TRACING INFLUENCE: UNBOXING LLM OUTPUTS WITH SPARSE LATENT SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

A critical step toward the reliable use of large language models (LLMs) in high-stakes domains such as healthcare is to attribute their predictions to training data, akin to a medical case study. This requires fine-grained precision: pinpointing not only which training examples influence a decision, but also which parts of them are responsible. While influence functions offer a principled framework for this, prior token-level approaches are restricted to losses that decompose additively across tokens and distribute credit over a strongly correlated token basis. We introduce RepInf, a framework that attributes influence to internal features through a latent mediation approach and is defined for any differentiable objective. Our method attaches a sparse autoencoder (SAE) to an intermediate layer of a finetuned LLM to obtain a sparse, less correlated attribution basis, derives an activation-weighted first-order attribution of the training gradient, and scales it with derivative swapping, reducing the per-feature Jacobian-vector products to a single additional reverse-mode pass (13–18$\times$ faster). Token-level influence is then obtained by projecting latent attributions back to the input space via token activation patterns. Experiments on multiple-choice QA with Llama-3.2-1B and Qwen2.5-1.5B show that influence-selected features are more necessary and sufficient for the prediction than activation-, frequency-, or randomly-selected features, including under on-distribution replacement, and are better than a gradient$\times$activation baseline in several settings while never significantly worse. The total feature influence preserves the magnitude ranking of training examples under direct influence (Spearman $\rho = 0.98$ on Qwen2.5-1.5B), and reweighting the selected training data shifts predictions more than reweighting random data, supporting RepInf as a practical tool for model auditing.

1 INTRODUCTION

Large language models (LLMs) are increasingly being explored for high-stakes domains such as healthcare (Singhal et al., 2023), where predictions must be auditable and grounded in reliable evidence. In clinical settings, it is not enough for a model to produce a plausible answer; practitioners need to understand which prior evidence or training examples shaped the prediction. This is especially important because LLMs can hallucinate (Ji et al., 2023), rely on spurious correlations, or generate convincing but unfaithful explanations (Turpin et al., 2023).

Existing interpretability methods only partially address this need. Chain-of-thought prompting may produce post-hoc rationales rather than faithful explanations of the model's computation (Turpin et al., 2023), while attention maps and gradient-based attributions mainly explain a single forward pass (Jain & Wallace, 2019; Sundararajan et al., 2017). These methods do not directly answer a training-data attribution question (Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023; Hammoudeh & Lowd, 2024): which training examples, and which parts of them, most influenced a given prediction?

Influence functions (IFs) provide a principled framework for this question by estimating how a model's prediction would change if a training example were infinitesimally upweighted (Hampel, 1974; Koh & Liang, 2017). Recent work has scaled IFs to LLMs and used them to study influence at the sequence and token level (Schioppa et al., 2022; Grosse et al., 2023; Kwon et al., 2024). Token-level influence (Grosse et al., 2023) decomposes the training loss across tokens; this decomposition

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

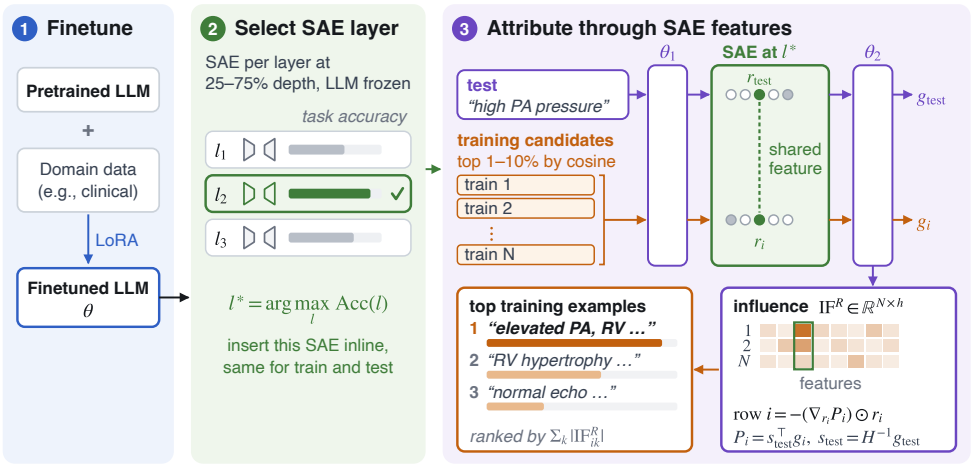


Figure 1: **Overview of RepInf.** (1) A pretrained LLM is finetuned on domain data with LoRA. (2) SAEs are trained at candidate layers (25%–75% of depth) with the LLM frozen; the SAE at the layer l^* that best preserves accuracy is inserted inline. (3) Training candidates are pre-filtered by gradient cosine similarity. For each candidate i, the feature scores $-(\nabla_{r_i} P_i) \odot r_i$, with $P_i = s_{\text{test}}^\top g_i$ and $s_{\text{test}} = H^{-1} g_{\text{test}}$, form one row of the influence matrix IF^R. They take two backward passes (Section 4.4). Ranking rows by $\sum_k |\text{IF}^R_{ik}|$ gives the top training examples. Inputs are illustrative.

is exact, and each token's gradient already reflects its context, but it is limited when the goal is to explain which parts of a training example matter. First, it exists only when the loss decomposes additively across tokens, and its credit is indexed by the loss terms of predicted tokens rather than by the content that carries the effect. Second, it distributes a well-defined total over token representations that are strongly correlated: in our models, the text-embedding and dense residual representations have stable ranks of only 2.1–4.5, compared with 6.8–7.7 for the SAE latent space (Table 1). On such a basis the total is stable, but the influence assigned to an individual token is difficult to identify, much like regression coefficients under collinearity.

We propose RepInf, a representation-level influence framework that attributes LLM predictions through sparse latent features rather than directly through raw tokens. Our key observation is that an influence score is an inner product between a fixed test signal and a training gradient, and that this training gradient is mediated by the model's intermediate representations. Our method therefore inserts a sparse autoencoder (SAE) (Cunningham et al., 2023; Bricken et al., 2023; Gao et al., 2024) at an intermediate layer of a finetuned LLM, producing a sparse latent representation that serves as a more structured attribution basis. We then attribute influence to these latent features and project the resulting attributions back to input tokens through activation patterns (Figure 1). This connects model predictions to both influential training examples and the internal features that mediate their effect.

A key challenge is that latent influence does not decompose additively across features in the same way token-level autoregressive losses decompose across tokens. To address this, we derive an activation-weighted latent influence formulation that measures how each active feature changes the downstream training gradient. A naive implementation would require one Jacobian-vector product per feature; we further show that, by swapping the derivative order, all latent-feature influences can be computed with a single additional reverse-mode pass, making the method practical for billion-parameter LLMs. Since this is a first-order attribution, we check its fidelity to direct influence and validate it behaviorally. Our contributions are as follows:

1. We introduce a representation-level influence framework that attributes training-data influence in LLMs to SAE features and tokens, and that is defined for any differentiable objective rather than only token-additive losses (Section 4).
2. We derive an activation-weighted latent influence estimator and an efficient derivative-swapping implementation that avoids per-feature JVP computation, reducing the cost per train–test pair by 13–18× (Section 4.4, Appendix G).
3. We validate the method on multiple-choice QA with Llama-3.2 and Qwen2.5 models, showing that the SAE latent space is better conditioned than the text and dense representations (Sec-

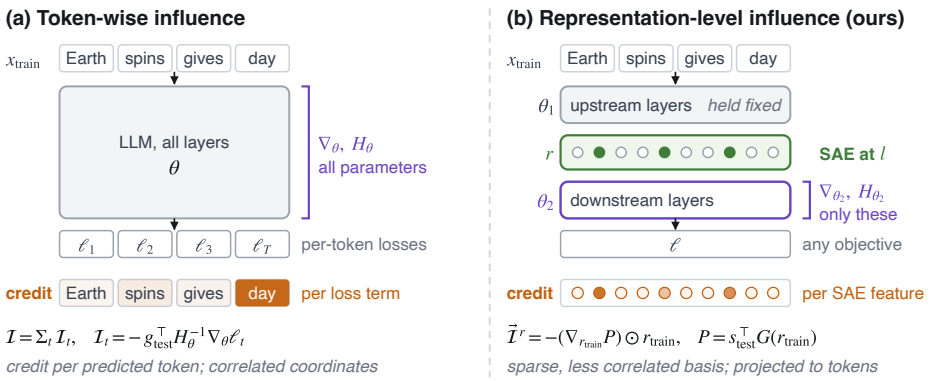


Figure 2: **Token-wise vs. representation-level influence.** (a) Token-wise influence (Grosse et al., 2023) splits $\mathcal{I} = \sum_t \mathcal{I}_t$ over per-token loss terms of correlated tokens. (b) RepInf holds θ_1 fixed, computes influence through θ_2 for any objective, and attributes it to the features of an SAE at layer l, which are projected back to tokens (Section 4).

tion 5.2), that influence-ranked latent features outperform activation magnitude, frequency, and random baselines in necessity and sufficiency tests, also on-distribution, and are better than test-side gradient×activation in several settings and never significantly worse (Section 5.3), and that reweighting the selected training data shifts predictions more than random data (Section 5.4). The total feature influence also preserves the magnitude ranking of training examples under a Grosse-style direct influence baseline ($\rho = 0.98$ on Qwen2.5-1.5B; Section 5.5).

2 Related Work

Training-data attribution. Influence functions originate in robust statistics (Hampel, 1974) and were brought to deep learning by Koh & Liang (2017); they can be fragile in non-convex models (Basu et al., 2021) and approximate a proximal Bregman response rather than leave-one-out retraining (Bae et al., 2022). Scalable variants use EK-FAC and other curvature approximations (Grosse et al., 2023; Schioppa et al., 2022), LoRA-specific closed forms (Kwon et al., 2024), random projections (Park et al., 2023), or gradient tracing (Pruthi et al., 2020); gradient similarity also drives data selection for instruction tuning (Xia et al., 2024). Related approaches include datamodels (Ilyas et al., 2022) and representer points (Yeh et al., 2018) (survey: Hammoudeh & Lowd 2024). All produce *example-level* scores; RepInf instead attributes an existing score to internal features and can be combined with any of these approximations.

Finer-grained influence. The closest work is the token-wise influence of Grosse et al. (2023); RepInf attributes along a sparse feature basis instead of loss terms and does not require a token-additive objective (Figure 2). Pezeshkpour et al. (2022) combine feature attribution and instance attribution to detect dataset artifacts. RepInf instead obtains the feature-level view *from* the influence score itself.

Feature attribution and SAEs. Gradient- and perturbation-based attribution methods (Sundararajan et al., 2017; Lundberg & Lee, 2017; Ribeiro et al., 2016; Smilkov et al., 2017) explain a single forward pass, and their deletion/insertion evaluations are confounded by off-distribution masking (Hooker et al., 2019), which we control for in Section 5.3. SAEs recover sparse, often interpretable features from LLM activations (Bricken et al., 2023; Cunningham et al., 2023; Templeton et al., 2024; Gao et al., 2024), and sparse feature circuits attribute behavior to SAE features on a given input (Marks et al., 2024). RepInf applies Integrated-Gradients-style path attribution to the *training gradient* rather than the output, and uses SAEs as a basis for training-data attribution, requiring a less correlated basis, not monosemanticity (extended discussion: Appendix A).

3 Preliminaries

Let $\mathcal{D} = \{z_i = (x_i, y_i)\}_{i=1}^n$ be a training set drawn i.i.d. from an unknown distribution, with inputs $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$. A model $h_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters $\theta \in \mathbb{R}^p$ is trained by empirical risk minimization, $\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^n \ell(h_\theta(x_i), y_i)$.

Influence functions. Influence functions (Hampel, 1974; Koh & Liang, 2017) measure the first-order effect on the fitted parameters of infinitesimally upweighting a training point $z_{\text{train}} = (x_{\text{train}}, y_{\text{train}})$. Under standard regularity conditions and the implicit function theorem (Krantz & Parks, 2002), this parameter response is $\text{IF}_{\hat{\theta}}(z_{\text{train}}) = -H_{\hat{\theta}}^{-1} \nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{train}}), y_{\text{train}})$, where $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 \ell(h_{\hat{\theta}}(x_i), y_i)$ is the Hessian of the empirical risk (Appendix B). Projecting the response onto the gradient of the loss at a test example $z_{\text{test}} = (x_{\text{test}}, y_{\text{test}})$ gives the scalar influence score

$$\mathcal{I}(z_{\text{train}}, z_{\text{test}}) = -\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{test}}), y_{\text{test}})^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{train}}), y_{\text{train}}). \tag{1}$$

Token-wise influence in autoregressive tasks. If the training loss is a sum of next-token terms, $\ell(z_{\text{train}}) = \sum_{t=1}^{T} \ell_t$ with $\ell_t = -\log p_\theta(x_t \mid x_{<t})$, then linearity of the gradient splits Eq. (1) into per-token contributions (Grosse et al., 2023):

$$\mathcal{I}(z_{\text{train}}, z_{\text{test}}) = \sum_{t=1}^{T} \mathcal{I}_t, \qquad \mathcal{I}_t = -\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{test}}), y_{\text{test}})^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell_t. \tag{2}$$

Remark 3.1 (Scope of Per-Token Influence). The decomposition in Eq. (2) is exact and does not assume that tokens are independent, since each $\nabla_\theta \ell_t$ is computed through the full causal context. However, it is restricted to losses that decompose additively across tokens, where credit is indexed by the predicted tokens (under an answer-only supervised loss, it falls entirely on the answer tokens), and it distributes the total over strongly correlated token representations (Section 5.2), where the credit of any single coordinate is not identifiable. To address this, we attribute influence in a sparse, less correlated SAE latent space (Section 4; Figure 2).

Jacobian–vector products. We use Jacobian–vector products (JVPs), computed with forward-mode automatic differentiation, to obtain directional derivatives without materializing a Jacobian (Baydin et al., 2018). The formal definition is given in Appendix B.

4 Methodology

We now detail our framework that attributes influence on test predictions to latent features and, through them, to tokens: 1) augmenting LLMs with SAEs (Section 4.1), 2) computing influence over latent representations (Section 4.2), 3) attributing the training gradient to latent features (Section 4.3) efficiently via derivative swapping (Section 4.4), and 4) projecting the scores back to examples and tokens (Section 4.5); see Figure 2b.

4.1 Augmenting LLMs with Sparse Autoencoders

We follow Bricken et al. (2023) and Gao et al. (2024) to define a sparse autoencoder that maps input $x^l \in \mathbb{R}^d$ at layer l into a sparse latent code $r \in \mathbb{R}^h$ through

$$r = \sigma(W_{\text{enc}}(x^l - b_{\text{pre}}) + b_{\text{enc}}), \qquad \tilde{x}^l = W_{\text{dec}} r + b_{\text{pre}}, \tag{3}$$

where $W_{\text{enc}} \in \mathbb{R}^{h \times d}$, $b_{\text{enc}} \in \mathbb{R}^h$, $W_{\text{dec}} \in \mathbb{R}^{d \times h}$, and $b_{\text{pre}} \in \mathbb{R}^d$. The nonlinearity $\sigma(\cdot)$ is ReLU in classical settings (Bricken et al., 2023) and TopK in modern settings (Gao et al., 2024); we use TopK. The SAE is trained with the LLM frozen, and its reconstruction $\tilde{x}^l$ replaces x^l in the forward pass. Let r_j be the activation of feature j, forming the basis for our feature-level influence analysis.

Remark 4.1 (SAE as an Attribution Basis). Attributing influence to individual coordinates is only meaningful when those coordinates are close to linearly independent (Remark 3.1). SAEs induce a latent representation that is sparse and markedly less correlated than the dense representation it replaces, which makes SAE latents a more suitable basis for influence attribution than raw token-level representations. We do not, however, require the feature activations to be statistically independent, nor do we infer either property from sparsity alone; both are measured in Section 5.2.[1]

[1]To promote orthogonality, we also experimented with adding an orthogonality constraint. However, since it changed the geometry only marginally (Appendix C.5), we ultimately excluded this term.

4.2 Influence Functions on Latent Representation

Rather than computing influence directly on correlated text tokens (Eq. (2)), we compute influence on *latent features*. We split the model parameters into two parts, $\theta = (\theta_1, \theta_2)$, where θ_1 comprises all parameters up to and including layer l (and the SAE encoder), mapping a raw input sequence x to an intermediate representation $r = h_{\theta_1}(x)$, and θ_2 comprises the remaining parameters, mapping r to the final prediction. By fixing θ_1 and treating r as the input, the IF can be computed with respect to θ_2 alone, effectively attributing influence to the latent representation r rather than to the original tokens. Let $r_{\text{train}} = h_{\theta_1}(x_{\text{train}})$ and $r_{\text{test}} = h_{\theta_1}(x_{\text{test}})$. The *representation-level influence function* is defined as

$$\mathcal{I}^r(z_{\text{train}}, z_{\text{test}}) = -\underbrace{\nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{test}}), y_{\text{test}})^\top}_{g_{\text{test}}^\top} H_{\theta_2}^{-1} \underbrace{\nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{train}}), y_{\text{train}})}_{g_{\text{train}}}, \tag{4}$$

where H_{θ_2} is the Hessian (or a positive semi-definite curvature surrogate) of the training loss w.r.t. θ_2. Restricting influence to θ_2 is an approximation to full-parameter influence (Eq. (1)), since it ignores how upweighting z_{train} would change θ_1; Section 5.5 checks that the total influence still ranks training examples as full-parameter influence does. Attribution is applied only to the training-side gradient g_{train}, while $s_{\text{test}} = H_{\theta_2}^{-1} g_{\text{test}}$ is treated as a fixed evaluation signal computed once per test example.[2]

4.3 Feature-Level Attribution via Integrated Gradients

The core challenge lies in attributing g_{train} to individual latent coordinates of r_{train}. To make this dependence explicit, we define a function G that maps any intermediate representation r to the corresponding gradient on θ_2, with the loss evaluated using the fixed training label:

$$G(r) \;=\; \nabla_{\theta_2}\ell(h_{\theta_2}(r), y_{\text{train}})\,, \qquad g_{\text{train}} = G(r_{\text{train}}). \tag{5}$$

In contrast to token attribution in autoregressive tasks, $G(r)$ is a single vector within the high-dimensional parameter space of θ_2; consequently, it cannot be linearly separated into distinct components for individual neurons. A naive directional derivative $\partial G / \partial r^{(j)}$ captures only local sensitivity, not how much a feature contributes when transitioning from inactivity to activity. For sparse SAE latents, we therefore use the baseline $r_0 = \mathbf{0}$ and study $\Delta G = G(r_{\text{train}}) - G(\mathbf{0})$. Following the axiomatic framework of Integrated Gradients (Sundararajan et al., 2017), the straight-line path from the origin gives an exact additive decomposition $\Delta G = \sum_j r_{\text{train}}^{(j)} \int_0^1 J_G(\alpha r_{\text{train}}) e_j \, d\alpha$ (Appendix H). Computing the exact integral is expensive. To obtain a scalable estimator, we evaluate the Jacobian at the realized activations, a backward first-order Taylor expansion that reuses activations already materialized during the forward pass:

$$\Delta G_j \;:=\; r_{\text{train}}^{(j)}\, J_G(r_{\text{train}})\, e_j. \tag{6}$$

Moreover, the multiplicative factor $r_{\text{train}}^{(j)}$ guarantees that contributions vanish for inactive features.

Definition 4.2 (Neuron-Level Influence)**.** The influence score attributed to neuron (SAE feature) j in the training representation r_{train} is defined as

$$\mathcal{I}_j^r(z_{\text{train}}, z_{\text{test}}) = -\, g_{\text{test}}^\top H_{\theta_2}^{-1}\, \Delta G_j = -\, r_{\text{train}}^{(j)}\; s_{\text{test}}^\top\, J_G(r_{\text{train}})\, e_j. \tag{7}$$

This formulation attributes influence to a neuron only if (i) it is active in the representation ($r_{\text{train}}^{(j)} \neq 0$) and (ii) its activation induces a change in the downstream parameter gradient along the test signal s_{test}; the sign of $\mathcal{I}_j^r$ follows the convention of the example-level score (Eq. (4)). Because Eq. (6) is a first-order approximation, $\sum_j \mathcal{I}_j^r$ need not equal $\mathcal{I}^r$ exactly; we therefore evaluate the fidelity of the aggregate empirically (Section 5.5) and treat the neuron-level scores as an *attribution* that ranks features rather than as an exact additive decomposition of $\mathcal{I}^r$.

[2] Decomposing g_{test} would answer a different question, namely which test features make the prediction sensitive to training data.

4.4 Scaling Influence From JVPs to Constant-Time Derivative Swapping

In Definition 4.2, the contribution of neuron j relies on the Jacobian column $J_G(r_{\text{train}})\, e_j$. For LLMs, materializing the full Jacobian $J_G \in \mathbb{R}^{|\theta_2| \times h}$ is memory-prohibitive, and a standard circumvention is to compute each column implicitly as a Jacobian-vector product (JVP) (Baydin et al., 2018), $J_G(r_{\text{train}})\, e_j = \frac{d}{d\varepsilon} G(r_{\text{train}} + \varepsilon e_j)\big|_{\varepsilon=0}$. However, this requires a separate forward-over-reverse pass for *every* active feature, costing 30–60 s per train–test pair at the 1B scale (Appendix G).

Constant-Time Influence via Derivative Swapping. The key insight is to exchange the order of differentiation (via Clairaut's Theorem), enabling us to obtain all feature influences from a single backward pass. Let $s_{\text{test}} = H_{\theta_2}^{-1} g_{\text{test}}$ and define $P(r_{\text{train}}) = s_{\text{test}}^{\top} G(r_{\text{train}})$. Since s_{test} is constant w.r.t. r_{train}, we have $s_{\text{test}}^{\top} J_G(r_{\text{train}}) e_j = \partial P / \partial r_{\text{train}}^{(j)}$, and the vector of latent influences is

$$\vec{\mathcal{I}}^{r}(z_{\text{train}}, z_{\text{test}}) = -\big(\nabla_{r_{\text{train}}} P\big) \odot r_{\text{train}}, \tag{8}$$

reducing the complexity from $\mathcal{O}(h)$ to $\mathcal{O}(1)$ backward passes (compute $\nabla_{\theta_2} \ell$, then backpropagate P to r_{train}). This identity is exact, i.e., a reverse-mode evaluation of Eq. (7) rather than a further approximation (Appendix H.2). In practice, it reduces the per-pair cost from about 40 s to 2.2 s on Llama-3.2-1B and from about 60 s to 4.4 s on Qwen2.5-1.5B (Appendix G.4).

4.5 From Latent Influence to Training Examples and Tokens

Gradient pre-filtering. Exact influence computation over the full training set is expensive. We therefore pre-filter training examples by the absolute cosine alignment of their aggregated latent gradients with the test example and retain only the top 1%–10% of candidates (Appendix F.4), a standard strategy for scaling influence to LLMs (Pruthi et al., 2020; Xia et al., 2024).

Aggregation. For a test sample, let $\text{IF}^R \in \mathbb{R}^{N \times h}$ denote the representation-level influence matrix over the N retained training examples and h SAE latents, obtained by stacking Eq. (8). Summing a row gives the total feature influence of a training example (the "summed influence" in our case studies), and summing its absolute values gives the RepInf mass used in Section 5.5. We average over training examples to obtain a test-conditioned importance vector with $s_k = \frac{1}{N} \sum_i \text{IF}^R[i,k]$, and rank features by s_k (Section 5.3).

Token projection. For a sequence x, let $A(x) \in \mathbb{R}^{T(x) \times h}$ denote its SAE activation matrix; the token-level influence on a training example x_i is then $\text{IFT}_{\text{train}}(i,t) = \sum_k \text{IF}^R[i,k]\, A(x_i)[t,k]$, and on the test input $\text{IFT}_{\text{test}}(t) = \sum_k s_k\, A(x_{\text{test}})[t,k]$. Token maps are thus projections through activation patterns, and the most reliable object is the latent feature attribution itself rather than the colored token map.

5 Experiments

We test the claims of the introduction in order: **RQ1** conditioning of the SAE basis; **RQ2** necessity and sufficiency of influence-selected features, including under on-distribution interventions and against a test-side attribution baseline; **RQ3** counterfactual effect of the selected training data and numerical stability; **RQ4** fidelity of the total feature influence to direct token-level influence; **RQ5** qualitative case studies.

5.1 Setup

Data and models. We use OpenBookQA (Mihaylov et al., 2018) (4,957 train / 500 test) as the primary quantitative benchmark, CommonsenseQA (Talmor et al., 2019) as a second one (Appendix C.4), and MedQA (Jin et al., 2021; Han et al., 2023) (9,160 / 1,018) for the SAE layer sweep and a clinical case study. We finetune Llama-3.2-1B-Instruct and Qwen2.5-1.5B-Instruct with LoRA. On OpenBookQA, finetuning raises accuracy from 42.0% to 72.8% for Llama and from 69.4% to 77.6% for Qwen (Table 3). The finetuning data are thus known and shape the predictions, the setting in which influence is well posed; all influence quantities use this checkpoint.

Table 1: **Basis geometry (RQ1)** on OpenBookQA (500 test examples) for the SAE used in all experiments (h=512, k=64; Llama layer 11, Qwen layer 17). Off-diagonal Gram statistics: mean absolute value, mean squared value, and normalized Frobenius norm. Results for larger SAEs and dense autoencoders, from a separate analysis configuration, are in Table 7.

Model	Space	Gram Abs ↓	Gram Sq ↓	Frob. Norm ↓	Stable Rank ↑	% < 0.1 ↑
Llama-3.2-1B	Text	0.151	0.056	0.236	4.47	57.7%
	Pre-Latent	0.200	0.065	0.255	4.43	32.5%
	SAE Latent	**0.090**	**0.029**	**0.171**	**7.65**	**79.2%**
Qwen2.5-1.5B	Text	0.199	0.087	0.294	3.44	47.4%
	Pre-Latent	0.407	0.234	0.483	2.11	14.4%
	SAE Latent	**0.081**	**0.030**	**0.173**	**6.80**	**80.1%**

SAE. A TopK SAE (h=512, k=64) is trained on the frozen model at the middle-half layer that best preserves accuracy: layer 11 for Llama (-2.4 points) and 17 for Qwen (-0.8) (sweeps in Appendix D; solver settings in Appendix F.4).

Evaluation scope. Zero-masking tests use 100 (Llama) and 500 (Qwen) OpenBookQA test examples; other analyses use the subsets stated with each result. Intervention results average three seeds; CIs are paired bootstraps over test examples (1,000 resamples).

5.2 RQ1: Geometry of the Attribution Basis

Per-coordinate credit is well posed only on a well-conditioned basis (Remark 3.1). For the SAE used in all experiments, we compare three spaces: input text embeddings (**Text**), the dense activations immediately before the SAE (**Pre-Latent**), and the SAE latent space (**SAE Latent**). Following Fel et al. (2025), we report off-diagonal Gram statistics (mean absolute value, mean squared value, and normalized Frobenius norm), the fraction of near-orthogonal pairs ($|\rho| < 0.1$), and the stable rank $\|A\|_F^2/\|A\|_2^2$. Table 1 shows that the SAE latent space is the least correlated and best conditioned of the three on every measure, for both models. Its stable rank is 7.65 (Llama) and 6.80 (Qwen), against 2.11–4.47 for the text and pre-latent spaces, and about 80% of its pairs are near-orthogonal, against 14–58%. It is not orthogonal, however: a fifth of its pairs still have $|\rho| \geq 0.1$.

Sparsity does not imply independence: centered correlations over nonzero activations (against a frequency-preserving permutation null with floor 0.006) average about 0.07 overall and 0.25 among co-active pairs, so SAE features are *not* statistically independent. The SAE supplies a markedly less correlated basis, not an orthogonal or independent one, so we validate attributions behaviorally (RQ2) and measure their agreement with direct influence (RQ4).

5.3 RQ2: Are Influence-Selected Features Necessary and Sufficient?

We rank the SAE features for each test example by the influence importance s_k (Section 4.5). We compare this ranking with three unsupervised baselines (activation magnitude, activation frequency, random) and one gradient-based single-pass baseline: **test-side gradient×activation** $|\partial \ell_{\text{test}}/\partial r_j \cdot r_j|$ on the test input. The last one asks whether training-data influence adds anything over explaining the test forward pass directly. For each method and $k \in [10, 200]$, **necessity** removes the top-k features and **sufficiency** keeps only them (metrics in Figure 3).

Zero masking (Figure 3). The separation from the unsupervised baselines is clearest at intermediate budgets. For necessity, removing 50–150 influence-ranked features drops the correct logit, raises the NLL, and flips the answer much faster than any of them on both models (Qwen: flip rate ≈1.0 at k=100 vs. at most 0.27). For sufficiency, keeping the top 100–150 influence-ranked features recovers most answers (Qwen: same-answer rate ≈0.95 at k=150 vs. at most ≈0.24). The gap closes at the extremes. At $k \leq 50$ no method is sufficient, and at k=200 the baselines catch up because most active features are removed or kept regardless of ranking (on Llama, activation magnitude even gives a larger ΔNLL). A three-seed rerun with CIs gives, at k=150: for Llama, the flip rate is 0.998 [0.994, 1.000] and the sufficiency same-answer rate is 0.526 [0.484, 0.568]. For Qwen they are 1.000 [1.000, 1.000] and 0.926 [0.902, 0.946]. CommonsenseQA shows the same pattern (Appendix C.4).

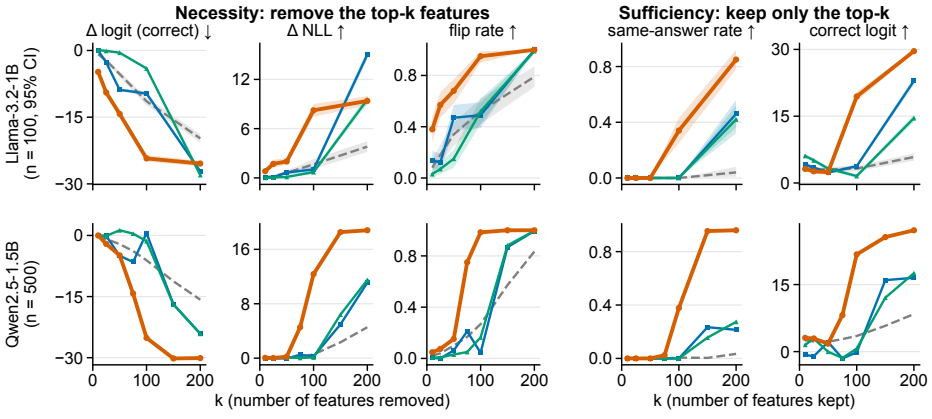


Figure 3: **Necessity and sufficiency under zero masking (RQ2)**, OpenBookQA. Llama: 100 test examples, 95% bootstrap CIs shaded; Qwen: all 500, point estimates. Seed-averaged CIs at k=150 are in the text; CommonsenseQA in Figure 4.

Table 2: **On-distribution interventions (RQ2)** with empirical-donor replacement (SAE h=512, k=64; n=100 OpenBookQA test examples; 3 seeds). Means shown. **Bold**: the paired difference between RepInf and gradient×activation has a 95% CI excluding zero in RepInf's favor; no difference is significant in the other direction. Arrows give the stronger direction. CIs: Table 5.

			Necessity (remove top-k)			Sufficiency (keep top-k)	
Model	k	Method	Δlogit ↓	ΔNLL ↑	Flip ↑	Same-ans. ↑	Corr. logit ↑
Llama (L11)	150	RepInf	**−12.86**	**5.53**	0.96	0.30	**18.14**
		Grad×act	−10.81	4.32	0.87	0.37	17.29
	200	RepInf	**−15.15**	**9.38**	0.98	0.46	**20.41**
		Grad×act	−14.10	7.93	0.95	0.43	19.74
Qwen (L17)	150	RepInf	**−9.64**	6.04	0.97	**0.65**	**17.65**
		Grad×act	−8.80	5.52	0.94	0.34	16.94
	200	RepInf	−10.01	8.11	0.99	**0.82**	18.35
		Grad×act	−10.04	8.74	0.99	0.70	18.58

On-distribution interventions and a test-side baseline. Zero masking can push the representation off the data manifold, so prediction changes may reflect distribution shift rather than feature importance (Hooker et al., 2019). We therefore repeat the tests with *empirical-donor replacement*: a removed feature takes the value the same feature has on another real example. This edit moves the representation by an L2 distance of only 0.022 (Qwen) and 0.026 (Llama), versus about 1.0 for zero masking, yet still collapses the answer. Table 2 compares RepInf with test-side gradient×activation under this replacement on 100 test examples (CIs: Table 5). RepInf is never significantly worse. On Llama it is significantly stronger on necessity (Δlogit, ΔNLL) and the retained correct logit at both budgets. On Qwen it retains the answer far more often (paired difference in same-answer rate +0.31 [0.20, 0.42] at k=150 and +0.12 [0.03, 0.21] at k=200), with several other metrics tied. Training-data influence thus adds information beyond a single-pass attribution of the test input in several settings, though not uniformly. A supplementary check with a larger 2048-dimensional SAE is given in Appendix C.3.

5.4 RQ3: Counterfactual Training-Data Effects and Estimator Stability

Counterfactual reweighting. To test the training-data claim itself, we up- or down-weight, from the converged adapter, the training groups RepInf selects for each of 20 test examples (two seeds) and compare the target-logit shift with that of random groups. Selected groups shift it more: 0.0294 vs. 0.0133, paired difference +0.0161 [0.0075, 0.0257]. The effect is small, as expected for one group in a converged model, but the CI excludes zero.

Solver diagnostics. Conditioning the reference checkpoint on the inserted SAE reduces its gradient norm from 0.888 to 0.0016. Because the raw Hessian can have non-positive curvature (Basu

et al., 2021), the fidelity and reweighting analyses use a PSD empirical Fisher, whose solves reach relative residuals below 1.9×10^{-7} with identical rankings and RepInf–direct agreement of $\rho = 0.973$–0.979 across a 3×3 grid of sample sizes and damping values (Table 9).

Layer dependence. Explanations are layer-specific (top-5 training examples overlap by only 0.13–0.18 across insertion layers), so we report them per layer (Appendix D).

5.5 RQ4: Fidelity to Direct Token-Level Influence

Since the feature scores rest on the θ_2 restriction and a first-order approximation, we check that their total preserves the example-level relation of classical influence, using Qwen2.5-1.5B-Instruct on OpenBookQA. As a direct baseline, we compute Grosse-style token-wise influence (Eq. (2)) over *all* trainable parameters (the 18.46M-parameter LoRA space, including layers before the SAE), with the same checkpoint, target-logit objective, damping, and candidates, over three curvature seeds; its per-token influences sum to the per-example influence within a mean relative error of 1.4×10^{-6}. Ranking each test example's candidates by RepInf mass and by the magnitude of direct influence, the mean Spearman correlation over 20 test examples is $\rho = 0.980$ [0.974, 0.984] for a fixed pool (480 pairs) and 0.983 [0.977, 0.989] for a random one (1,280 pairs). Classical influence thus continues to supply the signed example score, while RepInf's feature scores rank the features through which it acts, which RQ2–RQ3 validate behaviorally.

5.6 RQ5: Qualitative Attribution: Linking Predictions to Training Evidence

Protocol. Cases come from test instances the pretrained model gets wrong and the finetuned model gets right (OpenBookQA: 88/500 for Qwen, 187/500 for Llama). Features are described only as "associated with X in their top contexts" (e.g., feature 487 with birds, 1687 with deserts), based on top-activating contexts, attributed tokens, activation frequency, and decoder geometry; we do not claim monosemanticity.

In Case A (Figure 6), a question on what the Earth rotating on its axis causes is linked to a differently worded training example with the same causal relation, and attribution concentrates on "Earth rotating", "on its axis", and "sky" rather than option labels. Individual features further separate this evidence into components: feature 1338 fires on "Earth rotating on its axis" and "across the sky" (the physical mechanism), while feature 1835 fires on "causes", "appear", and the temporal options "day" and "daytime" (the question framing). In a MedQA case with Qwen (Figure 8), a question on small-airway inflammation is linked to a severe-asthma training example through shared features on "breath", "breathing", "distress", and "air". Appendix E gives these and two more cases; they are illustrative, and the quantitative claims rest on RQ1–RQ4.

6 Discussion and Limitations

Limitations. *Approximations:* feature scores are first-order and their fidelity is measured on one model (RQ4); influence describes infinitesimal upweighting rather than retraining (Bae et al., 2022); and the reweighting effect is reliable but small (20 tests). *SAE dependence:* attributions are specific to one SAE and layer; features are neither orthogonal, independent, nor all monosemantic, and descriptions are not human-evaluated. *Scope:* multiple-choice QA with ∼1B LoRA-finetuned models; the gain over gradient×activation holds only in some settings, and larger models and free-form generation are future work. A natural next step is detecting spurious training artifacts (Pezeshkpour et al., 2022).

7 Conclusion

RepInf attributes classical influence to SAE features and tokens at the cost of one extra backward pass. Across two model families, influence-selected features are necessary and sufficient under on-distribution interventions and never significantly worse than a test-side baseline, and the selected training data moves predictions more than random data. Since the first-order scores are not an exact additive decomposition, RepInf is best read as a behaviorally validated feature-level attribution, whose outputs are evidence for human review rather than ground truth.

Ethics Statement

RepInf is an auditing tool, not a source of ground truth. Attributions and token maps are evidence that requires human and counterfactual validation. They should not be presented as clinical justification, or as proof of why a model is correct. Plausible-looking explanations can create false confidence, particularly in healthcare and other high-stakes domains. We therefore recommend human oversight and do not claim clinical validity; the MedQA case study is illustrative only. Because attribution can surface specific training examples, it carries a privacy risk when the training data are sensitive. All datasets used in this work are public benchmarks.

Reproducibility Statement

The method is fully specified in Section 4, with derivations and the derivative-swapping proof in Appendix H. Training, SAE, curvature, and filtering hyperparameters are listed in Appendix F. Compute resources and runtime/memory profiles are given in Appendices F.2 and G. For every analysis, we state the evaluation subsets, the numbers of tests, pairs, and seeds, and the confidence-interval procedure (Section 5.1). Code, with instructions to reproduce the main results, is included in the supplementary material.

AI Use Statement

We used generative AI tools to edit the manuscript (grammar, wording, and restructuring of the presentation) and to help produce figures (visual layout and vector re-rendering of existing results). Generative AI was not used to generate experimental results. All reported numbers come from the authors' experiments. The authors reviewed all AI-assisted text and figures and take full responsibility for the content of this paper.

References

Ahmed Abdulaal, Hugo Fry, Nina Montaña-Brown, Ayodeji Ijishakin, Jack Gao, Stephanie Hyland, Daniel C Alexander, and Daniel C Castro. An x-ray is worth 15 features: Sparse autoencoders for interpretable radiology report generation. *arXiv preprint arXiv:2410.03334*, 2024.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *CoRR*, 2024. doi: 10.48550/ARXIV.2406.11717. URL https://arxiv.org/abs/2406.11717.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):e0130140, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140.

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B. Grosse. If influence functions are the answer, then what is the question? In *Advances in Neural Information Processing Systems*, 2022.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *CoRR*, June 2021. doi: 10.48550/arXiv.2106.07682. URL http://arxiv.org/abs/2106.07682.

Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL http://jmlr.org/papers/v18/17-468.html.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, March 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *Proc. Natl. Acad. Sci. U.S.A.*, 121(2):e2304406120, January 2024. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2304406120. URL http://arxiv.org/abs/2212.11870.

Sid Black, Lee Sharkey, Leo Grinsztajn, Eric Winsor, Dan Braun, Jacob Merizian, Kip Parker, Carlos Ramón Guevara, Beren Millidge, Gabriel Alfour, and Connor Leahy. Interpreting neural networks through the polytope lens. *CoRR*, November 2022. URL https://arxiv.org/abs/2211.12312.

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/index.html.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *ICLR*, 2023. URL http://arxiv.org/abs/2212.03827.

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

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, June 2020. URL https://distill.pub/2020/circuits/curve-detectors.

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. URL https://distill.pub/2020/circuits/curve-circuits/.

Giuseppe Casalicchio, Christoph Molnar, and Bernd Bischl. Visualizing the feature importance for black box models. *ECML PKDD*, 11051:655–670, 2018. doi: 10.1007/978-3-030-10925-7_40. URL http://arxiv.org/abs/1804.06620.

Lawrence Chan, Leon Lang, and Erik Jenner. Natural abstractions: Key claims, theorems, and critiques. *AI Alignment Forum*, March 2023. URL https://www.alignmentforum.org/posts/gvzW46Z3BsaZsLc25/natural-abstractions-key-claims-theorems-and-critiques-1.

David Chanin, Anthony Hunter, and Oana-Maria Camburu. Identifying linear relational concepts in large language models. *CoRR*, 2023. doi: 10.48550/ARXIV.2311.08968. URL https://arxiv.org/abs/2311.08968.

Ian C. Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: a unified framework for model explanation. *J. Mach. Learn. Res.*, 22(1):209:9477–209:9566, January 2021. ISSN 1532-4435. URL https://arxiv.org/abs/2011.14878.

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

Mingyang Deng, Lucas Tao, and Joe Benton. Measuring feature sparsity in language models. *CoRR*, 2023. doi: 10.48550/ARXIV.2310.07837. URL https://arxiv.org/abs/2310.07837.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=J6zHcScAo0.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Olsson Catherine, Nanda Neel, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022a. URL https://transformer-circuits.pub/2022/solu/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *Transformer Circuits Thread*, 2022b. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged bases in the transformer residual stream. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/privileged-basis/index.html.

Joshua Engels, Isaac Liao, Eric J. Michaud, Wes Gurnee, and Max Tegmark. Not all language model features are linear. *CoRR*, May 2024. doi: 10.48550/arXiv.2405.14860. URL http://arxiv.org/abs/2405.14860.

Thomas Fel et al. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *arXiv preprint*, 2025.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *TMLR*, 2023. URL https://arxiv.org/abs/2305.01610.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 2024.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Tianyu Han, Lisa C Adams, Jens-Michalis Papaioannou, Paul Grundmann, Tom Oberhauser, Alexander Löser, Daniel Truhn, and Keno K Bressem. Medalpaca–an open-source collection of medical conversational ai models and training data. *arXiv preprint arXiv:2304.08247*, 2023.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *EMNLP*, October 2023. doi: 10.48550/arXiv.2310.15916. URL http://arxiv.org/abs/2310.15916.

Tom Henighan, Shan Carter, Tristan Hume, Nelson Elhage, Robert Lasenby, Stanislav Fort, Nicholas Schiefer, and Christopher Olah. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/toy-double-descent/index.html.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. *CoRR*, August 2023. doi: 10.48550/arXiv.2308.09124. URL http://arxiv.org/abs/2308.09124.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Aapo Hyvärinen. Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110534, 2013.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning*, 2022.

M. Ivanitskiy, Alexander F. Spies, Tilman Rauker, Guillaume Corlouer, Chris Mathwin, Lucia Quirke, Can Rager, Rusheb Shah, Dan Valentine, Cecilia Diniz Behn, Katsumi Inoue, and Samy Wu Fung. Structured world representations in maze-solving transformers. *CoRR*, December 2023. URL https://arxiv.org/abs/2312.02566.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

janus. Simulators. *LessWrong*, September 2022. URL https://www.lesswrong.com/posts/vJFdjigzmcXMhNTsx/simulators.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

Adam Karvonen. Emergent world models and latent variable estimation in chess-playing language models. *COLM*, July 2024. doi: 10.48550/arXiv.2403.15498. URL http://arxiv.org/abs/2403.15498.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *ICML*, July 2019. doi: 10.48550/arXiv.1905.00414. URL http://arxiv.org/abs/1905.00414.

Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. DataInf: Efficiently estimating data influence in LoRA-tuned LLMs and diffusion models. In *International Conference on Learning Representations*, 2024.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023. URL https://arxiv.org/abs/2210.13382.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Giovanni Luca Marchetti, Christopher Hillar, Danica Kragic, and Sophia Sanborn. Harmonics of learning: Universal fourier features emerge in invariant networks. *CoRR*, December 2023. doi: 10.48550/arXiv.2312.08550. URL http://arxiv.org/abs/2312.08550.

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. *arXiv preprint arXiv:2403.19647*, 2024.

Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. Copy suppression: Comprehensively understanding an attention head. *CoRR*, October 2023. doi: 10.48550/arXiv.2310.04625. URL http://arxiv.org/abs/2310.04625.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1260.

Neel Nanda. 200 cop in mi: Interpreting algorithmic problems. *Neel Nanda's Blog*, 2022. URL https://www.lesswrong.com/posts/ejtFsvyhRkMofKAFy/200-cop-in-mi-interpreting-algorithmic-problems.

Neel Nanda. Actually, othello-gpt has a linear emergent world representation. *Neel Nanda's Blog*, March 2023. URL https://neelnanda.io/mechanistic-interpretability/othello.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, March 2018. URL https://distill.pub/2018/building-blocks.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Laura O'Mahony, Vincent Andrearczyk, Henning Muller, and Mara Graziani. Disentangling neuron representations with concept vectors. *CVPR Workshops*, April 2023. doi: 10.48550/arXiv.2304.09707. URL http://arxiv.org/abs/2304.09707.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. *ICML MI Workshop (Oral)*, June 2024. URL https://openreview.net/forum?id=KXuYjuBzKo.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *International Conference on Machine Learning*, 2023.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron C. Wallace. Combining feature and instance attribution to detect artifacts. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, 2020.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *CoRR*, April 2024. doi: 10.48550/arXiv.2404.16014. URL http://arxiv.org/abs/2404.16014.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *NAACL*, August 2016. doi: 10.48550/arXiv.1602.04938. URL http://arxiv.org/abs/1602.04938.

Adam Scherlis, Kshitij Sachan, Adam S. Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *CoRR*, July 2023. doi: 10.48550/arXiv.2210.01892. URL http://arxiv.org/abs/2210.01892.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *ICCV*, 2016. URL https://arxiv.org/abs/1610.02391.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. *Nature*, 623(7987):493–498, November 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06647-8. URL https://www.nature.com/articles/s41586-023-06647-8.

Lee Sharkey. Circumventing interpretability: How to defeat mind-readers. *CoRR*, December 2022. doi: 10.48550/ARXIV.2212.11415. URL https://arxiv.org/abs/2212.11415.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *ICML*, 2017. doi: 10.48550/arXiv.1704.02685. URL http://arxiv.org/abs/1704.02685.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, June 2017. doi: 10.48550/arXiv.1706.03825. URL http://arxiv.org/abs/1706.03825.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Viacheslav Surkov, Chris Wendler, Mikhail Terekhov, Justin Deschenaux, Robert West, and Caglar Gulcehre. Unpacking sdxl turbo: Interpreting text-to-image models with sparse autoencoders. In *Mechanistic Interpretability for Vision at CVPR 2025 (Non-proceedings Track)*, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4149–4158, 2019.

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*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Language models linearly represent sentiment. *ICML MI Workshop*, June 2024. URL https://openreview.net/forum?id=Xsf6dOOMMc.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. *CoRR*, 2023. doi: 10.48550/ARXIV.2310.15213. URL https://arxiv.org/abs/2310.15213.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. doi: 10.1162/tacl_a_00321. URL https://aclanthology.org/2020.tacl-1.25.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning*, 2024.

Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Qingyu Yin, Chak Tou Leong, Hongbo Zhang, Minjun Zhu, Hanqi Yan, Qiang Zhang, Yulan He, Wenjie Li, Jun Wang, Yue Zhang, et al. Direct preference optimization using sparse feature-level constraints. *arXiv preprint arXiv:2411.07618*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency. *CoRR*, October 2023. doi: 10.48550/arXiv.2310.01405. URL http://arxiv.org/abs/2310.01405.

A Extended Related Work

This section complements the compact discussion in Section 2 with a broader survey of interpretability work related to sparse autoencoders.

Interpretability in LLMs Interpretability methods range from black-box approaches like perturbation and sensitivity analysis (Casalicchio et al., 2018; Ribeiro et al., 2016; Covert et al., 2021; Warstadt et al., 2020), to gradient-based attribution methods (Smilkov et al., 2017; Sundararajan et al., 2017; Bach et al., 2015; Shrikumar et al., 2017; Selvaraju et al., 2016; Bilodeau et al., 2024), and concept-based representations probing (Belinkov, 2022; Kornblith et al., 2019; Bansal et al., 2021; Burns et al., 2023; Zou et al., 2023; Arditi et al., 2024). More recent work in mechanistic interpretability focuses on reverse-engineering internal model structures through circuit analysis (Olah et al., 2018; Elhage et al., 2021; 2022b), and feature discovery (Bricken et al., 2023; Sharkey, 2022; Cunningham et al., 2023; Deng et al., 2023). In addition to monosemanticity and disentanglement, this line of work has enabled analyses of motifs like induction heads or copy suppression (Olsson et al., 2022; McDougall et al., 2023; Cammarata et al., 2020; 2021), universality (Chan et al., 2023; Gurnee et al., 2023; Marchetti et al., 2023), and emergent world models (Li et al., 2023; Nanda, 2023; Ivanitskiy et al., 2023; Karvonen, 2024; Shanahan et al., 2023; janus, 2022). Unlike these approaches, which mostly explain a model's computation on a given input, RepInf connects internal features to the *training data* that shaped a prediction.

Sparse Autoencoders SAEs learn sparse, often interpretable features via sparsity constraints (e.g., an L1 penalty or a TopK activation). This approach relates to a long history of seeking structured data components, including classical linear methods like Principal Component Analysis (PCA) (Jolliffe & Cadima, 2016) and Independent Component Analysis (ICA) (Hyvärinen, 2013), as well as nonlinear probabilistic frameworks like Variational Autoencoders (VAEs) (Kingma & Welling, 2014). SAEs offer a deterministic and transparent way to learn such features while trading off sparsity against reconstruction fidelity. They are widely used for mechanistic interpretability in LLMs (Cunningham et al., 2023; Bricken et al., 2023; Templeton et al., 2024; Marks et al., 2024), with variants including k-sparse SAEs (Makhzani & Frey, 2013), gated and JumpReLU SAEs (Rajamanoharan et al., 2024), and TopK methods (Gao et al., 2024; Bussmann et al., 2024). Beyond language, SAEs extend to multimodal domains (Surkov et al., 2025), radiology and medical imaging (Abdulaal et al., 2024), and reinforcement learning alignment (Yin et al., 2024), demonstrating versatility across tasks. Recent work shows that transcoders (which approximate dense MLP behavior via wider, sparsely-activating networks) often match or exceed SAEs in interpretability and fidelity (Dunefsky et al., 2024). Extending RepInf to transcoder features is a natural direction that we leave to future work.

Monosemanticity and Disentanglement The pursuit of monosemantic features, where neurons respond to single coherent concepts, represents a major focus in interpretability research. This effort addresses the phenomenon of polysemanticity, explained through the superposition hypothesis (Olah et al., 2018; Elhage et al., 2021; 2023; Scherlis et al., 2023; Henighan et al., 2023). Solutions include both architectural modifications such as k-sparse autoencoders (Makhzani & Frey, 2013), softmax linear units (Elhage et al., 2022a; Rajamanoharan et al., 2024), as well as post-hoc methods like SAEs (Bricken et al., 2023; Sharkey, 2022; Cunningham et al., 2023; Deng et al., 2023). Studies have examined the linearity of representations (Nanda, 2022; Engels et al., 2024; O'Mahony et al., 2023; Hendel et al., 2023; Todd et al., 2023; Hernandez et al., 2023; Chanin et al., 2023; Tigges et al., 2024; Arditi et al., 2024), identified counterexamples such as circular features (Engels et al., 2024) and non-linear perspectives (Black et al., 2022). Geometry-aware analyses show structured organization (Park et al., 2024), and scaling studies (Templeton et al., 2024) suggest disentanglement improves with model size. While these works pursue monosemanticity, we use SAEs for a narrower purpose: to obtain a sparse, less correlated basis to which training-data influence can be attributed. RepInf does not require, and we do not claim, full monosemanticity or statistical independence of features (Section 5.2).

B Framework Overview

Influence Functions (IFs) In statistical estimation, the IF quantifies the sensitivity of an estimator to infinitesimal perturbations in the data, under the assumption that the data are independent. This

concept extends directly to machine learning, where the high-dimensional "parameter" is the set of weights $\hat{\theta}$ of a trained neural network—a complex function of the data shaped by the architecture, loss, and optimizer. Once training is complete and the model parameters $\hat{\theta}$ are fixed, we can analyze their local sensitivity to individual training samples. This is first formalized by the *response function*, $\hat{\theta}_{\epsilon,z_{\text{train}}}$, which describes what the optimal parameters *would be* if we were to infinitesimally upweight the loss (by ϵ) on a specific point $z_{\text{train}} = (x_{\text{train}}, y_{\text{train}})$ in the empirical risk. This perturbed objective is defined as:

$$\hat{\theta}_{\epsilon,z_{\text{train}}} = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \ell(h_\theta(x_i), y_i) + \epsilon\ell(h_\theta(x_{\text{train}}), y_{\text{train}}), \tag{9}$$

where the solution at $\epsilon = 0$ corresponds exactly to the original pre-trained parameters: $\hat{\theta}_{0,z_{\text{train}}} = \hat{\theta}$. The IF measures the sensitivity of these pre-trained parameters by computing the first-order Taylor approximation (i.e., the derivative) of the response function with respect to ϵ, at $\hat{\theta}$. Under standard regularity conditions, this can be computed using the Implicit Function Theorem (Krantz & Parks, 2002). Let $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \nabla^2_\theta \ell(h_{\hat{\theta}}(x_i), y_i)$ be the Hessian of the empirical risk evaluated at $\hat{\theta}$, then

$$\text{IF}_{\hat{\theta}}(z_{\text{train}}) = \left.\frac{d\hat{\theta}_{\epsilon,z_{\text{train}}}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{train}}), y_{\text{train}}). \tag{10}$$

The scalar influence score Eq. (1) and its token-wise decomposition Eq. (2) follow by projecting this response onto the test-loss gradient (Section 3).

Jacobian-Vector Products This is a key technical tool that we use. Given a function $F : \mathbb{R}^n \to \mathbb{R}^m$ and a direction $v \in \mathbb{R}^n$, the JVP at $x \in \mathbb{R}^n$ is the *directional derivative* of F at x along v:

$$\text{JVP}(F, x, v) \;=\; \left.\frac{d}{d\varepsilon}F(x + \varepsilon v)\right|_{\varepsilon=0} \;=\; J_F(x)\, v, \tag{11}$$

where $J_F(x)$ is the Jacobian of F at x. Intuitively, it answers the question: *"If I nudge the input by an infinitesimal step εv, how does the output change to first order?"*

Modern automatic differentiation libraries (e.g., PyTorch, JAX, TensorFlow) can compute JVPs directly without materializing the full Jacobian. Instead, they propagate the perturbation v forward through each primitive operation (forward-mode AD), making JVPs scalable to high-dimensional functions such as deep neural networks.

Framework Overview Figure 2 (Section 3) maps directly onto the notation in Section 4. We split the model at layer l as $\theta = (\theta_1, \theta_2)$, where the upstream subnetwork maps an input sequence x to an intermediate representation $r = h_{\theta_1}(x)$ and the downstream subnetwork h_{θ_2} maps r to the final prediction. For a training–test pair, this gives $r_{\text{train}} = h_{\theta_1}(x_{\text{train}})$ and $r_{\text{test}} = h_{\theta_1}(x_{\text{test}})$, and the corresponding latent-space samples $z^r_{\text{train}} = (r_{\text{train}}, y_{\text{train}})$ and $z^r_{\text{test}} = (r_{\text{test}}, y_{\text{test}})$. Panel (a) corresponds to classical token-wise influence, which splits the example-level score over per-token loss terms. Panel (b) shows our representation-level pipeline: we insert an SAE at the split point so that r is represented in a sparse latent basis, keep θ_1 fixed, and compute influence only through the downstream parameters θ_2. In the notation of Eq. (4), the test branch provides $g_{\text{test}} = \nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{test}}), y_{\text{test}})$ and the training branch provides $g_{\text{train}} = \nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{train}}), y_{\text{train}})$, which are combined through $H^{-1}_{\theta_2}$ to form the representation-level score $\mathcal{I}^r(z_{\text{train}}, z_{\text{test}})$. To localize this score to individual latent features, we define $G(r) = \nabla_{\theta_2}\ell(h_{\theta_2}(r), y_{\text{train}})$ and evaluate either the per-feature JVP form of Eq. (7) or, equivalently, the vectorized form in Eq. (8). Finally, these influential latent coordinates are projected back to tokens through SAE activations, yielding token-level heatmaps that explain which training evidence supports the test prediction.

C Full Experimental Results

This appendix reports results that support Section 5 but are not shown in full there.

C.1 Finetuning and SAE Insertion

Auditing with RepInf requires two things. Finetuning must produce meaningful task-specific updates, and inserting the SAE must not substantially degrade task performance. Table 3 shows that

LoRA-SFT yields large gains for all base models. The largest are for Llama-3.2-1B (+44.2 points on OpenBookQA and +40.2 on CommonsenseQA). Qwen2.5-1.5B starts from a stronger baseline and gains less. Table 4 shows that inserting the SAE largely preserves finetuned accuracy. Qwen2.5-1.5B-Instruct loses 0.8 points at its selected layer, which is also its best layer. Llama-3.2-1B-Instruct loses 2.4 points at the selected layer. A nearby layer loses only 0.2, so most of the cost comes from where the bottleneck is placed rather than from the SAE itself. We select the best layer within the middle half of the network (layers 4–12 for Llama and 7–22 for Qwen), because late layers yield less useful features for intervention (Table 8).

Table 3: Finetuning results across base models. Δ denotes LoRA-SFT minus pretrained accuracy.

	CommonsenseQA			OpenBookQA		
Model	Pretrained	LoRA-SFT	Δ	Pretrained	LoRA-SFT	Δ
Llama-3.2-1B	28.99%	69.21%	+40.22%	26.60%	70.80%	+44.20%
Llama-3.2-1B-IT	50.61%	72.40%	+21.79%	42.00%	72.80%	+30.80%
Qwen2.5-1.5B	67.49%	74.45%	+6.96%	64.80%	79.60%	+14.80%
Qwen2.5-1.5B-IT	72.81%	74.28%	+1.47%	69.40%	77.60%	+8.20%

Table 4: SAE insertion results on OpenBookQA (h=512, k=64). Δ denotes +SAE minus finetuned accuracy.

		Selected layer			Best layer		
Model	LoRA-SFT	Layer	+SAE	Δ	Layer	+SAE	Δ
Llama-3.2-1B-IT	72.80%	11	70.40%	-2.40%	13	72.60%	-0.20%
Qwen2.5-1.5B-IT	77.60%	17	76.80%	-0.80%	17	76.80%	-0.80%

C.2 Confidence Intervals for the Main Tables

Table 5 gives the 95% paired-bootstrap confidence intervals (1,000 resamples over test examples) for Table 2.

Table 5: Empirical-donor interventions with 95% CIs (OpenBookQA, n=100, 3 seeds). Paired differences (RepInf minus gradient×activation) whose CI excludes zero are listed below the table.

Model	k	Method	Nec. Δlogit	Nec. ΔNLL	Flip rate	Suff. same-ans.	Suff. corr. logit
Llama	150	RepInf	−12.858 [−13.982, −11.724]	5.534 [4.622, 6.473]	0.960 [0.920, 0.990]	0.300 [0.210, 0.390]	18.142 [17.418, 18.902]
Llama	150	Grad×act	−10.814 [−11.973, −9.640]	4.322 [3.244, 5.496]	0.870 [0.800, 0.930]	0.370 [0.270, 0.460]	17.292 [16.418, 18.135]
Llama	200	RepInf	−15.149 [−16.265, −14.146]	9.379 [8.116, 10.738]	0.980 [0.950, 1.000]	0.460 [0.370, 0.560]	20.409 [19.917, 20.937]
Llama	200	Grad×act	−14.102 [−15.119, −13.012]	7.934 [6.704, 9.294]	0.950 [0.900, 0.990]	0.430 [0.330, 0.520]	19.738 [19.066, 20.371]
Qwen	150	RepInf	−9.643 [−10.405, −8.825]	6.044 [5.026, 7.051]	0.970 [0.930, 1.000]	0.650 [0.560, 0.740]	17.649 [17.054, 18.330]
Qwen	150	Grad×act	−8.803 [−9.612, −7.966]	5.515 [4.518, 6.584]	0.940 [0.890, 0.980]	0.340 [0.240, 0.430]	16.941 [16.205, 17.716]
Qwen	200	RepInf	−10.011 [−10.849, −9.142]	8.108 [6.932, 9.483]	0.990 [0.970, 1.000]	0.820 [0.740, 0.890]	18.348 [17.756, 18.911]
Qwen	200	Grad×act	−10.044 [−10.899, −9.145]	8.735 [7.318, 10.258]	0.990 [0.970, 1.000]	0.700 [0.610, 0.780]	18.575 [17.931, 19.161]

Significant paired differences in RepInf's favor. **Llama**, k=150: necessity Δlogit 2.045 [1.324, 2.755]; ΔNLL 1.212 [0.631, 1.809]; sufficiency correct logit 0.850 [0.278, 1.387]. **Llama**, k=200: 1.047 [0.519, 1.635]; 1.445 [0.773, 2.211]; 0.671 [0.291, 1.045]. **Qwen**, k=150: necessity Δlogit 0.840 [0.210, 1.509]; same-answer rate 0.310 [0.200, 0.420]; correct logit 0.708 [0.020, 1.393]. **Qwen**, k=200: same-answer rate 0.120 [0.030, 0.210]. All other paired differences have CIs that include zero; none is significant in gradient×activation's favor.

C.3 Supplementary Missingness Check with a Larger SAE

In addition to the matched analysis of Table 2, we ran a missingness check with a 2048-dimensional analysis SAE on 12 reference-correct test examples, using empirical-donor replacement on the top-512 features. Because the SAE dimension and budget differ from Figure 3, these numbers are not directly comparable to the main results; we report them only as a consistency check (Table 6). Both paired intervals exclude zero.

Table 6: Empirical-donor intervention on the top-512 features of a 2048-dimensional analysis SAE (n=12 reference-correct test examples). Paired differences with bootstrap 95% CIs.

Intervention on top-512 features	RepInf	Random	Paired diff. (95% CI)
Necessity: NLL increase from removal (↑)	2.319	0.190	+2.129 [0.228, 4.006]
Sufficiency: degradation when retaining (↓)	8.963	12.118	−3.155 [−4.787, −1.664]

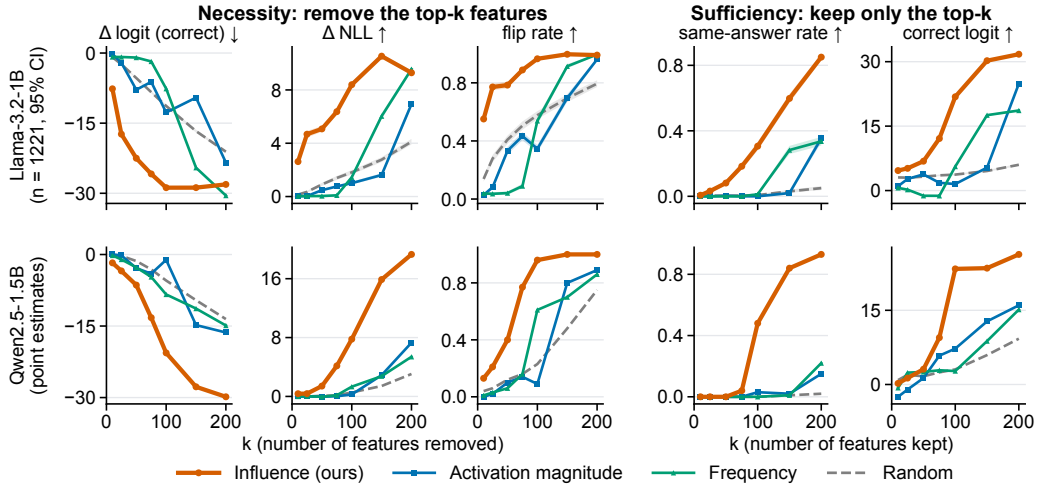


Figure 4: CommonsenseQA necessity (remove the top-k features) and sufficiency (keep only the top-k features), comparing influence-selected features to the baselines under zero masking. Llama uses the full 1,221-example evaluation set, with 95% bootstrap CIs shaded. Qwen curves are point estimates.

C.4 CommonsenseQA Quantitative Results

We provide additional necessity and sufficiency results on CommonsenseQA, following the same protocol as in Section 5.3. These extended results show that the qualitative pattern from the main text also holds on a second benchmark, supporting the generalizability of the framework across datasets and model families.

C.5 Orthogonality Ablations Across SAE/AE Variants

Orthogonality across SAE/AE variants. To study how architecture and regularization affect disentanglement, Table 7 compares dense AE (D) and sparse SAE (S) variants across orthogonality metrics and task accuracy (Acc.). Here, OW denotes the orthogonality-penalty weight; OW = – indicates no orthogonality penalty for that row.

Overall, sparse SAEs yield the strongest near-orthogonality (up to 99.10% at k=16384/256) and lowest off-diagonal correlation magnitudes, while dense AEs improve substantially as OW increases (e.g., better stable rank and lower mean-squared correlation at OW = 1.0). Across settings, this indicates an orthogonality–accuracy tradeoff: improving feature orthogonality is generally accompanied by some drop in task accuracy (within roughly 70.8–73.4 in our runs).

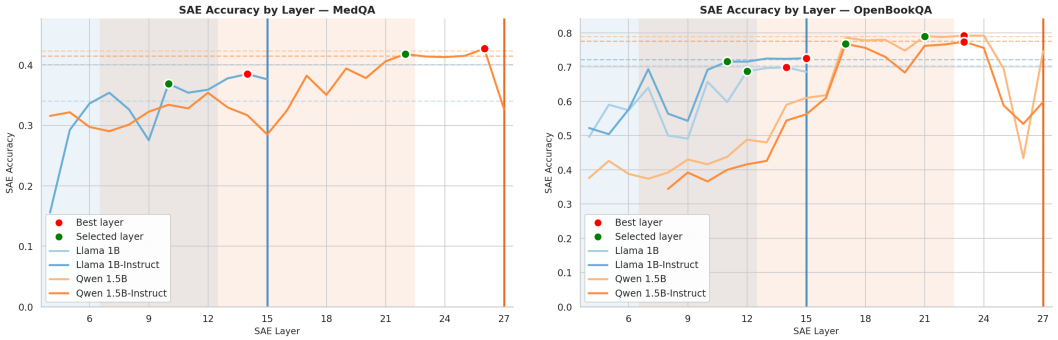
D SAE Layer Sweep and Training Results

We sweep SAE insertion layers and SAE configurations across benchmarks. Accuracy is typically best preserved in intermediate layers, and can degrade when inserting too early (low-level representations) or too late (near task-head computations). In all plots, the *shaded region* denotes the *middle-half* of the network layers considered in the main experiments (Section C.1). Concretely, for a model with L transformer blocks, the middle-half range is

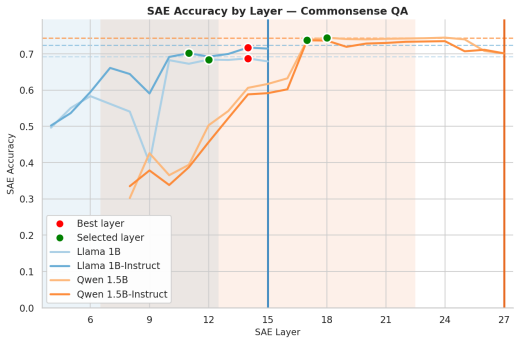
$$\mathcal{L}_{\text{mid}} = \{\lceil L/4 \rceil, \ldots, \lfloor 3L/4 \rfloor\}.$$

Table 7: Orthogonality comparison across sparse SAE and dense AE variants (Llama-3.2-1B, OpenBookQA). This comes from a separate analysis configuration and is not directly comparable to Table 1, which re-exports all spaces for the SAE used in the experiments. Lower is better for mean/squared/Frobenius off-diagonal metrics; higher is better for stable rank and near-orthogonality. G-Abs and G-Sq stands for the mean absolute/squared value of the gram off-diagnoal entries

Sparse Setup	Space	OW	G-Abs ↓	G-Sq ↓	F-norm ↓	SR ↑	%< 0.1 ↑	Acc.
D 512	Text	–	0.153	0.057	0.238	4.37	56.81%	72.6
D 2048	Pre-latent	–	0.227	0.083	0.289	3.76	28.90%	72.6
D 2048	Latent	0.1	0.124	0.055	0.235	4.68	65.11%	71.2
D 2048	Latent	1.0	0.090	0.016	0.125	16.95	65.56%	70.8
S 2048/256	Latent	–	0.095	0.028	0.167	7.79	75.43%	73.4
S 16384/256	Latent	–	0.008	0.002	0.047	25.02	98.67%	73.2
S 16384/256	Latent	0.1	0.006	0.002	0.040	22.34	99.10%	72.6
D 16384	Latent	0.1	0.100	0.029	0.169	7.61	66.41%	71.2
D 16384	Latent	1.0	0.090	0.015	0.124	22.51	65.98%	71.4



(a) MedQA. (b) OpenBookQA.



(c) CommonsenseQA.

Figure 5: Post-insertion task accuracy versus SAE insertion layer for Llama-3.2-1B(-Instruct) and Qwen2.5-1.5B(-Instruct). Shaded regions mark the middle half of each model's layers; green dots mark the selected layer (best within the middle half) and red dots the best layer over the full sweep. Dashed horizontal lines show finetuned accuracy without the SAE.

The *red dot* marks the best recovered accuracy (smallest drop) over the full sweep. The *selected layer* is the best layer *restricted* to $\mathcal{L}_{\text{mid}}$.

Takeaways. Across all three datasets: (i) there typically exists a contiguous band of intermediate layers where the post-insertion accuracy drop is small; (ii) the optimal insertion layer can shift by several blocks across model families and datasets; and (iii) larger SAEs (higher latent dimension H) tend to be more forgiving to insertion, while more aggressive sparsity (smaller k) increases layer sensitivity. These trends motivate the layer-selection rule in Section C.1: we search within

Table 8: Necessity effects across insertion depth (Llama-3.2-1B-Instruct, OpenBookQA). Removing the top-k influence-ranked features at the intermediate layer (L10) generally yields larger ΔNLL and higher flip rate (FR) than at the deeper layer (L13).

k	ΔNLL (L10)	ΔNLL (L13)	FR (L10)	FR (L13)
10	**0.6498**	0.2897	**0.384**	0.182
25	**1.3853**	0.8533	**0.558**	0.354
50	**2.1468**	2.1452	**0.646**	0.460
75	**4.4718**	2.9893	**0.836**	0.582
100	**8.2704**	4.4470	**0.964**	0.656
150	**10.3935**	9.8442	**0.998**	0.756
200	9.4899	**10.3416**	**1.000**	0.862

$\mathcal{L}_{\text{mid}}$ to avoid both input-adjacent and head-adjacent regimes, and we report both the selected layer (restricted to $\mathcal{L}_{\text{mid}}$) and the best layer over the full sweep.

Depth trade-off: downstream preservation vs. feature utility. We compare necessity-test effects at two insertion depths for Llama-3.2-1B-Instruct on OpenBookQA. Layer 10 is an intermediate layer, and Layer 13 is the layer that best preserves accuracy after insertion (both are marked in Figure 5b). Table 8 shows that removing influence-ranked features at L10 produces a larger NLL increase at most budgets, and consistently higher flip rates, than at L13. Intermediate-layer features are therefore more consequential under intervention, while the deeper layer preserves accuracy better after insertion. This trade-off supports inserting the SAE at an intermediate layer for auditing.

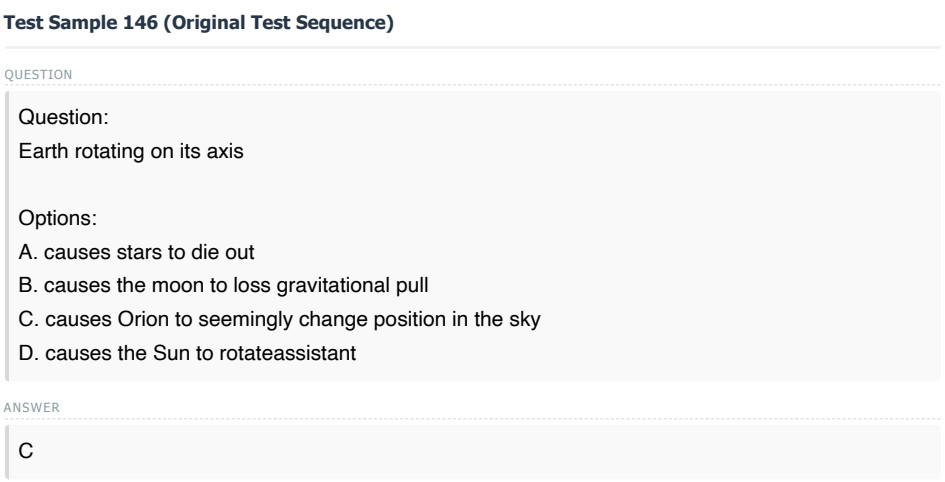
E Qualitative Case Studies

This appendix gives the figures for the case studies in Section 5.6, together with three additional cases. All figures are vector renderings of the output of our visualization tool.

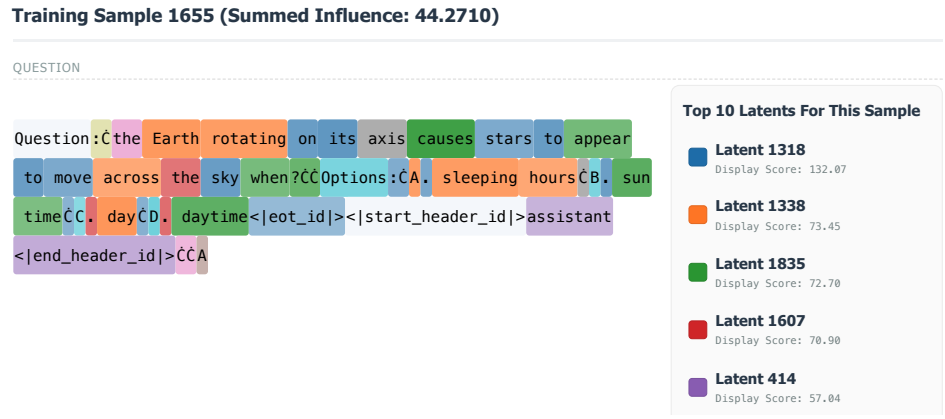
Protocol. We select test instances that the pretrained model answers incorrectly and the finetuned model answers correctly; on OpenBookQA there are 88 of 500 such instances for Qwen and 187 of 500 for Llama. For each selected test example, we compute the representation-level influence matrix IF^R and use it to rank influential training examples and latent features. We then visualize token-level influence by projecting the influential latent signal onto SAE activations (Section 4.5). Each token is colored by its dominant influential latent, with intensity given by activation $\times$ influence, normalized per sequence. The legend lists the top latents for the sample with their display scores. Feature descriptions follow the four-source protocol of Section 5.6. They are statements about association in top-activating contexts, not claims of monosemanticity.

Case Study A: OpenBookQA; shared astronomical cues (Llama-3.2-1B-Instruct). The test question asks what the Earth rotating on its axis causes (answer: Orion seemingly changes position in the sky). RepInf retrieves a training example with summed influence 44.27, "the Earth rotating on its axis causes stars to appear to move across the sky when?", whose wording differs from the test question but encodes the same causal relation. The dominant features concentrate on "Earth rotating", "on its axis", "causes stars", "appear", "move", and "sky", rather than on option labels or isolated lexical overlap (Figure 6). Panel (c) isolates latent 1338, which primarily activates on the concept-bearing phrase "Earth rotating on its axis" together with "across the sky", matching the causal relation required by the test question. Latent 1835 captures a complementary pattern involving "Earth", "causes", "appear", and temporal answer choices such as "day" and "daytime". Together, these latents suggest that the retrieved training example supports the prediction through two components: the physical mechanism of Earth's rotation and the question-specific temporal framing. This gives more fine-grained evidence than a single aggregate influence score.

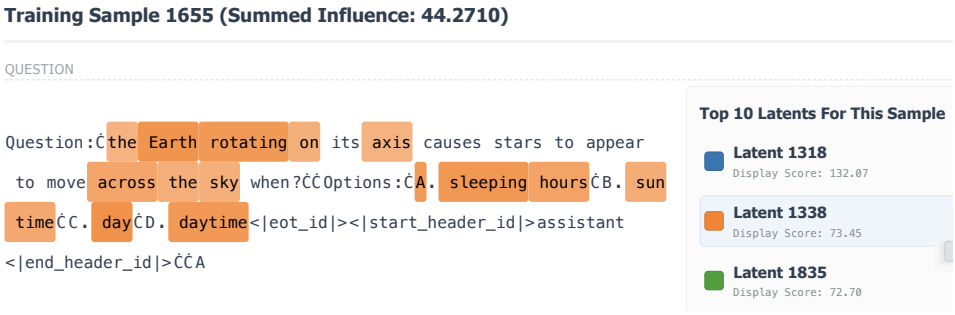
Case Study B: OpenBookQA; shared physical mechanism (Llama-3.2-1B-Instruct). This example shows that RepInf can retrieve evidence at the level of physical mechanisms rather than exact answer strings. The test question asks which object can serve as an electrical conductor; the correct answer is "a penny". The most influential training example instead asks what flows when one



(a) Original test input.



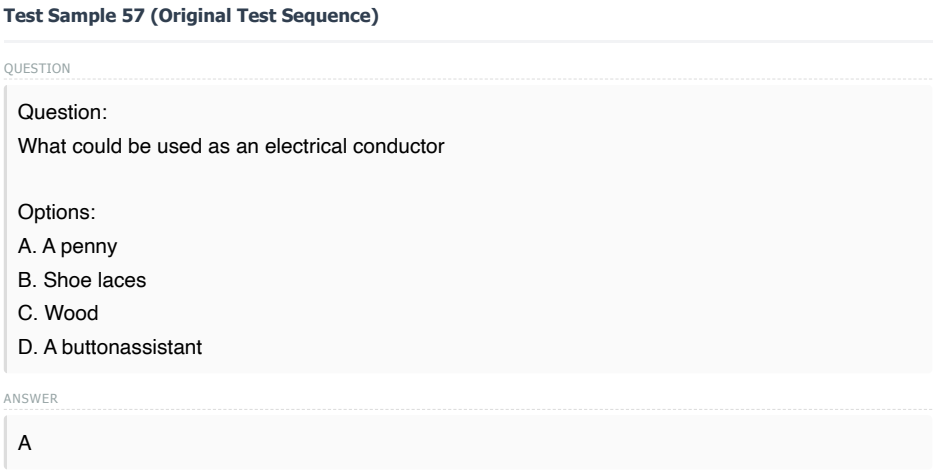
(b) Most influential training example (token-level influence, colored by dominant latent).



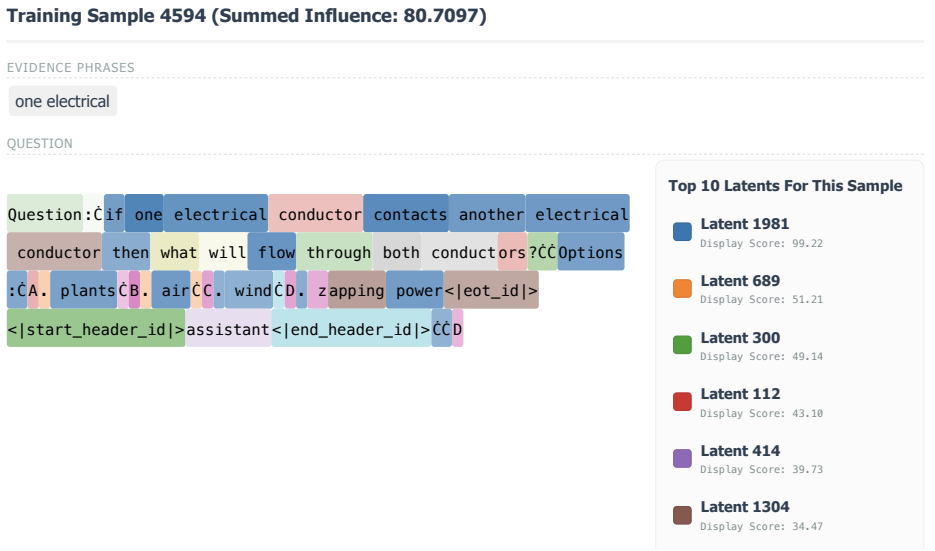
(c) The same training example, showing only latent 1338.

Figure 6: **Case Study A** (OpenBookQA, Llama-3.2-1B-Instruct): a test question (top), its most influential training example (middle), and one influential latent feature on that example (bottom). Colors denote the dominant influential latent feature; intensities denote per-token influence.

electrical conductor contacts another electrical conductor. It activates the same underlying concept: conductors permit the flow of electricity or power. The highlighted latent features concentrate on "one electrical conductor", "another electrical conductor", "flow", and "power". The influence signal is therefore localized to the mechanism-bearing phrase rather than to superficial overlap with the options.



(a) Test input.

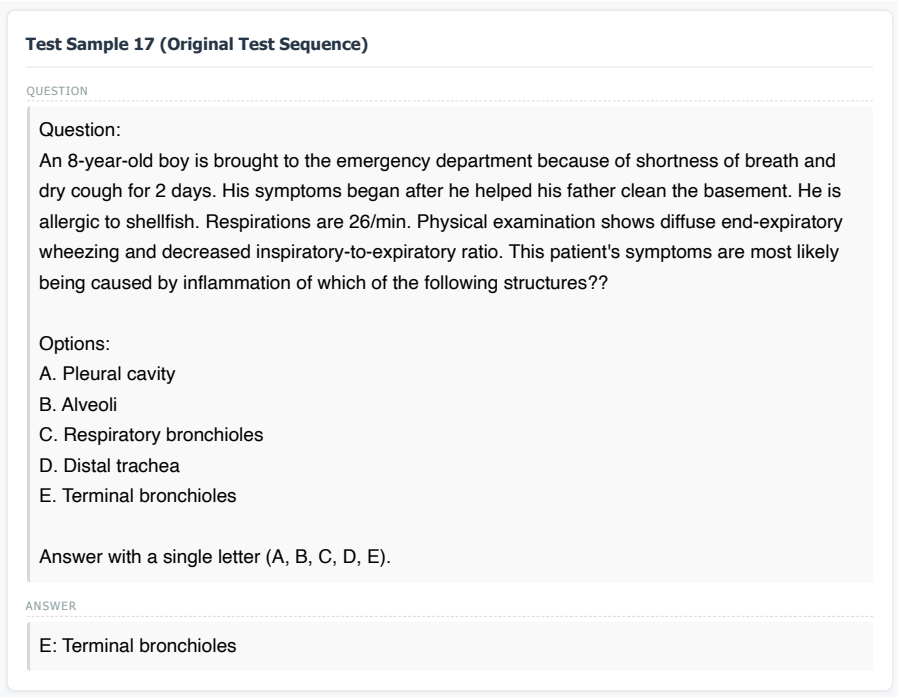


(b) Most influential training example (token-level influence).

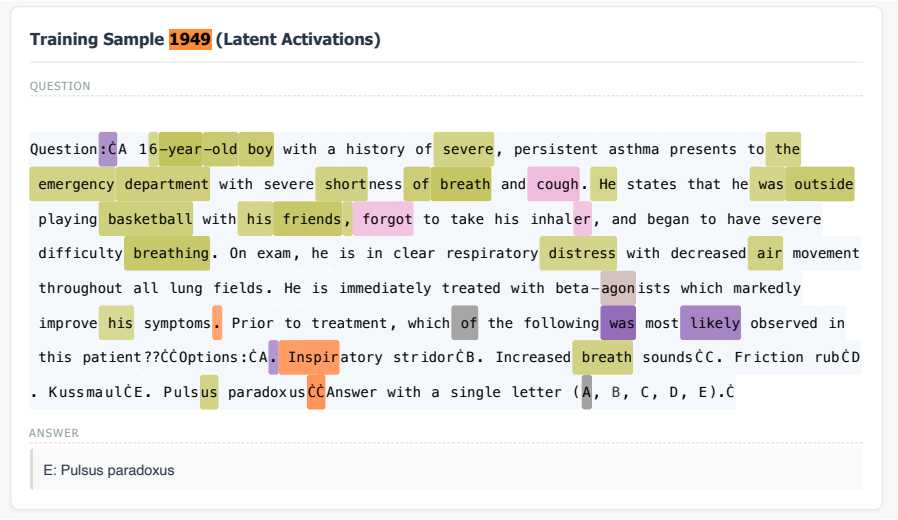
Figure 7: **Case Study B** (OpenBookQA, Llama-3.2-1B-Instruct): electrical conductor.

Case Study C: MedQA; asthma exacerbation and small-airway obstruction (Qwen2.5-1.5B-Instruct). The test example in Figure 8 describes an 8-year-old with shortness of breath and dry cough after environmental exposure, diffuse end-expiratory wheezing, and a decreased inspiratory-to-expiratory ratio. The correct choice is *terminal bronchioles*, consistent with inflammation and narrowing of the small airways in an obstructive process. The retrieved training example concerns a severe asthma presentation and asks for a classic physiologic finding (pulsus paradoxus). Although the questions differ, influential latents activate on shared features related to bronchospasm and respiratory distress (e.g., “breath”, “breathing”, “distress”, “air” movement). Representation-level influence can thus connect a test prediction to training evidence through a shared obstruction motif, without exact surface-form matching. This is an illustration of what an auditor would inspect; it is not a clinical validation.

Case Study D: reasoning augmentation sharpens attribution (deep-sea animals). In this case we alter the training example. Keeping the original question–answer pair fixed, we append a short rationale that makes the relevant semantic relation explicit. We then recompute representation-level influence and compare it with the baseline heatmap, to test whether the added reasoning shifts attribution from surface overlap toward concept-level evidence. The test question asks why frilled sharks



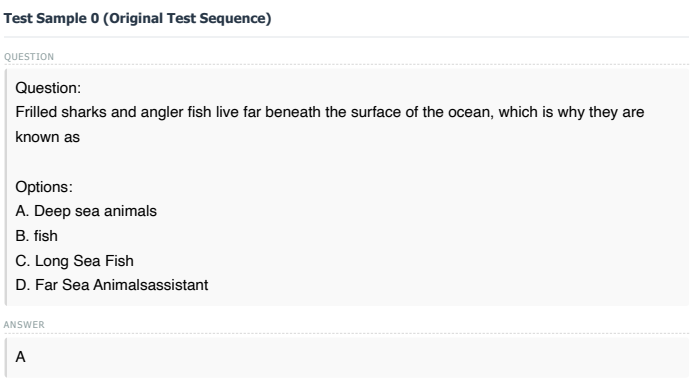
(a) Test input.



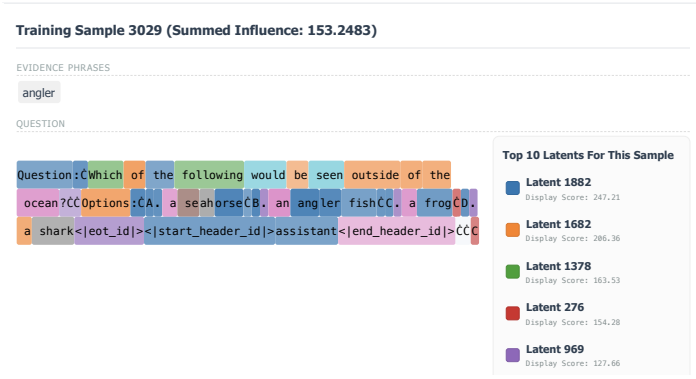
(b) Most influential training example (latent activations).

Figure 8: **Case Study C** (MedQA, Qwen2.5-1.5B-Instruct): asthma exacerbation and small-airway obstruction.

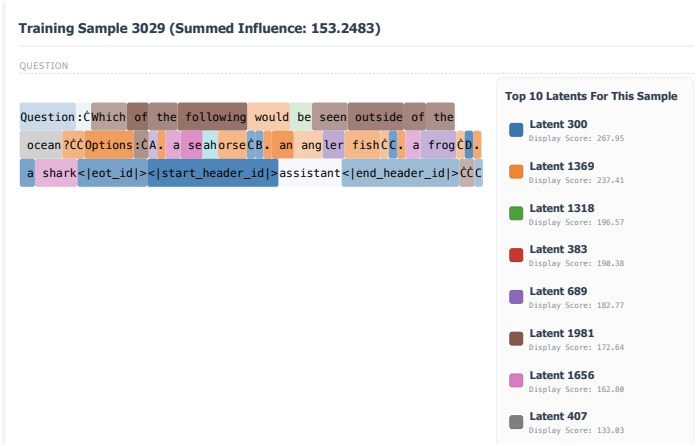
and angler fish are known as deep-sea animals, which requires connecting these species to organisms living far beneath the ocean surface. In the baseline heatmap (Figure 9b), the retrieved training example is relevant mainly because "angler fish" appears among its answer choices, and the attribution is diffuse across the multiple-choice sequence. After the rationale is added (Figure 9c), the influence becomes more concentrated: highlighted regions focus on "outside of the ocean" and on the animal tokens, especially "angler fish" and "shark". This suggests that reasoning augmentation helps the latent features align animal identity with the ocean-depth context.



(a) Test input.



(b) Most influential training example (token-level influence).



(c) Most influential training example with reasoning augmentation.

Figure 9: **Case Study D**: token-level influence before and after reasoning augmentation of the training example.

F Experimental Details

This appendix records the settings needed to reproduce the experiments in Section 5.

F.1 Models and fine-tuning

- **Base models.** Qwen2.5-1.5B-Instruct and Llama-3.2-1B-Instruct.

- **Fine-tuning.** We experiment with both full fine-tuning and LoRA. LoRA is faster and more memory-efficient; in our setting it also yields better task performance, consistent with reduced catastrophic forgetting under constrained compute.
- **Optimization.** AdamW with learning rate 2×10^{-4}, batch size 32, 4 epochs, maximum sequence length 512.
- **Evaluation.** Accuracy over multiple-choice options $\{A, B, C, D, E\}$, extracted from the model generation using a deterministic option parser.

F.2 Compute Resources

All fine-tuning, SAE training, and influence-computation experiments were run on GPU clusters equipped with NVIDIA A100 GPUs with 80 GB memory per GPU. Depending on the experiment, jobs used one or more A100 devices; this hardware was sufficient for the reported runs and associated ablations.

F.3 SAE training

- **Insertion layers.** Llama: layers 4–12; Qwen: layers 7-22. The reported "selected layer" is the best-performing layer within $\mathcal{L}_{\text{mid}}$.
- **Objective.** Reconstruction MSE on the instrumented hidden state, optionally trained jointly with the downstream task loss when enabled (see Table 10).
- **SAE size and sparsity.** Latent dimension $H \in \{2048, 1024, 512\}$ and Top-k sparsity $k \in \{256, 128, 64\}$.
- **Training data and steps.** `2 epochs`.

F.4 Influence computation and filtering

Gradient pre-filtering. Pre-filtering does not use only the last token. For each example z, we differentiate the supervised answer loss with respect to the SAE state at every sequence position. Because the answer loss is causal, earlier positions contribute. We aggregate the resulting latent-gradient vectors over positions and normalize them to obtain $\bar{g}(z)$. Candidates are then ranked by absolute cosine alignment with the test example,

$$\text{sim}(z_i, z_{\text{test}}) = \left| \frac{\langle \bar{g}(z_i), \bar{g}(z_{\text{test}}) \rangle}{\|\bar{g}(z_i)\| \, \|\bar{g}(z_{\text{test}})\|} \right|, \tag{12}$$

and we retain the top {1% / 5% / 10%} for exact influence computation.

Curvature and iHVP. For the intervention experiments (Section 5.3), we compute $H_{\theta_2}^{-1} g_{\text{test}}$ by conjugate gradient (CG) on the empirical Hessian, with damping 10^{-3} and a budget of 20 iterations (Appendix G.2). The raw local Hessian can carry non-positive curvature, and additional CG iterations did not help. The fidelity, counterfactual, and solver analyses (Sections 5.5 and 5.4) therefore use a shared per-test positive semi-definite empirical Fisher, evaluated over a grid of $\{8, 32, 128\}$ samples and damping $\{0.1, 1, 3\}$ (Table 9).

Table 9: **Solver diagnostics** (Section 5.4). Relative residual $\|Hx - g\|_2 / \|g\|_2$. RepInf–direct ρ is the agreement between RepInf mass and direct influence; rank ρ and top-5 Jaccard measure the stability of the influence ranking across the grid settings.

Solver setting	Rel. residual	RepInf–direct ρ	Rank ρ	Top-5 Jaccard
Empirical Fisher, matched solve	3.4×10^{-9}	–	–	–
Fisher grid: $\{8, 32, 128\}$ samples $\times$ damping $\{0.1, 1, 3\}$	$< 1.9 \times 10^{-7}$	0.973–0.979	1.000	1.000

Influence target. The intervention experiments compute influence on the test loss. The fidelity, counterfactual, and solver analyses use the target-logit objective, matched between RepInf and the direct baseline.

Table 10: Key experimental hyperparameters used throughout the appendix.

Category	Setting
Fine-tuning	LoRA; AdamW, lr 2×10^{-4}, batch size 32, 4 epochs, max length 512.
SAE	Layer sweep: Llama 4–15, Qwen 4–26; objective: reconstruction MSE; $H \in \{2048, 1024, 512\}$, $k \in \{256, 128, 64\}$.
Filtering	Latent-gradient cosine pre-filtering; retained fraction 1%–10%.
Influence	Zero-masking tests: iHVP via CG on batched empirical Hessian (batch size 8), damping 10^{-3}, 20 iterations. Fidelity/counterfactual/solver analyses: PSD empirical Fisher, $\{8, 32, 128\}$ samples, damping $\{0.1, 1, 3\}$.

G Scaling to 1B-Parameter LLMs

Scaling representation influence to 1B-parameter LLMs is challenging due to: (i) memory blow-ups from naively materializing second-order objects or large JVP tensors; and (ii) numerical instability when the empirical Hessian exhibits negative curvature, which can break iterative solvers.

G.1 Memory scaling via contraction order

We compute the representation influence for latent dimension j as

$$\mathcal{I}_j^r(z_{\text{train}}, z_{\text{test}}) = -\underbrace{g_{\text{test}}^\top H_{\theta_2}^{-1}}_{\text{iHVP term}} \underbrace{\text{JVP}(G, r_{\text{train}}, e_j)}_{\text{per-latent JVP}}, \tag{13}$$

where H_{θ_2} is the Hessian of the training objective restricted to downstream parameters θ_2 and g_{test} is the test gradient.

Avoiding Hessian materialization. We never explicitly construct H_{θ_2}. Instead, we rely on Hessian-vector products (HVPs) computed via automatic differentiation, enabling CG solves without storing the Hessian.

Avoiding Jacobian materialization. A naive implementation computes all $\{\text{JVP}(G, r_{\text{train}}, e_j)\}_{j=1}^H$ in one autograd call by concatenating basis vectors; this leads to catastrophic memory usage. Profiling shows the per-feature JVP stage can allocate $\sim$10–15 GB on GPT-2, and aggregation can add another $\sim$10–15 GB, pushing peak memory toward $\sim$40 GB. Dense offloading is worse: storing the full Jacobian scales as $\mathcal{O}(H \cdot |\theta_2|)$ and can reach hundreds of GB at 1B scale.

We therefore compute influence in a *streamed contraction* manner:

1. **Precompute iHVP once per test input.** Compute $s_{\text{test}}^\top \triangleq g_{\text{test}}^\top H_{\theta_2}^{-1}$ once and reuse it across all latent dimensions.
2. **Fuse JVP with contraction.** For each j, compute $r_j \triangleq \text{JVP}(G, r_{\text{train}}, e_j)$ and immediately contract $\mathcal{I}_j^r = -s_{\text{test}}^\top r_j$, then discard r_j.

This reduces storage from $\mathcal{O}(H \cdot |\theta_2|)$ to $\mathcal{O}(|\theta_2|)$ and avoids Jacobian storage. In our Llama-3.2-1B profiling, this reduces the peak from an estimated dense-materialization regime ($\sim$400 GB) to a practical tens-of-GB regime.

G.2 Stabilizing iHVP under noisy curvature

We obtain $s_{\text{test}}^\top$ by solving $H_{\theta_2} v = g_{\text{test}}$ using CG, where Hv is computed via autograd HVPs. Although CG assumes symmetric positive definite curvature, too-small batches yield noisy empirical Hessians with occasional negative curvature, manifested as $p^\top H p < 0$ and early termination or degenerate solutions. Empirically, batch size ≤ 4 can cause a large fraction of influence entries to collapse to near-zero due to solver instability. We therefore define H_{θ_2} using a batched empirical loss with batch size 8 for stable curvature in all 1B runs.

Table 11: Runtime and memory breakdown for 1B-scale influence computation per $(z_{\text{train}}, z_{\text{test}})$ pair, based on our profiling. "JVP sweep" corresponds to streamed contraction (no Jacobian storage).

Component	Time (s)	Peak GPU mem
Forward/backward	$\sim$1	–
CG iHVP solve	$\sim$5	–
Llama-3.2-1B JVP sweep	$\sim$35	$\sim$50 GB
Qwen-2.5-1.5B JVP sweep	$\sim$60	$\sim$100 GB

Table 12: Runtime breakdown (seconds) per $(z_{\text{train}}, z_{\text{test}})$ pair using the derivative-swapping formulation on Qwen-2.5-1.5B.

Operation	Time (s)
Train forward	0.2388
Compute batched iHVP	3.4927
Gradient projection	0.2228
Sensitivity backprop	0.4834
Attribution & aggregation	0.0001
Total	**4.4378**

G.3 Compute-time scaling and accelerations

Compute is dominated by the latent loop if we explicitly sweep per-feature JVPs. Without further reductions, on Llama-3.2-1B the per-$(z_{\text{train}}, z_{\text{test}})$ cost decomposes into roughly $\sim$1 s (forward/backward), $\sim$5 s (CG iHVP), and $\sim$30–60 s (full JVP sweep, depending on model and SAE size), which is infeasible when repeated over many training candidates.

We apply two practical reductions:

1. **Task-sized SAE.** Since MedQA fine-tuning has $\sim$9k training samples, we train effective SAEs with latent size $H = 512$ and Top-$k = 64$, reducing the JVP sweep to $\sim$30 s and total to $\sim$40 s per candidate training example.
2. **Gradient-similarity pre-filtering.** We compute exact influence only on the top 1%–10% training candidates ranked by gradient cosine similarity.

With these reductions, end-to-end influence computation is $\sim$1 hour per test sample in our setting.

G.4 Further acceleration via derivative swapping (Section 4.4)

Beyond streamed JVP contraction, we exploit the derivative-swapping reformulation of Section 4.4, which eliminates the explicit per-feature JVP sweep. Recall that representation influence can be written as

$$\mathcal{I}^r(z_{\text{train}}, z_{\text{test}}) = -s_{\text{test}}^\top G(r_{\text{train}}), \tag{14}$$

where $s_{\text{test}}^\top \triangleq g_{\text{test}}^\top H_{\theta_2}^{-1}$ and $G(r_{\text{train}}) = \nabla_{\theta_2} \ell(h_{\theta_2}(r_{\text{train}}), y_{\text{train}})$. Instead of iterating over j and computing $\text{JVP}(G, r_{\text{train}}, e_j)$, we directly differentiate the scalar projection with respect to r_{train}:

$$\nabla_{r_{\text{train}}} \big(s_{\text{test}}^\top G(r_{\text{train}})\big). \tag{15}$$

This single reverse-mode pass produces the full latent-level influence vector $\{\mathcal{I}_j^r\}_{j=1}^H$, replacing an $\mathcal{O}(H)$ JVP loop with one sensitivity backpropagation.

Overall compute comparison. Table 14 compares the streamed-JVP pipeline with the derivative-swapping formulation. The latter removes the $\sim$30–60 s JVP sweep (Table 11) and reduces total per-pair runtime to $\approx$ 2–4 seconds, corresponding to a 13–18$\times$ practical speedup.

Combined with batched iHVP solves and gradient-similarity pre-filtering, this reduction makes representation-level influence computation practical for 1B–1.5B parameter LLMs under commodity multi-GPU constraints.

Table 13: Runtime breakdown (seconds) per $(z_{\text{train}}, z_{\text{test}})$ pair using the derivative-swapping formulation on Llama-3.2-1B.

Operation	Time (s)
Train forward	0.0947
Compute batched iHVP	1.6803
Gradient projection	0.1443
Sensitivity backprop	0.2460
Attribution & aggregation	0.0001
Total	**2.1654**

Table 14: End-to-end per-$(z_{\text{train}}, z_{\text{test}})$ runtime comparison (seconds).

Model	Streamed JVP	Derivative swapping
Llama-3.2-1B	$\sim$40	2.17
Qwen-2.5-1.5B	$\sim$60	4.44

H Derivations

H.1 Exact Path-Integral Decomposition

This section expands the presentation in Section 4.3 with two details. First, a naive directional derivative $\partial G/\partial r_{\text{train}}^{(j)}$ measures only local sensitivity; for sparse SAE latents, the natural baseline is $r_0 = \mathbf{0}$, corresponding to the inactive-feature state, so $\Delta G = G(r_{\text{train}}) - G(\mathbf{0})$ captures the transition from inactive to active features. Second, the exact path-integral derivation below makes explicit why the activation multiplier appears in the final attribution: each feature contributes in proportion to both its realized activation magnitude and its directional sensitivity along the path, so inactive coordinates contribute zero even when the Jacobian itself is nonzero.

Let $G : \mathbb{R}^h \to \mathbb{R}^{|\theta_2|}$ be continuously differentiable. Fix $r_0, r_{\text{train}} \in \mathbb{R}^h$ and define the straight-line path

$$r(\alpha) = r_0 + \alpha(r_{\text{train}} - r_0), \qquad \alpha \in [0, 1]. \tag{16}$$

Path integral identity. By the chain rule,

$$\frac{d}{d\alpha} G(r(\alpha)) = J_G(r(\alpha)) \, \frac{dr(\alpha)}{d\alpha} = J_G(r(\alpha))(r_{\text{train}} - r_0), \tag{17}$$

where $J_G(r)$ denotes the Jacobian of G. Applying the fundamental theorem of calculus,

$$\begin{aligned} G(r_{\text{train}}) - G(r_0) &= \int_0^1 \frac{d}{d\alpha} G(r(\alpha)) \, d\alpha \\ &= \int_0^1 J_G(r(\alpha))(r_{\text{train}} - r_0) \, d\alpha. \end{aligned} \tag{18}$$

Coordinate decomposition. Using the canonical basis expansion

$$r_{\text{train}} - r_0 = \sum_{j=1}^{h} (r_{\text{train}}^{(j)} - r_0^{(j)}) e_j, \tag{19}$$

and linearity of integration,

$$G(r_{\text{train}}) = G(r_0) + \sum_{j=1}^{h} (r_{\text{train}}^{(j)} - r_0^{(j)}) \left(\int_0^1 J_G(r(\alpha)) e_j \, d\alpha \right). \tag{20}$$

Equations Eq. (18)–Eq. (20) hold exactly under the sole assumption that G is continuously differentiable.

H.2 Derivative Swapping with Activation Weighting

While the Jacobian-vector product (JVP) formulation in Eq. (7) provides a rigorous attribution mechanism, computing it naively is computationally prohibitive. Evaluating the JVP for each latent dimension $j \in \{1, \dots, h\}$ requires h separate forward-mode passes. For a sparse autoencoder with thousands of features, this $\mathcal{O}(h)$ complexity leads to severe computational bottlenecks and memory allocation limits.

We remove this bottleneck by exploiting the structure of the computation graph and the symmetry of mixed partial derivatives (Clairaut's Theorem). We transition from evaluating h directional derivatives to performing a single reverse-mode gradient pass.

The Independence of Upstream Activations. Recall that the model is split at layer l, such that the latent representation $r_{\text{train}} = h_{\theta_1}(x_{\text{train}})$ is produced entirely by upstream parameters θ_1, while we differentiate with respect to the downstream parameters $\theta_2 = \{\theta : \theta_{>l}\}$. Because r_{train} serves as an input to the downstream computation and does not depend on θ_2, it is treated as a constant with respect to θ_2. Therefore, the parameter gradient of the activation is strictly zero:

$$\frac{\partial r_{\text{train}}^{(j)}}{\partial \theta_2} = \mathbf{0}. \tag{21}$$

Equivalence of Inside and Outside Weighting. Let $\Delta^{(j)} = \frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}$ denote the sensitivity of the training loss to the j-th latent feature. In our implementation, we define a feature's loss contribution as $s^{(j)} = r_{\text{train}}^{(j)} \Delta^{(j)}$. Taking the gradient of this contribution with respect to the downstream parameters θ_2, we apply the product rule:

$$\frac{\partial s^{(j)}}{\partial \theta_2} = \frac{\partial}{\partial \theta_2}\left(r_{\text{train}}^{(j)} \Delta^{(j)}\right) = \underbrace{\frac{\partial r_{\text{train}}^{(j)}}{\partial \theta_2}}_{=\mathbf{0}} \Delta^{(j)} + r_{\text{train}}^{(j)} \frac{\partial \Delta^{(j)}}{\partial \theta_2}. \tag{22}$$

Because the first term vanishes due to the computation graph cut Eq. (21), we are left with:

$$\frac{\partial s^{(j)}}{\partial \theta_2} = r_{\text{train}}^{(j)} \frac{\partial}{\partial \theta_2}\left(\frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right) = r_{\text{train}}^{(j)} \frac{\partial G(r_{\text{train}})}{\partial r_{\text{train}}^{(j)}}, \tag{23}$$

where $G(r_{\text{train}}) = \nabla_{\theta_2} \ell$. This proves that multiplying the activation "inside" the gradient tracker is mathematically identical to multiplying by the activation "outside" the mixed-partial term, aligning the implementation with the first-order attribution of Definition 4.2.

Order Swapping for Constant-Time Evaluation. We now apply the influence contraction with the inverse-HVP test signal $v = H_{\theta_2}^{-1} g_{\text{test}}$. The influence for neuron j becomes:

$$\mathcal{I}_j^r(z_{\text{train}}, z_{\text{test}}) = -\left(\frac{\partial s^{(j)}}{\partial \theta_2}\right)^\top v = -\, r_{\text{train}}^{(j)} \left(\frac{\partial}{\partial \theta_2} \frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right)^\top v. \tag{24}$$

By the symmetry of mixed partial derivatives (assuming standard regularity conditions on the loss surface), we can swap the order of differentiation:

$$\left(\frac{\partial}{\partial \theta_2} \frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right)^\top v = \frac{\partial}{\partial r_{\text{train}}^{(j)}}\left(\left(\frac{\partial \ell}{\partial \theta_2}\right)^\top v\right) = \frac{\partial}{\partial r_{\text{train}}^{(j)}}\Big(G(r_{\text{train}})^\top v\Big). \tag{25}$$

Define the scalar projection $P = G(r_{\text{train}})^\top v$. This scalar represents the alignment between the training parameter gradients and the test signal. Eq. (25) reveals that the sensitivity for neuron j is exactly the partial derivative of P with respect to $r_{\text{train}}^{(j)}$.

Consequently, the entire vector of sensitivities for all h neurons can be computed simultaneously by taking a single gradient of the scalar P with respect to the latent representation r_{train}:

$$\text{Sensitivity Vector} = \nabla_{r_{\text{train}}} P = \nabla_{r_{\text{train}}}\Big(G(r_{\text{train}})^\top v\Big). \tag{26}$$

Combining Eq. (24) and Eq. (26), the final influence vector for all latent features is obtained via an element-wise product (Hadamard product, denoted by $\odot$) with the realized activations:

$$\vec{\mathcal{I}}^{r}(z_{\text{train}}, z_{\text{test}}) = -\left(\nabla_{r_{\text{train}}} P\right) \odot r_{\text{train}}. \tag{27}$$

This formulation requires only *two* backward passes total: one to construct the computation graph for $G(r_{\text{train}})$, and a second to compute the gradient of the scalar projection P with respect to r_{train}. The time complexity with respect to the feature dimension drops from $\mathcal{O}(h)$ to $\mathcal{O}(1)$. In practice, this reverse-mode (vector–Jacobian product) formulation enables the simultaneous computation of influences across all latent features and batch samples, allowing our method to scale to 1B-parameter models and SAEs with tens of thousands of features without materializing explosive Jacobians.