CROSS-LAYER TRANSCODERS AS HYPOTHESIS GENERATORS FOR CAUSAL CIRCUIT ANALYSIS IN MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-layer transcoders (CLTs) replace transformer MLP computation with sparse features that can be assembled into prompt-specific attribution graphs. These graphs make candidate mechanisms easier to inspect, but a large attribution score does not by itself establish the effect of intervening on a feature. We study this distinction in Gemma-3-4B-IT. We introduce a latent-mixing CLT whose decoder uses $17.4\times$ fewer parameters than a direct cross-layer parameterization, and train it to reconstruct all 34 MLP outputs with $100{,}800$ features per layer; it is sparse and transfers across five held-out distributions, with reconstruction MSE between 0.0059 and 0.0083. Direct-effect and multi-hop attribution graphs recover compact hypotheses for factual retrieval and arithmetic completion, which we refine using prompt replacements and feature ablations. Recurrence across AIME reasoning prompts then identifies `L22:F31850`. Intervening on it at sentence-ending periods reveals a bidirectional sentence-boundary effect: amplification lowers the generated newline rate from 65.8% to 24.4% and promotes continuations led by `Since` and `We`, while reversal has the opposite effect. The edit substantially changes solution layout, but its effect on correctness is heterogeneous across problems. Our results support CLT attribution as a method for generating mechanistic hypotheses, with explicit intervention required to determine their causal and behavioral meaning.

1 INTRODUCTION

Mechanistic interpretability seeks explanations of language-model behavior in terms of internal computation rather than input–output correlation alone. Transformer components provide a natural computational graph, but not necessarily the right explanatory units: attention heads and MLP blocks are coarse, while individual neurons can participate in multiple unrelated features because information is represented in superposition (Elhage et al., 2021; 2022; Bricken et al., 2023). Sparse dictionary methods address this problem by learning feature directions that are more interpretable than the model's native basis (Templeton et al., 2024b). For circuit analysis, transcoders are especially useful because they reconstruct a component's output from its input, turning an MLP computation into sparse feature activations and decoder writes (Dunefsky et al., 2024).

Cross-layer transcoders extend this idea by allowing one feature to contribute to multiple downstream MLP outputs, reducing the layer-by-layer duplication that can obscure persistent computations (Lindsey et al., 2024; Ameisen et al., 2025). Once the model is expressed in this feature basis, attribution graphs can connect input tokens, active features, and output logits in a readable prompt-specific circuit. However, these graphs are constructed from a local replacement model in which attention patterns, normalization factors, and feature masks are frozen. Their edge weights and multi-hop path sums therefore describe influence within that approximation; they need not equal the change produced by an isolated intervention in autoregressive generation. This motivates the central question of this work: how should CLT attribution be used to discover causal mechanisms without treating attribution scores themselves as causal effects?

We study this question in Gemma-3-4B-IT. First, we introduce a parameter-efficient latent-mixing CLT that factorizes each source-to-output decoder direction into a feature-specific cross-layer coefficient and an output-layer base direction. At our scale, this reduces the decoder from 153.54 billion to 8.83

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.

billion parameters while retaining cross-layer feature identities. We then construct direct-effect and truncated Neumann attribution graphs. In factual retrieval and arithmetic completion, the graphs provide candidate circuits that can be narrowed using controlled prompt replacements and tested by feature ablation. These examples establish a workflow in which graph structure determines where to look, while response to controlled input changes and interventions determines how the selected nodes should be interpreted.

We next apply the same workflow to a feature that recurs across AIME reasoning prefixes. Attribution recurrence, training-corpus activation examples, and a fixed-prefix intervention select `L22:F31850`; per-token activations then identify sentence-ending periods as its natural intervention site. Scaling the feature during generation shifts probability bidirectionally between newline tokens and a restricted family of prose continuations. This local effect persists in sampled continuations and changes the global layout of solutions. Its accuracy effect, however, varies substantially by problem, separating reliable control of generation form from uniform improvement in task performance.

In summary, we contribute a latent-mixing CLT for Gemma-3-4B-IT that supplies cross-layer sparse features at a $17.4\times$ reduction in decoder parameters; a direct-effect and multi-hop attribution procedure with factual and arithmetic case studies that pair graph-based candidate selection with prompt replacement and ablation; and a generation-time intervention study showing that a recurrently attributed feature causally controls sentence-boundary continuation and solution layout while producing heterogeneous effects on mathematical accuracy.

2 Related Work

Sparse features and transcoders. Sparse autoencoders and related dictionary-learning methods decompose dense model activations into features that are often more semantically coherent than individual neurons (Bricken et al., 2023; Templeton et al., 2024b). Automated descriptions based on maximally activating examples provide one way to characterize such features at scale (Bills et al., 2023). Whereas an autoencoder reconstructs an activation state, a transcoder predicts the output of a model component from its input, making its sparse features units of computation rather than only units of representation (Li et al., 2023; Dunefsky et al., 2024; Templeton et al., 2024a). Crosscoders and CLTs further share features across layers so that information persisting through the residual stream need not be rediscovered in an independent dictionary at every depth (Lindsey et al., 2024; Ameisen et al., 2025). Our latent-mixing parameterization targets the memory cost of direct cross-layer decoding by sharing an output-layer direction across source layers while retaining featurewise cross-layer coefficients.

Circuit discovery and causal validation. Transformer circuit analysis represents behavior through interactions among internal components (Elhage et al., 2021; Wang et al., 2023). Automated methods use ablation, attribution patching, or learned edge masks to identify compact subgraphs (Conmy et al., 2023; Syed et al., 2023; Bhaskar et al., 2024; Yu et al., 2024). Attribution graphs built from CLT replacement models instead expose prompt-specific interactions among sparse features and have been used to study factual recall, planning, multilingual processing, and reasoning (Ameisen et al., 2025; Lindsey et al., 2025). Across these approaches, a candidate circuit must be distinguished from evidence that the circuit is functionally faithful. Ablations can trigger compensatory computation (McGrath et al., 2023), and a score computed in a local approximation need not equal a component's marginal effect in the original model. We therefore use attribution to rank hypotheses, then use prompt replacements, feature interventions, and autoregressive cache propagation to evaluate the proposed mechanism.

Causal interventions on transformer activations and parameters can also localize or alter specific associations (Meng et al., 2022). Our goal differs from knowledge editing: rather than a permanent parameter change or a single corrected fact, we intervene on a sparse feature only where it is naturally active and follow the effect from the immediate next-token distribution through sampled continuations, solution layout, and final-answer accuracy.

3 CROSS-LAYER TRANSCODER FORMULATION

3.1 DESIGN GOAL AND LATENT-MIXING PARAMETERIZATION

Our goal is to replace the MLPs of Gemma-3-4B-IT with sparse feature dictionaries that satisfy three requirements: they should reconstruct the frozen model's MLP computation, represent a feature's downstream contributions across layers without introducing a new copy at every depth, and remain tractable at model scale.

Standard transcoders. Let $\mathbf{x}_{\ell,t} \in \mathbb{R}^d$ and $\mathbf{m}_{\ell,t} \in \mathbb{R}^d$ denote the input and output of the MLP at layer ℓ and token position t. A transcoder (Templeton et al., 2024a; Li et al., 2023; Dunefsky et al., 2024) encodes the input into a sparse feature vector $\mathbf{z}_{\ell,t} \in \mathbb{R}^M$ and decodes it into an approximation of the MLP output:

$$\mathbf{z}_{\ell,t} = f_\ell(\mathbf{W}_\ell^{\text{enc}}\mathbf{x}_{\ell,t} + \mathbf{b}_\ell^{\text{enc}}), \qquad \widehat{\mathbf{m}}_{\ell,t} = \mathbf{W}_\ell^{\text{dec}\top}\mathbf{z}_{\ell,t} + \mathbf{b}_\ell^{\text{dec}}. \tag{1}$$

Here $M \gg d$, $\mathbf{W}_\ell^{\text{enc}}, \mathbf{W}_\ell^{\text{dec}} \in \mathbb{R}^{M\times d}$, and f_ℓ is a sparsifying nonlinearity whose concrete form is specified with the training implementation. Equation 1 learns an independent dictionary at every layer. A feature that persists through the residual stream may therefore be represented by multiple layer-local copies, obscuring whether an attribution path reflects persistence or a new computation (Lindsey et al., 2024).

Direct cross-layer transcoders. A cross-layer transcoder (CLT) instead allows a feature encoded at source layer s to contribute to the reconstructed MLP outputs of layer s and all later layers (Ameisen et al., 2025). In the direct parameterization,

$$\widehat{\mathbf{m}}_{\ell,t} = \sum_{s\le\ell}\sum_{a=1}^{M} z_{s,t,a}\,\mathbf{w}_a^{s\to\ell} + \mathbf{b}_\ell^{\text{dec}}, \tag{2}$$

where $\mathbf{w}_a^{s\to\ell} \in \mathbb{R}^d$ is an independent decoder direction for feature a and source–output pair (s,ℓ). This avoids layer-local duplication, but its decoder requires $\frac{1}{2}L(L+1)Md$ parameters. At the scale of our trained model ($L = 34$, $d = 2560$, and $M = 100{,}800$), this is 153.54 billion decoder parameters.

Latent mixing. Our parameterization first mixes source-layer activations in a shared feature space and then applies one base decoder per output layer. For each output layer ℓ,

$$\widetilde{\mathbf{z}}_{\ell,t} = \sum_{s\le\ell} \boldsymbol{\gamma}^{s\to\ell} \odot \mathbf{z}_{s,t}, \qquad \boldsymbol{\gamma}^{s\to\ell} \in \mathbb{R}^M, \tag{3}$$

$$\widehat{\mathbf{m}}_{\ell,t} = \mathbf{W}_\ell^{\text{dec}\top}\widetilde{\mathbf{z}}_{\ell,t} + \mathbf{b}_\ell^{\text{dec}}, \qquad \mathbf{W}_\ell^{\text{dec}} \in \mathbb{R}^{M\times d}. \tag{4}$$

Here $\odot$ denotes elementwise multiplication. Equivalently, latent mixing factorizes every direct decoder vector as

$$\mathbf{w}_a^{s\to\ell} = \gamma_a^{s\to\ell}\mathbf{w}_{\ell,a}^{\text{dec}}. \tag{5}$$

The output-layer direction $\mathbf{w}_{\ell,a}^{\text{dec}}$ is shared across source layers, while the feature-specific scalar $\gamma_a^{s\to\ell}$ sets the strength and sign of the contribution from layer s. Features can therefore learn distinct cross-layer profiles, but one feature's contributions into a fixed output layer must share a direction.

The latent-mixing decoder contains $LMd + \frac{1}{2}L(L+1)M = 8.83$ billion parameters at the same scale, a $17.4\times$ reduction from the direct parameterization. This saving is a structured approximation rather than a free compression: for a fixed feature and output layer, different source layers may change the magnitude and sign of the write but not its direction.

3.2 TRAINING AND INTRINSIC VALIDATION

Training. We train the CLT to reconstruct all MLP outputs of the frozen Gemma-3-4B-IT model (Gemma Team, Google DeepMind, 2025). Each training step samples B token instances and minimizes the per-coordinate mean-squared error over all L layers,

$$\mathcal{L} = \frac{1}{BLd}\sum_{t=1}^{B}\sum_{\ell=1}^{L}\|\widehat{\mathbf{m}}_{\ell,t} - \mathbf{m}_{\ell,t}\|_2^2. \tag{6}$$

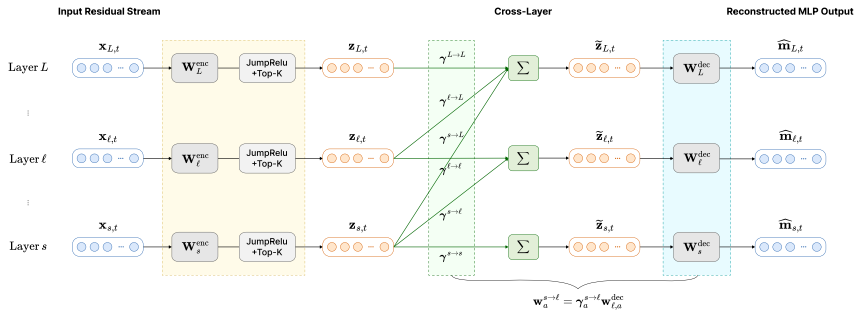


Figure 1: Latent-mixing CLT architecture. Layer-specific encoders produce sparse feature activations, featurewise coefficients mix contributions from source layers, and one base decoder per output layer reconstructs the corresponding MLP output.

Prompts are drawn from StackExchange questions parsed from the uncopyrighted Pile (Gao et al., 2020), and the frozen Gemma-3-4B-IT writes each answer in its own chat format (Gemma Team, Google DeepMind, 2025). The resulting corpus contains $6{,}398{,}447$ question–answer pairs and roughly 5.5 billion content tokens. We mask padding and repeated chat-template tokens from the reconstruction loss. We instantiate f_ℓ in Equation 1 as a JumpReLU with threshold $\theta = 0.025$, followed by a global top-k cap with $k = 500$. The analyzed CLT contains $M = 100{,}800$ features per layer and reaches a training MSE of 0.0064 after $50{,}000$ steps. Full optimization and systems details are provided in Appendix A.

Table 1: Held-out CLT reconstruction.

Evaluation set	MSE	Rel. L_2
StackExchange	0.0059	0.30
AIME 2024–2025	0.0060	0.29
NaturalReasoning	0.0063	0.31
Medical-o1	0.0082	0.36
WikiQA	0.0083	0.35

Intrinsic validation. We assess whether the trained CLT provides an adequate sparse surrogate for downstream circuit analysis along three axes. First, reconstruction transfers beyond the StackExchange training distribution: across five held-out evaluation sets, MSE ranges from 0.0059 to 0.0083 and relative L_2 error from 0.29 to 0.36 (Table 1). Second, activations are sparse. On held-out StackExchange data, each token activates an average of 58.6 of the $100{,}800$ features per layer, a density of 0.06%; the median layer activates only 10.4 features per token. Third, inspection of maximally activating OpenWebMath passages (Paster et al., 2023) shows that many features admit coherent descriptions spanning lexical, formatting, and reasoning patterns. We report the per-layer reconstruction and sparsity profiles, the feature-labeling protocol, and representative activation examples in Appendix A.

These diagnostics establish that the CLT is sparse and reconstructs MLP outputs consistently across distributions. Whether quantities derived from this surrogate identify causal mechanisms is a separate question, which we evaluate through intervention in the following sections.

4 Attribution as Hypothesis Generation

We next use the trained CLT to turn a model prediction into a feature-level circuit hypothesis. Attribution identifies candidate features and paths, while controlled prompt replacements and targeted local interventions characterize what the selected features respond to. Section 5 then applies the same discovery-to-intervention logic to a recurring generation behavior.

4.1 Direct Effects and Multi-Hop Attribution

For a fixed prompt and target logit, we construct an attribution graph whose nodes are input tokens, active CLT features, and the output logit. An edge records the *direct effect* of a source node on a downstream node before influence is routed through any other feature. We compute these effects in a locally linearized replacement model: the transformer's MLPs are replaced by the trained CLT, while attention probabilities, RMSNorm scale factors, and active-feature masks are frozen at their clean forward-pass values. Gradients are blocked from traversing other CLT encode–decode paths when a direct edge is measured.

Let $\mathbf{J}_{i\to j}$ be the transformer Jacobian mapping a residual-stream perturbation at source node i to the read location of target feature j. The direct feature-to-feature effect is

$$g_{i\to j} = \mathbf{w}_j^{\text{enc}} \mathbf{J}_{i\to j} \left(z_i \mathbf{w}_i^{\text{dec}}\right). \tag{7}$$

Read from right to left, the source activation scales its decoder write, the Jacobian propagates that write through the frozen computation, and the target encoder reads out the resulting perturbation. Token and logit endpoints replace the decoder or encoder with the corresponding embedding or unembedding direction. Collecting all admissible pairwise effects gives a prompt-specific adjacency matrix $\mathbf{G}$, with $(\mathbf{G})_{j,i} = g_{i\to j}$.

Direct edges capture only one-step influence. Partitioning $\mathbf{G}$ by token (T), feature (F), and logit (L) endpoints separates terminating logit edges from edges that can propagate onward:

$$A = \begin{pmatrix} \mathbf{G}_{L\leftarrow T} & \mathbf{G}_{L\leftarrow F} \end{pmatrix}, \qquad B = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{G}_{F\leftarrow T} & \mathbf{G}_{F\leftarrow F} \end{pmatrix}. \tag{8}$$

The zero token rows encode that tokens can start a path but cannot be reached from another node. The contribution through exactly k intermediate features is AB^k, giving the truncated Neumann score

$$S_n = A \sum_{k=0}^{n} B^k = A + AB + AB^2 + \cdots + AB^n. \tag{9}$$

We use $n = 10$ throughout; on the analyzed graphs, the final retained term is more than four orders of magnitude smaller than the direct term. Full endpoint definitions, cross-layer Jacobian expansions, and convergence diagnostics are given in Appendix B.

4.2 Attribution Graphs and Prompt-Replacement Case Studies

To turn attribution scores into testable circuit hypotheses, we retain the 100 active features with the largest positive S_{10} scores for each prompt–target pair. Figure 10 places these features by token and layer and connects them with positive direct effects from $\mathbf{G}$.

We use prompt replacement to characterize candidate inputs. At each screened position, we replace the relevant entity, relation, or operand with matched alternatives, rerun an unmodified forward pass, and compare feature activation at the corresponding position. We retain candidates whose activation tracks the replaced item rather than the shared prompt frame. For the arithmetic example, we then follow these screened inputs through the frozen local replacement model to recover additional feature-to-feature hops. We apply this workflow to factual retrieval (`France` $\to$ `Paris`) and arithmetic completion ($42 + 7 = 49$).

Factual retrieval: France $\to$ Paris. *Candidate selection.* For `The capital of France is` $\to$ `Paris`, we first collect features at the literal `capital`, literal `France`, and final `is` positions whose aggregated Neumann attribution exceeds 0.01. This produces 1,034 graph candidates.

France input. Applying the prompt-replacement test at the `France` position leaves three layer-0 candidates. Among them, `L0:F26582` is the clearest: across 24 France sentences, 24 other-country controls, and 120 unrelated sentences, `France` is its maximally activating token in $23/24$ France contexts, and sentence-level maxima separate France sentences from unrelated text with AUC 0.972. Because unrelated text can still activate it weakly, we use the narrow provisional label *literal-France feature*, rather than a France-only detector.

Capital input. Applying the same prompt-replacement test at the `capital` position with seven alternative relations yields six candidates. Among them, `L0:F81670` combines strong direct and

multi-hop attribution to `Paris` with activation that peaks on `capital` in all 24 capital-reference contexts.

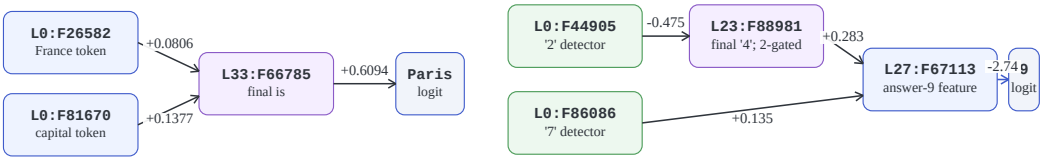
Shared mediator. Searching for a downstream feature that receives positive edges from both retained inputs and emits one to the target logit, the strongest candidate is `L33:F66785` at the final `is`, with incoming weights $+0.0806$ and $+0.1377$ and an outgoing weight of $+0.6094$ to `Paris`. Those incoming edges sit at the 97.46th and 94.03rd percentiles of their sources' positive outgoing edges, and at the 98.29th and 99.43rd percentiles of positive edges entering the mediator. Attribution and screening therefore yield the graph-supported hypothesis `L0:F26582`/`L0:F81670` $\rightarrow$ `L33:F66785` $\rightarrow$ `Paris`.

Arithmetic completion: $42 + 7 = 49$. *Candidate selection.* For `calc:` `(4+17)*2+7` = `21*2+7` = `42+7` = `4` $\rightarrow$ `9`, we apply the graph-to-trace procedure to the final `2` and `7` operand positions and the final `4` prediction position. We screen operand inputs, an intermediate prediction-position hop, and the answer feature separately as described below.

Operand inputs. At each operand position, we zero the token embedding and retain the 20 CLT features with the largest activation decrease, giving 40 candidates. Sweeping that operand over all ten digits leaves seven candidates that activate on the original digit but are inactive on the other nine. Among them, `L0:F44905` at `2` and `L0:F86086` at `7` have the largest frozen-model effects on the downstream answer feature for their respective operands. Ablating `L0:F44905` reduces `L23:F88981` at the final `4` from 1.203 to 0.728. Both input features also connect directly to `L27:F67113`: their ablations change its activation by $+0.131$ and $+0.135$, respectively.

Prediction-position hop. Among nodes changed by the operand-feature ablations, `L23:F88981` is the only prediction-position node in the layer-22–26 computation block. Ablating it increases `L27:F67113` from 8.953 to 9.237, identifying a suppressive path from the `2` input to the prediction position. Together, these measurements support the compact trace shown in Figure 2b.

Answer-`9` feature. We exclude the two `calc:` prefix positions, rank the remaining 43,404 active features by absolute S_{10} attribution, and ablate the top 60 individually. `L27:F67113` has the largest measured relative-margin effect. Across controlled prompts it activates strongly when an arithmetic completion has answer digit `9` (6.47–12.79 over 20 prompts), but is nearly silent for other arithmetic answers (0–0.34 over six prompts). Ablating it reduces the `9` relative margin by 2.74.



(a) `France` $\rightarrow$ `Paris` (b) `42+7` $\rightarrow$ `9`

Figure 2: Compact traces for the two case studies. In (a), edge labels are signed direct-effect weights. In (b), labels report the relative change in the downstream feature after ablating the source; the final edge reports the change in the `9` relative logit margin.

5 Recurrent Attribution and Sentence-Boundary Intervention

We next ask whether a feature that recurs across attribution graphs also produces a consistent causal effect. Our discovery set is six AIME reasoning prefixes for which the model assigns substantial probability to a connective continuation (`However`, `Since`, or `So`). Attribution recurrence forms the candidate set, training-corpus activation examples supply independent semantic evidence, and the behavioral interpretation is reserved for intervention.

5.1 Feature and Intervention-Site Selection

For each discovery prefix we run Neumann attribution toward its connective logit and retain the top 100 features by S_{10} score. Identifying features by layer and index while ignoring token position, and

keeping those that recur in at least five of the six lists, yields five candidates. Recurrence alone does not determine what a candidate represents, so we compare two further signals: automated labels from training-corpus top activations, and a fixed-prefix scale intervention on one discovery example.

Feature	Rec.	Activation label	Representative top activation	Pred.	Obs.
`L22:F31850`	5/6	Mathematical logic/proof	`7.41`: cyclic free modules whose generating element may not form a basis	0.707	0.210
`L29:F60066`	5/6	Math exercises	`4.78`: a textbook exercise asking students to sketch a family of functions	1.800	0.192
`L32:F53761`	6/6	General academic	`3.36`: a mathematics-department seminar calendar	1.636	0.089
`L25:F20384`	6/6	STEM Q&A	`7.72`: a physics-forum question about vertical circular motion	0.351	−0.064
`L7:F20827`	6/6	Conversational openings	`9.06`: an informal invitation beginning "We're talking about organizing and maps"	0.504	−0.027

Table 2: Recurrent candidates with independent semantic evidence and fixed-prefix effects. Labels and examples come from training-corpus top activations; the leading number is the passage's peak activation. Predicted and observed columns give the `However` logit slope per unit feature activation.

For an active feature value z_f, the intervention sets $z'_f = s_f z_f$, where s_f is a dimensionless scale factor. The clean baseline is $s_f = 1$; $s_f = 0$ ablates the feature, $s_f > 1$ amplifies it, and $s_f < 0$ reverses its contribution. On the AIME 2024 Problem 5 discovery sentence, we freeze attention probabilities, RMSNorm scale factors, and active-feature masks at their clean values, then measure the finite-difference `However` logit slope per unit change in z_f. Table 2 compares this observed slope with the Neumann-predicted slope. We select `L22:F31850` because its top activations are the most specifically associated with mathematical proof and logical exposition, and because it has the largest positive observed-to-predicted ratio among the recurrent candidates. Under this protocol the response scales linearly with the edit, but Neumann scores do not predict causal magnitude: none of the observed-to-predicted ratios approaches one, and two Neumann-positive features have negative measured slopes. This discrepancy is not a nonlinearity artifact, since it persists even in the frozen linear regime. The mismatch is structural: the Neumann score credits a feature with the entire multi-hop path sum toward the logit, while the measured slope perturbs that feature alone. When negative routes dominate at the perturbed position, the marginal effect can point opposite to the aggregate score. Attribution scores are rankings, not causal magnitudes, and must be checked by intervention.

Inspection across the six discovery traces shows that `L22:F31850` is concentrated at sentence boundaries rather than on a particular connective. Its mean activation on sentence-ending period tokens is 3.1–8.6× its mean on other tokens, with a pooled ratio of 5.3× (Figure 3). This observation defines the intervention site without reference to the next token: during autoregressive generation, we trigger whenever the decoded token ends in a period and the feature has positive baseline activation at that position.

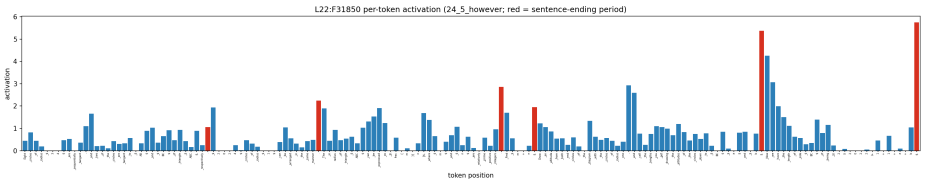


Figure 3: Per-token activation of `L22:F31850` across the AIME 2024 Problem 5 discovery sentence. Each bar is the activation at one token position; red bars mark tokens whose decoded text ends in a period. Figure 9 shows a second discovery sentence.

5.2 Generation-Time Intervention and Results

We evaluate all 30 problems from AIME 2024 and all 30 from AIME 2025. For each problem and setting, we sample 128 solutions at temperature 1.0 with a 2,048-token budget under the unedited model, amplification ($s_f = 3$), and reversal ($s_f = -1$). Comparisons use the 7,141 problem–sample triplets for which all three settings produce a solution.

Next-token probabilities. Each intervened period contributes a paired clean/edited measurement on the same prefix, so the change is an isolated effect of that edit. Each setting is compared against its own clean baseline. Amplification reduces the combined probability of `\n` and `\n\n` by 29.8 points, from 54.9% to 25.0% (Figure 11). The largest increases are `Since` (+14.4 points) and `We` (+10.6), with smaller gains for `So`, `Let`, `In`, and `However`. Reversal raises combined newline probability by 10.3 points, from 75.0% to 85.3%, while `Since` and `We` fall by 1.8 and 3.0 points. Because `Then`, `Also`, and `Thus` decrease under amplification, the effect is not a uniform increase in sentence-initial words.

Generated continuations. The probability shifts persist under sampling (Figure 4). Amplification reduces the generated newline share from 65.8% to 24.4%; `Since` rises from 2.8% to 18.3%, `We` from 4.6% to 17.1%, and the pooled share of `So`, `Let`, `In`, and `However` from 4.0% to 14.8%. Reversal raises the newline share to 83.1% and lowers these continuation groups to 0.1%, 0.5%, and 0.4%, respectively.

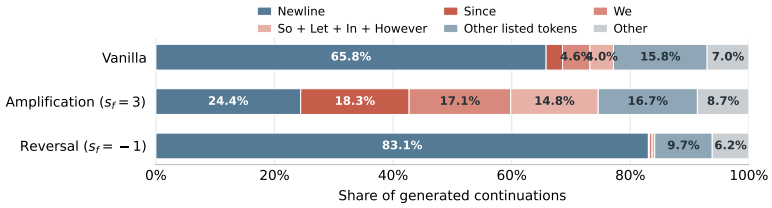


Figure 4: Tokens generated after sentence-ending periods in the 7,141 matched solutions. "Other listed tokens" combines `Then`, `The`, `This`, `Thus`, `$`, and `Also`.

Solution layout. Accumulated over a solution, the same shift changes its line structure. Amplification reduces the mean number of lines from 37.6 to 20.0 and increases line length from 40.5 to 81.8 tokens; reversal moves both measures in the opposite direction (Table 3).

Setting	Samples	Tokens/solution	Lines/solution	Newlines/1k	Tokens/line
Unedited	7,141	1,526	37.6	27.8	40.5
$s_f = 3$	7,141	1,634	20.0	14.5	81.8
$s_f = -1$	7,141	1,490	44.2	32.8	33.7

Table 3: Line statistics over the matched AIME 2024 and 2025 solutions.

Final-answer accuracy. Accuracy effects are modest in aggregate and heterogeneous across problems (Figure 5). Amplification raises AIME 2024 Problem 12 from 17.3% to 41.3% and Problem 2 from 70.3% to 82.8%, but lowers Problem 23 by 4.7 points; on Problem 23, reversal instead raises accuracy from 18.8% to 27.3%. Thus the intervention reliably changes how solutions continue, but no setting is best on every problem.

A representative trajectory-level effect. Table 4 examines that Problem 12 gain at the level of a single trajectory. At the displayed period the edit moves the top next-token prediction from a line break to `Since`, and the sampled continuation then performs the hours-to-minutes conversion that a representative unedited rollout omits. The clean and edited probabilities are paired on an identical prefix inside the amplified rollout; the unedited text is a separate rollout, included as an illustrative failure rather than a common-prefix counterfactual.

6 Conclusion

The intervention isolates a direct behavioral role for `L22:F31850`: at sentence-ending periods, increasing its activation suppresses newline and stop behavior and promotes a restricted set of connective continuations, especially `Since` and `We`, while reversing it does the opposite. Its effect on correctness is conditional rather than uniformly beneficial. Continuing the argument helps when

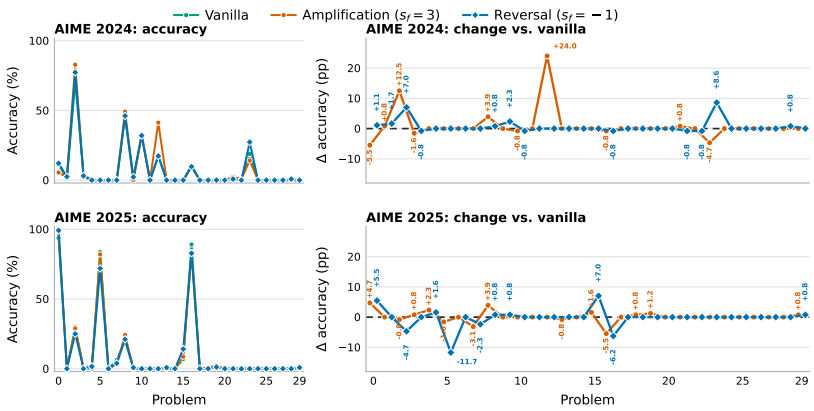


Figure 5: Per-problem AIME accuracy. Left: absolute accuracy under the three settings. Right: amplification and reversal minus the unedited baseline in percentage points.

Table 4: AIME 2024 Problem 12 case study. Amplification raises matched-sample accuracy from 17.3% to 41.3% (+24.0 points) and supplies an omitted unit conversion.

Measured prefix: "... The time taken at $s + \frac{1}{2}$ km/h is $\frac{9}{3} = 3$ hours."

Clean distribution; unedited failure

At the measured prefix, top-1 is `\n` (p=0.93); `Since` has p=0.03. A separate unedited rollout (sample 104, incorrect) reads:

"... The walk time is $\frac{9}{3} = 3$ hours. ↩
The total time is $3 + 24 = 27$ minutes."

Final Answer: $\boxed{27}$: hours were not converted.

Edited distribution and rollout (s_f=3)

At the same prefix, top-1 is `Since` (p=0.41), while `\n` falls to 0.37; the rollout samples `Since`:

"... The time taken at $s + \frac{1}{2}$ km/h is $\frac{9}{3} = 3$ hours. **Since** the coffee shop takes $t = 24$ minutes, the total time taken is $3 \times 60 + 24 = 180 + 24 = 204$ minutes."

Final Answer: $\boxed{204}$: the continuation performs the conversion.

the unedited model breaks or concludes before a necessary step, as in AIME 2024 Problem 12, but can hurt when a problem benefits from committing to an answer. A large and reproducible behavioral effect therefore yields only a modest aggregate accuracy change and heterogeneous per-problem effects. The operative distinction is between controlling *how* the model continues its reasoning and improving *whether* it reaches the correct answer: this feature causally steers the former and is not a general-purpose accuracy knob.

The case studies make the same methodological point. Attribution ranked `L22:F31850` into a five-feature candidate set that no activation statistic alone would have produced, and per-token activations located its intervention site; yet the Neumann score overstated the measured logit slope for every candidate in Table 2. CLT attribution is thus well suited to deciding where to intervene and poorly suited to predicting what the intervention will do, and the two should be reported as separate quantities rather than one as an estimate of the other.

Limitations. Three limitations bound these conclusions. First, the surrogate is weakest exactly where much of the interesting computation sits: per-layer relative L_2 error is inverted-U-shaped, reaching 0.67 at layer 13 and 0.49 at layer 22, the layer of the analyzed feature, so roughly half of mid-network MLP output energy is unexplained (Appendix A). Second, all results come from one model, one CLT checkpoint, and a six-prompt discovery set, and the sentence-boundary mechanism was characterized at a single trigger type. Third, we report accuracy under a fixed 2,048-token budget and a fixed sampling temperature; because amplification lengthens solutions, part of the per-problem accuracy variation may reflect budget interaction rather than reasoning quality.

REPRODUCIBILITY STATEMENT

Section 3.2 and Appendix A give the CLT objective, sparsification, optimizer settings, hardware, corpus composition, and training budget. Appendix B defines every attribution-graph endpoint, the direct-effect expression actually implemented, and the Neumann truncation diagnostics. The evaluation protocol for Table 1 (sequence length, top-k, and the question-hash filter used to construct held-out sets) and the intervention protocol of Section 5 (trigger rule, cache propagation, sample count, temperature, and token budget) are specified in place. The trained CLT, the attribution-graph artifacts for all prompts analyzed here, and the generation and scoring code will be released.

THE USE OF LARGE LANGUAGE MODELS

Large language models enter this work in four distinct roles, which we state separately.

As the object of study. The frozen Gemma-3-4B-IT is the system analyzed throughout. Its weights are never updated, and every activation, attribution score, ablation result, and accuracy figure reported here is a measurement taken on that model by the authors' own code.

As the generator of the training and evaluation corpora. The CLT is trained to reconstruct the frozen model's MLP outputs on chat-formatted data whose answers are written by that same frozen model (Section 3.2); the held-out and transfer evaluation sets of Table 1 are constructed the same way (Appendix A). This data is therefore model-generated by construction, which is the point of the surrogate: the reconstruction target is the frozen model's own computation on its own outputs.

As a component of the feature-labeling method. Following Bills et al. (2023), a language model was prompted to describe what a feature's maximally activating examples have in common, producing the activation labels in Table 2 and Table 5; the protocol is given in Appendix A. These labels are descriptive summaries of activation evidence only. We report them alongside the raw top-activating passages so that readers can check them directly, and no causal claim in this paper rests on a label: the selection of `L22:F31850` is settled by the intervention measurements, not by its label.

As a writing aid. The authors used an LLM-based assistant for prose editing and LaTeX formatting.

We did not use language models for the remaining tasks on the disclosure list: no theoretical model or conceptual framework was developed with LLM assistance, no mathematical claim was formulated or proved with it, and no translation was involved. We have reviewed all AI-assisted work in this paper and take full responsibility for its final content, including all text, claims, and artifacts.

REFERENCES

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

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning, 2024.

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

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary

learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 2023.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. *arXiv preprint arXiv:2406.11944*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling, 2020. Copyright-filtered variant: monology/pile-uncopyrighted.

Gemma Team, Google DeepMind. Gemma 3 technical report, 2025. Base model: google/gemma-3-4b-it.

Max Li, Samuel Marks, and Aaron Mueller. dictionary_learning. GitHub repository, 2023. https://github.com/saprmarks/dictionary_learning.

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. 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, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Thomas McGrath, Matthew Rahtz, János Kramár, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text, 2023.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery, 2023.

Adly Templeton, Joshua Batson, Adam Jermyn, and Chris Olah. Predicting future activations. *Transformer Circuits Thread*, 2024a. https://transformer-circuits.pub/2024/jan-update/index.html#predict-future.

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, 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*, 2024b. https://transformer-circuits.pub/2024/scaling-monosemanticity/.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Lei Yu, Jingcheng Niu, Zining Zhu, and Gerald Penn. Functional faithfulness in the wild: Circuit discovery with differentiable computation graph pruning, 2024.

A Additional CLT Training and Validation Details

Loss masking. We exclude chat-template tokens and padding from the reconstruction loss of Equation 6. Their repetition makes them trivially reconstructable and heavily over-represented, so including them rewards template scaffolding rather than content computation. Concretely, every occurrence of <start_of_turn>, <end_of_turn>, and the role labels user and model is masked, along with padding. These tokens still pass through the frozen model normally and are masked only when the loss is computed.

Optimization and systems. Training pairs are extracted with forward hooks on the frozen Gemma-3-4B-IT's MLPs: each unmasked token contributes the residual input $\mathbf{x}_{\ell,t}$ and MLP output $\mathbf{m}_{\ell,t}$ across all layers. Feature activations are sparsified by a JumpReLU with threshold $\theta = 0.025$, followed by a global top-k gate that keeps the $k = 500$ largest features per token and layer; "global" here means across the tensor-parallel shards, not across layers. We train the analyzed $100{,}800$-feature latent-mixing CLT with AdamW at a constant learning rate of 10^{-4}, weight decay 10^{-2}, PyTorch-default betas, and seed 42, with $B = 64$ token instances per step. The run uses 8 NVIDIA A100 80 GB GPUs, one rank generating activations and seven training, at 1.96 steps per second. The reconstruction loss falls from 0.911 at initialization to 0.0064 by step $50{,}000$ (Figure 6), about 7.1 hours; with the loss converged we stop there.

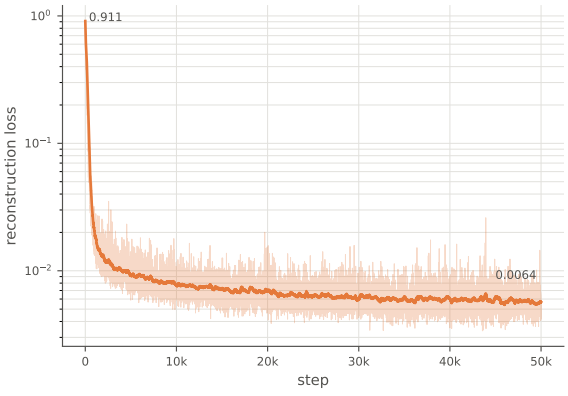


Figure 6: Training trajectory of the analyzed CLT.

Held-out evaluation protocol. Table 1 evaluates at sequence length $2{,}048$ with top-$k = 500$, matching training. Held-out StackExchange questions (200 pairs) are drawn from corpus shards absent from the training split and verified to have zero normalized-question-hash overlap with the full $6{,}398{,}447$-pair corpus; the AIME row uses 60 pairs and the three transfer rows 50 pairs each, each filtered against the same hash set. Answers for every evaluation set are generated by the same frozen model under the generation configuration used to build the training corpus.

Reconstruction quality varies with depth. The pooled relative L_2 error reported in Table 1 is energy-weighted and therefore dominated by the high-norm boundary layers. Per layer, the error $\|h_\ell - \hat{h}_\ell\|_2 / \|h_\ell\|_2$ is inverted-U-shaped: it rises from 0.15 at layer 0 to 0.67 at layer 13, then falls to 0.28 at layer 33, with 14 of 34 layers above 0.5 and layer 22 at 0.49 (Figure 7). The peak coincides with a trough in the activation norm itself ($\|h_\ell\|_2 \approx 43$ at layer 0, 2.5 at layer 13, 27 at layer 33), so the middle layers are the thinnest rather than the worst reconstructed in absolute terms. All five evaluation distributions trace the same curve to within 0.05 over layers 0–21, so this is a property of the CLT rather than of any evaluation set. Mid-network, the surrogate leaves roughly 45% of MLP output energy unexplained, which is the quantitative form of the first limitation stated in the Conclusion.

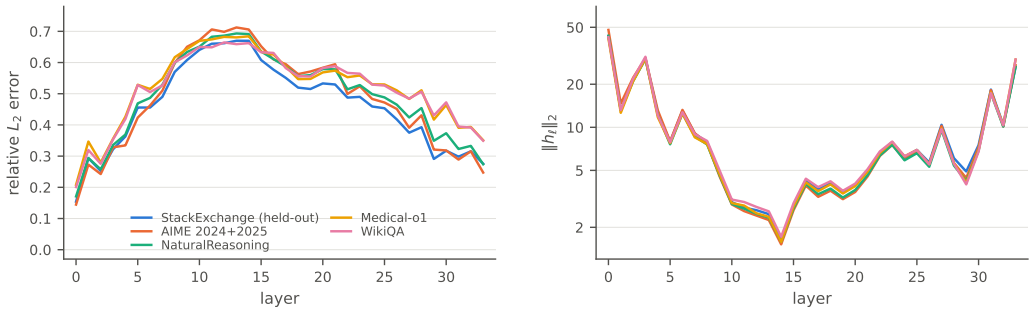


Figure 7: Left: relative L_2 reconstruction error $\|h_\ell - \hat{h}_\ell\|_2 / \|h_\ell\|_2$ by layer. Right: the activation norm $\|h_\ell\|_2$ (log scale), whose trough coincides with the error peak.

Sparsity is depth-skewed. The mean of 58.6 active features per token and layer is not representative of a typical layer. The median layer activates 10.4 features and 24 of the 34 layers stay below 20, while the deepest layers rise steeply toward the top-k cap: approximately 79 at layer 29, 214 at layer 30, 367 at layer 31, 452 at layer 32, and 487 at layer 33 against $k = 500$ (Figure 8). The JumpReLU threshold is therefore the binding sparsity constraint in the shallow and middle layers, whereas the top-k gate binds in the deepest ones. Over this evaluation sample, 41.3% of dictionary entries never fire.

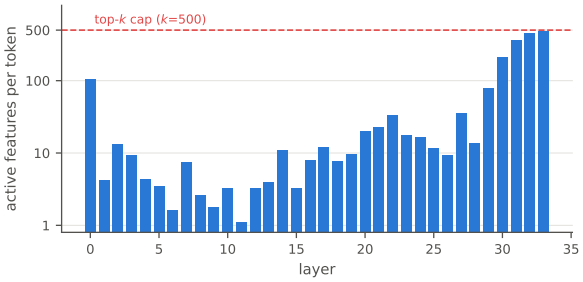


Figure 8: Active features per token (mean hard ℓ_0) by layer on the held-out set, against the top-k cap.

Feature-labeling protocol. We run the CLT over a random sample of 20,000 OpenWebMath (Paster et al., 2023) passages and, for each of the $34 \times 100{,}800 \approx 3.4$M features, record its firing rate together with the ten passages that activate it most strongly and the token position of each peak. We discard the first-token artifact (position < 2), retain features whose peak activation is at least 4 in normalized activation units, and label each retained feature by automated interpretability (Bills et al., 2023), prompting a language model to describe what its maximally activating examples share. Table 5 reports six retained features spanning layers 0 through 31 and the kinds of structure the dictionary learns: early-layer lexical concepts, mid-layer formatting and reasoning features, and deep-layer register features.

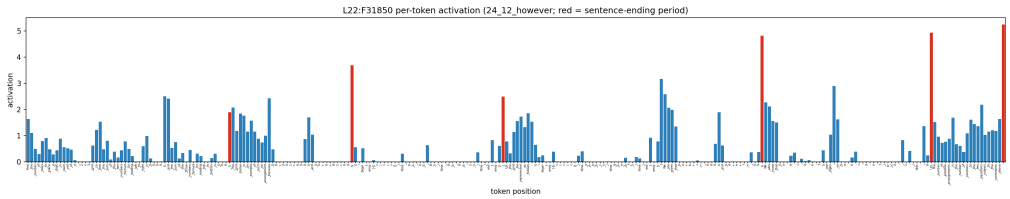
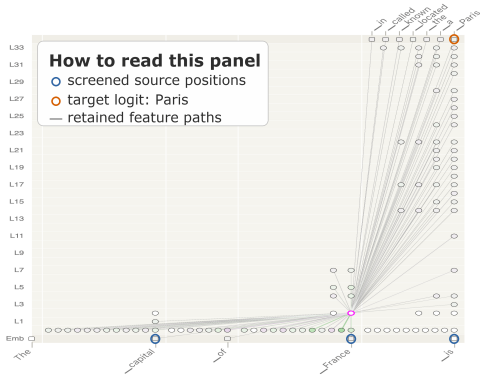


Figure 9: Per-token activation of `L22:F31850` across the AIME 2024 Problem 12 discovery sentence, the second of the two shown in Section 5. Red bars mark tokens whose decoded text ends in a period.

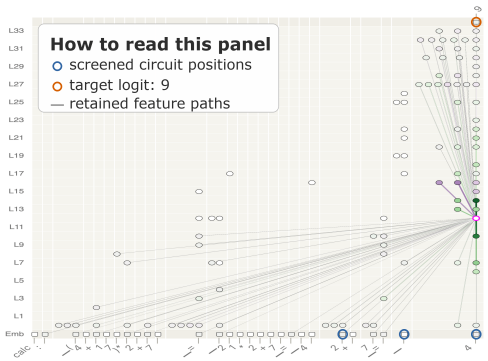
What magnitude-based selection does and does not surface. Selection by activation magnitude is not neutral, and we report the negative result because it affects how Table 5 should be read. On this corpus every magnitude-based ranking we tried (peak activation, peak-to-mean selectivity, and z-score, with and without the first-token exclusion) returns essentially the same population: code-syntax token detectors, because code tokens are the most repetitive and most selective items in the corpus. The high-level features that matter for our case studies have the opposite statistical profile. On the attribution-validated working set, they fire on 97–100% of passages, are diffuse rather than spiky (peak-to-mean between 1.5 and 2.5), and anchor on structural tokens such as periods rather than on content words. `L22:F31850` is one of these: it fires on 99.97% of passages with a peak near 7, so any sparsity or peak-magnitude filter excludes it by construction. Table 5 is accordingly a stratified sample across depth and structure type, not the top of a single ranking, and we do not claim it is an unbiased draw from the dictionary.

B Direct-Effect and Multi-Hop Attribution Details

This appendix expands Equation 7 into the form actually implemented, and gives the truncation diagnostics for Equation 9.



(a) `The capital of France is` → `Paris`



(b) `42+7 = 4` → `9`

Figure 10: Attribution graphs for factual retrieval and arithmetic completion. Blue rings mark the screened positions: `capital`, `France`, and the final `is` in (a); the operands `2` and `7`, and the final `4` prediction position in (b). Orange rings mark the target logits. Gray edges show retained positive direct-effect paths; magenta nodes and colored edges mark graph-selected local neighborhoods, which are subsequently refined by the tracing experiments in the text.

Node types and endpoint vectors. We write s for the layer at which the *source* node reads and r for the layer at which the *target* node reads, with corresponding token positions t_s and t_r; t indexes a token position and a indexes features. A node i is one of three types: a *feature node*, one per active (s, t, a) triple; a *token node*, one per non-special input token position t; or a *logit node*, one per output logit k in the subset whose cumulative softmax probability reaches a threshold (default 0.95). Every

Table 5: Automated-interpretability evidence that CLT features are monosemantic. Features are denoted `L`ℓ`:F`a, where ℓ is the layer and a the feature index. Token backgrounds are shaded in proportion to activation strength, with darker colors indicating larger activations.

Feature	Label	Top-activating OpenWebMath excerpts (token tint $\propto$ activation)
`L0:F30117`	Enterprise-software names	### Moving from MediaWiki to SharePoint O365 - part 4 … … · This is an example of SharePoint list lookup column in PowerApps. Power … … intranet local site, and sites in the intranet are configured (this is necessary in order …
`L0:F28233`	Troubleshooting & diagnosis	… errors out ...anyone willing to PM to troubleshoot ? • rooted, took me a … … the Design and Analysis of Experiments for process troubleshooting and discovering ways to improve process output. … … prompt. I do this because a) troubleshooting database issues is far outside of my skillset …
`L20:F14399`	Section headings	… ## Stochastic Oscillator 10 1 The fundamental premise … … ## Defining Definite Integrals What is a definite … … ## Countable and Uncountable Sets Definition. …
`L22:F31850`	Mathematical logic/proof	… show that m forms a basis. However, I am unsure as to whether … … set $$f(U)$$ is open. I saw related questions where other users … … a mono- and an epimorphism. Note that the usual examples fo
`L30:F38145`	Academic citation	… theorem in the theory of binary relations}, journal = {Compositio Mathematica}, … … illi}, journal={Celestial mechanics}, year={1985}, … … }, journal={J. Complex.}, year={2004}, …
`L31:F31060`	Problem-solving write-ups	… Relevant equations 3. The attempt at a solution I don't know where … … })$$ 3. The attempt at a solution Going back to basics with this … … 7:53 There is a tricky substi

edge is built from two ingredients: the *output vector* $\mathbf{v}_i^{\text{out}} \in \mathbb{R}^d$ that the source node writes into the residual stream, and the *input vector* $\mathbf{v}_i^{\text{in}} \in \mathbb{R}^d$ along which the target node reads a scalar off the stream. A feature node has both; a token node has only $\mathbf{v}_i^{\text{out}}$ and a logit node only $\mathbf{v}_i^{\text{in}}$.

For a feature node, $\mathbf{v}_i^{\text{out}}$ is the activation-weighted decoder contribution, spread into every downstream output layer $\ell \geq s$, one block per write layer,

$$\mathbf{v}^{\text{out},\,\ell}_{(s,t_s,a)} = z_{s,t_s,a}\,\mathbf{w}_a^{s\to\ell}, \qquad \ell \geq s, \tag{10}$$

with $z_{s,t_s,a}$ and $\mathbf{w}_a^{s\to\ell}$ as in Equation 2. For a token node it is the base-model embedding $\mathbf{v}_t^{\text{out}} = \mathbf{W}^E_{\text{tok}_t}$ of the input token at position t. For a feature node, $\mathbf{v}_i^{\text{in}}$ is encoder row a of its layer r, namely $\mathbf{W}_r^{\text{enc}}[a,:]$ from Equation 1. For a logit node it is the demeaned unembedding column

$$\mathbf{v}_k^{\text{in}} = \mathbf{u}_k - \bar{\mathbf{u}}, \tag{11}$$

where $\mathbf{u}_k$ is the unembedding column for token k and $\bar{\mathbf{u}}$ its mean over the vocabulary. Demeaning removes the component shared across all logits, so the score reflects deviation from average output behavior rather than absolute logit magnitude.

The direct effect. Under the linearization, the direct effect of source i on target j is

$$g_{i\to j} = \mathbf{v}_i^{\text{out}} \cdot \frac{\partial \left(\mathbf{v}_j^{\text{in}} \cdot \mathbf{x}_j\right)}{\partial \mathbf{x}_i}, \tag{12}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the residual streams at the source and target locations, the partial derivative is the Jacobian $\mathbf{J}_{i\to j}$ of Equation 7, and "$\cdot$" sums over the d model dimensions. Because a feature's output is spread over write layers, a feature-source edge sums the per-layer blocks of Equation 10, each derivative taken at the layer-ℓ MLP output at the source position. Fully indexed on both ends, the feature-to-feature edge is

$$g_{(s,t_s,a)\to(r,t_r,a')} = \sum_{s\le\ell<r} \left(z_{s,t_s,a}\, \mathbf{w}_a^{s\to\ell}\right) \cdot \frac{\partial \left(\mathbf{W}_r^{\text{enc}}[a',:] \cdot \mathbf{x}_{r,t_r}\right)}{\partial \mathbf{m}_{\ell,t_s}}, \tag{13}$$

where $\mathbf{W}_r^{\text{enc}}[a',:]\cdot\mathbf{x}_{r,t_r}$ is the pre-activation of target feature a' read from the layer-r residual at the target position. It omits the constant encoder bias $\mathbf{b}_r^{\text{enc}}$, which drops out of the derivative. The sum stops at $\ell < r$ because a write at or after the target's read layer never reaches the residual it reads. Substituting $\mathbf{v}_k^{\text{in}} = \mathbf{u}_k - \bar{\mathbf{u}}$ instead gives the feature-to-logit edge.

Implementation. Rather than evaluating Equation 13 independently per edge, all incoming edges of a target are computed in one backward pass: fixing the target j fixes the scalar $\mathbf{v}_j^{\text{in}}\cdot\mathbf{x}_j$, whose gradient with respect to the residual stream is available at every layer and position after a single backward pass, and the inner product of that gradient with $\mathbf{v}_i^{\text{out}}$ at each source location yields all $g_{i\to j}$. The forward pass caches the residual stream of every layer as a live computation-graph node and the scalar readout is never materialized: the target's input vector is written directly into the residual-stream gradient at position t_r with all other gradient seeds zero, so the backward pass propagates only the injected direction (Figure **??**). These derivatives are taken on the original Gemma-3-4B-IT transformer, not on a CLT surrogate: each layer's true MLP output $\mathbf{m}_{\ell,t}$ is a gradient leaf, and the CLT enters only through the direction vectors $\mathbf{v}^{\text{in}}$ and $\mathbf{v}^{\text{out}}$.

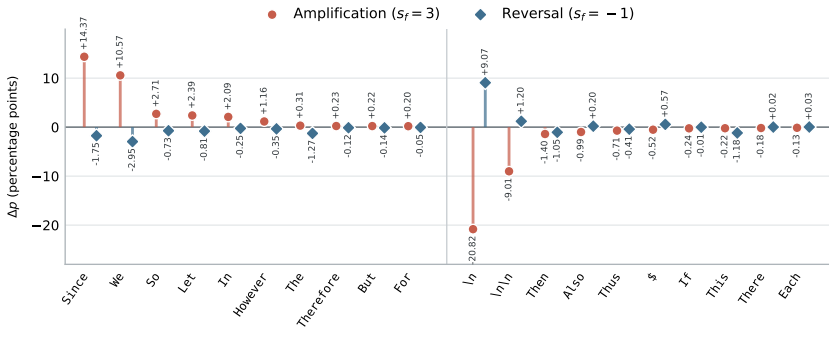


Figure 11: Change in mean next-token probability after intervention at sentence-ending periods, pooled over AIME 2024 and 2025. Tokens are ordered by their change under amplification.

Neumann truncation. Writing $T_0 = A$ and $T_i = T_{i-1}B = AB^i$, the entry $(T_i)_{\ell,s}$ collects every path from source s to logit ℓ through exactly i intermediate features, and $S_n = \sum_{i=0}^{n} T_i$ is Equation 9. These terms shrink quickly. Figure 12 plots the total magnitude $\|T_i\|_1$, the sum of absolute values of all entries of T_i, on the full attribution graph of the `However` prompt: after the first two hops the terms fall off geometrically at roughly a quarter per hop, from $\|T_0\|_1 \approx 6.4 \times 10^3$ to $\|T_{10}\|_1 \approx 0.14$. The last term retained at $n = 10$ is thus more than four orders of magnitude below the direct term, and the dropped tail is negligible at the precision of any claim we make.

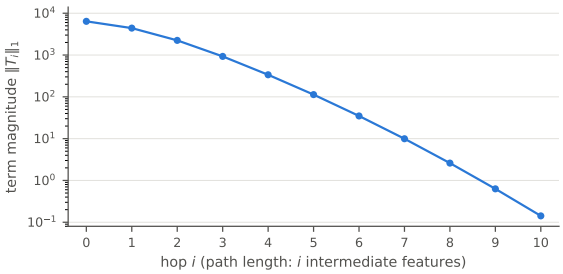


Figure 12: Geometric decay of the Neumann-series terms. Each point is the total magnitude $\|T_i\|_1$ of the hop-i contribution on the full attribution graph of the `However` prompt.