Correcting Influence: Unboxing LLM Outputs with Orthogonal Latent Spaces

Shixing Yu
Electrical and Computer Engineering
Cornell Tech
sy774@cornell.edu

Promit Ghosal
Department of Statistics
University of Chicago
promit@uchicago.edu

Kyra Gan
Operations Research and Industrial Engineering
Cornell Tech
kyragan@cornell.edu

Abstract

A critical step for reliable large language models (LLMs) use in healthcare is to attribute predictions to their training data, akin to a medical case study. This requires token-level precision: pinpointing not just which training examples influence a decision, but which tokens within them are responsible. While *influence functions* offer a principled framework for this, prior work is restricted to *autoregressive* settings and relies on an implicit assumption of *token independence*, rendering their identified influences unreliable. We introduce a flexible framework that infers *token-level influence* through a latent mediation approach for *general prediction tasks*. Our method attaches *sparse autoencoders* to any layer of a pretrained LLM to learn a basis of approximately independent latent features. Unlike prior methods where influence decomposes additively across tokens, influence computed over latent features is inherently *non-decomposable*. To address this, we introduce a novel method using *Jacobian-vector products*. Token-level influence is obtained by propagating latent attributions back to the input space via token activation patterns. We scale our approach using efficient inverse-Hessian approximations. Experiments on medical benchmarks show our approach identifies sparse, interpretable sets of tokens that *jointly* influence predictions. Our framework enhances trust and enables model auditing, generalizing to any high-stakes domain requiring transparent and accountable decisions.

1 Introduction

The deployment of LLMs in high-stakes domains like healthcare hinges on a critical and unmet requirement: the ability to audit a model's reasoning by tracing its predictions directly to the evidence in its training data. This need for verifiability is urgent, as LLMs are increasingly explored for clinical tasks such as diagnostic support and treatment planning, where errors can have severe consequences [Singhal et al., 2023, Topol, 2019]. Without this capability—akin to a clinician demanding the source for a medical decision—LLMs remain unverifiable black boxes. Their tendency to hallucinate [Ji et al., 2023] and their susceptibility to spurious correlations present in training data [Oberst and Sontag, 2019] pose significant safety risks, undermining the trust required for clinical adoption [Futoma et al., 2020, Ghassemi et al., 2021].

This fundamental need for evidence-based reasoning is not adequately addressed by prevailing interpretability methods. Techniques like Chain-of-Thought prompting generate rationales that are

Preprint.

often post hoc justifications rather than faithful reflections of the model's true decision process [Turpin et al., 2023, Barez et al., 2025]. Other popular approaches, such as attention visualization [Wiegreffe and Pinter, 2019, Jain and Wallace, 2019] or gradient-based feature attribution [Sundararajan et al., 2017a], are limited to explaining a single forward pass of a model. They operate within the context of a given input, providing no insight into how prior training experiences shaped the model's fundamental behavioral patterns and knowledge [Feldman and Zhang, 2020]. This represents a critical limitation for clinical deployment, where the ability to pinpoint the exact training evidence behind a prediction—not just generate plausible-sounding rationales—is essential for medical professionals to validate the model's logic against established knowledge, fact-check its conclusions, and ultimately build the trust required for adoption in safety-critical settings.

A principled framework for addressing this question lies in *influence functions* (IFs), a tool from robust statistics that explains how a model's predictions depend on its training data [Hampel, 1974]. This approach treats the model as an empirical entity shaped by its dataset, enabling one to trace a final prediction back to influential training points [Koh and Liang, 2017]. Recent work has successfully scaled this approach to modern LLMs, demonstrating its potential to reveal generalization patterns by attributing influence down to the token level [Grosse et al., 2023]. However, a key limitation persists: the IF framework assumes independence among the components of the objective (e.g., tokens in an autoregressive prediction task in prior work). This assumption is necessary for influence scores to be meaningfully interpretable, as it ensures that the relative difference in influence between components is well-defined. In practice, the tokens within LLMs are highly correlated. Thus, prior implementations, while powerful, produce influence estimates that are theoretically unsound and difficult to interpret [Basu and Echenique, 2020, Tsimpoukelli et al., 2021].

We introduce a robust framework that infers *token-level* influence on test predictions via latent mediation, enabling more reliable influence estimation. Building on recent monosemanticity research [Bricken et al., 2023, Templeton et al., 2024a] and disentangled representation learning [Wang et al., 2024], our method leverages that neural networks decompose into semantically meaningful, independent components. Our method generalizes to *general prediction tasks* by propagating influence through disentangled latent spaces where features exhibit statistical independence, critical for reliable influence estimation. Our contributions are fourfold:

1. **Unified sample- and feature-level influence**: We extend influence analysis beyond the isolated-token paradigm of prior work to model the *joint influence of tokens* within training sample-label pairs. By propagating influence from latent features to input tokens through their joint activation patterns, we attribute predictions to specific token combinations in the training data while leveraging monosemantic structure. Unlike methods treating neurons as atomic units, we recognize meaningful computation occurs at interpretable feature level spanning multiple neurons.
2. **Stable, independent feature extraction via sparse autoencoders (SAEs)**: We use SAEs [Gao et al., 2024, Cunningham et al., 2023, Marks et al., 2024, Cong et al., 2023] as an interpretability component to produce sparse, approximately orthogonal latent features at an intermediate layer. We then compute influence scores with respect to these latent features, improving the stability and interpretability of training-data attributions.
3. **Scalable non-decomposable influence estimation via derivative swapping**: Latent-level influence is holistically interdependent and lacks the additive decomposition of token-level approaches. While naive Jacobian-vector product (JVP) evaluation requires an $\mathcal{O}(d_l)$ forward-mode pass per feature , we exploit Clairaut's Theorem to swap the derivative order. This gradient-derivative formulation restructures the computation into a single reverse-mode pass, reducing complexity to $\mathcal{O}(1)$ and achieving a $10\times$ to $20\times$ practical speedup.
4. **Large-scale empirical validation on medical and general reasoning**: We evaluate our framework on 1B and 1.5B parameter models (Llama-3.2 and Qwen2.5) across multiple QA datasets. Rigorous necessity and sufficiency ablations show our method isolates compact, highly influential circuits that systematically outperform activation magnitude and frequency baselines. Furthermore, qualitative evaluations successfully map predictions back to mechanistically aligned training evidence.

By unifying data-level and feature-level attribution, our approach offers a principled pathway toward transparent, trustworthy, and deployable LLMs for high-stakes domains, with additional potential for large-scale training data auditing and diagnostics, which we further discuss in Section 5. Section 2 introduces our notation and preliminaries on IF and JVP. We then describe our method in Section 3 and evaluate its performance in Section 4. Additional related works is included in Appendix A.

2 Preliminaries

Given a training dataset $\mathcal{D} = \{z_i = (x_i, y_i)\}_{i=1}^n$ i.i.d. drawn from an unknown distribution, with input $x_i \in \mathcal{X}$ and label $y_i \in \mathcal{Y}$. A model $h_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters $\theta \in \mathbb{R}^p$ is trained by minimizing the empirical risk $\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^n \ell(h_\theta(x_i), y_i)$, where $\ell(\cdot, \cdot)$ is the loss function.

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:

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

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 and Parks, 2002]. Let $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 \ell(h_{\hat{\theta}}(x_i), y_i)$ be the Hessian of the empirical risk evaluated at $\hat{\theta}$, then

$$\text{IF}_{\hat{\theta}}(z_{\text{train}}) = \left. \frac{d\hat{\theta}_{\epsilon, z_{\text{train}}}}{d\epsilon} \right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{train}}), y_{\text{train}}). \tag{2}$$

Influential Training Samples on Test Prediction Since $\text{IF}_{\hat{\theta}}(z_{\text{train}})$ is a high-dimensional vector, it is often difficult to interpret directly. To obtain a more concrete measure, we convert this parameter-space influence into a scalar quantity by measuring its effect on a specific model output. This is done by projecting the influence vector onto the gradient of a chosen function, such as the loss or the logits for a test example $z_{\text{test}} = (x_{\text{test}}, y_{\text{test}})$. Applying the Chain Rule, we can compute the scalar *influence* of upweighting z_{train} on the loss at z_{test} as follows:

$$\mathcal{I}(z_{\text{train}}, z_{\text{test}}) = -\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{test}}), y_{\text{test}})^\top H_{\hat{\theta}}^{-1} \quad \nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{train}}), y_{\text{train}}). \tag{3}$$

This provides an interpretable measure to trace predictions back to influential training samples.

Influential Tokens on Test Prediction in Autoregressive Tasks In *autoregressive* tasks, the loss function decomposes additively across tokens, which enables the direct computation of token-level influence. This additive structure permits the gradient and Hessian in the influence function to be similarly decomposed, allowing the influence of individual training tokens to be derived explicitly. Let $\{x_1, \cdots, x_T\}$ to denote the T tokens in x_{train}. Then, Eq. (3) can be rewritten as

$$\mathcal{I}(z_{\text{train}}, z_{\text{test}}) = -\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{test}}), y_{\text{test}})^\top H_{\hat{\theta}}^{-1} \quad \nabla_\theta \sum_{t=1}^T \ell(h_{\hat{\theta}}(x_t), y_t). \tag{4}$$

Thus, the per-token influence score is defined as [Grosse et al., 2023]:

$$\mathcal{I}_t(z_{\text{train}}, z_{\text{test}}) = -\nabla_\theta \ell(h_{\hat{\theta}}(x_{\text{test}}), y_{\text{test}})^\top H_{\hat{\theta}}^{-1} \quad \nabla_\theta \ell(h_{\hat{\theta}}(x_t), y_t). \tag{5}$$

Remark 2.1 (Problems with Existing Per-Token Influence). However, the decomposition in Eq. (5) is restricted to an autoregressive task and implicitly assumes that the tokens in each training sample are independent. This is violated in text, as tokens are highly correlated. Consequently, the influence score for a token captures not only its own effect but also the confounded effects of correlated tokens in its context. This entanglement breaks the core interpretation of the score as measuring the isolated effect of a single token, rendering the estimates unreliable. To address this, we propose augmenting the LLM with modified SAEs (Section 3), which enable influence estimation in a structured latent space where these dependencies can be better controlled.

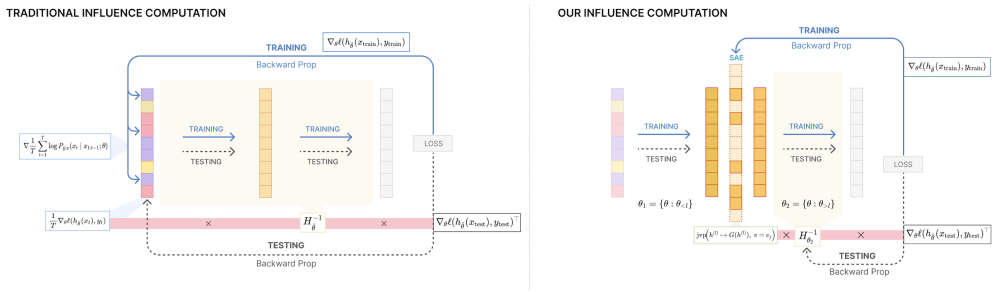


Figure 1: **Framework overview.** Traditional influence functions operate in the input space, assuming token independence and decomposable losses. Our method introduces a sparse autoencoder at an intermediate layer, splitting the model into upstream and downstream parts. Influence is then computed at the representation level using JVPs, enabling stable per-feature attributions and linking test predictions to interpretable sparse features.

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:

$$\mathrm{JVP}(F, x, v) \;=\; \frac{d}{d\varepsilon} F(x + \varepsilon v)\Big|_{\varepsilon=0} \;=\; J_F(x)\, v, \tag{6}$$

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.

3 Methodology

We now detail our framework that infers *token-level* influence on test predictions via a latent mediation approach, enabling more reliable influence estimation for general prediction tasks. This section presents the core components of our approach: 1) augmenting LLMs with SAEs to obtain more interpretable latent representations (Section 3.1), 2) computing influence scores over these latent features rather than directly on input tokens (Section 3.2), and 3) efficiently implementing this computation via Jacobian-vector products (Section 3.4) while maintaining the ability to propagate attributions back to the input space. Figure 1 provides an overview of the complete framework.

3.1 Augmenting LLM with Sparse Autoencoders for Independent Features

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

$$r = \sigma(W_{\text{enc}}(x^l - b_{\text{pre}}) + b_{\text{enc}}), \tag{7}$$

$$\tilde{x}^l = W_{\text{dec}} r + b_{\text{pre}}, \tag{8}$$

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].

Let r_j be the activation of feature j, forming the basis for our feature-level influence analysis.

Remark 3.1 (SAE for Improved Influence Estimation). Classical influence functions assume independent samples, an assumption broken by token-level representations, where strong sequential correlations invalidate estimates. SAEs, by contrast, induce a latent representation comprised of approximately independent features,[1] each corresponding to a semantically meaningful concept.

[1]To promote orthogonality, we also experimented with adding an orthogonality constraint. However, since it led to only negligible performance gains, we ultimately excluded this term.

While these features are not strictly independent, their distributions are regularized toward comparable sparsity patterns—significantly closer to the independent structure assumptions underlying influence estimation. This alignment makes SAE-based latents far more suitable for influence score interpretation than raw token-level attribution, where strong sequential dependencies violate core requirements of the influence framework. Although influence functions technically require features to be identically distributed for comparable scaling across components, this represents a second-order concern. We expect SAEs to produce latent representations with more comparable distribution scales than raw tokens, thereby providing more reliable influence estimates.

3.2 Influence Functions on Latent Representation

Classical influence functions applied directly to correlated text tokens (Eq. (5)) are problematic due to strong sequential dependencies. To address this, we compute influence on *latent features* rather than raw tokens. Consider an intermediate activation $h^{(l)} \in \mathbb{R}^{d_l}$ at layer l, which may correspond to the output of an attention head, MLP block, or residual stream in a transformer [Elhage et al., 2021]. Mechanistic interpretability studies suggest that such representations often encode semantically meaningful features causally linked to final predictions [Wang et al., 2023, Meng et al., 2022]. To attribute influence to individual neurons (latent coordinates) within the representation r_{train} (corresponding to $h^{(l)}$), we must relate neuron-level effects directly to the *training gradient* that appears in influence functions.

As illustarated in Figure 1, we split the model parameters into two parts: $\theta = (\theta_1, \theta_2)$, where $\theta_1 = \{\theta : \theta_{<l}\}$ comprises all parameters *up to and including layer* l, mapping a raw input sequence x to an intermediate representation $r = h_{\theta_1}(x)$; $\theta_2 = \{\theta : \theta_{>l}\}$ comprises the remaining parameters *after layer* l, 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 the original tokens.

Let $r_{\text{train}} = h_{\theta_1}(x_{\text{train}})$ and $r_{\text{test}} = h_{\theta_1}(x_{\text{test}})$ denote the latent representation of the training and testing inputs, x_{train} and x_{test}, respectively. Define the corresponding latent-space data points as $z^r_{\text{train}} = (r_{\text{train}}, y_{\text{train}})$ and $z^r_{\text{test}} = (r_{\text{test}}, y_{\text{test}})$.

The *representation-level influence function* is defined as:

$$\mathcal{I}(z^r_{\text{train}}, z^r_{\text{test}}) = -\underbrace{\nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{test}}), y_{\text{test}})}_{=:\, g_{\text{test}}}^{\top} H_{\theta_2}^{-1} \quad \underbrace{\nabla_{\theta_2}\ell(h_{\theta_2}(r_{\text{train}}), y_{\text{train}})}_{=:\, g_{\text{train}}}, \tag{9}$$

where H_{θ_2} is the Hessian of the training loss w.r.t. θ_2.

A crucial **philosophical asymmetry** now arises: IFs measure how a *training point* affects the loss on a *test point*. The test point serves as a fixed evaluation context—we care about its loss, but we do not attribute influence to its internal structure. Consequently, while both z^r_{train} and z^r_{test} are latent representations, our goal is to decompose the training-side gradient g_{train} into contributions from individual latent coordinates of r_{train}. In contrast, g_{test} requires no decomposition; it is obtained by a single backward pass through θ_2 after computing r_{test} via θ_1. To make this asymmetry explicit in notation, we use the notation $\mathcal{I}^r(z^r_{\text{train}}, z_{\text{test}})$:

$$\mathcal{I}^r(z^r_{\text{train}}, z_{\text{test}}) = \mathcal{I}(z^r_{\text{train}}, (h_{\theta_1}(x_{\text{test}}), y_{\text{test}})) = \mathcal{I}(z^r_{\text{train}}, z^r_{\text{test}}), \tag{10}$$

where the superscript r on $\mathcal{I}$ signals that attribution targets the *training representation*. The core challenge lies in attributing g_{train} to individual latent coordinates of r_{train} in a manner that reflects their actual computational attribution,[2] which we articulate in Section 3.3. Now, by mapping these influential features back to the specific words that *activate* them, our explanations more faithfully capture the model's internal reasoning—moving beyond isolated token attributions toward coherent, feature-driven interpretability.

3.3 Feature-Level Attribution via Integrated Gradients

Recall that our goal is to attribute the training-side gradient to individual latent coordinates of r_{train}. To make this dependence explicit and to evaluate it at different inputs, we define a function G that

[2]Decomposing g_{test} would answer a different question (e.g., "which test features make it sensitive to training data"). The Hessian H_{θ_2} depends solely on training data, and the perturbation is applied to the training point; thus, r_{test}'s internal composition is irrelevant for the attribution we seek.

any intermediate representation r to the corresponding gradient on θ_2:

$$G(r) \;=\; \nabla_{\theta_2}\ell(h_{\theta_2}(r), y_{\text{train}})\,, \tag{11}$$

with the loss evaluated using the fixed training label y_{train}. By construction, $g_{\text{train}} = G(r_{\text{train}})$.

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 approach would be to consider the directional derivative $\partial G/\partial r^{(j)}$, which measures how an infinitesimal perturbation to neuron j affects the gradient. While this captures local sensitivity, it does not tell us how much that neuron's *activation* contributes to the actual value of $G(r)$ when transitioning from an inactive to an active state. This issue is commonly addressed by defining contributions relative to a baseline representing the absence of the features (e.g., integrated gradients [Sundararajan et al., 2017b], Shapley values [Lundberg and Lee, 2017]). For sparse representations learned by SAEs, the natural baseline is $r_0 = \mathbf{0}$, corresponding to the manifold where "all features are inactive." We therefore consider the change $\Delta G = G(r_{\text{train}}) - G(\mathbf{0})$, which captures the effect of turning on the active features.

To decompose ΔG into per-neuron attributions, we adopt the axiomatic framework of integrated gradients [Sundararajan et al., 2017b]. This method attributes the output of a scalar model to its input features by integrating the gradients along a straight-line path from a baseline to the realized representation. We parametrize this straight-line path by $\alpha \in [0, 1]$, and, abusing the notation, denote it as $r(\alpha)$. Because our baseline is the origin ($r_0 = \mathbf{0}$), this straight-line path is the scaling function $r(\alpha) = \alpha r_{\text{train}}$. By the chain rule of calculus, the rate of change of the gradient along this path is driven entirely by the representation itself:

$$\frac{d}{d\alpha}G(r(\alpha)) = J_G(\alpha r_{\text{train}})\frac{dr(\alpha)}{d\alpha} = J_G(\alpha r_{\text{train}})r_{\text{train}}. \tag{12}$$

Applying the fundamental theorem of calculus, we integrate this derivative to express the total gradient change:

$$G(r_{\text{train}}) - G(\mathbf{0}) = \int_0^1 J_G(\alpha r_{\text{train}})\, r_{\text{train}}\, d\alpha, \tag{13}$$

where $J_G(r) \in \mathbb{R}^{|\theta_2|\times d_l}$ is the Jacobian of G with respect to r. Expanding the product coordinate-wise yields an exact additive latent decomposition:

$$G(r_{\text{train}}) = G(\mathbf{0}) + \sum_{j=1}^{d_l} r_{\text{train}}^{(j)} \left(\int_0^1 J_G(\alpha r_{\text{train}})\, e_j\, d\alpha\right). \tag{14}$$

We provide a detailed proof of this decomposition in Appendix E.1.

Computing the exact integral is expensive. To obtain a scalable estimator, we approximate the integral by evaluating the Jacobian at a particular point along the path, $r^\star$. This represents a backward first-order Taylor expansion from the realized activations, defining our per-neuron latent contribution as:

$$\Delta G_j(r^\star) \;:=\; r_{\text{train}}^{(j)}\, J_G(r^\star)\, e_j. \tag{15}$$

By choosing $r^\star = r_{\text{train}}$, this yields a computationally-efficient approximation because it evaluates the Jacobian using activations already materialized during the standard forward pass. Moreover, the multiplicative factor $r_{\text{train}}^{(j)}$ guarantees that contributions correctly vanish for inactive features, satisfying a desirable *activation* sensitivity property.

Definition 3.2 (Neuron-Level Influence)**.** Using Eq. (15) to adjust for activation attribution, the influence score attributed to neuron j in the training representation r_{train} is defined as

$$\mathcal{I}_j^r(z_{\text{train}}^r, z_{\text{test}}) = -\, g_{\text{test}}^\top H_{\theta_2}^{-1}\, \Delta G_j(r_{\text{train}}), \tag{16}$$

where $g_{\text{test}} = \nabla_{\theta_2}\ell(h_{\theta_1}(x_{\text{test}}), y_{\text{test}})$ and H_{θ_2} is the downstream Hessian.

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, as quantified by the scaled Jacobian column $J_G(r_{\text{train}})\, e_j$. The sign convention ensures that positive influence corresponds to improved train–test gradient alignment. Consequently, the score provides a causal, influence-based measure of each neuron's contribution, offering a rigorous alternative to correlational metrics like raw activation magnitude [Koh and Liang, 2017, Geiger et al., 2021].

Table 1: Finetuning results across base models. Δ denotes LoRA-SFT minus baseline 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%

3.4 Scaling Influence From JVPs to Constant-Time Derivative Swapping

In Definition 3.2, the contribution of neuron j relies on the Jacobian column $J_G(r_{\text{train}})\, e_j$. For LLMs, materializing the full Jacobian $J_G \in \mathbb{R}^{|\theta_2| \times d_l}$ is memory-prohibitive. A standard circumvention is to compute this column implicitly using a Jacobian-vector product (JVP) [Baydin et al., 2018]. By the definition of the directional derivative, evaluating the Jacobian along the standard basis vector e_j yields:

$$J_G(r_{\text{train}})\, e_j = \frac{d}{d\varepsilon} G(r_{\text{train}} + \varepsilon e_j)\Big|_{\varepsilon=0} = \text{JVP}(G, r_{\text{train}}, e_j). \tag{17}$$

Putting Eq.s (15)-(17) together, we obtain a computable form for the neuron-level influence:

$$\mathcal{I}_j^r(z^r_{\text{train}}, z_{\text{test}}) = -\, g_{\text{test}}^\top H_{\theta_2}^{-1}\, r^{(j)}_{\text{train}}\, \text{JVP}(G, r_{\text{train}}, e_j). \tag{18}$$

Here, the JVP term quantifies the downstream gradient sensitivity, while the activation factor ($r^{(j)}_{\text{train}}$) enforces that only active circuits contribute. This directly connects data-level influence attribution to feature-level monosemanticity [Bricken et al., 2023].

The Computational Bottleneck. While Eq. (18) avoids storing the full Jacobian, its evaluation requires computing a separate JVP for *every* active feature j, making it computationally impractical. Because G maps to the high-dimensional parameter space θ_2, each JVP requires an expensive forward-over-reverse differentiation pass ($\mathcal{O}(d_l)$ passes). For SAEs where thousands of features may fire simultaneously across a sequence, iterating over neurons individually becomes computationally intractable. We therefore need a method that computes all neuron influences in constant time, independent of the number of active features.

Constant-Time Influence via Derivative Swapping. The key insight is to exchange the order of differentiation and summation (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 $P(r_{\text{train}}) = s_{\text{test}}^\top g_{\text{train}}$. Since s_{test} is constant w.r.t. r_{train}, we have

$$\nabla_{r_{\text{train}}} P = \nabla_{r_{\text{train}}} \left(s_{\text{test}}^\top g_{\text{train}} \right), \tag{19}$$

and the vector of latent influences is

$$\vec{\mathcal{I}}^r(z^r_{\text{train}}, z_{\text{test}}) = -\left(\nabla_{r_{\text{train}}} P \right) \odot r_{\text{train}}, \tag{20}$$

reducing complexity from $\mathcal{O}(d_l)$ to $\mathcal{O}(1)$ backward passes (compute $\nabla_{\theta_2} \ell$, then backpropagate P to r_{train}).

This constant-time formulation enables simultaneous computation of influence for all latent features and batch samples, allowing the method to scale to billion-parameter models and sparse autoencoders with tens of thousands of features, which significantly improves the efficiency of the framework. The full proof is provided in Appendix E.2. For speedup and memory-saving analysis, we redirect readers to Appendix D.

4 Experiments

We evaluate on three multiple-choice QA benchmarks: MedQA Han et al. [2023], OpenBookQA Mihaylov et al. [2018], and CommonsenseQA Talmor et al. [2019]. All experiments start from task-finetuned models, and influence is measured with respect to the resulting task-adapted model at

Table 2: SAE insertion results on OpenbookQA. 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%

Table 3: Orthogonality statistics across representation spaces on OpenbookQA, Llama-3.2-1B. Lower is better for entanglement metrics, higher is better for rank and near-orthogonality.

Space	Mean Abs $\downarrow$	Mean Sq $\downarrow$	Frob. Norm $\downarrow$	Stable Rank $\uparrow$	% Pairs < 0.1 $\uparrow$
Text	0.121	0.039	0.198	5.35	64.9%
Pre-Latent	0.824	0.730	0.854	1.17	2.0%
SAE Latent	**0.074**	**0.022**	**0.149**	**7.93**	**83.5%**

inference time. We use OpenBookQA and CommonsenseQA as the primary quantitative benchmark in the main text, and defer additional dataset results and full experimental details to the appendix (Appendix C). We study small open-weight, instruction-tuned LLMs from the Llama and Qwen families ($\sim$1B parameters).

Setup and Pipeline For a given task, the pipeline proceeds as follows: (1) finetune a pretrained LLM on the task dataset (Sec 4.1); (2) insert an SAE at an intermediate layer and train it with frozen LLM weights to obtain sparse latent features (Sec 4.1); (3) pre-filter candidate training examples using gradient similarity to reduce influence computation cost and evaluate pre-filtering stability (Sec 4.3). (4) compute representation-level influence scores at the SAE layer (defined in Sec. 3.4) to rank training examples and latent features, and presents the quantitative evaluation (Sec. 4.4).

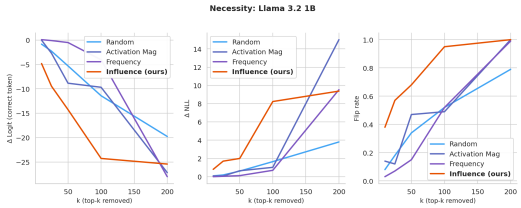
4.1 Task Performance After Finetuning and SAE Insertion

A basic requirement for practical auditing is that 1) finetuning yields meaningful task-specific updates and 2) inserting the SAE layer does not significantly degrade the task performance. We therefore report (i) pretrained and LoRA-SFT accuracy across base models in Table 1 and (ii) the accuracy change from inserting an SAE at the chosen layer in Table 2. We additionally report full sweeps over SAE insertion layers across datasets and models in Appendix B, showing that performance is sensitive to the placement of the representation bottleneck.

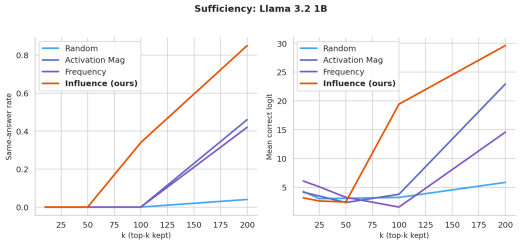
LoRA Supervised Finetuning. Tables 1 show that LoRA-SFT yields substantial gains across all base models, with the largest improvements for Llama-3.2-1B (+44.2%/40.22%) and Llama-3.2-1B-IT (+30.8%/21.79%). Qwen2.5-1.5B starts from a stronger baseline and thus shows smaller but still meaningful gains. These results confirm that finetuning produces large, measurable task specific updates, which is a prerequisite for downstream influence analysis.

Layer selection logic. Rather than selecting the globally best post insertion layer, we choose the best layer within the middle half of the network (late layers can be less informative): layers 4–12 for Llama-3.2-1B-Instruct and 7–22 for Qwen2.5-1.5B-Instruct. We report both the *selected* and *best* layers, and provide the full sweep in the Appendix B.

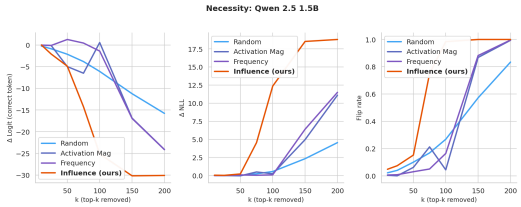
SAE insertion. Table 2 shows that inserting an SAE at an intermediate layer largely preserves finetuned performance: Qwen2.5-1.5B-IT drops by only 0.8 points at its selected (and best) layer, whereas Llama-3.2-1B-IT drops by 2.4 points at the selected layer. However, the "best layer" results indicate that most of Llama's degradation can be recovered by a nearby layer choice (down to a 0.2 point drop), suggesting that the main cost comes from bottleneck placement rather than the SAE mechanism itself; layer selection is therefore a key hyperparameter for balancing interpretability and accuracy.



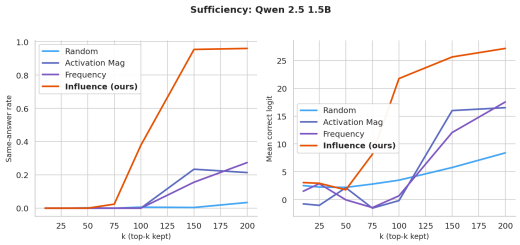
(a) Llama-3.2-1B-Instruct (necessity).



(b) Llama-3.2-1B-Instruct (sufficiency).



(c) Qwen2.5-1.5B-Instruct (necessity).



(d) Qwen2.5-1.5B-Instruct (sufficiency).

Figure 2: Necessity and sufficiency tests on OpenbookQA. For necessity, we rank and remove top-k SAE features using RepInf and three baselines, then report Δlogit, ΔNLL, and flip rate after masking. For sufficiency, we keep only top-k features and report the retained correct logit and correct rate.

4.2 Latent Space Orthogonality Analysis

We evaluate representation disentanglement by comparing feature orthogonality across three spaces: input text embeddings, dense pre-latent activations (immediately before SAE insertion), and the k-sparse SAE latent space. Following prior work, we summarize orthogonality using the off-diagonal Gram statistics (mean absolute value, mean squared value, and Frobenius norm), the fraction of near-orthogonal feature pairs ($|\rho| < 0.1$), and stable rank $||A||_F^2/||A||_2^2$ Fel et al. [2025]. Table 3 shows a clear separation: pre-SAE latents are highly entangled (stable rank 1.17; 2.0% near-orthogonal pairs), whereas SAE latents are substantially more disentangled (stable rank 7.93; 83.5% near-orthogonal pairs). Compared with text embeddings (stable rank 5.35; 64.9% near-orthogonal pairs), the SAE recovers more orthogonal directions while reducing pairwise correlation magnitude, supporting the use of sparse latents for influence estimation.

4.3 Gradient Pre-Filtering for Scalable Influence Computation

Exact influence computation over the full training set is expensive. We therefore pre-filter training examples using gradient similarity and retain only the top 1%–10% candidates per test example. Concretely, for a test example z_{test} and each training example z_i, we score

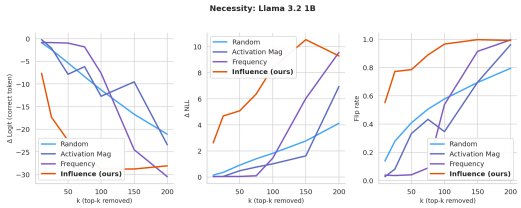
$$s_i \;=\; \langle \nabla_\theta \mathcal{L}(z_i;\theta)\,,\, \nabla_\theta \mathcal{L}(z_{\text{test}};\theta) \rangle\,, \tag{21}$$

and keep $\mathcal{C}_K(z_{\text{test}}) = \text{TopK}_i(s_i)$. We defer additional experiments on filtering thresholds and stability to Appendix C.3.

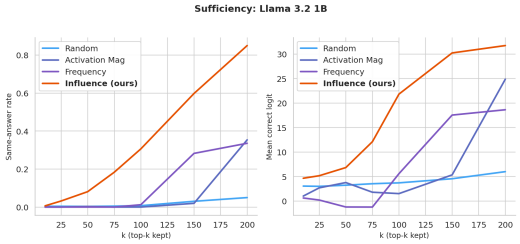
4.4 Quantitative Evaluation: Necessity and Sufficiency Tests

To evaluate the faithfulness of sparse feature influence, we run two deletion/insertion style tests. For each evaluation sequence, we record the baseline prediction and correct-token statistics, then apply a binary mask at the SAE layer using a chosen ranking over latent features.

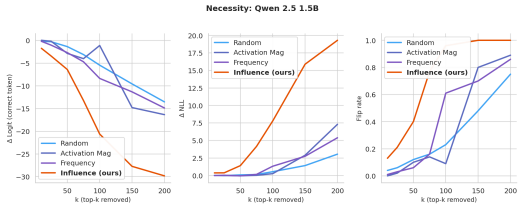
Our core signal is the representation level influence matrix $\text{IFR} \in \mathbb{R}^{N\times H}$ for a test sample, where N is the number of retained training examples after gradient pre filtering and H is the SAE latent dimension; $\text{IFR}[i,k]$ measures the influence of training example i on latent feature k. We aggregate over training examples (mean over i) to obtain a test conditioned importance vector $s \in \mathbb{R}^H$ with $s_k = \frac{1}{N}\sum_{i=1}^{N} \text{IFR}[i,k]$, and take the top-$k$ features under s as the *most influential* for that test sample.



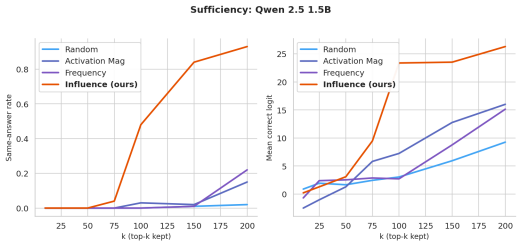
(a) Llama-3.2-1B-Instruct (necessity).



(b) OpenbookQA Llama-3.2-1B-Instruct (sufficiency).



(c) Qwen2.5-1.5B-Instruct (necessity).



(d) Qwen2.5-1.5B-Instruct (sufficiency).

Figure 3: CommonsenseQA necessity (remove top-k features) and sufficiency (keep top-k features) tests comparing influence-selected features to baselines.

We compare our influence score against three simple baselines: (i) ranking by activation magnitude, (ii) ranking by feature frequency, and (iii) a random set. Concretely, for each method we induce an ordering over latent dimensions and construct:

- **Necessity (Remove Top-k):** A mask that zeroes out the top-k features, leaving other features intact.
- **Sufficiency (Keep Top-k):** A mask that preserves only the top-k ranked features, zeroing all others.

We evaluate both settings across multiple values of $k = \{25,50,100,125,150,175,200\}$. For necessity, we report the average change in the correct token's logit (Δlogit), the average change in negative log likelihood (ΔNLL), and prediction flip rate relative to the baseline. For sufficiency, we report the same answer rate and the retained mean correct logit under masking.

Figures 2 and 3 demonstrate a consistent and quantitative separation between influence-based ranking and all baselines across both Llama-3.2-1B and Qwen2.5-1.5B.

Necessity. Under the remove top-k intervention, influence produces the steepest and most systematic degradation. On both models and both datasets, removing as few as 50 - 100 influence-ranked features induces a large negative shift in the correct token logit, a substantial increase in NLL, and a rapid rise in flip rate, often approaching saturation at moderate k. In contrast, random masking leads to gradual degradation, while activation magnitude and frequency exhibit intermediate behavior but consistently weaker impact. The approximately monotonic dependence of Δlogit, ΔNLL, and flip rate on k suggests that influence induces a meaningful global ordering over latent features in terms of their causal contribution.

Sufficiency. The keep top-k experiment exhibits the complementary pattern. Influence retains substantially higher same answer rates and larger correct logits at every k. Notably, a relatively small subset of influence-ranked features suffices to recover a large fraction of the original predictive confidence, whereas random and heuristic rankings require many more features and still fail to match the retained performance. This indicates that predictive information is concentrated in a compact set of influence identified latent directions.

Interpretation. Taken together, the dual behavior: sharp degradation under removal and strong recovery under retention, provides evidence that influence ranking captures features that are not merely highly active or frequent, but structurally implicated in the model's decision. The consistency

of this pattern across two distinct architectures further suggests that the effect reflects a representation level property rather than model specific artifacts.

5 Discussion

We do not report direct comparisons with standard data attribution methods because most prior approaches operate in input space and evaluate instance-level rankings. In contrast, our framework estimates representation-level influence in a sparse latent space and only subsequently projects attributions back to text; conventional ground-truth protocols therefore do not transfer directly.

A more appropriate validation would involve causal interventions: deactivating influential latents and measuring accuracy drop, or removing the corresponding training signal during finetuning and evaluating downstream shifts. We leave such intervention-based validation to future work.

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.

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.

Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, Adel Bibi, Robert Trager, Damiano Fornasiere, John Yan, Yanai Elazar, and Yoshua Bengio. Chain-of-thought is not explainability, 2025.

Pathikrit Basu and Federico Echenique. On the falsifiability and learnability of decision theories. *Theoretical Economics*, 15(4):1279–1305, 2020.

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.

Lin William Cong, Guanhao Feng, Jingyu He, and Junye Li. Sparse modeling under grouped heterogeneity with an application to asset pricing. Technical report, National Bureau of Economic Research, 2023.

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.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Joseph Futoma, Morgan Simons, Trishan Panch, Finale Doshi-Velez, and Leo Anthony Celi. The myth of generalisability in clinical research and machine learning in health care. *The Lancet Digital Health*, 2(9):e489–e492, 2020.

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.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *NeurIPS*, 34:9574–9586, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html.

Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The lancet digital health*, 3(11):e745–e750, 2021.

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.

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.

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

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.

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*, pages 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.

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*, pages 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.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 2022. doi: 10.48550/arXiv.2202.05262. URL http://arxiv.org/abs/2202.05262.

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)*, pages 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.

Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890. PMLR, 2019.

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.

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.

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*, pages 3319–3328. PMLR, 2017a.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *ICML*, June 2017b. doi: 10.48550/arXiv.1703.01365. URL http://arxiv.org/abs/1703.01365.

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)*, pages 4149–4158, 2019.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, and Brian Chen. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024a. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024b. 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.

Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44–56, 2019.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

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.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *ICLR*, 2023. doi: 10.48550/arXiv.2211.00593. URL http://arxiv.org/abs/2211.00593.

Xin Wang, Hong Guo, Sumit Jha, and Ruishan Gao. Disentangled representation learning. *arXiv preprint arXiv:2211.11695*, 2024.

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.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

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.

Appendix

A Related Works

Interpretability in LLM 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., 2017a, 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 often prioritize global model understanding, our method emphasizes actionable, testable attributions tailored for high-stakes domains like healthcare, where rapid fact-checking and validation of model decisions are critical for reliability and trust.

Sparse Autoencoders and Independent Features SAEs learn disentangled, interpretable features via sparsity constraints (e.g., L1 penalty), promoting statistical independence in latent representations. This approach builds upon a long history of seeking independent data components, including classical linear methods like Principal Component Analysis (PCA) [Jolliffe and Cadima, 2016] and Independent Component Analysis (ICA) [Hyvärinen, 2013], as well as nonlinear probabilistic frameworks like Variational Autoencoders (VAEs) [Kingma and Welling, 2014]. However, SAEs offer a uniquely transparent and deterministic pathway to feature learning that balances sparsity and reconstruction fidelity. They are widely used for mechanistic interpretability in LLMs [Cunningham et al., 2023, Bricken et al., 2023, Templeton et al., 2024b, Marks et al., 2024], with variants including k-sparse SAEs [Makhzani and 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 our framework to handle independent logits from a transcoder is promising but beyond the scope of this 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 and 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., 2024b] suggest disentanglement improves with model size. While these works aim for complete monosemanticity, our approach uses SAEs to obtain approximately independent features specifically to enable more reliable influence estimation, prioritizing practical interpretability over full disentanglement.

B 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

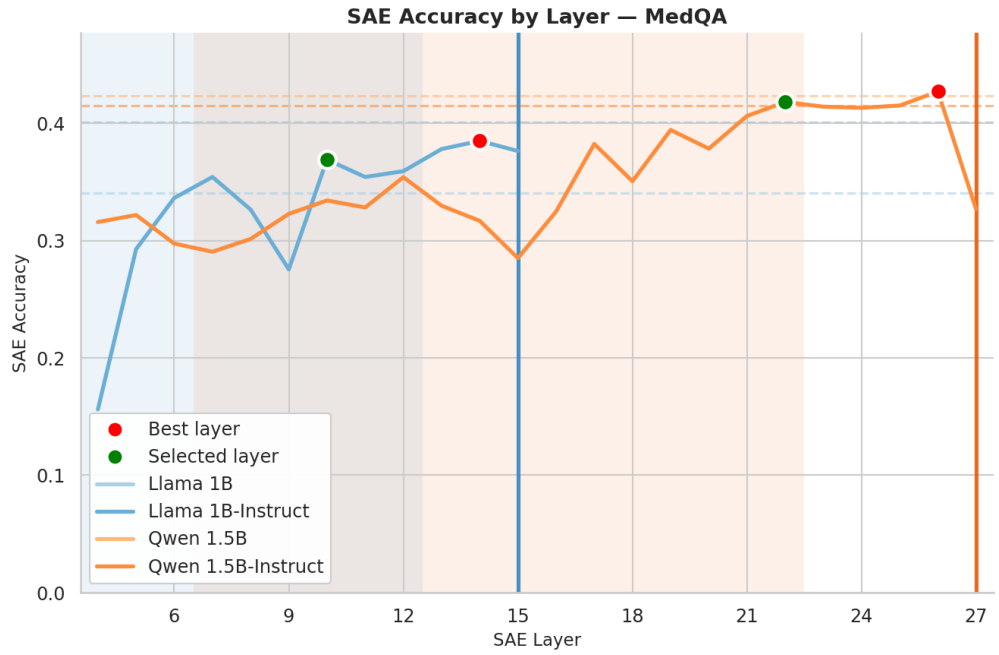


Figure 4: Accuracy change Δ versus SAE insertion layer on MedQA across SAE configurations.

the network layers considered in the main experiments (Section 4.1). Concretely, for a model with L transformer blocks, the middle-half range is

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

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 4.1: we search within $\mathcal{L}_{\text{mid}}$ 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. To quantify the trade-off between preserving downstream behavior and obtaining useful/interpretable features, we compare necessity test scores at two insertion depths (L10 and L13). For Llama-3.2-1B-Instruct on OpenBookQA, Layer 10 is the selected layer under our middle-half selection policy, while Layer 13 is the best performance layer; both points are marked in Figure 5. Table 4 shows that the deeper layer (L13) consistently yields smaller NLL increase for most k, indicating worse task behavior preservation, while also exhibiting substantially lower flip rates, indicating weaker controllable feature effects. This supports the practical choice of intermediate layers for auditing: deeper insertion can preserve accuracy better, but the resulting features tend to be less interpretable/usable for intervention-based analysis.

C Experimental Details

This appendix records the settings needed to reproduce the experiments in Section 4.

C.1 Models and fine-tuning

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

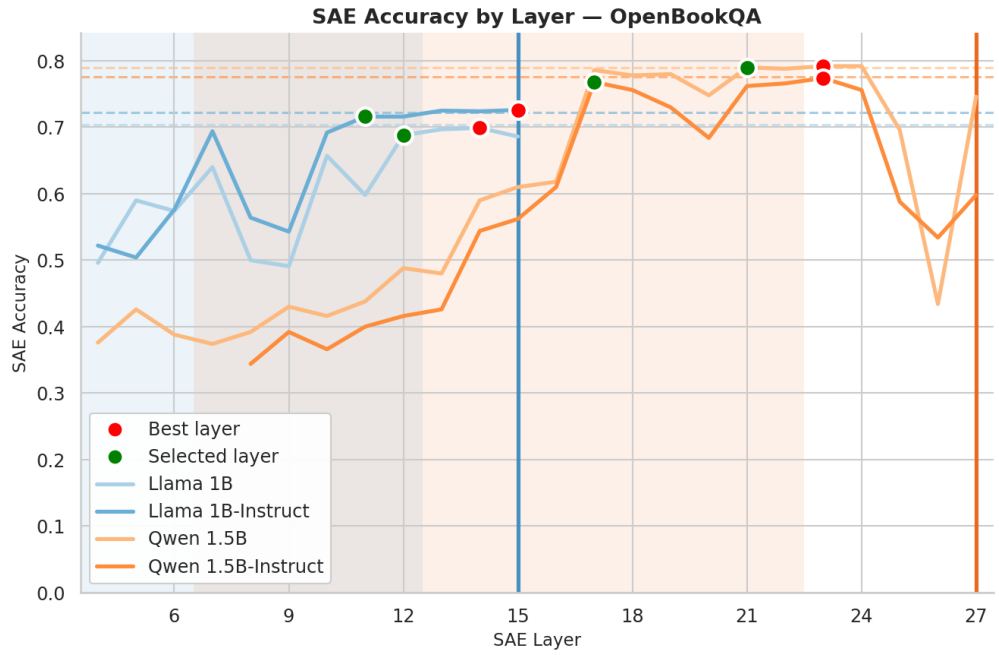


Figure 5: Accuracy change Δ versus SAE insertion layer on OpenBookQA across SAE configurations.

Table 4: Trade-off across insertion depth: intermediate insertion layer better preserves downstream behavior and yields stronger intervention effects with lower ΔNLL and higher flip rate(FR), suggesting higher feature usefulness for interpretability.

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

- **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.

C.2 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 5).
- **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`.

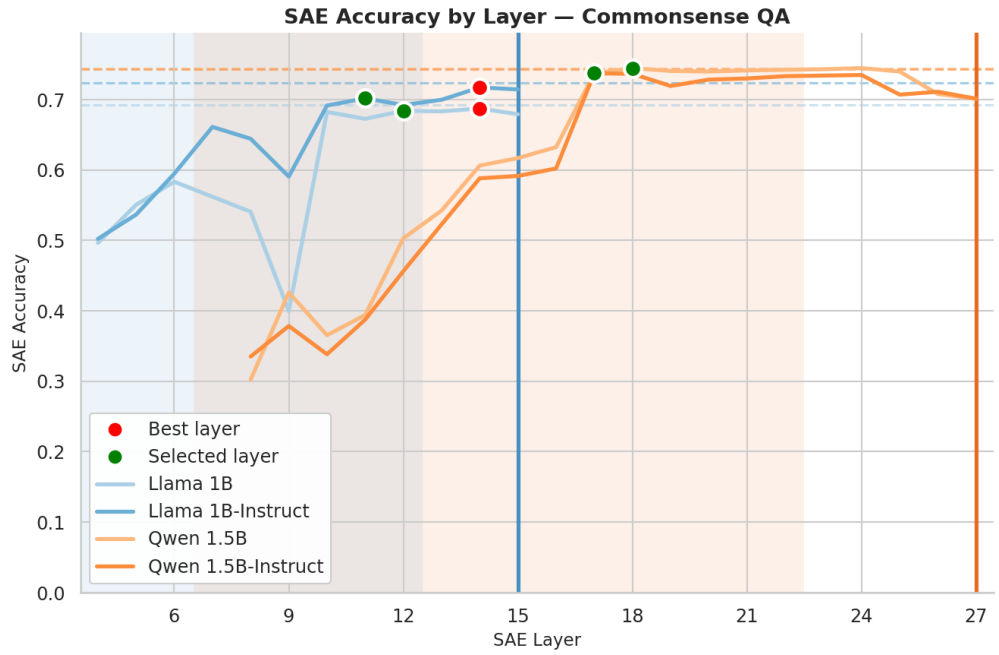


Figure 6: Accuracy change Δ versus SAE insertion layer on CommonsenseQA across SAE configurations.

Table 5: 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	Gradient similarity pre-filtering; retained fraction 1%–10%.
Influence	iHVP via CG using batched empirical loss (batch size 8); damping 10^{-3}; 20 iterations.

C.3 Influence computation and filtering

Gradient pre-filtering. We rank training examples by cosine similarity between their gradients and the test gradient:

$$\begin{aligned} \mathrm{sim}(z_i, z_{\text{test}}) &= \frac{\langle g_i, g_{\text{test}} \rangle}{\|g_i\| \, \|g_{\text{test}}\|}, \\ g_i &= \nabla_\theta \ell(h_\theta(x_i), y_i), \\ g_{\text{test}} &= \nabla_\theta \ell(h_\theta(x_{\text{test}}), y_{\text{test}}). \end{aligned} \tag{22}$$

We retain the top {1% / 5% / 10%} candidates for exact influence computation.

iHVP approximation. We compute $H_{\theta_2}^{-1} g_{\text{test}}$ using conjugate gradient (CG), with damping `1e-3` and iteration budget 20. Unless otherwise stated, the empirical Hessian is formed using a batched loss with batch size 8 to improve curvature stability.

Influence target. Unless otherwise stated, influence is computed on the test loss.

C.4 Qualitative Attribution: Linking Predictions to Training Evidence

We now present case studies illustrating how our method links a test prediction to specific training evidence. For each case, we report the model prediction, the most influential training examples, and token-level heatmaps obtained by projecting representation-level influence back to the input space.

Case study protocol. We select test instances that are predicted incorrectly by the pretrained model but become correct after finetuning, so that the attributions reflect task-specific learning rather than generic priors. We run this procedure for both Llama-3.2-1B-Instruct and Qwen2.5-1.5B-Instruct; for readability, the main text shows one representative example and additional cases (including Qwen-based results) are provided in Appendix C.4.

For each selected test example, we compute representation-level influence IFR and use it to rank influential latent features and influential training examples. We then visualize token-level influence by projecting the influential latent signal onto SAE activations and color tokens by their dominant influential latent, with intensity given by activation $\times$ influence (normalized per sentence).

Case study A: Evidence retrieval highlights clinically salient cues. Figure 7 presents a test question involving low urine output following major surgery together with its most influential training example. Tokens are colored according to the dominant latent feature they project onto, while color intensity reflects the product of token-level activation and latent influence, normalized within each sentence. In both the test and training samples, the same latent features concentrate on tokens describing reduced urine output, low blood pressure, and abnormal kidney-related measurements. Notably, these shared latent activations span different surface descriptions, including postoperative fluid loss in the test case and cardiac-related stress in the training example. This suggests that the model internally represents a common physiological condition: reduced blood flow to the kidneys leading to compensatory salt and water retention, and reuses this representation to support its prediction. The observed alignment indicates that the model's reasoning is mediated by consistent latent features rather than direct lexical overlap between the two samples.

We run the full influence-and-retrieval pipeline on all correctly predicted test samples, using gradient-filtered training candidates, for both Qwen2.5-1.5B-Instruct and Llama-3.2-1B-Instruct. In Section 4, we report a single representative case study for readability; here we provide additional examples (including Qwen-based results) to support the claim that the qualitative patterns hold across model families.

Each case includes a test question and its most influential retrieved training example, visualized with the same token-level influence heatmap scheme as in the main paper.

C.5 Case Study B: Seizure, temporal lobe hemorrhage, and vasogenic edema

The test question in Figure 8 describes fever, headache, behavioral changes, and a seizure with MRI evidence of temporal lobe edema and hemorrhage. The correct mechanism is endothelial tight-junction breakdown, i.e., blood–brain barrier disruption causing vasogenic edema.

The retrieved training example also involves seizure and marked cerebral edema. The heatmaps highlight influential latent features concentrating on shared mechanistic cues (edema on imaging and tight-junction loss) rather than incidental details, yielding an auditable link from the test-time prediction to training evidence through a coherent latent motif (BBB disruption $\rightarrow$ vasogenic edema).

C.6 Case Study C: Post-operative oliguria and prerenal physiology

The test question in Figure 9 concerns post-operative oliguria with dry mucous membranes and rising BUN/creatinine, consistent with prerenal physiology; the correct choice is *low urine sodium*.

The retrieved training example also discusses low urine output and renal injury markers (e.g., BUN:Cr ratio, muddy brown casts). The heatmaps emphasize urine output and prerenal diagnostic features, while down-weighting tokens tied to intrinsic injury, clarifying why the model favors the prerenal "low urine sodium" conclusion.

C.7 Case Study D: Asthma exacerbation and small-airway obstruction

The test example in Figure 10 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 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., wheeze, reduced airflow, difficulty breathing). This illustrates that representation-level influence can connect test predictions to training evidence through a shared obstruction motif, without requiring exact surface-form matching.

D Scaling to 1B-Parameter LLMs

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

D.1 Memory scaling via contraction order

We compute the representation influence for latent dimension j as

$$\mathcal{I}_j^r(z_{\text{train}}^r, z_{\text{test}}) = -\underbrace{g_{\text{test}}^\top H_{\theta_2}^{-1}}_{\text{iHVP term}} \underbrace{\text{JVP}(G, r_{\text{train}}, e_j)}_{\text{per-latent JVP}}, \tag{23}$$

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.

D.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 $p^\top Hp < 0$ 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.

D.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:

Table 6: 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 7: Runtime breakdown (seconds) per $(z_{\text{train}}, z_{\text{test}})$ pair using the gradient-derivative 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**

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.

D.4 Further acceleration via the gradient-derivative formulation (Sec. 3.4)

Beyond streamed JVP contraction, we exploit the gradient-derivative reformulation from Sec. 3.4, which eliminates the explicit per-feature JVP sweep. Recall that representation influence can be written as

$$\mathcal{I}^r(z^r_{\text{train}}, z_{\text{test}}) = -s_{\text{test}}^\top G(r_{\text{train}}), \tag{24}$$

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}:

$$\nabla_{r_{\text{train}}} \big(s_{\text{test}}^\top G(r_{\text{train}})\big). \tag{25}$$

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

Overall compute comparison. Table 9 compares the streamed-JVP pipeline with the gradient-derivative formulation. The latter removes the $\sim$30–60 s JVP sweep (Table 6) and reduces total per-pair runtime to $\approx$ 2–4 seconds, corresponding to a $\sim 10\times$–$20\times$ 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.

E Proofs

E.1 Exact Path-Integral Decomposition

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

$$r(\alpha) = r_0 + \alpha(r_{\text{train}} - r_0), \qquad \alpha \in [0, 1]. \tag{26}$$

Table 8: Runtime breakdown (seconds) per $(z_{\text{train}}, z_{\text{test}})$ pair using the gradient-derivative 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 9: End-to-end per-$(z_{\text{train}}, z_{\text{test}})$ runtime comparison (seconds).

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

Path integral identity. By the chain rule,

$$\frac{d}{d\alpha} G(r(\alpha)) = J_G(r(\alpha)) \frac{dr(\alpha)}{d\alpha} = J_G(r(\alpha))(r_{\text{train}} - r_0), \tag{27}$$

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

$$\begin{aligned} G(r_{\text{train}}) - G(r_0) &= \int_0^1 \frac{d}{d\alpha} G(r(\alpha)) \\ &= \int_0^1 J_G(r(\alpha))(r_{\text{train}} - r_0)\, d\alpha. \end{aligned} \tag{28}$$

Coordinate decomposition. Using the canonical basis expansion

$$r_{\text{train}} - r_0 = \sum_{j=1}^{d_l} (r_{\text{train}}^{(j)} - r_0^{(j)}) e_j, \tag{29}$$

and linearity of integration,

$$G(r_{\text{train}}) = G(r_0) + \sum_{j=1}^{d_l} (r_{\text{train}}^{(j)} - r_0^{(j)}) \left(\int_0^1 J_G(r(\alpha)) e_j \, d\alpha \right). \tag{30}$$

Equations Eq. (28)–Eq. (30) hold exactly under the sole assumption that G is continuously differentiable.

E.2 Derivative Swapping with activation

While the Jacobian-vector product (JVP) formulation in Eq. (18) provides a rigorous attribution mechanism, computing it naively is computationally prohibitive. Evaluating the JVP for each latent dimension $j \in \{1, \ldots, d_l\}$ requires d_l separate forward-mode passes. For a sparse autoencoder with thousands of features, this $\mathcal{O}(d_l)$ complexity leads to severe computational bottlenecks and memory allocation limits.

We can achieve a massive optimization by exploiting the structure of the computation graph and the symmetry of mixed partial derivatives (Clairaut's Theorem). We transition from evaluating d_l 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:

$$\frac{\partial r_{\text{train}}^{(j)}}{\partial \theta_2} = \mathbf{0}. \tag{31}$$

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 operationalized code, 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:

$$\frac{\partial s^{(j)}}{\partial \theta_2} \;=\; \frac{\partial}{\partial \theta_2}\left(r_{\text{train}}^{(j)} \Delta^{(j)}\right) \;=\; \underbrace{\frac{\partial r_{\text{train}}^{(j)}}{\partial \theta_2}}_{=\mathbf{0}} \Delta^{(j)} + r_{\text{train}}^{(j)} \frac{\partial \Delta^{(j)}}{\partial \theta_2}. \tag{32}$$

Because the first term vanishes due to the computation graph cut Eq. (31), we are left with:

$$\frac{\partial s^{(j)}}{\partial \theta_2} \;=\; r_{\text{train}}^{(j)} \frac{\partial}{\partial \theta_2}\left(\frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right) \;=\; r_{\text{train}}^{(j)} \frac{\partial G(r_{\text{train}})}{\partial r_{\text{train}}^{(j)}}, \tag{33}$$

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, perfectly aligning the empirical implementation with the theoretical Taylor decomposition from Definition 3.2.

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:

$$\mathcal{I}_j^r(z_{\text{train}}^r, z_{\text{test}}) = -\left(\frac{\partial s^{(j)}}{\partial \theta_2}\right)^\top v = -\, r_{\text{train}}^{(j)} \left(\frac{\partial}{\partial \theta_2} \frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right)^\top v. \tag{34}$$

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

$$\left(\frac{\partial}{\partial \theta_2} \frac{\partial \ell}{\partial r_{\text{train}}^{(j)}}\right)^\top v \;=\; \frac{\partial}{\partial r_{\text{train}}^{(j)}}\left(\left(\frac{\partial \ell}{\partial \theta_2}\right)^\top v\right) \;=\; \frac{\partial}{\partial r_{\text{train}}^{(j)}}\Big(G(r_{\text{train}})^\top v\Big). \tag{35}$$

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. (35) reveals that the sensitivity for neuron j is exactly the partial derivative of P with respect to $r_{\text{train}}^{(j)}$.

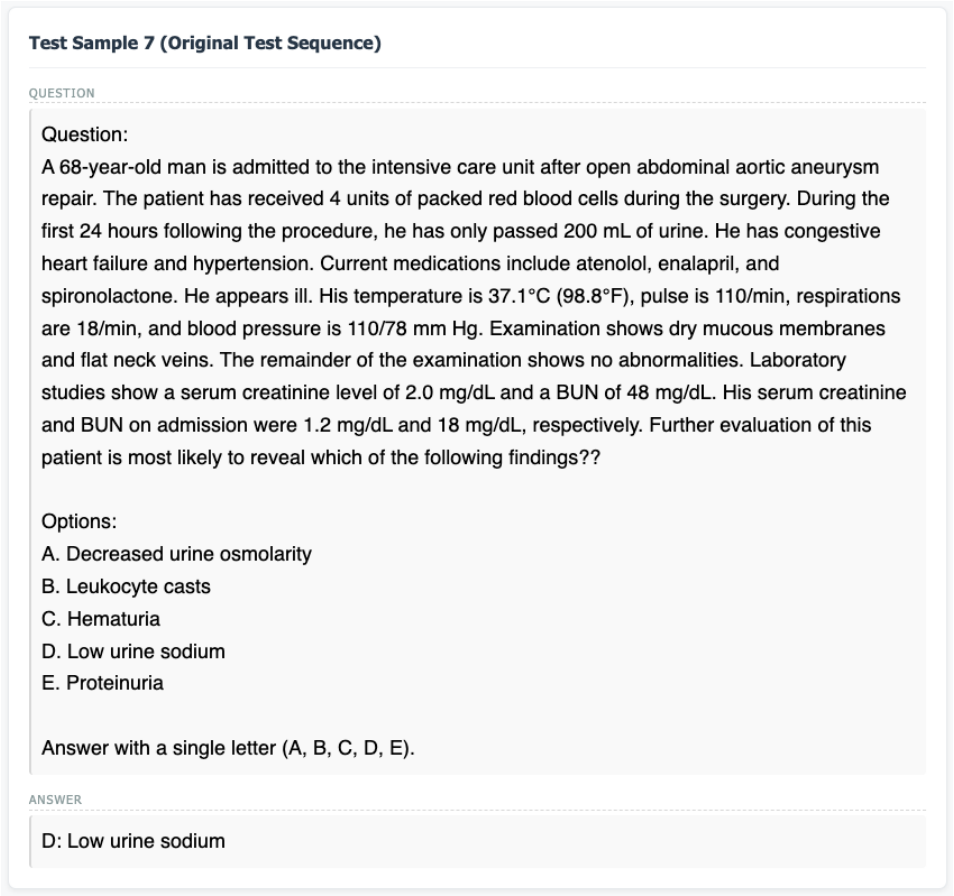
Consequently, the entire vector of sensitivities for all d_l neurons can be computed simultaneously by taking a single gradient of the scalar P with respect to the latent representation r_{train}:

$$\text{Sensitivity Vector} = \nabla_{r_{\text{train}}} P = \nabla_{r_{\text{train}}}\Big(G(r_{\text{train}})^\top v\Big). \tag{36}$$

Combining Eq. (34) and Eq. (36), the final influence vector for all latent features is obtained via an element-wise product (Hadamard product, denoted by $\odot$) with the realized activations:

$$\vec{\mathcal{I}}^r(z_{\text{train}}^r, z_{\text{test}}) = -\Big(\nabla_{r_{\text{train}}} P\Big) \odot r_{\text{train}}. \tag{37}$$

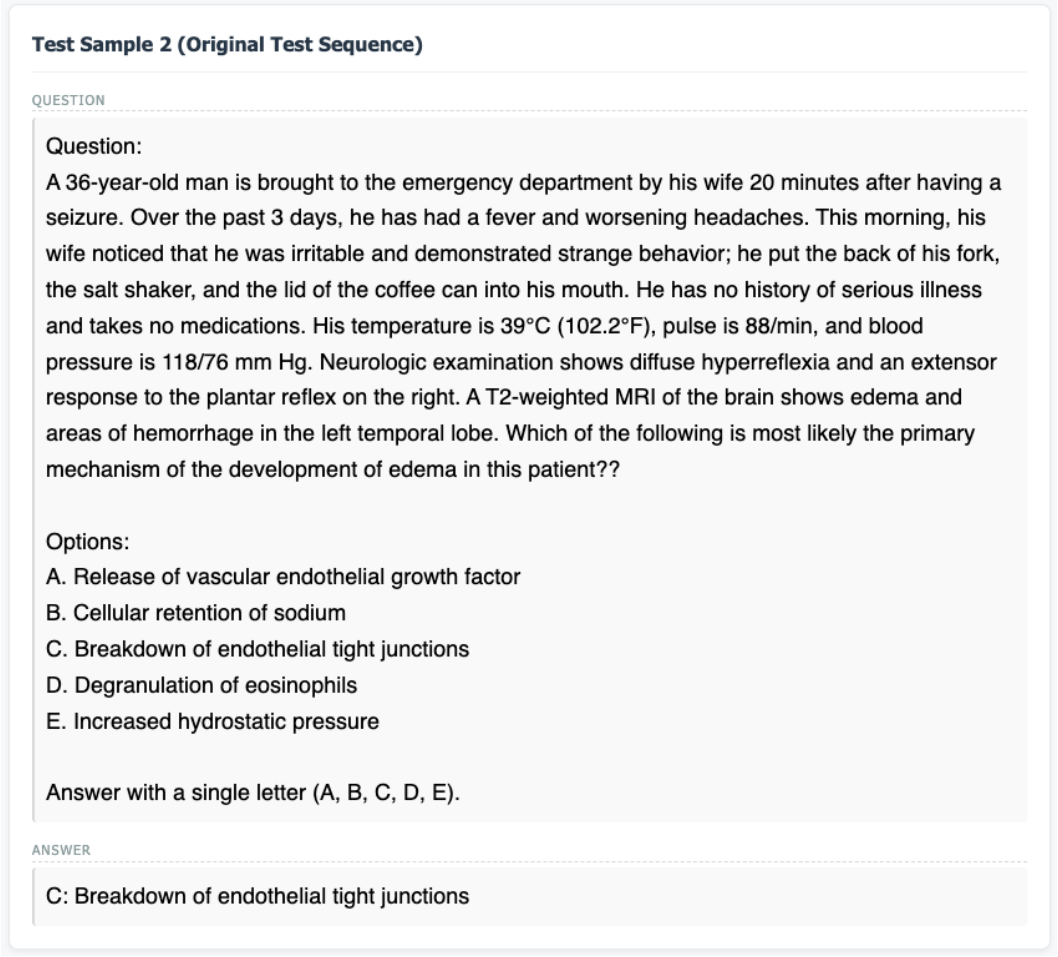
This formulation requires only *two* backward passes total: one to construct the computation graph for $G(r_{\text{train}})$, and a second to compute the gradient of the scalar projection P with respect to r_{train}. The time complexity with respect to the feature dimension drops from $\mathcal{O}(d_l)$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



(a) Original test input (token-level influence).

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

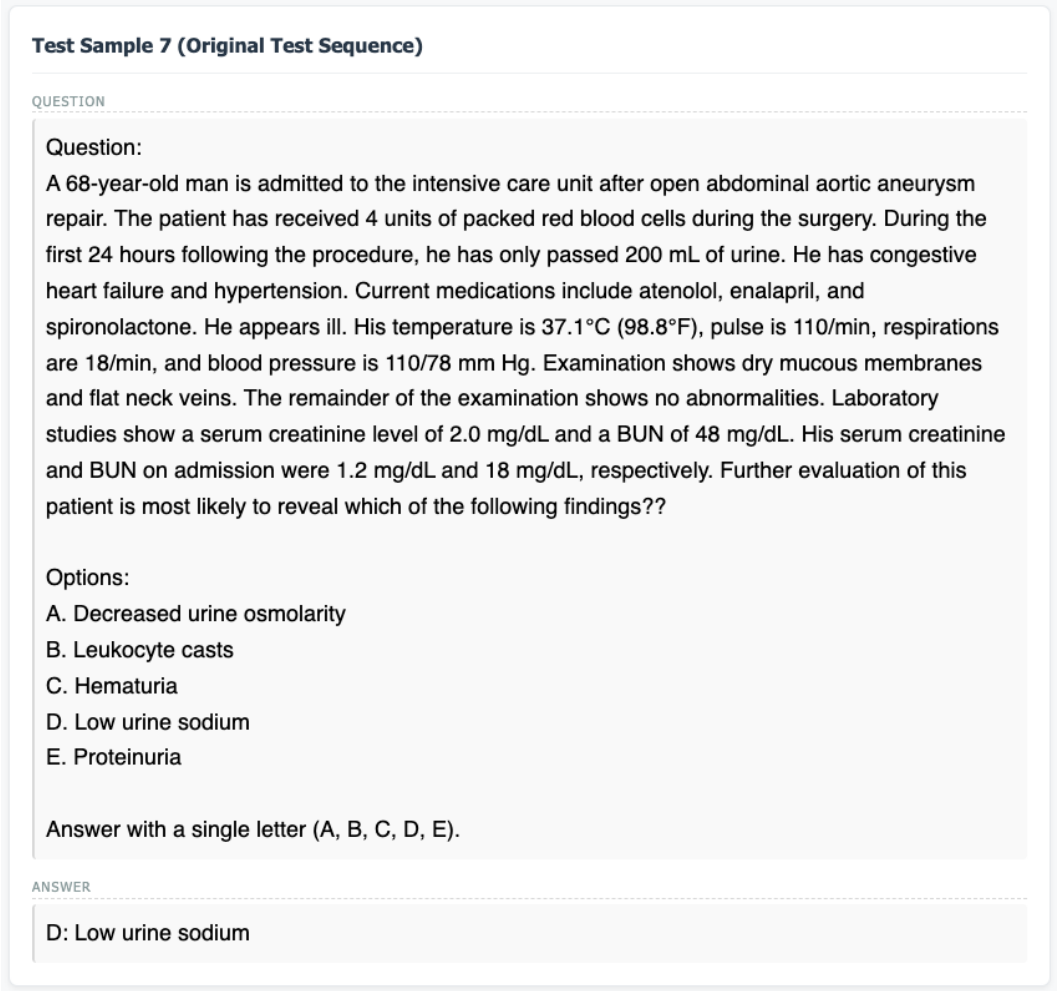
Figure 7: Token-level influence visualizations for a representative MedQA test question (top) and its most influential training example (bottom). Colors denote the dominant influential latent feature and intensities denote per-token influence.



(a) Test input (token-level influence).

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

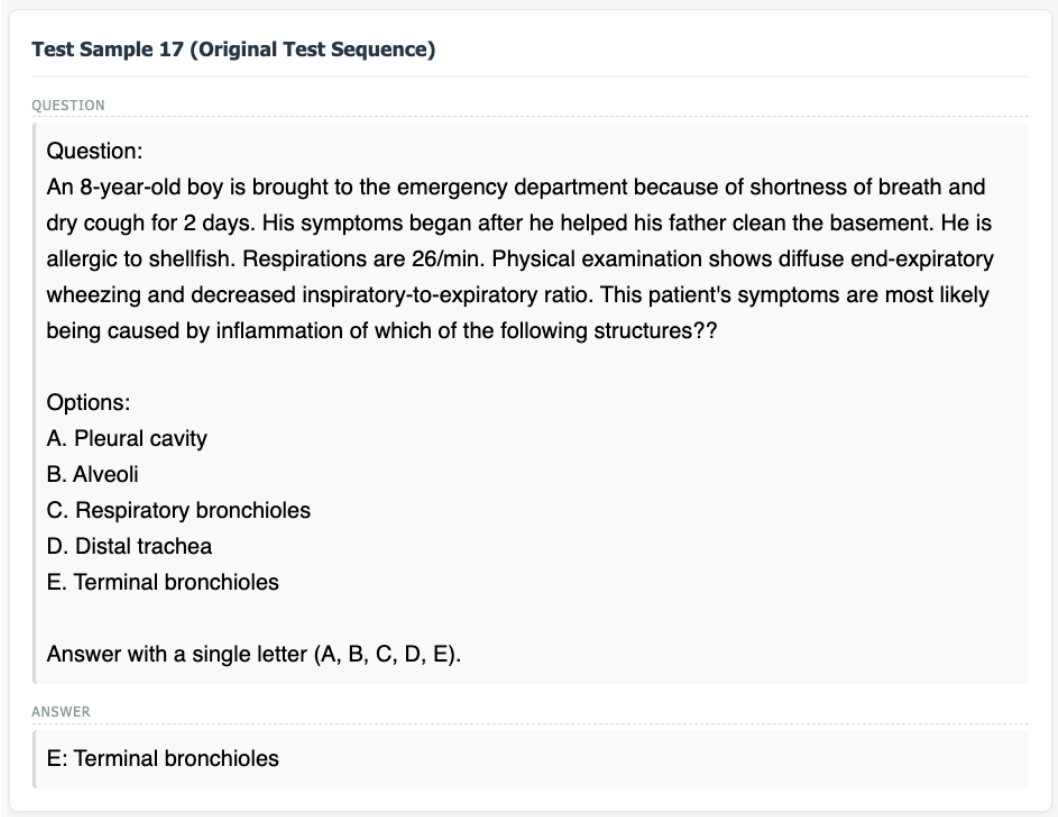
Figure 8: Case Study A (Llama-3.2-1B-Instruct): seizure, temporal lobe hemorrhage, and vasogenic edema.



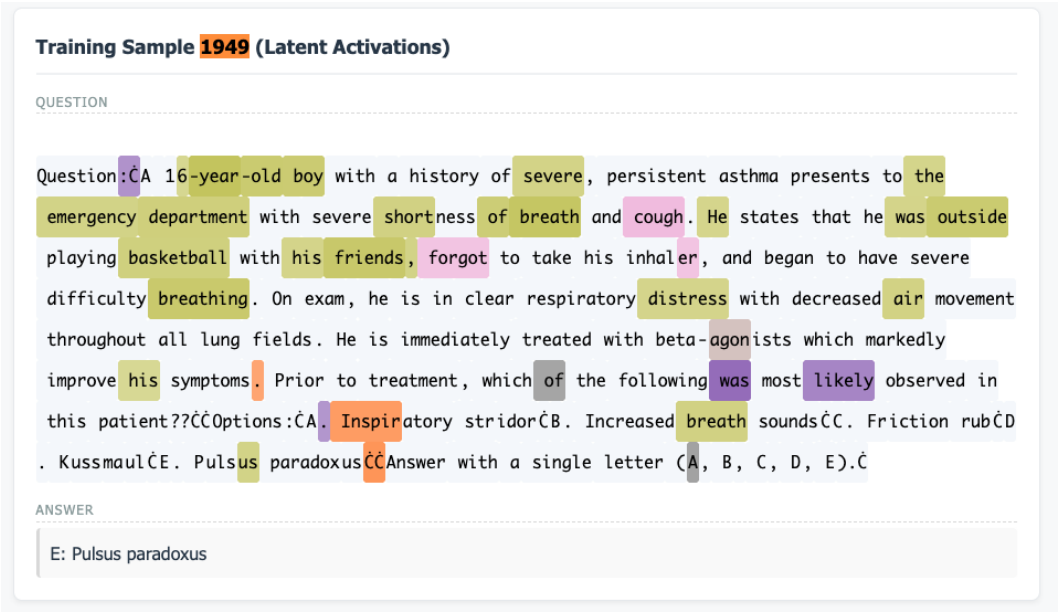
(a) Test input (token-level influence).

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

Figure 9: Case Study B (Llama-3.2-1B-Instruct): post-operative oliguria and prerenal physiology.



(a) Test input (token-level influence).



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

Figure 10: Case Study C (Qwen2.5-1.5B-Instruct): asthma exacerbation and small-airway obstruction.