it. A share of 0.1 of the target's inactive atoms admitted at random recovers 0.11 of the gap between seeds, in proportion to the atoms admitted, and a null permuting the target's positions recovers -0.94. The same share taken nearest the cutoff first recovers 0.73 of the gap between seeds and 0.76 on the retraining pair, so most of the loss sits on the inactive atoms nearest the cutoff.

7 Related work

Maps across seeds. Meta-dictionaries write one dictionary's decoder directions as sparse combinations of another's on one model and one layer (Leask et al., 2025; Bussmann et al., 2024b; Rubinstein et al., 2010; Li et al., 2016), model merging carries weights through a permutation or a transport plan (Ainsworth et al., 2023; Singh & Jaggi, 2020; Blondel et al., 2018; Liu et al., 2023; Cuturi et al., 2023), and causal abstraction composes exact bijective translations (Geiger et al., 2025). Features are matched across layers (Balagansky et al., 2025; Cao et al., 2026; Chen et al., 2025; Sweeney, 2026) and dictionaries trained jointly across seeds or models (Thasarathan et al., 2025; Váradi & Kmetty, 2026; Wang et al., 2025). Identifiability results make two dictionaries coincide up to a permutation with scaling or signs (Aharon et al., 2006; Nelson et al., 2026; Song et al., 2026), and a permutation at positive scales is the case of Section 6's maps with one atom per column and row.

Writes within a model. Feature interventions run at high strengths (Lindsey et al., 2025; Templeton et al., 2024), and their side effects are forecast or validated by ablation (Duan, 2026; Chalnev et al., 2024; Droste et al., 2026). JumpReLU and batch-TopK dictionaries select by a threshold, so the margin is defined beside TopK (Rajamanoharan et al., 2024; Lieberum et al., 2024; Bussmann et al., 2024a), and benchmarks score features at rest (Karvonen et al., 2025; Chanin et al., 2025).

First-order graphs. Attribution patching and its corrections (Nanda, 2023; Syed et al., 2024; Kramár et al., 2024; Sundararajan et al., 2017; Hanna et al., 2024; Marks et al., 2025), circuit searches (Conmy et al., 2023; Bhaskar et al., 2024; Mueller et al., 2025), replacement models (Ge et al., 2024; Dunefsky et al., 2024; Ameisen et al., 2025) and crosscoders (Lindsey et al., 2024) build the first-order graphs Section 4 re-ranks. For an MLP, Kramár et al. (2024, p. 42) linearize to the next node's pre-activation and recompute the nonlinearity, and the margin graph applies that factored form to a TopK readout, whose recomputed rule is a sort. It names the inactive features one ablation away that Ameisen et al. (2025) propose to consider, at the product count of the graph they build. A network of ReLU blocks is a max-affine spline (Balestriero & Baraniuk, 2018; Hanin & Rolnick, 2019a;b), while a TopK code jumps across the cells Theorem 1 names.

8 Limitations

No JumpReLU dictionary was read. The one public release with adjacent residual layers stands on a model behind an access agreement this study lacks. No transcoder was read. The largest model read is pythia-160m. The transport stands on one cell, between seeds and on its retraining pair, because no public release carries a second seed. Integrated gradients were read on cell F alone. No runtime or memory was measured, and the cost model counts Jacobian-vector products.

9 Conclusion

A zero ablation moves the downstream support in most cases on every TopK pair read, and the margin graph predicts the code and that change. A reader who builds a graph keeps every downstream row and re-ranks at no added product. A reader who ablates a feature reads off which features enter and leave, and audits each predicted membership after the write by the band inside the regime its norm sets. On 4 sampled features per position at three doses that audit accepted no wrong membership at 0.59 to 0.87 of a single-feature write's memberships. On the one triple read every decoded middle code failed at the third layer while one product through the residual stream carried it. A reader who compares two seeds compares the operator, or the source's co-active block with the target's mask widened by its inactive atoms nearest the cutoff, since a masked graph carries each token's sign pattern into its edges. Which nodes of a circuit across several blocks does a residual-stream chain predict at each layer's code, and do decoded middle codes fail on other triples?

AI USE STATEMENT

In this work, we used generative AI tools for "assist in the writing of proofs", "propose or refine hypotheses", and "clean and reformat dataset". We have not used generative AI tools for "generate synthetic data sets", "help develop theoretical models or conceptual frameworks", "formulate mathematical claims", "provide critical ingredients for proving mathematical claims", "design or provide feedback on research methodology or experiments", "implement methods", "support qualitative and thematic data analysis", or "interpret results", and "assist with translation" is not applicable to this work. Additionally, we used generative AI tools for "create or modify scientific figures or images", "create or edit software code", "summarize or analyse existing literature", "brainstorming", "sourcing/searching for information", "edit a research paper to improve readability", "identify relevant literature", "format references", and "propose a title or keywords for a research paper". We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A carries the proof of each theorem and proposition with its assumptions stated. Appendix B gives each dictionary's revision and licence with the library it is read under, the corpus and its sampling, every count of interventions, the operator checks, the field's devices as implemented and the testbed's training configuration. Appendix C specifies the transport's fit, its matching, its nulls and its interval. Appendix E gives every threshold with what the reading returned. The supplementary material carries the code written for this paper and every result file the numbers print from.

ETHICS STATEMENT

The public dictionaries were read at the pinned revisions Appendix B gives, for evaluation only. Their repositories give the licence identifier `mit` for the releases on pythia-70m and pythia-160m and for the ReLU release, while the deduplicated TopK release's repository states no licence. The corpus card gives the licence identifier `other` for the collection it serves.

REFERENCES

Michal Aharon, Michael Elad, and Alfred M. Bruckstein. On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them. *Linear Algebra and its Applications*, 416(1):48–67, 2006. doi: 10.1016/j.laa.2005.06.035.

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. arXiv:2209.04836.

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`.

Nikita Balagansky, Ian Maksimov, and Daniil Gavrilov. Mechanistic permutability: Match features across layers. In *International Conference on Learning Representations*, 2025. arXiv:2410.07656.

Randall Balestriero and Richard G. Baraniuk. A spline theory of deep learning. In *International Conference on Machine Learning*, pp. 374–383, 2018.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.16778.

Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. In *International Conference on Artificial Intelligence and Statistics*, 2018. arXiv:1710.06276.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris 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, 2024a.

Bart Bussmann, Michael Pearce, Patrick Leask, Joseph Bloom, Lee Sharkey, and Neel Nanda. Showing SAE latents are not atomic using meta-SAEs. AI Alignment Forum, 2024b. URL https://www.lesswrong.com/posts/TMAmHh4DdMr4nCSr5.

E.J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005. doi: 10.1109/TIT.2005.858979.

Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus. Mathématique*, 346(9-10):589–592, 2008. doi: 10.1016/j.crma.2008.03.014.

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

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features, 2024.

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*, 2025. arXiv:2409.14507.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2506.06609.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, 2023. arXiv:2304.14997.

Marco Cuturi, Michal Klein, and Pierre Ablin. Monge, bregman and occam: Interpretable optimal transport in high-dimensions with feature-sparse maps. In *International Conference on Machine Learning*, 2023. arXiv:2302.04065.

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.

D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006. doi: 10.1109/TIT.2005.860430.

Hendrik Droste, Christian Medeiros Adriano, Kathrin Korte, and Holger Giese. Where decoder cosine similarity fails for SAE feature flow discovery, 2026.

Evan Duan. Pre-intervention prediction of sparse autoencoder steering side effects, 2026.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.11944.

EleutherAI. sae-pythia-160m-32k. Hugging Face model repository EleutherAI/sae-pythia-160m-32k, revision 2046768ae0c8cb69a2e8ed64f2eafb9f8c5fa294, 2024a. URL https://huggingface.co/EleutherAI/sae-pythia-160m-32k/tree/2046768ae0c8cb69a2e8ed64f2eafb9f8c5fa294. licence mit, last modified 2024-08-05.

EleutherAI. sae-pythia-70m-32k. Hugging Face model repository EleutherAI/sae-pythia-70m-32k, revision 63999ba8720bb4817d02466d7fe11b887ad4e3cc, 2024b. URL https://huggingface.co/EleutherAI/sae-pythia-70m-32k/tree/63999ba8720bb4817d02466d7fe11b887ad4e3cc. licence mit, last modified 2024-08-05.

EleutherAI. sae-pythia-70m-deduped-32k. Hugging Face model repository EleutherAI/sae-pythia-70m-deduped-32k, revision 7a64bade597212176dfe0782f9d839b94f0addaf, 2024c. URL https://huggingface.co/EleutherAI/sae-pythia-70m-deduped-32k/tree/7a64bade597212176dfe0782f9d839b94f0addaf. licence none stated on the repository, last modified 2024-08-15.

EleutherAI. sparsify. GitHub repository EleutherAI/sparsify, licence MIT, read at commit 42c064525b1c, 2026. URL https://github.com/EleutherAI/sparsify.

Lucy Farnik, Tim Lawson, Conor Houghton, and Laurence Aitchison. Jacobian sparse autoencoders: Sparsify computations, not just activations. In *International Conference on Machine Learning*, 2025. arXiv:2502.18147.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, 2025. arXiv:2406.04093, OpenReview forum tcsZt9ZNKD.

Xuyang Ge, Fukang Zhu, Wentao Shu, Junxuan Wang, Zhengfu He, and Xipeng Qiu. Automatically identifying local and global circuits with linear computation graphs, 2024.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83), 2025. URL https://jmlr.org/papers/v26/23-0058.html.

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

R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, 2003. doi: 10.1109/TIT.2003.820031.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pp. 2596–2604, 2019a. arXiv:1901.09021.

Boris Hanin and David Rolnick. Deep ReLU networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, volume 32, 2019b. arXiv:1906.00904.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *Conference on Language Modeling*, 2024. arXiv:2403.17806.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Isaac Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum Stuart McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *International Conference on Machine Learning*, 2025. arXiv:2503.09532, OpenReview forum qrU3yNfX0d.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing LLM behaviour to components, 2024.

Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. In *International Conference on Learning Representations*, 2025. arXiv:2502.04878.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representations*, 2016. arXiv:1511.07543.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on Gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.19. arXiv:2408.05147.

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/index.html.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, 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. On the biology of a large language model. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Tianlin Liu, Joan Puigcerver, and Mathieu Blondel. Sparsity-constrained optimal transport. In *International Conference on Learning Representations*, 2023. arXiv:2209.15466.

Samuel Marks, Adam Karvonen, and Aaron Mueller. dictionary_learning. GitHub repository saprmarks/dictionary_learning, licence MIT, read at commit 60ec6bf52649, 2024. URL https://github.com/saprmarks/dictionary_learning.

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. arXiv:2403.19647, OpenReview forum I4e82CIDxv.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In *International Conference on Machine Learning*, 2025. arXiv:2504.13151.

Neel Nanda. Attribution patching: Activation patching at industrial scale. Blog post, 2023. URL https://www.neelnanda.io/mechanistic-interpretability/attribution-patching.

Walter Nelson, Theofanis Karaletsos, and Francesco Locatello. Toward identifiable sparse autoencoders. In *International Conference on Machine Learning*, 2026. arXiv:2605.31245.

Thanh Thi Nguyen, Jerome Idier, Charles Soussen, and El-Hadi Djermoune. Non-negative orthogonal greedy algorithms. *IEEE Transactions on Signal Processing*, 67(21):5643–5658, 2019. doi: 10.1109/TSP.2019.2943225.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. In *International Conference on Learning Representations*, 2026. arXiv:2501.16615, version of record confirmed at iclr.cc/virtual/2026/poster/10010650.

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

R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3):1553–1564, 2010. doi: 10.1109/TSP.2009.2036477.

saprmarks. pythia-70m-deduped-saes. Hugging Face model repository saprmarks/pythia-70m-deduped-saes, revision 50a434461d36ed78d1b0b901944e6edc829f1dce, 2024. URL https://huggingface.co/saprmarks/pythia-70m-deduped-saes/tree/50a434461d36ed78d1b0b901944e6edc829f1dce. licence mit, last modified 2024-07-29.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, 2020. arXiv:1910.05653.

Xiangchen Song, Aashiq Muhamed, Yujia Zheng, Lingjing Kong, Zeyu Tang, Mona T. Diab, Virginia Smith, and Kun Zhang. Mechanistic interpretability should prioritize feature consistency in sparse autoencoders. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2172–2210, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.99. URL https://aclanthology.org/2026.acl-long.99/.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328, 2017. arXiv:1703.01365.

John Sweeney. Signed-permutation coordinate transport for RMSNorm transformers, 2026.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.25. arXiv:2310.10348.

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.

Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos G. Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment. In *International Conference on Machine Learning*, 2025. arXiv:2502.03714.

J.A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004. doi: 10.1109/TIT.2004.834793.

Bendegúz Váradi and Zoltán Kmetty. Cross-seed explainability using procrustes-conditioned joint end-to-end top-k sparse autoencoders, 2026.

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

A PROOFS

Every statement below was also checked in double precision on constructed instances from declared seeds. The checks from seed 9200 confirm Theorem 1's exact prediction on affine blocks and the fixed-support graph's difference on the entering and leaving coordinates. They also confirm the code's linearity between rank changes and Theorem 2's bound and band, with the mask term of Theorem 3 and the write that flips it, and they hold at 8 of 8. The checks from seed 9000 confirm Theorem 3's two identities with the atom that breaks the masked one and its decomposition with

its bound. They also confirm that a representation's class, the count of atoms at the first step that meets the tolerance as Appendix C sorts it, depends on the tolerance, and they hold at 6 of 6. The checks from seed 9500 confirm the single-swap bound of Theorem 4 on affine blocks, the growth of the ratio across tokens whose gap at the cutoff shrinks, the residual's absorption of the excess, the residual chain's identity, Proposition 2's conditions and the band's remainder, at 12 of 12. The checks from seed 9700 confirm the set where the graphs differ, the entering set's share, the band at a cutoff at or below zero, the clipped margin, the entrant's jump against the write's reach, the tie and the chain identity on pre-activations, at 12 of 12. The checks of Theorem 4 as stated here run from seed 9800 on residual streams of width 24, dictionaries of width 60 at a TopK of 5 and hand-built instances, at a relative tolerance of 1.0×10^{-9}, and hold at 6 of 6. They confirm both bounds on writes through a nonlinear block, the affine form and the floor on a swapping write's reach. With coincident decoder columns the ratio stays at one half, and with a cutoff that shrinks with the gap it stays at the square root of five over two. At one token with a fixed cutoff and gap, no write whose reach is under half the gap swaps.

Throughout, x is the upstream activation at the token, v the write, $f(x)$ the downstream activation and $p = E(f(x) - b_{\mathrm{pre}}) + b_{\mathrm{enc}}$ the downstream pre-activations, which are affine in the downstream activation. The exact pre-activations after the write are p' and the predicted ones are $\hat{p} = p + \widehat{\Delta p}$, with $\widehat{\Delta p} = EJv$. The live support before the write is S, the exact support after it is S' and the predicted support is $\hat{S}$. A strict gap is assumed throughout, that no two coordinates of a pre-activation vector a statement reads are equal, none is zero and none equals its feature's threshold under a JumpReLU rule.

The statements proved here that the main text does not carry are these.

Theorem 4 (the readout overshoot and the residual chain)**.** *Write $\Delta x = f(x + v) - f(x)$ for the block's image of the write, $R = \max_i |p'_i - p_i|$ for the write's exact reach, the largest absolute change of a downstream pre-activation, and γ for the largest downstream decoder column norm. Assume that the token carries at least k positive pre-activations before and after the write, and that the write swaps s pairs, the entrants I entering the live support and the leavers O leaving it. Then the entering-and-leaving part of the decoded code change and the whole decoded change satisfy*

$$\|G\,\Delta z_{I\cup O}\| \geq \theta_k \Big\| \sum_{l\in I} g_l - \sum_{j\in O} g_j \Big\| - 3sR\gamma, \qquad \|G\,\Delta z\| \geq \theta_k \Big\| \sum_{l\in I} g_l - \sum_{j\in O} g_j \Big\| - (k+2s)R\gamma,$$

and the ratio of each to $\|\Delta x\|$ is at least the same right-hand side over $\|\Delta x\|$. On a block affine along the write, $\Delta x = Jv$ and $R \leq \|EJ\|\,\|v\|$. Where $\theta_k > 0$ and the gap $\theta_k - \theta_{k+1}$ at the cutoff is positive, a write whose reach is under $\min(\theta_k, (\theta_k - \theta_{k+1})/2)$ leaves the support unchanged. The reconstruction residual's change satisfies $\|\Delta x - G\,\Delta z\| \geq \|G\,\Delta z\| - \|\Delta x\|$. Now let the write cross two blocks with Jacobians J_1 and J_2, the middle dictionary's decoder G_2 and the third layer's encoder E_3, and assume the first block is affine along the write and the second along the middle change Δy it induces. Then $p_3 + E_3 J_2 J_1 v$ are the exact third-layer pre-activations, and the residual chain's code under the third layer's rule is the exact code. A chain that carries a decoded middle code change $G_2\,\Delta\hat{z}$ through the second block forms third-layer pre-activations that differ from the residual chain's by $E_3 J_2(\Delta y - G_2\,\Delta\hat{z})$, which for the exact middle code is the second block's image of the middle reconstruction residual's change.

Proposition 1 (the maps are determined and stable)**.** *Let y be an atom of A and c a column of M that represents it with a support of s atoms. If every set of at most $2s$ columns of D_B is linearly independent, which holds when the spark of D_B exceeds $2s$, then no other representation of y uses at most s atoms, and likewise for each row of N over the rows of E_B. If two representations of one atom y meet a relative residual τ on supports S and S', they differ by at most $2\tau\|y\|/\sigma$, where σ is the smallest singular value of the columns of D_B indexed by $S \cup S'$.*

Proposition 2 (the band per feature and the partner condition)**.** *Assume an affine block, and call the size of a feature's own first-order pre-activation change its own reach, with the write's reach r the largest of them. A live feature whose margin exceeds its own reach plus the largest reach among the inactive features keeps its membership, and an inactive feature the size of whose negative margin exceeds its own reach plus the largest reach among the live features stays out. A live feature whose own move is under its margin leaves only where an inactive partner's move and its own together reach its margin and the size of the partner's. The decoded first-order change of a live feature that keeps its membership is the rank-one operator $g_j e_j^\top$ applied to the activation change.*

Proof of Theorem 1. On each cell of the partition cut by the hyperplanes $p_j = p_l$ and $p_j = 0$ the order of the pre-activations and their signs are fixed, so the support is a fixed set S_0 and the code is $P_{S_0}\, p$, which is linear. Each cell is an intersection of half-spaces and so convex. The cells are finitely many, so the code is piecewise linear. Across $p_j = 0$ a coordinate enters or leaves at the value zero, so the code is continuous there. Across a hyperplane $p_j = p_l$ that separates a member from a non-member, one coordinate goes from zero to the shared value and the other from that value to zero. The shared value is positive, so the code jumps there. These cells are the activation regions of Hanin & Rolnick (2019b) for one dictionary, and the spline view of Balestriero & Baraniuk (2018) states the partition into cells and the linearity on each for a network of ReLU blocks.

Suppose $f(x + tv) = f(x) + tJv$ for $t \in [0, 1]$. Then $f(x + v) = f(x) + Jv$, and since p is affine in the downstream activation, $p' = p + EJv = \hat{p}$. The margin graph's prediction is the code of $\hat{p}$, which is the code of p', the exact code, on every coordinate, the entering and leaving ones included. The ranking plays no part in the argument, since the prediction is exact before any ranking is read.

On a feature j in $S \cap \hat{S}$ the margin graph predicts $\hat{p}_j$, and the fixed-support graph predicts $z_j + \widehat{\Delta p}_j = p_j + \widehat{\Delta p}_j = \hat{p}_j$, since the code equals the pre-activation on the live support. On a predicted leaver, in S and not in $\hat{S}$, the fixed-support graph predicts $\hat{p}_j$ and the margin graph zero. On a predicted entrant l, in $\hat{S}$ and not in S, the fixed-support graph predicts zero and the margin graph $\hat{p}_l$, which is positive. Everywhere else both predict zero. So the set where the two differ lies inside the entering and the leaving coordinates, it holds every entrant, and it holds a leaver exactly when $\hat{p}_j$ is not zero. On an affine segment the exact change on an entrant l is $\hat{p}_l$ and the fixed-support graph's change there is zero, so the squared norm of its error is at least the exact response's energy on the entering set, and dividing by the response's energy gives the bound on its relative error. The energy on the leaving coordinates enters no such bound, since the fixed-support graph predicts a leaver's change up to its jump.

Where the token carries at least k positive pre-activations the cutoff θ_k is positive and every inactive feature stands outside the top k. On an affine segment every pre-activation moves by at most r, so the k-th largest pre-activation after the write is at least $\theta_k - r$. An entrant stands among the k largest after the write, so its code rises from zero to at least $\theta_k - r$. A leaver stood among the k largest before it, so its code was at least θ_k and falls to zero. □

Proof of Theorem 2. Along the segment,

$$\|f(x+v) - f(x) - Jv\| = \Big\| \int_0^1 \big(J(x+tv) - J(x)\big)v\, dt \Big\| \le \int_0^1 L_v\, t\, \|v\|^2\, dt = \frac{L_v}{2}\|v\|^2,$$

by integrating the derivative along the segment and taking the norm under the integral, which is the mean-value form of Taylor's theorem on one segment. Reading the remainder through the encoder row e_j by the Cauchy and Schwarz inequality gives $|p'_j - \hat{p}_j| \le \|e_j\|(L_v/2)\|v\|^2 = \varepsilon_j(v)$.

Read the margins on $\hat{p}$, whose k-th and k plus first largest values are $\hat{\theta}_k$ and $\hat{\theta}_{k+1}$. Take j in $\hat{S}$ with $\hat{p}_j - \max(\hat{\theta}_{k+1}, 0) > \varepsilon_j(v) + \bar{\varepsilon}_{\text{out}}$. Every l outside $\hat{S}$ has $\hat{p}_l \le \max(\hat{\theta}_{k+1}, 0)$, because l either stands outside the top k of $\hat{p}$, where $\hat{p}_l \le \hat{\theta}_{k+1}$, or stands inside it with $\hat{p}_l \le 0$. So $p'_j \ge \hat{p}_j - \varepsilon_j(v) > \max(\hat{\theta}_{k+1}, 0) + \bar{\varepsilon}_{\text{out}} \ge \hat{p}_l + \varepsilon_l(v) \ge p'_l$, and $p'_j > 0$. Then j outranks every feature outside $\hat{S}$ in p' and is positive, at most $k - 1$ features can outrank it, and it stands in S'.

Take l outside $\hat{S}$ with $\max(\hat{\theta}_k, 0) - \hat{p}_l > \varepsilon_l(v) + \bar{\varepsilon}_{\text{in}}$. Where $\hat{\theta}_k > 0$, every one of the k largest predicted pre-activations is positive, so $\hat{S}$ holds exactly k features each with $\hat{p}_j \ge \hat{\theta}_k$, and $p'_l \le \hat{p}_l + \varepsilon_l(v) < \hat{\theta}_k - \bar{\varepsilon}_{\text{in}} \le \hat{p}_j - \varepsilon_j(v) \le p'_j$ for each of them, so k features outrank l in p' and l stands outside S'. Where $\hat{\theta}_k \le 0$, the condition reads $-\hat{p}_l > \varepsilon_l(v) + \bar{\varepsilon}_{\text{in}} \ge \varepsilon_l(v)$, so $p'_l \le \hat{p}_l + \varepsilon_l(v) < 0$ and l is not positive, so it stands outside S'. Where one constant L bounds L_v for every write, $\varepsilon_j(v) \le \|e_j\|(L/2)\|v\|^2$, so every membership the band admits under L it admits under the write's own constant as well, and the uniform statement follows. □

Proof of Theorem 3. If $D_A = D_B M$ and $E_A = N E_B$, then $W_A = E_A J D_A = (N E_B) J (D_B M) = N (E_B J D_B) M = N W_B M$. Nothing about J, the masks or the widths enters, so the identity holds at every token and for any two widths, and it asks nothing of a matching.

The two conditions of the masked identity read $P_A N(I - P_B) = 0$ and $(I - Q_B)MQ_A = 0$, so $P_A N = P_A N P_B$ and $MQ_A = Q_B MQ_A$. Under both, $P_A W_A Q_A = P_A N W_B MQ_A = (P_A N P_B)W_B(Q_B MQ_A) = P_A N(P_B W_B Q_B)MQ_A$.

Without the two conditions and under exact representation, $P_A W_A Q_A - P_A N(P_B W_B Q_B)MQ_A = P_A N(W_B - P_B W_B Q_B)MQ_A$. In general the masked difference carries this term beside $P_A(W_A - NW_B M)Q_A$. Since $W_B - P_B W_B Q_B = (I - P_B)W_B + P_B W_B(I - Q_B)$, the term is the transport carried by the weight of N on the downstream atoms of B that $I - P_B$ keeps, plus the transport carried by the weight of M on the upstream atoms of B that $I - Q_B$ keeps, through B's live downstream atoms. A projector $I - P_B$ keeps exactly the atoms that are inactive at the token, and an atom is inactive exactly when its margin is not positive. The term therefore depends on the margins only through their signs.

For the residual, $\Delta E\, JD_A + NE_B J\, \Delta D = (E_A - NE_B)JD_A + NE_B J(D_A - D_B M) = E_A JD_A - NE_B JD_B M = W_A - NW_B M$. Taking norms by the triangle inequality and $\|XYZ\|_F \le \|X\|_F \|Y\|_2 \|Z\|_2$ with its counterpart $\|XYZ\|_F \le \|X\|_2 \|Y\|_2 \|Z\|_F$ gives the bound. □

Proof of Theorem 4. Since p is affine in the downstream activation, $p' - p = E\,\Delta x$ exactly on any block, so $R = \|E\,\Delta x\|_\infty$. Every order statistic of the pre-activations moves by at most R. At least i coordinates of p stand at or above its i-th largest value θ_i and each moves by at most R, so $\theta'_i \ge \theta_i - R$. At least $m - i + 1$ coordinates stand at or below θ_i and end at or below $\theta_i + R$, so $\theta'_i \le \theta_i + R$.

Under the assumption both supports hold k features, so as many leave as enter. An entrant l stood outside the top k, so $p'_l \le p_l + R < \theta_k + R$, and it stands inside after the write, so $p'_l \ge \theta'_k \ge \theta_k - R$ and $|p'_l - \theta_k| \le R$. A leaver j had $p_j \ge \theta_k$ and ends with $p'_j \le \theta'_k \le \theta_k + R$, so $0 \le p_j - \theta_k \le 2R$. The entrant's code rises from zero to p'_l, which is positive, the leaver's falls by p_j, and each of the $k - s$ stable members changes by at most R, so

$$G\,\Delta z = \theta_k\Big(\sum_{l\in I} g_l - \sum_{j\in O} g_j\Big) + \sum_{l\in I}(p'_l - \theta_k)g_l - \sum_{j\in O}(p_j - \theta_k)g_j + \sum_{i\in S\cap S'}(p'_i - p_i)g_i,$$

whose first three terms are $G\,\Delta z_{I\cup O}$. The second and third terms have norm at most $3sR\gamma$ together and the fourth at most $(k - s)R\gamma$, and the triangle inequality gives both bounds. A swap needs $R > 0$, so $\|\Delta x\| > 0$, and dividing gives the ratios. On a block affine along the write, $\Delta x = Jv$ and $R = \|EJv\|_\infty \le \|EJ\|\,\|v\|$.

Let $R < \min(\theta_k, (\theta_k - \theta_{k+1})/2)$. Every member of S ends at or above $\theta_k - R$, which is positive and above $\theta_{k+1} + R$, and every other feature ends at or below $\theta_{k+1} + R$. So the k largest pre-activations after the write are the members of S, all positive, and $S' = S$. Since $\Delta x - G\,\Delta z$ is the change of the reconstruction residual, $\|\Delta x - G\,\Delta z\| \ge \|G\,\Delta z\| - \|\Delta x\|$.

Across two blocks the first block's affinity along the write gives the middle change $\Delta y = J_1 v$ exactly, and the second block's affinity along Δy gives the third layer's activation change $J_2 J_1 v$. The third-layer pre-activations are therefore $p_3 + E_3 J_2 J_1 v$ exactly, which is what one product through both blocks forms, and the code of those pre-activations under the third layer's rule is the exact code by the argument of Theorem 1. A chain that carries $G_2\,\Delta\hat{z}$ forms $p_3 + E_3 J_2 G_2\,\Delta\hat{z}$, which differs from the residual chain's $p_3 + E_3 J_2 \Delta y$ by $E_3 J_2(\Delta y - G_2\,\Delta\hat{z})$. For the exact middle code, $\Delta y - G_2\,\Delta z_2$ is the change of the middle reconstruction residual. The identity is on the third layer's pre-activations, and the third layer's rule applied to the two vectors need not differ by that vector, which is why the chains are read at the third layer's code. □

Proof of Proposition 1. If $D_B c = D_B c'$ for two representations of y on supports of at most s atoms each, then $D_B(c - c') = 0$ with $c - c'$ supported on at most $2s$ columns, which are linearly independent, so $c = c'$. That is the uniqueness theorem of Donoho & Elad (2003) and Gribonval & Nielsen (2003) applied column by column, and its restricted-isometry form is that of Candes & Tao (2005). The same argument on the rows of E_B gives each row of N. For stability, $\|D_B(c - c')\| \le \|D_B c - y\| + \|D_B c' - y\| \le 2\tau\|y\|$. Since $c - c'$ is supported on $S \cup S'$, $\sigma\|c - c'\| \le \|D_B(c - c')\| \le 2\tau\|y\|$. The classical forms of the same bound are those of Donoho et al. (2006) under coherence and Candès (2008) under a restricted isometry. □

Proof of Proposition 2. On an affine block the first-order change of each pre-activation is its exact change, so a feature's own reach is the size of its move. The argument of Theorem 2, with each feature's own reach in place of its error bound, then gives the first two conclusions. Take a live feature j whose own move is under its margin, $-\Delta p_j < p_j - \max(\theta_{k+1}, 0)$. Then $p'_j > \max(\theta_{k+1}, 0) \geq 0$, so j is positive after the write, and if it leaves it leaves by rank, which needs k features above it in p' and so an inactive feature l with $p'_l \geq p'_j$. Then $\Delta p_l - \Delta p_j \geq p_j - p_l \geq p_j - \max(\theta_{k+1}, 0)$, the margin of j, because an inactive l has $p_l \leq \max(\theta_{k+1}, 0)$, and by the same inequality read from the other side $\Delta p_l - \Delta p_j \geq \max(\theta_k, 0) - p_l$, the size of l's margin, since a live j has $p_j \geq \max(\theta_k, 0)$. A live feature whose own move is not under its margin can leave with no partner, by falling to zero. A live feature that keeps its membership changes by $\Delta p_j = e_j^\top J v = e_j^\top \Delta y$, so its decoded change is $g_j e_j^\top \Delta y$, the rank-one operator $g_j e_j^\top$ applied to the activation change. □

B THE DICTIONARIES READ, THE SAMPLES, THE COUNTS AND THE CHECKS

Each public dictionary is read at a pinned revision of its repository, on the model and the model revision its card names, and under the arithmetic of the library that trained it, which subtracts the decoder bias before the encoder and adds it after the decoder (EleutherAI, 2026; Marks et al., 2024). Table 4 gives every revision and licence, and the bibliography entry of each release carries the full commit. The EleutherAI releases on pythia-70m and pythia-160m state a budget of 8.2 billion tokens of the Pile at a context of 2,049, and the deduplicated TopK release states none. The ReLU release's library states 2 billion tokens, and its configuration states 120,000 steps at 16,384 activations per batch and a context of 128, with an ℓ_1 penalty of 0.1.

Table 4: The dictionaries read, in full, one row per pair or configuration. Each row gives the repository the dictionaries were read from and its revision, the model and its revision, the two layers, the rule and the k it keeps at a token, the training budget the release states and the licence the repository states, and the last two rows are the further configurations trained here that the appendices read.

repository	revision	model	model revision	layers	rule	k	budget	licence
EleutherAI/sae-pythia-70m-32k	63999ba8720b	pythia-70m	a39f36b100fe	3, 4	topk	16	8.2 B	mit
EleutherAI/sae-pythia-70m-32k	63999ba8720b	pythia-70m	a39f36b100fe	1, 2	topk	16	8.2 B	mit
EleutherAI/sae-pythia-70m-32k	63999ba8720b	pythia-70m	a39f36b100fe	4, 5	topk	16	8.2 B	mit
EleutherAI/sae-pythia-160m-32k	2046768ae0c8	pythia-160m	50f5173d932e	6, 7	topk	32	8.2 B	mit
EleutherAI/sae-pythia-70m-deduped-32k	7a64bade5972	pythia-70m-deduped	e93a9faa9c77	3, 4	topk	16	none stated	none stated
saprmarks/pythia-70m-deduped-saes	50a434461d36	pythia-70m-deduped	e93a9faa9c77	3, 4	relu	none	2 B	mit
cell F, trained here	none	pythia-70m	a39f36b100fe	3, 4	topk	32	199,999,488	none
narrower, trained here	none	pythia-70m	a39f36b100fe	3, 4	topk	25	100,941,824	none
small block, trained here	none	pythia-70m	a39f36b100fe	3	topk	32, 25	12,288 distinct	none

A revision is the first twelve characters of the commit read. A budget in B is in billions of tokens, and a budget of a dictionary trained here is in activations. The narrower configuration has width 2,048 with 25 features kept, and the small-block configurations of both widths fit their distinct activations over 32,768,000 presentations, at the upstream layer alone, and stand in Appendices C and D.

The readings take the validation file val.jsonl.zst of the corpus repository monology/pile-uncopyrighted at revision 3be90335b66f. Its card names the subsets removed from the collection, Books3, BookCorpus2, OpenSubtitles, YTSubtitles and OWT2. Each model's own tokenizer at its pinned revision reads the documents in file order. A document under 16 characters is skipped, which is the rule of the packer written for the testbed, and each document is followed by the end-of-text token. The rule skipped 2 of the 1,666 documents the readings read, and the skipped documents held 4 tokens. It skipped 83 of the 144,467 documents the testbed's training and its held-out block read, and those held 335 tokens. The stream is cut into sequences of 2,048 tokens, and each sequence gives one position, drawn uniformly from the positions after the first up to position 2,047 of the sequence indexed from zero, from seed 20260921. Both rules and the draw were fixed before any reading. The first 256 sequences are the measurement sample, which chose the transport's sparsity level and nothing else, and the next 1,024 are the held-out sample. The first 512 positions of the held-out sample carry every single-feature write, at 10 doses for the crossing rate and 3 for the predictors. The transport reads all 1,024.

Cell F was trained for this paper on pythia-70m at the layers of its first public pair, with a TopK rule keeping 32 features and an expansion of 32 over the residual stream's width of 512, for 199,999,488 activations for each of 8 runs. Its trainer is this paper's own and follows the recipe of Gao et al. (2025). It streams the same repository's training split in file order under the packer's rules and drops the first position of every packed sequence, the one position that attends to nothing but itself, so a sequence of 2,048 tokens yields 2,047 activations. The first 1,048,576 activations form the block held out before training, on which the mean μ and the scale s of Section 2 are read. Every run shares μ and s, with s at 12.424 at layer 3 and 13.093 at layer 4, so in the gauge of Section 2 every code, pre-activation and cutoff is s times its stored value and a write's norm in cutoff units is 0.949 times its stored ratio. The fraction of variance unexplained is read on the first 65,536 activations of that block at each checkpoint, where Table 1 prints the last. On a second block of 1,048,576 drawn after training it stands from 0.1128 to 0.1131 at layer 3 and from 0.1543 to 0.1559 at layer 4 over the 8 runs.

The loss is the squared reconstruction error summed over coordinates and averaged over the batch, plus one over 32 times an auxiliary term. A latent is dead when it fired on none of the last 10,000,000 activations. None counts as dead before that many were trained. The auxiliary term decodes the positive parts of the 256 largest pre-activations among the dead latents, or of every dead latent where fewer are dead. It fits them to the residual of the live reconstruction,

$$\mathcal{L}_{\text{aux}} = c\,\frac{\sum_b \|G\,a_b - (x_b - \hat{x}_b)\|^2}{\sum_b \|x_b - \bar{x}\|^2},$$

where a_b keeps those pre-activations at row b, $\hat{x}_b$ is the live reconstruction, $\bar{x}$ is its run's batch mean, and both sums run over the rows of the 8 runs trained together. The coefficient c is one for all 8 runs, their summed count of dead latents over 8 times the smaller of 256 and the largest count among them, capped at one, and the term is off where no run holds a dead latent. Table 5 prints c and every run's dead count at the checkpoints where any run held one. The term was off at the first 12 of the 14 checkpoints of each layer, through 50,970,624 activations. At the last two c stood from 0.25 to 0.44, above 0.01 at 2 of the 14. The coupling it carried reached at most 3 of a run's 16,384 latents at a checkpoint, the dead ones, through their encoder and decoder weights and the run's pre-encoder bias. Each run uses Adam at a learning rate of 1.0×10^{-4} with PyTorch's default betas of 0.9 and 0.999 and epsilon of 1.0×10^{-8}, on batches of 4,096 activations. The model's activations are taken in bfloat16 and cast to single precision before centring. Every parameter and optimizer state is in single precision, under PyTorch 2.8.0. Each run permutes each chunk of 1,000,000 collected activations by its own generator and consumes it in batches of 4,096. That generator is seeded at the run's shuffle seed times 1,000,003 plus the layer times 10,007 plus the chunk's index. The rate warms up linearly over 1,000 steps as the sparsify library does at its pinned commit, and it stays constant after the warmup where that library decays (EleutherAI, 2026). Decoder columns are renormalised to unit norm after every step. The encoder starts from a uniform draw, from a generator seeded at the run's initialization seed plus 7,919 times the layer, bounded by the inverse square root of the stream's width and the decoder from its normalised transpose, with both biases at zero. Each run writes a checkpoint at each of 14 targets, one, two and five times each power of ten from 10,000 to 200,000,000 activations, and Appendix D reads them as the 9 budgets the training reached. The six runs of independent seeds take initialization seeds 100 to 105 and shuffle seeds 200 to 205. The retraining pair shares the initialization seed 106 and takes the shuffle seeds 206 and 207. The narrower configuration at width 2,048 with 25 features kept trains on the same stream to 100,941,824 activations, and the small-block configurations of both widths fit 12,288 distinct activations over 32,768,000 presentations. The remote computations behind the paper took 31.8 accelerator-hours, summed over calls that each ran on one NVIDIA A100 80GB, of which 20.8 trained the testbed, 1.7 read the public releases, 5.4 fitted and read the transport and 3.9 read the readout and its audits.

Table 5: The auxiliary coefficient the 8 runs of cell F, the testbed on pythia-70m, shared in training. Each run's count of dead latents, the latents that entered its support at no activation of the last 10,000,000, at the two checkpoints of each layer where any run held one, and the one coefficient the runs shared there, their summed count over 8 times the smaller of 256 and the largest count. At the 12 earlier checkpoints of each layer every count is zero and the auxiliary term is off.

run	layer 3 at 100,941,824	layer 3 at 199,999,488	layer 4 at 100,941,824	layer 4 at 199,999,488
run 0	0	1	0	0
run 1	0	0	0	1
run 2	0	0	2	0
run 3	1	1	3	0
run 4	0	1	0	1
run 5	0	2	0	2
run 6	2	1	0	1
run 7	2	1	1	1
coefficient	0.3125	0.4375	0.2500	0.3750

The block between two layers is the model's next transformer block, from the residual stream after the upstream layer to the residual stream after the downstream one. Its Jacobian is taken at one position with every earlier position's keys and values held. Every exact response and every Jacobian is taken in double precision, on a model loaded in double precision. Before any reading uses the operator it is checked four ways on the first 4 positions of the held-out sample, with 3 random directions each, from seed 1 for the check at the release's single-precision softmax and seed 2 for the three in double precision. The block function is compared with the model's own forward pass at the release's softmax and in double precision, the Jacobian with the Jacobian-vector product through the whole layer, and the Jacobian with a central difference at a step of 1.0×10^{-6}. Each reading is the largest over positions and directions of the Euclidean norm of the difference over the reference's norm, and Table 6 gives each check's tolerance and reading on every pair. On cell F two ways to read one edge were compared as well. The first multiplies the encoder row by the forward-mode product of the decoder column, and the second pulls the encoder row back in reverse mode and multiplies it by the column. They differed by 1.1×10^{-4} against a tolerance of 1.0×10^{-4}, since the block computes its softmax in single precision, and no reading of this paper uses either.

Table 6: The operator checks, the largest relative error each returned on each pair under the tolerance it was held to, read on the first 4 held-out positions of the pinned Pile validation file with 3 random directions each before any reading used the operator. A reading at or under its tolerance held, and lower is better.

dictionaries	model	layers	the function at the release's softmax	the function in double precision	the Jacobian against a product	the Jacobian against a central difference
tolerance			1.0×10^{-10}	1.0×10^{-3}	2.0×10^{-3}	1.0×10^{-5}
TopK release	pythia-70m	3, 4	1.4×10^{-15}	4.8×10^{-5}	2.7×10^{-4}	1.2×10^{-7}
TopK release	pythia-70m	1, 2	9.3×10^{-16}	1.0×10^{-7}	1.8×10^{-6}	2.1×10^{-10}
TopK release	pythia-70m	4, 5	1.2×10^{-15}	1.9×10^{-4}	8.1×10^{-4}	1.2×10^{-7}
TopK release	pythia-160m	6, 7	1.1×10^{-15}	5.0×10^{-6}	1.5×10^{-5}	2.2×10^{-8}
TopK release	pythia-70m-deduped	3, 4	1.5×10^{-15}	4.9×10^{-5}	1.7×10^{-4}	8.4×10^{-8}
ReLU release	pythia-70m-deduped	3, 4	1.5×10^{-15}	4.9×10^{-5}	1.7×10^{-4}	8.4×10^{-8}
cell F	pythia-70m	3, 4	1.4×10^{-15}	4.8×10^{-5}	2.7×10^{-4}	1.2×10^{-7}

Cell F's operator on run 0 carries three more checks. The one-position function agrees with the block in double precision at every one of the 1,280 positions of the measurement and held-out samples, at 2.1×10^{-4} under 1.0×10^{-3}. On the first 4 held-out positions one edge, from the first live upstream feature to the first live downstream one, is read two ways through the release's single-precision softmax, forward and in reverse, and the two readings differ by 1.1×10^{-4} of the Cauchy and Schwarz bound on that edge, the encoder row's norm times the pushed write's, under 1.0×10^{-4}, which failed. The edge the stored double-precision Jacobian gives agrees with both at 1.8×10^{-4} of the same bound under 2.0×10^{-3}.

Table 7 gives every pair's writes and crossings per dose beside its zero responses, which stand at zero at every dose, and Tables 8 and 9 give both graphs' code errors at each dose, per pair and per run of cell F. Each share of Section 3 is the mean over the writes that cross at the zero ablation of a per-write share. For a write whose code change is Δz, with entering set I and leaving set O, the

crossing share is $\|\Delta z_{I\cup O}\|^2/\|\Delta z\|^2$ and the entering share is $\|\Delta z_I\|^2/\|\Delta z\|^2$, where Δz_U keeps the change on the set U.

Table 7: The interventions read on every pair. The single-feature writes at each predictor dose, one on every live upstream feature at each of the 512 positions of the pinned Pile validation file, the writes under which the downstream support changed at each dose, the writes at any of the three doses whose exact response is zero, and at the zero ablation the share of the response's energy on the entering coordinates alone and on the entering and leaving coordinates together, each the mean over the crossing writes. Cell F is the sum over its 8 runs of each count and the mean over the runs of each share. A share is higher where the crossing coordinates carry more of the response.

dictionaries	model	layers	writes per dose	cross at zero	cross at 0.5	cross at 0.9	zero responses at every dose	entering share	crossing share
TopK release	pythia-70m	3, 4	8,192	6,867	5,347	1,765	0	0.33	0.77
TopK release	pythia-70m	1, 2	8,192	6,078	4,553	1,528	0	0.39	0.84
TopK release	pythia-70m	4, 5	8,192	7,251	5,799	1,970	0	0.34	0.75
TopK release	pythia-160m	6, 7	16,384	14,519	11,467	3,853	0	0.33	0.73
TopK release	pythia-70m-deduped	3, 4	8,192	7,171	5,673	2,012	0	0.35	0.80
ReLU release	pythia-70m-deduped	3, 4	44,343	30,404	23,689	9,485	0	0.01	0.05
cell F	pythia-70m	3, 4	131,072	120,462	100,034	38,829	0	0.37	0.79

No position of any pair carried no live upstream feature, so no position was excluded, and no other exclusion applies.

Table 8: The code errors of the fixed-support graph and the margin graph at each predictor dose. The median relative error on the downstream code at the zero ablation, at the doses 0.5 and 0.9, and pooled over the three, on every pair read, over one single-feature write on every live upstream feature at each of the 512 positions of the pinned Pile validation file, with cell F, the testbed on pythia-70m, as the mean over its 8 runs of each run's median. Lower is better.

dictionaries	model	layers	fixed at zero	margin at zero	fixed at 0.5	margin at 0.5	fixed at 0.9	margin at 0.9	fixed pooled	margin pooled
TopK release	pythia-70m	3, 4	0.739	0.121	0.702	0.042	0.017	0.010	0.564	0.033
TopK release	pythia-70m	1, 2	0.803	0.073	0.612	0.036	0.013	0.010	0.247	0.023
TopK release	pythia-70m	4, 5	0.784	0.106	0.780	0.036	0.020	0.010	0.663	0.033
TopK release	pythia-160m	6, 7	0.774	0.032	0.759	0.016	0.007	0.005	0.617	0.013
TopK release	pythia-70m-deduped	3, 4	0.794	0.128	0.781	0.039	0.018	0.010	0.669	0.031
ReLU release	pythia-70m-deduped	3, 4	0.075	0.022	0.027	0.011	0.003	0.002	0.017	0.008
cell F	pythia-70m	3, 4	0.842	0.050	0.866	0.020	0.012	0.006	0.786	0.017

Table 9: The code errors of the fixed-support graph and the margin graph on cell F, run by run. The median relative error on the downstream code at the zero ablation, at the doses 0.5 and 0.9 and pooled over the three, over one single-feature write on every live upstream feature at each of the 512 positions of the pinned Pile validation file, 16,384 writes per dose per run of the testbed on pythia-70m. Lower is better.

run	fixed at zero	margin at zero	fixed at 0.5	margin at 0.5	fixed at 0.9	margin at 0.9	fixed pooled	margin pooled
run 0	0.839	0.050	0.866	0.020	0.013	0.006	0.790	0.017
run 1	0.838	0.052	0.865	0.021	0.012	0.006	0.782	0.017
run 2	0.843	0.051	0.872	0.020	0.013	0.006	0.794	0.017
run 3	0.843	0.050	0.870	0.020	0.013	0.006	0.796	0.017
run 4	0.838	0.051	0.855	0.021	0.012	0.006	0.770	0.017
run 5	0.842	0.048	0.861	0.021	0.012	0.006	0.782	0.017
run 6	0.844	0.047	0.867	0.020	0.012	0.006	0.785	0.016
run 7	0.848	0.049	0.869	0.020	0.012	0.006	0.790	0.017

A bootstrap over the positions, declared before it ran, bounds the two graphs' medians and their difference. It draws the 512 positions with replacement 2,000 times from seed 20260924 and keeps every write at a drawn position once per draw, and Table 10 gives the interval of each statistic. No resample on any pair puts the difference at or under zero.

Table 10: The bootstrap over the positions. The interval of each graph's median relative error on the downstream code and of their difference, the fixed-support graph's less the margin graph's, over 2,000 resamples of the 512 positions of the pinned Pile validation file with every write at a drawn position kept once per draw, pooled over the zero ablation and the doses 0.5 and 0.9, at the 2.5th and 97.5th percentiles, and the share of resamples whose difference stands at or under zero. Cell F, the testbed on pythia-70m, draws the same positions for every run and reads the mean over its 8 runs of each run's median. A lower error and a larger difference are better.

dictionaries	model	layers	fixed support	margin graph	difference	resamples at or under zero
TopK release	pythia-70m	3, 4	0.529 to 0.596	0.0316 to 0.0342	0.496 to 0.564	0.0
TopK release	pythia-70m	1, 2	0.162 to 0.433	0.0219 to 0.0240	0.138 to 0.410	0.0
TopK release	pythia-70m	4, 5	0.639 to 0.686	0.0315 to 0.0340	0.606 to 0.654	0.0
TopK release	pythia-160m	6, 7	0.590 to 0.643	0.0125 to 0.0135	0.577 to 0.630	0.0
TopK release	pythia-70m-deduped	3, 4	0.644 to 0.693	0.0294 to 0.0317	0.612 to 0.663	0.0
ReLU release	pythia-70m-deduped	3, 4	0.0154 to 0.0185	0.0075 to 0.0080	0.0076 to 0.0107	0.0
cell F	pythia-70m	3, 4	0.777 to 0.794	0.0164 to 0.0173	0.760 to 0.777	0.0

The support predictors of Section 4 rank the features inactive before the write. The margin graph is read at its own count, and its ordering at the true count n of entrants ranks those features by their predicted pre-activations. Each predictor below admits the true count of entrants and removes the true count of leavers, which differ under the ReLU rule, whose live count moves. The margin alone lets the inactive features whose margins are smallest in size enter and the live features whose margins are smallest leave. The edge alone lets the inactive features with the largest first-order increase enter and the live features with the largest first-order decrease leave. The null swaps inactive features drawn uniformly at random into the support and live features out of it, from seed 20260921 plus the run on cell F and from seed 20260921 on the releases. The exact oracle re-ranks the exact pre-activations at the true count, and it gives the entering set at a precision and a recall of 1.00 on every pair. The implementation reads a feature live where its pre-activation is positive and at or above the k-th largest, and no exact pre-activation tied at that boundary on any of the 673,701 interventions the oracle read. The readings of the margin alone, the edge alone and the null carry each one's hits beside its predicted and true counts, so its true positives are the hits and its false positives and false negatives are the two counts less the hits. Table 11 gives the confusion counts of all six on the entering set.

Table 11: The confusion counts of six support predictors on the entering set, on every pair read, pooled over the zero ablation and the doses 0.5 and 0.9 at one write on every live upstream feature at each of the 512 positions of the pinned Pile validation file. The exact oracle re-ranks the exact pre-activations and the margin graph is read at its own count. At the true count, of entrants on the entering side and of leavers on the leaving side, the ordering ranks the predicted pre-activations, the margin alone takes the margins smallest in size, the edge alone the largest signed first-order change and the null a uniform draw. Cell F, the testbed on pythia-70m, is the sum over its 8 runs of each count and the mean of the per-run precision and recall.

dictionaries	model	layers	predictor	true positives	false positives	false negatives	precision	recall
TopK release	pythia-70m	3, 4	exact oracle	26,520	0	0	1.000	1.000
			margin graph	22,142	5,007	4,378	0.816	0.835
			at the true count	22,493	4,027	4,027	0.848	0.848
			margin alone	11,369	15,151	15,151	0.429	0.429
			edge alone	681	25,839	25,839	0.026	0.026
			null	1	26,519	26,519	3.8×10^{-5}	3.8×10^{-5}
TopK release	pythia-70m	1, 2	exact oracle	20,557	0	0	1.000	1.000
			margin graph	17,656	2,945	2,901	0.857	0.859
			at the true count	18,139	2,418	2,418	0.882	0.882
			margin alone	10,175	10,382	10,382	0.495	0.495
			edge alone	260	20,297	20,297	0.013	0.013
			null	2	20,555	20,555	9.7×10^{-5}	9.7×10^{-5}
TopK release	pythia-70m	4, 5	exact oracle	28,282	0	0	1.000	1.000
			margin graph	24,202	3,935	4,080	0.860	0.856
			at the true count	24,999	3,283	3,283	0.884	0.884
			margin alone	12,451	15,831	15,831	0.440	0.440
			edge alone	572	27,710	27,710	0.020	0.020
			null	0	28,282	28,282	0.000	0.000
TopK release	pythia-160m	6, 7	exact oracle	56,903	0	0	1.000	1.000
			margin graph	52,151	5,559	4,752	0.904	0.916
			at the true count	52,793	4,110	4,110	0.928	0.928
			margin alone	27,121	29,782	29,782	0.477	0.477
			edge alone	1,766	55,137	55,137	0.031	0.031
			null	4	56,899	56,899	7.0×10^{-5}	7.0×10^{-5}
TopK release	pythia-70m-deduped	3, 4	exact oracle	28,133	0	0	1.000	1.000
			margin graph	23,157	5,751	4,976	0.801	0.823
			at the true count	23,578	4,555	4,555	0.838	0.838
			margin alone	11,946	16,187	16,187	0.425	0.425
			edge alone	530	27,603	27,603	0.019	0.019
			null	2	28,131	28,131	7.1×10^{-5}	7.1×10^{-5}
ReLU release	pythia-70m-deduped	3, 4	exact oracle	105,653	0	0	1.000	1.000
			margin graph	103,002	37,725	2,651	0.732	0.975
			at the true count	100,381	5,272	5,272	0.950	0.950
			margin alone	53,862	51,791	51,791	0.510	0.510
			edge alone	245	105,408	105,408	0.002	0.002
			null	30	105,623	105,623	2.8×10^{-4}	2.8×10^{-4}
cell F	pythia-70m	3, 4	exact oracle	578,104	0	0	1.000	1.000
			margin graph	512,550	75,675	65,554	0.871	0.887
			at the true count	518,182	59,922	59,922	0.896	0.896
			margin alone	254,746	323,358	323,358	0.441	0.441
			edge alone	15,768	562,336	562,336	0.027	0.027
			null	137	577,967	577,967	2.4×10^{-4}	2.4×10^{-4}

The precision is the true positives over the predicted count and the recall the true positives over the true count, each micro-averaged over the writes, and higher is better.

Table 12 gives the same counts on the leaving set, where the margin graph's precision runs from 0.89 to 0.93 and its recall from 0.92 to 0.95 on the five public TopK pairs.

Table 12: The confusion counts of six support predictors on the leaving set, on every pair read, pooled over the zero ablation and the doses 0.5 and 0.9 at one write on every live upstream feature at each of the 512 positions of the pinned Pile validation file. The exact oracle re-ranks the exact pre-activations and the margin graph is read at its own count. At the true count, of entrants on the entering side and of leavers on the leaving side, the ordering ranks the predicted pre-activations, the margin alone takes the margins smallest in size, the edge alone the largest signed first-order change and the null a uniform draw. Cell F, the testbed on pythia-70m, is the sum over its 8 runs of each count and the mean of the per-run precision and recall.

dictionaries	model	layers	predictor	true positives	false positives	false negatives	precision	recall
TopK release	pythia-70m	3, 4	exact oracle	26,520	0	0	1.000	1.000
			margin graph	24,536	2,613	1,984	0.904	0.925
			at the true count	25,091	1,429	1,429	0.946	0.946
			margin alone	14,526	11,994	11,994	0.548	0.548
			edge alone	9,627	16,893	16,893	0.363	0.363
			null	5,016	21,504	21,504	0.189	0.189
TopK release	pythia-70m	1, 2	exact oracle	20,557	0	0	1.000	1.000
			margin graph	18,824	1,777	1,733	0.914	0.916
			at the true count	19,375	1,182	1,182	0.943	0.943
			margin alone	11,546	9,011	9,011	0.562	0.562
			edge alone	6,168	14,389	14,389	0.300	0.300
			null	3,333	17,224	17,224	0.162	0.162
TopK release	pythia-70m	4, 5	exact oracle	28,282	0	0	1.000	1.000
			margin graph	25,916	2,221	2,366	0.921	0.916
			at the true count	26,822	1,460	1,460	0.948	0.948
			margin alone	15,448	12,834	12,834	0.546	0.546
			edge alone	9,899	18,383	18,383	0.350	0.350
			null	5,053	23,229	23,229	0.179	0.179
TopK release	pythia-160m	6, 7	exact oracle	56,903	0	0	1.000	1.000
			margin graph	53,783	3,927	3,120	0.932	0.945
			at the true count	54,476	2,427	2,427	0.957	0.957
			margin alone	28,149	28,754	28,754	0.495	0.495
			edge alone	14,083	42,820	42,820	0.247	0.247
			null	5,637	51,266	51,266	0.099	0.099
TopK release	pythia-70m-deduped	3, 4	exact oracle	28,133	0	0	1.000	1.000
			margin graph	25,806	3,102	2,327	0.893	0.917
			at the true count	26,396	1,737	1,737	0.938	0.938
			margin alone	15,886	12,247	12,247	0.565	0.565
			edge alone	9,374	18,759	18,759	0.333	0.333
			null	5,298	22,835	22,835	0.188	0.188
ReLU release	pythia-70m-deduped	3, 4	exact oracle	110,264	0	0	1.000	1.000
			margin graph	102,358	2,168	7,906	0.979	0.928
			at the true count	107,542	2,722	2,722	0.975	0.975
			margin alone	61,850	48,414	48,414	0.561	0.561
			edge alone	24,232	86,032	86,032	0.220	0.220
			null	9,599	100,665	100,665	0.087	0.087
cell F	pythia-70m	3, 4	exact oracle	578,104	0	0	1.000	1.000
			margin graph	538,639	49,586	39,465	0.916	0.932
			at the true count	545,962	32,142	32,142	0.944	0.944
			margin alone	298,295	279,809	279,809	0.516	0.516
			edge alone	139,116	438,988	438,988	0.241	0.241
			null	71,301	506,803	506,803	0.123	0.123

The precision is the true positives over the predicted count and the recall the true positives over the true count, each micro-averaged over the writes, and higher is better.

Figure 5 draws the precision and the recall of the six predictors on the entering set.

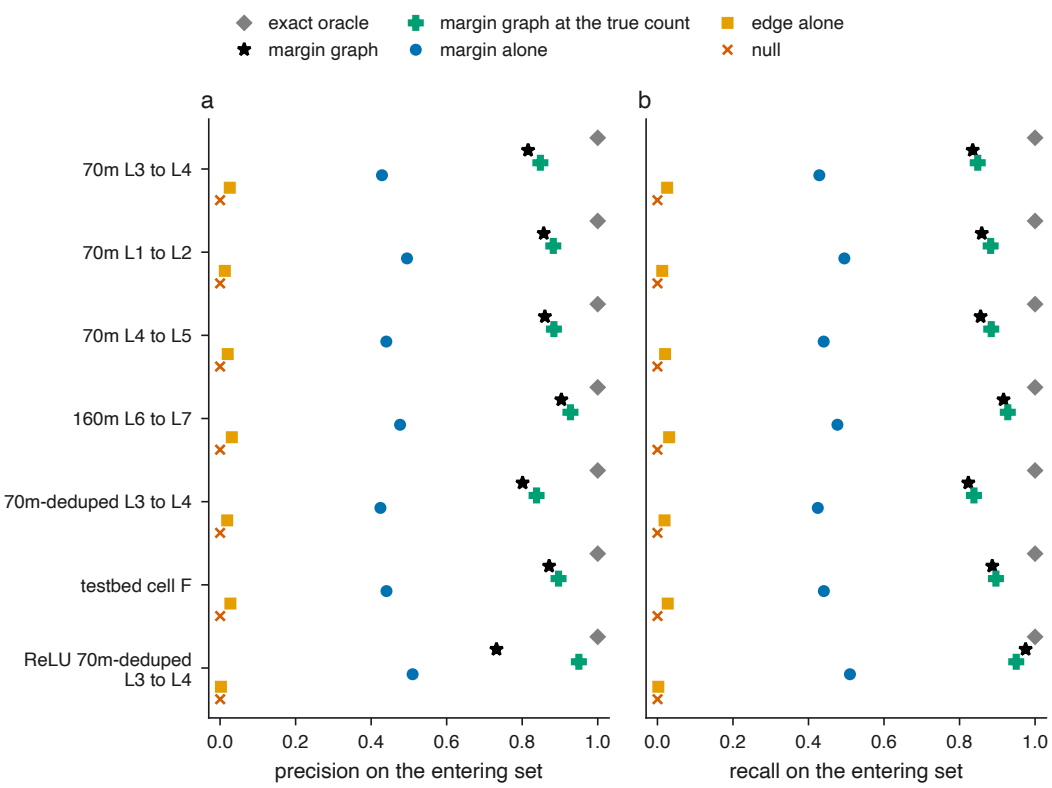

Figure 5: Predicting which features enter the support under a write. The precision in panel a and the recall in panel b, higher being better, on the entering set of six predictors on every pair read, micro-averaged over one single-feature write on every live upstream feature at each of the 512 positions of the pinned Pile validation file and pooled over the zero ablation and the doses 0.5 and 0.9, with cell F, the testbed on pythia-70m, as the mean over its 8 runs. The exact oracle re-ranks the exact post-write pre-activations at the true count of entrants and stands at one. The margin graph is read at its own count, and its ordering of the predicted pre-activations at the true count stands beside it. At the true count the margin alone admits the inactive features whose margins are smallest in size, the edge alone those with the largest signed first-order change and the null a uniform draw. Each predictor is offset on its pair's row so no two markers overprint, and pairs are labelled by Pythia model size and layers.

The field's devices of Section 4 are implemented as follows. Integrated gradients average the first-order term at n points along the write, at the fractions $(s - \frac{1}{2})/n$ for s from one to n, at 5 and 10 products. The softmax device recomputes the attention weights exactly at the written input, through the first layer norm, the query and key projections of the written position with the earlier positions' keys held, the rotary rotation, the scores and the softmax. The exact weights then combine the earlier positions' held values with the position's own value taken at first order. The MLP, which reads the block's input through the second layer norm, is taken at first order by the same Jacobian-vector product along the write, and the output projection applies exactly. Each device is read on the fixed support and re-ranked. The two softmax devices were also read on the five public TopK pairs, over one write on every live upstream feature at each of the 512 positions at the three doses, and Table 13 gives their errors beside both graphs' with each device's products and retained rows.

Table 13: The softmax device beside the two graphs on every TopK pair read. The median relative error on the downstream code, lower being better, at the zero ablation, at the doses 0.5 and 0.9 and pooled over the three, over one single-feature write on every live upstream feature at each of the 512 positions of the pinned Pile validation file, for the fixed-support graph, the margin graph and the softmax device read on the fixed support and re-ranked. The paired median is the median over the writes of the re-ranked device's error less the margin graph's, and the writes lower are the share on which the re-ranked device errs less, both pooled over the doses. Cell F, the testbed on pythia-70m, is the mean over its 8 runs of each run's reading.

dictionaries	layers	device	at zero	at 0.5	at 0.9	pooled	paired median	writes lower
pythia-70m release	3, 4	fixed support	0.739	0.702	0.017	0.564		
		margin graph	0.121	0.042	0.010	0.033		
		softmax, fixed support	0.731	0.699	0.013	0.552		
		softmax, re-ranked	0.072	0.031	0.008	0.024	−0.0017	0.77
pythia-70m release	1, 2	fixed support	0.803	0.612	0.013	0.247		
		margin graph	0.073	0.036	0.010	0.023		
		softmax, fixed support	0.797	0.608	0.010	0.191		
		softmax, re-ranked	0.049	0.025	0.007	0.016	−0.0036	0.95
pythia-70m release	4, 5	fixed support	0.784	0.780	0.020	0.663		
		margin graph	0.106	0.036	0.010	0.033		
		softmax, fixed support	0.777	0.775	0.016	0.654		
		softmax, re-ranked	0.075	0.028	0.008	0.026	−0.0005	0.64
pythia-160m release	6, 7	fixed support	0.774	0.759	0.007	0.617		
		margin graph	0.032	0.016	0.005	0.013		
		softmax, fixed support	0.769	0.753	0.006	0.610		
		softmax, re-ranked	0.027	0.014	0.004	0.011	−0.0004	0.71
pythia-70m-deduped release	3, 4	fixed support	0.794	0.781	0.018	0.669		
		margin graph	0.128	0.039	0.010	0.031		
		softmax, fixed support	0.787	0.777	0.012	0.660		
		softmax, re-ranked	0.058	0.026	0.007	0.021	−0.0023	0.82
cell F	3, 4	fixed support	0.842	0.866	0.012	0.786		
		margin graph	0.050	0.020	0.006	0.017		
		softmax, fixed support	0.839	0.864	0.009	0.783		
		softmax, re-ranked	0.032	0.015	0.005	0.013	−0.0009	0.80

The fixed-support graph and the margin graph take one Jacobian-vector product per live feature at a token, which every dose shares. Each softmax device takes one per write beside a partial forward pass from the first layer norm to the attention softmax. The fixed-support graph and the softmax device on the fixed support retain the live rows of each column, the k of Table 1, and the margin graph and the re-ranked device every row, the width there.

C THE ANATOMY OF THE MAPS, THE FITS AND THE NULLS

Each source vector and each target atom is set to unit norm, so the tolerance is a relative residual and a coefficient is read against a unit atom. The tolerance of 0.436 is the residual a unit vector leaves on one unit atom at a cosine of 0.9. At each step the solver takes the target atom with the largest positive inner product with the current residual and refits the coefficients on every chosen atom by nonnegative least squares. The refit is an accelerated proximal gradient in double precision. Its step is the inverse of 1.01 times the exact largest eigenvalue of the chosen atoms' Gram matrix, and its momentum follows $t' = (1+\sqrt{1+4t^2})/2$ and $y = x' + ((t-1)/t')(x'-x)$, restarted at $y = x'$ and $t = 1$ whenever the objective rises. Each step's refit starts from the previous step's coefficients, with the new atom at its inner product with the residual floored at zero. The refit checks the optimality violation every 25 iterations and stops once it falls to 1.0×10^{-8} or at 1,000 iterations, where the violation is $\max_i \nu_i / \|b\|$. Here G is the chosen atoms' Gram matrix and b their inner products with the vector. The entry ν_i is $|(Gc-b)_i|$ where $c_i > 0$ and $\max(b_i - (Gc)_i, 0)$ where $c_i = 0$. A vector stops at the first step that meets the tolerance or at the cap of 32 atoms, and sooner where no atom lowers its residual. A vector that never meets the tolerance keeps its closest fit. A coefficient above 1.0×10^{-4} counts as an atom, and the count deletes no coefficient.

The fit carries its coefficients back to each dictionary's rows in the gauge of Section 2 by their norms. The decoder columns have unit norm in that gauge, so M is the fitted coefficients themselves. A source encoder row is rebuilt as its own norm times the combination of unit target rows, so $N_{jl} = C_{jl}\,\|e_j^A\|/\|e_l^B\|$, with C the coefficients fitted on unit rows.

The level is the cap on a representation's atoms. It was read at one, two, four, eight, sixteen and thirty-two atoms on the measurement sample, where the mean unmasked explained fraction between seeds stands at 0.246, 0.336, 0.403, 0.463, 0.521 and 0.568. The declared rule chose the smallest level within 0.01 of the best, which is 32. At one atom each source keeps the target atom of largest positive inner product, with no constraint that two sources take different atoms. That level is the unconstrained nearest-atom map, and it explains 0.246 against 0.221 for the one-to-one matching.

The one-to-one matching is an optimal assignment of the source's vectors to the target's atoms that maximizes the summed positive cosine, with a negative cosine counting as zero, and each matched vector takes its atom at the nonnegative projection while an unmatched vector takes none. An ordered pair of distinct runs of independent seeds, source a and target b, takes the index $5a + b$, less one where b exceeds a. The random maps of matched sparsity give each source vector the count of atoms its fitted map carries, on that many target atoms drawn uniformly without replacement, with the coefficients refitted by the same nonnegative least squares, at 20 draws per pair from seed 6,000 plus the pair's index. The permutation null relabels the target's atoms uniformly at random under the fitted maps, independently at each layer, keeping the maps' sparsity and coefficients, at 200 draws per pair from seed 7,000 plus the pair's index. The rotation control rotates the target's atoms by a Haar-random orthogonal matrix of the residual coordinates, which keeps every internal angle of the target and breaks every correspondence with the source. It reads each run a of an independent seed against run $a+1$ modulo six rotated, from seed 3,000 plus a at the upstream layer and 4,000 plus a at the downstream one. The position null of the admission curve permutes the target's positions from seed 20260921, so the target's live sets and margins are read at a permuted position. The admission curve admits, at a share of the inactive atoms, the target's live atoms and the floor of that share times the count of its inactive atoms nearest the boundary, per position and on each side. The random-admission control admits the same count of inactive atoms drawn uniformly without replacement, at 3 draws per pair from seed 20,260,923 plus the pair's index, on the same 1,024 positions.

Table 14 gives every reading of Table 3 run by run, each run of an independent seed as the source of 5 ordered pairs. The interval under each central value of Table 3 takes 2,000 draws from seed 20260925. Draw b leaves out run b modulo six, keeps the ordered pairs among the other five runs and resamples the 1,024 positions with replacement once for every pair, and the retraining pair's value takes the same drawn positions with no run left out. The ends are the 2.5 and 97.5 percentiles of the draws.

Table 14: The transport's readings run by run on cell F, the testbed on pythia-70m, on the 1,024 held-out positions of the pinned Pile validation file. Each value of a run of an independent seed is the mean over the 5 ordered pairs with that run as the source, so the mean over the six runs is the between-seed value of Table 3, and each direction of the retraining pair stands alone. The explained fractions are one less the residual energy over the reference's energy summed over the positions, higher being closer. The gap is the unmasked-target reading less the masked one, larger meaning more lost to the masks, and it is the mask term's energy plus the cross term, twice the signed inner product of the mask term with the representation residual, each over the block's energy, while the residual's own energy, larger meaning a looser fit, cancels in the gap.

reading	run 0	run 1	run 2	run 3	run 4	run 5	retraining, 6 from 7	retraining, 7 from 6
matching, unmasked	0.221	0.221	0.220	0.222	0.221	0.220	0.349	0.349
sparse, unmasked	0.571	0.569	0.569	0.570	0.570	0.570	0.640	0.638
matching, masked	0.335	0.371	0.357	0.370	0.360	0.381	0.431	0.436
sparse, masked	0.445	0.465	0.464	0.471	0.467	0.480	0.506	0.511
sparse, target unmasked	0.833	0.849	0.853	0.851	0.854	0.861	0.844	0.853
gap	0.388	0.384	0.389	0.380	0.387	0.380	0.338	0.342
mask term	0.342	0.335	0.338	0.334	0.339	0.328	0.294	0.289
cross term	0.046	0.048	0.051	0.046	0.048	0.053	0.044	0.053
residual	0.167	0.151	0.147	0.149	0.146	0.139	0.156	0.147

The number of atoms at the first step that meets the tolerance sorts a vector into a class. One atom makes a copy, two to four atoms make a few, five up to the cap make many, and a vector that never meets the tolerance within the cap is unreachable. The first step at which the greedy path meets the tolerance bounds the smallest representation from above, and the class counts that step's coefficients above 1.0×10^{-4}, which can be fewer. Near a coherent target the class also depends on the tolerance,

since an atom equal to the normalized sum of two target atoms is represented exactly by those two and closely by a third atom near their sum, which is the failure of the exact recovery condition of Tropp (2004). Between seeds the upstream decoder columns fall into copies at 0.20, a few atoms at 0.15, many at 0.36 and unreachable at 0.29 (Figure 6). The downstream encoder rows fall into copies at 0.12, a few atoms at 0.05, many at 0.08 and unreachable at 0.76. On the retraining pair the decoder columns are copies in 0.30 of cases and unreachable in 0.21, and the encoder rows in 0.26 and 0.58. A control that rotates the target's atoms represents 0.00 of its source's decoder columns, leaves 1.00 of them unreachable and explains 0.13 of the unmasked operator.

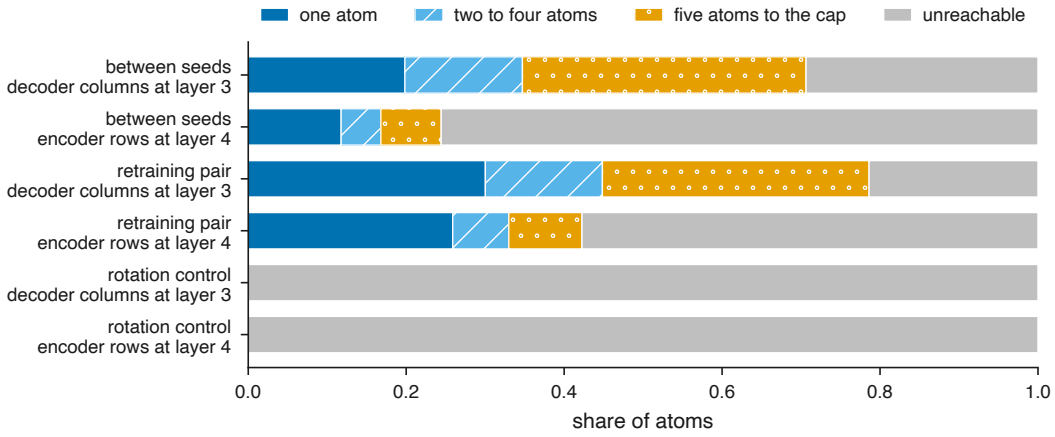

Figure 6: The anatomy of the maps on cell F, the testbed of 8 runs on pythia-70m. Each atom of one dictionary is written as a nonnegative sparse code over the decoder columns at layer 3, or the encoder rows at layer 4, of another, to a relative residual of 0.436 within a cap of 32 atoms, and sorted by how many atoms the code takes when it first meets the tolerance, one, two to four or five to the cap, a larger share of copies meaning more vectors carried by a single atom each. A vector that never meets the tolerance keeps its closest fit and is counted as unreachable. The rotation control rotates the target's atoms by a Haar-random orthogonal matrix of the residual coordinates, which keeps every angle within the target and breaks every correspondence with the source. The shares are drawn over the 30 ordered pairs between the six independent seeds, over the retraining pair's two directions and over the 6 pairs of the rotation control, each as the mean over the pairs in its class.

The maps recombine atoms more often than they copy them. They represent 0.71 of the upstream decoder columns at a median of 5 atoms each and 0.24 of the downstream encoder rows at a median of 2 atoms. Feature by feature, a live downstream feature's decoder partner in the other seed is that seed's decoder column of largest cosine, standing where the cosine reaches 0.9. Over every live feature with a partner at every position, with each pair read in both directions, the partner is active at the same position in 0.84 of cases against 0.08 under the position null. Written over cell F's atoms, a narrower dictionary of width 2,048 trained on the same stream takes two or more atoms for a mean share of 0.855 of its represented columns over its six pairs, which range from 0.852 to 0.860. Between two seeds of cell F at the same budget the mean share is 0.878, ranging from 0.871 to 0.886. The declared rule set the mean across widths against the largest share between seeds, 0.89. The mean across widths also stands under the mean between seeds.

D THE BUDGET CURVE

The nearest-neighbour overlap of two dictionaries at a cosine of 0.9 is the share of one dictionary's decoder columns, among the features that fired at least once in training, whose largest absolute cosine with such a column of the other exceeds 0.9, which lets several columns share one partner. Each pair is read from each of its two dictionaries in turn, and the curve averages the two directions. Between seeds of cell F's configuration at layer 3 the overlap rises with the training budget and stands at 0.198 at 199,999,488 activations, while on the retraining pair, which starts from one initialization, it falls to 0.300 (Figure 7). A narrower configuration at width 2,048 with 25 features kept reaches

0.307 between seeds at 100,941,824 activations. Dictionaries of the two configurations fitted over many passes to 12,288 distinct activations read 3.7×10^{-5} and 0.0034 between seeds. A nearest-neighbour overlap is therefore a reading of the training budget as well as of the model, and the maps of Section 6 are read at cell F's full budget. A checkpoint target is read at the first chunk boundary at or past it, and a chunk trains 999,424 activations, its 1,000,000 cut to whole batches. So the 6 targets below one chunk are read together at the first boundary and each larger target at the first boundary past it, with the last read at the end of training. The 14 targets thus give the 9 budgets the curve reads.

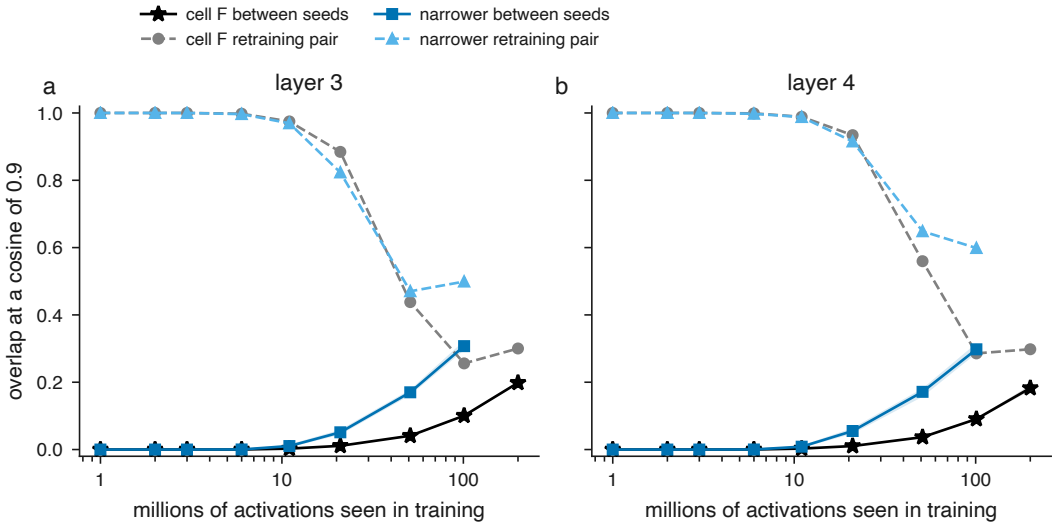

Figure 7: The budget curve. The nearest-neighbour overlap at a cosine of 0.9 between two dictionaries' decoder columns, higher meaning more shared directions, against the activations seen in training, at the upstream layer in panel a and the downstream layer in panel b, for cell F's configuration and the narrower one at width 2,048 with 25 features kept, both trained on pythia-70m. The overlap reads the columns of features that fired at least once in training at absolute cosine, lets several columns share one partner and averages the two directions. Between seeds is the mean over the 15 unordered pairs of the six independent seeds with their range shaded, beside the retraining pair. The 9 budgets run from 999,424 to 199,999,488 activations, of which the narrower configuration reached 100,941,824. The horizontal axis is logarithmic.

E THE DECLARATIONS

Each rule below fixes the quantity it reads and the threshold at which it fires before the reading it governs. Table 15 sets each beside what the reading returned.

Table 15: The rules declared before the readings they govern. Each rule's name, what it reads, when it fires, what the reading returned and whether it fired.

rule	quantity read	fires when	reading	fired
G1	crossing rate at the zero ablation, each public TopK pair	under 0.1	0.74 to 0.89	no
G2	energy share of the crossing coordinates, each public TopK pair	under 0.1	0.73 to 0.84	no
G2	the same on the ReLU pair	under 0.1	0.05	yes
C2	the ReLU pair's energy share	over 0.5	0.05	no
G3	the margin graph's error, each public TopK pair	not under 0.75 of the fixed-support graph's, which is 0.423, 0.186, 0.498, 0.462 and 0.502 on the pairs in the order of Table 1	0.013 to 0.033	no
F5	the margin graph's error, cell F	not under 0.75 of the fixed-support graph's, which is 0.590	0.017	no
F6	runs whose regressor on six inputs, the pre-activation, its first-order change and the two order statistics at the cut-off before and after the change, has a median error inside the margin graph's interquartile range	5 runs or more	0 runs	no
F7	energy share of the crossing coordinates, cell F	under 0.1	0.79	no
F8	the margin graph's precision and recall over the margin alone's and the edge alone's, cell F	a lead under 0.1	precision 0.87 and recall 0.89 against 0.44 and 0.027 for both	no
B1	memberships inside the band under the run's largest constant that flip	any	0	no
band rule	accepted memberships the exact pre-activations contradict on the scoring half, every population	any	0	no
M1	the once-built graph on subsets against the fixed-support graph	not under 0.75 of the fixed-support graph's error at a size	not under at the largest sizes	yes
P1	the path predictor at 5 products, re-ranked, on subsets	not under 0.75 of the fixed-support graph's error at a size	at most 0.0043 on cell F and 0.124 on the release	no
P2	the ratio of the two regime boundaries at the level 0.3	over 2.0	1.66	no
R1	the entering-and-leaving part, each TopK pair and cell F	norm ratio under 0.5 or share over 0.5	1.24 to 1.72 and 0.21 to 0.29	no
R2	the ReLU pair's decoded change over every intervention	share under 0.5 or ratio outside 0.50 to 1.25	0.38 and 0.64	yes
H1	the decoded margin chain against the decoded fixed-support chain	not under 0.75 of the fixed-support chain's error	0.999 against 0.884	yes
H2	the residual chain	over 0.663, which is 0.75 of the fixed-support chain's error, or over 0.114, which is 4.0 times the exact activation change chain's	0.053	no
H3	the projection of the decoded change beyond one write's reach of the cutoff above the change within it	a difference under 0.2	0.16	yes
F1	the sparse maps' gain over one-to-one matching, unmasked	under 0.19, a quarter of what the matching leaves	0.35	no
F2	pairs whose maps beat every one of 20 random draws and 200 null draws	any pair short	30 of 30	no
F3	maps of at most 4 atoms at the median that represent half the vectors, the decoder columns at layer 3 and the encoder rows at layer 4	a median atom count over 4 or a represented share under 0.5 on either family	a median of 5 atoms on the decoder columns and a represented share of 0.24 on the encoder rows	yes
F4	the mean multi-atom share over the six pairs from the narrower configuration into cell F against the largest share among the pairs between cell F's seeds	at or under	0.85 against 0.89	yes
C1	the rotation control's represented share of the decoder columns at layer 3, and its unmasked explained fraction	at or over half the between-seed represented share, or at or over the one-to-one matching's fraction	0.00 and 0.13	no
K1	the nearest-margin admission curve at a share of 0.1 of the inactive atoms	under 0.5 of the gap	0.73	no

2 rules fired on the ReLU pair, where they read the control both ways. The other 5 that fired read the once-built graph on the largest subsets, the decoded chains, the mediation and the maps' atom counts. Section 4 states the result M1 reads, and Section 5 those of H1 and H3. Section 6 states those of F3 and F4. The supplementary material carries every threshold beside the code that read it.

The regressors of F6 read what the margin graph reads and nothing more. The six-input regressor predicts each coordinate's code change from that coordinate's own six inputs with one set of weights

shared across coordinates. The inputs are the pre-activation before the write, its first-order change, the k-th and the k plus first largest pre-activations before the write, and the same two order statistics of the predicted pre-activations. The two-input regressor reads the first two of these. The margin graph is one function of the same six inputs, so the rule reads whether a fitted learner reaches that function within its declared budget. Each regressor is a multilayer perceptron of two hidden layers of 64 units with rectifiers, trained per run on 4,096 interventions drawn uniformly from the first 256 positions at the three predictor doses, every coordinate of each a training row, 67,108,864 rows in all. The inputs are standardized by the training rows' mean and standard deviation, and each regressor minimizes a mean squared error by Adam at a learning rate of 1.0×10^{-3} and the default betas and epsilon of Appendix B on batches of 65,536 rows for 2 passes, 2,048 steps. The draw of interventions and the order of the training rows take seed 20260921 plus the run, and the networks' weights start from the global generator seeded at 20260921 before the first run and drawn from in run order. Each layer takes PyTorch's default initialization, its weights and biases drawn uniformly within the inverse square root of its input width. On the scoring half the six-input regressor's median error runs from 2.23 to 2.45 over the runs and the two-input regressor's from 1.44 to 1.53. Both stand above one, the error of predicting no change, against a mean over the runs of 0.016 for the margin graph's median on the same half (Table 16).

Table 16: The fitted regressors against the margin graph on cell F, run by run. The relative error on the downstream code over the scoring half, the last 256 of the 512 positions of the pinned Pile validation file at the zero ablation and the doses 0.5 and 0.9, as the margin graph's quartiles and each regressor's median, where predicting no change errs by exactly one. The six-input regressor reads a coordinate's pre-activation, its first-order change and the two order statistics at the cutoff before and after the change, and the two-input regressor the first two. Cell F is the testbed on pythia-70m. Lower is better.

run	margin graph first quartile	margin graph median	margin graph third quartile	six-input regressor	two-input regressor
0	0.006	0.016	0.054	2.37	1.47
1	0.006	0.017	0.057	2.45	1.53
2	0.007	0.017	0.057	2.44	1.47
3	0.006	0.016	0.056	2.36	1.45
4	0.006	0.017	0.056	2.42	1.49
5	0.006	0.016	0.054	2.38	1.48
6	0.006	0.016	0.053	2.23	1.44
7	0.006	0.016	0.056	2.35	1.51

The constant of Theorem 2 is read on 2,048 zero-ablation writes per run, 4 live upstream features drawn without replacement at each of the 512 positions from seed 20260921 plus the run. Its median runs from 15.0 to 15.8 over the 8 runs, and its largest value from 283 to 552. Under the run's largest constant 267 of 524,288 predicted live memberships stand inside the band over the runs, and none of them flipped, which the rule $\mathrm{B1}$ reads. The subsets are drawn 16 per size at each position from seed 20260921 plus the run, each of distinct features, with the draws independent. On cell F's first run, at 1,024 writes per size under each write's own constant, the share of memberships inside the band falls from 0.230 on the subsets of two features to 1.2×10^{-5} on the subsets of eight and 6.0×10^{-8} on the subsets of sixteen (Figure 8b). No membership inside the band flips at any size, and every first-order remainder stands within its own bound at every size, a share of 1.00 on cell F's first run and on the release.

The regime of Figure 8a bins the write's norm geometrically from 0.0625 to 64 cutoff units at 2 bins per doubling. A bin under 20 writes stands empty, and the median error is read in each bin. The boundary at a level is interpolated linearly in the logarithm of the bin centres between the last bin below the level and the first at or above it.

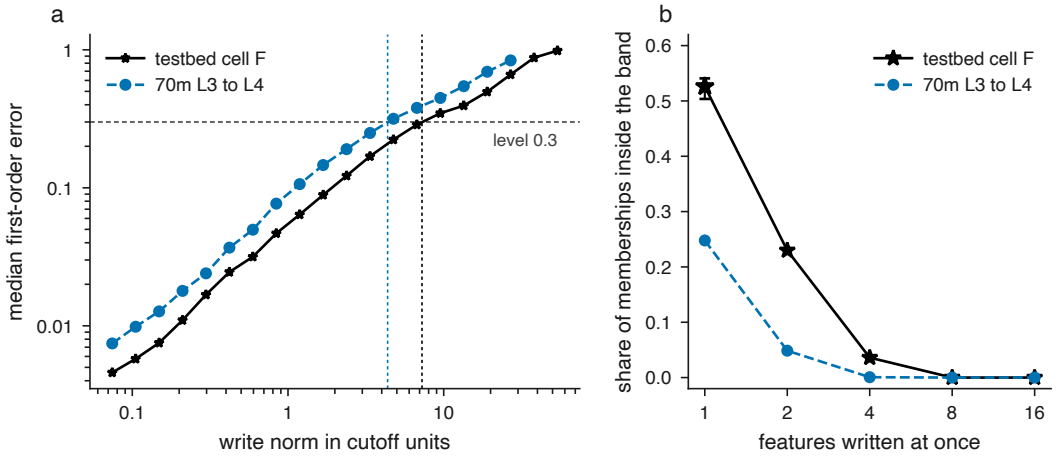

Figure 8: The regime of the first-order term and the band. Panel a draws the median first-order error of the downstream pre-activations against the write's norm in units of the downstream cutoff, over single-feature writes and subset writes together, on cell F with its 8 runs pooled at 655,360 writes and on the pythia-70m release at layers 3 and 4 at 57,344, with both axes logarithmic and the dotted lines marking where each curve first reaches 0.3. Panel b draws the share of a write's predicted memberships, in or out over every feature of the dictionary, that stand inside the band of Theorem 2 under each write's own constant, read from three Jacobians and so a check after the fact, against the number of features written at once on an axis logarithmic in base two, on cell F and on the same release. Cell F's single writes stand as the mean over its runs with their range as the bar, its subset points read run 0 at 1,024 writes per size, and the release's single-feature point reads 4 sampled features at each of the 256 positions of the scoring half the band rule below reads. Both panels read the 512 positions of the pinned Pile validation file.

The band rule of Section 4 is scored on every population it reads in Table 17. Its constant is the three-point constant above, and the regime boundary of each run and each pair is read on the calibration half of the positions, one permutation from seed 20260925 putting 256 of the 512 positions in each half, with the unit the calibration half's median cutoff. No accepted membership is wrong on any population, and the band alone on the same writes, with no regime condition, admits no wrong membership either.

Table 17: The rule of the band and the regime on writes held out from its setting. For each run of cell F, the testbed on pythia-70m, and each public TopK pair, the boundary at the level 0.3 is the write's norm in cutoff units, in the gauge of Section 2, set on the calibration half of the 512 positions of the pinned Pile validation file, and every other column is read on the other 256 positions. The held-out error is the median first-order error of the pre-activations over the scored writes inside the boundary. A write inside it has a membership accepted where the band of Theorem 2 holds under its own constant. The coverage is the accepted memberships over the writes times the width, and the live coverage the accepted memberships the write predicts live over the memberships it predicts live, both higher being better, and a wrong membership is one the exact pre-activations contradict. Single writes are zero ablations of 4 live upstream features drawn at each position, and subsets the first two of the 16 drawn per size at sizes of two, four, eight and sixteen.

dictionaries	layers	writes	boundary	held-out error	scored	inside	coverage	live coverage	wrong
cell F, run 0	3, 4	single	5.56	0.039	1,024	957	0.512	0.181	0
cell F, run 1	3, 4	single	5.59	0.038	1,024	947	0.503	0.170	0
cell F, run 2	3, 4	single	5.75	0.039	1,024	956	0.525	0.186	0
cell F, run 3	3, 4	single	5.57	0.038	1,024	933	0.541	0.173	0
cell F, run 4	3, 4	single	5.84	0.038	1,024	938	0.518	0.187	0
cell F, run 5	3, 4	single	5.77	0.038	1,024	952	0.524	0.174	0
cell F, run 6	3, 4	single	5.39	0.038	1,024	945	0.520	0.181	0
cell F, run 7	3, 4	single	5.70	0.038	1,024	935	0.519	0.174	0
cell F, run 0	3, 4	subsets	7.73	0.189	2,048	1,131	0.060	0.020	0
cell F, run 1	3, 4	subsets	7.46	0.185	2,048	1,083	0.066	0.020	0
cell F, run 2	3, 4	subsets	7.64	0.187	2,048	1,136	0.067	0.023	0
cell F, run 3	3, 4	subsets	7.65	0.183	2,048	1,102	0.061	0.018	0
cell F, run 4	3, 4	subsets	7.63	0.186	2,048	1,135	0.063	0.022	0
cell F, run 5	3, 4	subsets	7.51	0.182	2,048	1,086	0.058	0.018	0
cell F, run 6	3, 4	subsets	7.50	0.183	2,048	1,110	0.065	0.020	0
cell F, run 7	3, 4	subsets	7.52	0.186	2,048	1,099	0.062	0.019	0
pythia-70m release	3, 4	single	3.98	0.064	1,024	852	0.248	0.068	0
pythia-70m release	1, 2	single	5.14	0.042	1,024	983	0.710	0.242	0
pythia-70m release	4, 5	single	3.22	0.054	1,024	908	0.321	0.076	0
pythia-160m release	6, 7	single	9.60	0.026	1,024	1,005	0.706	0.287	0
pythia-70m-deduped release	3, 4	single	3.68	0.064	1,024	836	0.211	0.055	0
pythia-70m release	3, 4	subsets	4.60	0.221	2,048	555	0.012	0.004	0

Table 18 scores the same rule at the doses 0.5 and 0.9 on the same 4 features per position of the scoring half, under the same boundaries. No accepted membership is wrong at any dose, and the coverage rises with the dose, from 0.21 to 0.71 at the zero ablation to 0.97 to 1.00 at the dose 0.9.

Table 18: The rule of Table 17 at three doses. For each run of cell F, the testbed on pythia-70m, and each public TopK pair, the same 4 live upstream features drawn at each of the 256 positions of the scoring half are written at the zero ablation and at the doses 0.5 and 0.9, 39,936 writes in all, under their population's calibration boundary of Table 17. The coverage is the accepted memberships over the writes times the width and the live coverage the accepted memberships the write predicts live over the memberships it predicts live, both higher being better. Inside counts the writes inside the boundary out of the 3,072 a population carries, and a wrong membership is one the exact pre-activations contradict.

dictionaries	layers	coverage at zero	coverage at 0.5	coverage at 0.9	coverage pooled	live coverage pooled	inside pooled	wrong pooled
cell F, run 0	3, 4	0.512	0.787	0.995	0.765	0.481	2,996	0
cell F, run 1	3, 4	0.503	0.785	0.991	0.760	0.469	2,978	0
cell F, run 2	3, 4	0.525	0.796	0.995	0.772	0.487	2,991	0
cell F, run 3	3, 4	0.541	0.793	0.991	0.775	0.471	2,961	0
cell F, run 4	3, 4	0.518	0.789	0.994	0.767	0.486	2,971	0
cell F, run 5	3, 4	0.524	0.805	0.995	0.775	0.477	2,988	0
cell F, run 6	3, 4	0.520	0.808	0.993	0.774	0.486	2,983	0
cell F, run 7	3, 4	0.519	0.793	0.990	0.767	0.474	2,968	0
pythia-70m release	3, 4	0.248	0.613	0.982	0.614	0.350	2,870	0
pythia-70m release	1, 2	0.710	0.889	1.000	0.866	0.537	3,031	0
pythia-70m release	4, 5	0.321	0.627	0.972	0.640	0.342	2,935	0
pythia-160m release	6, 7	0.706	0.858	0.989	0.851	0.575	3,052	0
pythia-70m-deduped release	3, 4	0.211	0.579	0.984	0.591	0.321	2,861	0

Table 19 gives the errors by subset size behind the rules M1 and P1.

Table 19: The code errors by subset size. The median relative error on the downstream code under a zero ablation of a random subset of the live upstream features, 16 subsets per size at each of the 512 positions of the pinned Pile validation file, for the fixed-support graph, the once-built margin graph and the path predictor with a re-ranking at 5 and at 10 products per subset, on cell F, the testbed on pythia-70m, as the mean over its 8 runs of each run's median and on the pythia-70m release at layers 3 and 4 at its own median. The release keeps 16 features at every position read, so a subset of sixteen is its whole live set and the sixteen draws at a position repeat one intervention. Lower is better.

features written	cell F fixed support	cell F once built	cell F path at 5	cell F path at 10	release fixed support	release once built	release path at 5	release path at 10
2	0.770	0.287	0.0002	4.2×10^{-5}	0.685	0.347	0.0004	0.0001
4	0.676	0.371	0.0005	0.0001	0.613	0.425	0.0011	0.0003
8	0.590	0.405	0.0012	0.0003	0.578	0.498	0.0034	0.0006
16	0.551	0.458	0.0043	0.0007	0.631	0.670	0.1237	0.0191

The rule H3 asks which middle features mediate the path. Write v for the write, Δy for the exact middle activation change, $\Delta\hat{y} = J_1 v$ for its first-order change and $\Delta\hat{z}$ for the margin graph's middle code change. Let i^* be the token's live upstream feature of largest code, and let $\rho_j = |u_{i^*}\, e_{2,j}^{\top} J_1 d_{i^*}|$, with $e_{2,j}$ the middle encoder's row, be the reach of its zero ablation at middle feature j. A middle feature stands within reach when the size of its margin is at most ρ_j, and beyond it otherwise. The parts are

$$\Delta y_{\mathrm{b}} = G_2(\Delta\hat{z} \odot \mathbf{1}_{\mathrm{beyond}}), \qquad \Delta y_{\mathrm{w}} = G_2(\Delta\hat{z} \odot \mathbf{1}_{\mathrm{within}}), \qquad \Delta\hat{y} - \Delta y_{\mathrm{b}} - \Delta y_{\mathrm{w}},$$

the last the remainder, and each part's share is its inner product with Δy over $\|\Delta y\|^2$. Over every two-hop write at the three doses, the rule asked the median share of the part beyond to exceed that of the part within by 0.2. The part beyond projects at 0.30 and the part within at 0.14, a difference of 0.16, so the rule fired. Carried alone through the second block, the part beyond errs at the third layer by 0.92, the part within by 1.02 and the remainder by 1.06. Figure 3c draws the chains behind H1 and H2.