Scalable Circuit Learning for Interpreting Large Language Models

Naiyu Yin [1] **Dennis Wei** [2] **Tian Gao** [2] **Amit Dhurandhar** [2] **Karthikeyan Natesan Ramamurthy** [2] **Yue Yu** [1]

Abstract

A prominent research direction in mechanistic interpretability is learning sparse circuits over LLM components to reveal how they jointly produce model behavior. However, raw neurons are polysemantic, making learned circuits hard to interpret. Sparse autoencoder (SAE) features alleviate this, but their high dimensionality makes existing intervention-based circuit learning methods computationally prohibitive. We propose `CircuitLasso`, a scalable circuit learning approach based on sparse linear regression. `CircuitLasso` recovers circuits whose structural accuracy matches that of state-of-the-art intervention-based methods on the benchmark data, at a fraction of the computational cost. For interpretability, `CircuitLasso` efficiently uncovers relationships among SAE features, showing how human-interpretable semantic features propagate through the model and influence its predictions. Finally, we validate the utility of our learned circuits by leveraging their insights to achieve comparable performance at substantially lower cost on a domain-generalization task.

1. Introduction

The fundamental challenge of mechanistic interpretability is to understand the "why" behind the behaviors of large language models (LLMs). A key technique involves discovering circuits, which are compact subgraphs connecting key components within the model (such as attention heads and neurons) that drive a specific behavior or capability (Meng et al., 2022; Geva et al., 2023; Nanda, 2023). However, existing methods for circuit learning often face a bottleneck. The raw components of an LLM, such as individual neurons, are known to be polysemantic (Elhage et al., 2022), meaning that a single neuron can be activated by and contribute to multiple, seemingly unrelated concepts. This polysemanticity renders the learned circuits dense, noisy, and challenging for humans to interpret, undermining the very goal of mechanistic interpretability.

The limitations of using raw, polysemantic neurons have motivated a shift toward a more promising foundation for circuit analysis based on Sparse Autoencoders (SAEs) and related tools (Bricken et al., 2023; Cunningham et al., 2023). SAEs are neural networks trained to reconstruct the activations of an LLM's raw components using a high-dimensional but sparse set of "features". Remarkably, these SAE features tend to be monosemantic, i.e., each feature consistently activates for a single, human-interpretable concept, such as "related to sports," "a specific emotion," or "a particular grammatical structure." The monosemanticity of SAE features has the potential not only to enhance interpretability in itself but also to yield sparser, cleaner structural graphs, and perhaps more faithful representations of the model's internal processing.

Our work is motivated by the above potential of SAE features to achieve greater interpretability in LLM circuit analysis. However, existing circuit learning methods (Vig et al., 2020; Meng et al., 2022; Nanda, 2023; Syed et al., 2024; Kramár et al., 2024; Hanna et al., 2024), many of which are designed for the lower-dimensional space of raw neurons, struggle to scale to the high-dimensional feature space of SAEs. The computational complexity and the risk of finding spurious correlations increase dramatically. To address this, we introduce a novel approach to handle the high dimensionality. Our method, `CircuitLasso`, utilizes the Lasso (i.e., ℓ_1-penalized linear regression) to find a sparse set of connections between features that explains the model's behavior. Sparse linear regression is well-suited for high-dimensional data, as it is computationally efficient and the sparsity translates to more interpretable circuits.

An advantage of our approach is its use of observational data only. This broadens its applicability and addresses the scalability issue of the existing intervention-based approaches, whose cost scales with LLM size. We quantify the efficiency advantage of our regression-based method through a theoretical analysis of its computational cost compared to state-of-the-art intervention-based approaches, establishing conditions under which our method guarantees greater

[1]Lehigh University, Mathematics Department, Bethlehem, USA
[2]IBM Research, Yorktown Heights, USA. Correspondence to: Naiyu Yin <nay224@lehigh.edu>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

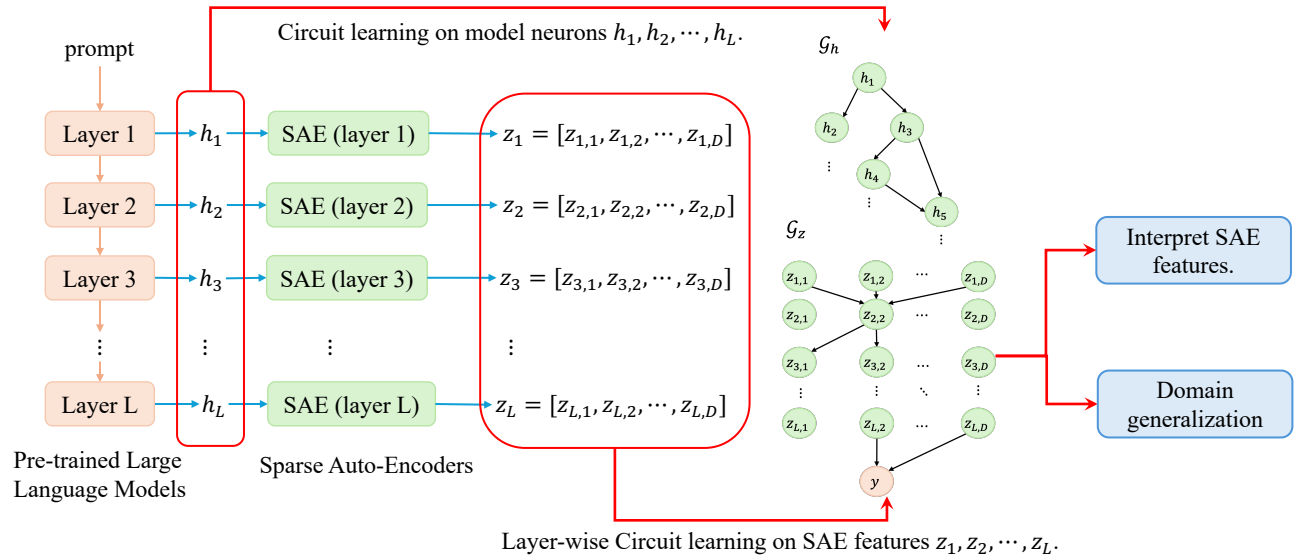


Figure 1. An illustration of our model neuron activation and SAE feature collection procedure, learned circuits, and potential downstream tasks.

efficiency.

We empirically evaluate `CircuitLasso` against state-of-the-art baselines on a circuit learning benchmark, demonstrating substantial improvements in efficiency while accurately capturing circuits involving LLM components. Leveraging its scalability, we then apply our method to SAE features and obtain human-interpretable circuits for the CoLA dataset, which has not been used before in mechanistic interpretability studies. Finally, leveraging insights from the learned circuits, we achieve comparable accuracy at substantially lower cost on a downstream domain-generalization task.

Our primary contributions are as follows: 1) Inspired by the continuous formulation of causal discovery, we propose a sparse regression surrogate for circuit discovery, `CircuitLasso`, and theoretically analyze its computational cost compared to intervention-intensive approaches. 2) `CircuitLasso` facilitates operating on monosemantic but high-dimensional SAE features, potentially offering clearer explanations of how human-interpretable concepts propagate through LLMs. 3) Extensive experiments across LLMs (up to 9B parameters), SAEs, and datasets demonstrate that `CircuitLasso` achieves efficiency at parity of accuracy on circuit discovery benchmarks and yields useful insights for a downstream generalization task.

2. Related Work

Mechanistic Interpretability. The established work in mechanistic interpretability explains behaviors in terms of raw or coarse-grained model components. Olsson et al. (2022) implicated induction heads in in-context learning, while others (Meng et al., 2022; Geva et al., 2023; Nanda et al., 2023) examined MLP modules for factual recall. However, due to the polysemantic nature of raw neurons and coarse-grained components (Elhage et al., 2022), the resulting mechanistic insights are often difficult to apply to downstream tasks. Some prior methods (Geiger et al., 2025; Zou et al., 2023) attempt to address this issue by fitting model internals to pre-defined hypotheses using curated data, but these approaches fail to generalize to scenarios where researchers lack expert knowledge or cannot anticipate how models implement specific behaviors. Recent work (Bricken et al., 2023; Cunningham et al., 2023) leverages advances in dictionary learning for interpretability and introduces sparse autoencoders (SAEs) to identify sparse, disentangled features in high-dimensional spaces that align with human-interpretable concepts. Building on this, a number of advanced strategies for learning SAE features have been proposed (Rajamanoharan et al., 2024; Gao et al., 2025; Bussmann et al., 2024; Dunefsky et al., 2024). Despite this progress, existing mechanistic interpretability methods continue to face challenges in scaling to the high-dimensional SAE feature space.

Circuit Learning. Intervention-based circuit learning approaches, including causal mediation analysis (Vig et al., 2020; Geva et al., 2023; Hanna et al., 2024), causal tracing (Meng et al., 2022), and activation patching (Nanda, 2023; Syed et al., 2024), quantify the influence between model components via counterfactual interventions. However, these methods are computationally intensive and struggle to scale to large component sets, particularly the high-dimensional space of sparse autoencoder (SAE) features.

Jafari et al. (2025) improves the *faithfulness* of attribution patching by replacing local gradients with Layer-wise Relevance Propagation coefficients, yet the efficiency bottleneck remains. Marks et al. (2025) propose efficient approximations for SAE features but rely on heuristic pre-processing such as clustering in high-dimensional settings. As an alternative direction, Laptev et al. (2025) construct circuit graphs in a data-free manner from SAE decoder weights. Although many prior works borrow causal concepts (Meng et al., 2022), the broader causal discovery literature remains largely unexplored for circuit learning; a notable exception is Conmy et al. (2023), who iteratively prune edges from the computation graph, reminiscent of constraint-based algorithms such as the PC algorithm (Pearl, 2000).

Per-prompt vs. population-level circuits. All data-driven circuit-learning approaches above, including Marks et al. (2025), Syed et al. (2024), and Hanna et al. (2024), aggregate effects across a dataset of prompts and recover a single population-level circuit per task; recent work also explores attribution graphs at per-prompt granularity. Ameisen et al. (2025) introduce a circuit-tracing methodology that constructs *per-prompt* attribution graphs over transcoder features, and Lindsey et al. (2025) apply this methodology to a series of case studies tracing how a frontier language model processes specific inputs. `CircuitLasso` is complementary to these per-prompt methods: by aggregating observational sparse-regression coefficients across many prompts, it recovers a single *population-level* dependency skeleton, which is the appropriate object when the downstream target is dataset-wide interpretation, model editing, or domain generalization (as in Section 4.2.2). The two granularities, per-prompt mechanism and population-level skeleton, answer different questions about the same model and can be used together to triangulate when, for which inputs, and through which features a given behavior arises.

3. Circuit Learning Framework and Methodology

3.1. Circuit Discovery via Sparse Linear Regression

State-of-the-art approaches quantify the importance of hidden representations or computational graph edges by estimating their causal effects, particularly indirect effects, using techniques such as causal mediation analysis (Vig et al., 2020), causal tracing (Meng et al., 2022), attribution patching (Nanda, 2023; Syed et al., 2024), and related variants (Kramár et al., 2024; Hanna et al., 2024). These approaches share some similarities with constraint-based causal discovery, which assesses potential edges among variables via independence tests and retains those with strong dependencies. However, constraint-based causal discovery methods are known to face scalability challenges, and circuit discovery methods share this limitation since they must separately quantify the importance of every hidden representation and edge in the computational graph, which can quickly become infeasible with larger models.

We adopt a sparse linear regression framework, structurally analogous to a linear SEM in the continuous causal discovery literature, as a tractable surrogate for the LLM's nonlinear computational graph. **The goal of this surrogate is not to recover fine-grained per-edge causal effects of the underlying nonlinear computation, but to identify, at scale, the dependency skeleton of the circuit, i.e., which components influence which others.** Assume we extract N components (which may be MLP neurons, attention heads, or SAE features) from all the desired locations in the LLMs and concatenate their activations to form a vector $\boldsymbol{x} = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^N$. Our goal is to learn the DAG $\mathcal{G}$ with the N components as its nodes. The structural equation models (SEMs) model the structural relations between a component x_i and its parents $Pa_{\mathcal{G}}(x_i)$: $x_i = f_i(Pa_{\mathcal{G}}(x_i)) + \varepsilon_i$, where $f_i(\cdot)$ is the structural function and ε_i is the residual. In this work, we assume the structural relations between components are linear. Given M observations of the N components, i.e., input matrix $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, we can then obtain the linear SEM in its matrix form:

$$\boldsymbol{X} = A^\top \boldsymbol{X} + \boldsymbol{\varepsilon}, \tag{1}$$

with continuous parameters $A \in \mathbb{R}^{N\times N}$, a weighted adjacency matrix; $\boldsymbol{\varepsilon} \in \mathbb{R}^{N\times M}$ is a matrix of linearization errors. $A[i,j] \neq 0$ indicates the directed dependency $x_i \to x_j$. We aim to learn A by minimizing the reconstruction error between $\boldsymbol{X}$ and $A^\top \boldsymbol{X}$ subject to sparsity and acyclicity constraints:

$$\begin{aligned} \widehat{A} = \arg\min_{A} \|\boldsymbol{X} - A^\top \boldsymbol{X}\|_F^2 + \lambda \|A\|_1, \\ \text{subject to } \mathcal{G}(A) \in \mathbb{D} \end{aligned} \tag{2}$$

where $\|\cdot\|_F$ denotes Frobenius norm; $\|A\|_1$ is the sparsity penalty with tuning coefficient λ; $\mathcal{G}(A)$ is the circuit structure inferred from A; and $\mathbb{D}$ is the space of acyclic graphs with N nodes.

Discussion of Assumptions. We use Eq. (1) as a sparse-regression *surrogate* for the LLM's nonlinear computational graph, not as a strict SEM. Accordingly, we do not appeal to identifiability theorems that would require causal sufficiency, independent noise, or a correctly specified linear functional form: all three hold only approximately in transformer LLMs given the residual stream, SAE reconstruction error, and the inherent nonlinearity of attention, LayerNorm, and MLP layers. In particular, ε should be interpreted as *linearization error* rather than exogenous noise, since LLM activations are deterministic. Our goal is to recover the dependency *skeleton* over modeled components efficiently. The dependency skeleton is a sparse summary of the linear-projection structure among the modeled components, with

unmodeled contributions absorbed into the residual term. The ℓ_1 penalty filters out weak dependencies regardless of whether they originate from omitted parents or from genuine but small effects, so the recovered skeleton remains a useful and conservative map of the strongest dependencies that exist among the components included in the model. We also provide a nonlinear extension of `CircuitLasso`, which empirically yields similar topological skeletons at higher computational cost. Please refer to Appendix B for a detailed discussion.

The main computational challenge of the optimization in Eq. (2), as in many continuous causal discovery approaches, lies in enforcing the acyclicity constraint. Such a constraint is essential to prevent self-loops and cycles, which are unsuitable for interpreting the transmission, aggregation, and evolution of model components. To address this, we make simplifying assumptions that bypass the explicit enforcement of this constraint and reduce the optimization to sparse linear regression problems (i.e., Lasso), enabling a scalable solution.

3.2. Circuit Discovery on Neurons

To better understand how models encode and process information, mechanistic interpretability research (Conmy et al., 2023; Cao et al., 2021; Syed et al., 2024) has focused on identifying graphical structures (circuits) connecting pretrained language model neurons, including outputs from attention and MLP modules. To evaluate the effectiveness of our proposed method in Section 3.1, we follow the same setting as these prior works and treat model neurons as the components of interest. We first collect neuron activations $[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L]$ from L target locations, each with dimension d, and for M LLM inputs (observations), resulting in $\boldsymbol{H} \in \mathbb{R}^{L \times d \times M}$. Existing circuit discovery methods typically assume that circuit structures respect the model locations' computation order, meaning that neurons from layer i precede those from layer j if $i < j$, and within each layer, attention activations come before MLP activations. We adopt this assumption to simplify the acyclicity constraint in Eq. (2). Accordingly, we reorder $\boldsymbol{H}$ to match the computational graph and reshape it to obtain $\tilde{\boldsymbol{H}} \in \mathbb{R}^{N \times M}$, whereby $N = Ld$. Substitute $\boldsymbol{X}$ in Eq. (2) with $\tilde{\boldsymbol{H}}$ to estimate the weighted adjacency matrix A as

$$\widehat{A} = \arg\min_{A} \|\tilde{\boldsymbol{H}} - A^\top \tilde{\boldsymbol{H}}\|_F^2 + \lambda \|A\|_1, \quad \text{subject to } A \text{ being block upper triangular.} \tag{3}$$

Specifically, each block $A[i,j]$ is now a $d \times d$ square matrix, and $A[i,j] = 0^{d \times d}$ whenever $i \geq j$. This block upper triangular structure ensures that each block of variables depends only on preceding blocks, so later layers cannot influence earlier ones, thereby preserving the computational ordering without requiring an explicit acyclicity constraint. This constraint exploits the known, human-engineered computational order of the LLM. The underlying architectural insight provides a justifiable acyclic constraint that aligns with the inherent feed-forward nature of the network (activations in later layers are computed after, and depend on, those in earlier layers). Circuit learning with such a justified acyclicity leads to more accurate identification of the underlying structural dependencies between model components, as demonstrated by empirical results in Figure 2. In practice, we enforce this constraint by initializing the lower-triangular blocks to zero matrices and keeping them fixed throughout optimization. The resulting circuit structure $\mathcal{G}$ is then inferred from $\widehat{A}$.

We now provide complexity analysis of our proposed circuit discovery approach on model neurons versus the existing intervention-based approaches. For our optimization problem in Eq. (3), we have:

Proposition 3.1. ***Computational complexity of `CircuitLasso` on model neurons.*** *Consider the block upper-triangular Lasso problem in Eq. (3), with $\tilde{\boldsymbol{H}} \in \mathbb{R}^{N \times M}$ containing M observations of $N = Ld$ neuron activations from L positions of width d. The objective is the sum of a smooth quadratic data-fit term and a non-smooth ℓ_1 regularizer; on this composite objective, FISTA (Beck & Teboulle, 2009) reaches ϵ-suboptimality in $\mathcal{O}(1/\sqrt{\epsilon})$ iterations, matching the optimal rate of Nesterov's accelerated gradient method on smooth convex problems. The total computational cost is therefore $\mathcal{O}\Big(\frac{M\, L(L-1)\, d^2}{2\sqrt{\epsilon}}\Big)$.*

Please refer to the detailed proof in Appendix A.1. We compare the cost of `CircuitLasso` against EAP-ig[1] (Hanna et al., 2024), the state-of-the-art intervention-based circuit discovery method. EAP-ig estimates indirect effects (IEs) over L component locations of dimension d via *attribution patching*, which linearizes and parallelizes the computation so its additional bookkeeping scales only as $\mathcal{O}(MLd)$. The dominant cost is therefore the two forward and one backward passes through the LLM that EAP-ig requires per observation. `CircuitLasso`, by contrast, runs no backward passes and shares only the initial forward pass (used for activation collection) with EAP-ig. We next quantify this gap on the same problem (Eq. 3) and identify regimes in which `CircuitLasso` is provably faster.

Proposition 3.2. *Consider a transformer-based large language model with S blocks and model neuron dimension d. n_{token} is the token sequence length, h is the number of attention heads, and f is the feedforward expansion factor. Beyond the shared cost, EAP-ig requires an additional cost*

[1]We omit the theoretical and empirical comparison to RelP (Jafari et al., 2025), as it inherits the runtime of EAP-ig and our reported efficiency advantage over EAP-ig already characterizes the expected gap against RelP.

of $\mathcal{O}\big(12MSn_{token}d^2(2+f) + 12MSn_{token}^2 d + MLd\big)$*, while* `CircuitLasso` *with computational complexity* $\mathcal{O}\big(ML(L-1)d^2 \,/\, 2\sqrt{\epsilon}\big)$ *has guaranteed better efficiency compared to EAP-ig if one of the following conditions holds: (1)* $L(L-1) \,/\, 24\, S(2+f)\sqrt{\epsilon} < n_{token} \ll d$*; (2)* $n_{token} \gg d$ *and* $n_{token} > \sqrt{L(L-1)\, d \,/\, 24\, S\, \sqrt{\epsilon}}$*.*

Please refer to the detailed proof in Appendix A.2. Intuitively, **Proposition 3.2** provides a guideline for selecting the number of component locations of interest (L) and determining which circuit discovery approach is more efficient.

3.3. Circuit Discovery on Sparse Features

Recent work (Marks et al., 2025; Laptev et al., 2025) has shifted circuit discovery from raw neurons to monosemantic SAE features, which yield cleaner, more human-interpretable circuits. However, high SAE dimensions push the cost of intervention-based methods beyond practical limits. We address this by extending our sparse-regression formulation to operate directly on SAE features.

3.3.1. Preliminaries on Sparse Autoencoders

Given a model with d-dimensional latent space and neuron activations $\boldsymbol{h} \in \mathbb{R}^d$, an SAE can represent $\boldsymbol{h}$ as a linear combination of sparse features $\boldsymbol{z} \in \mathbb{R}^D$, $D \gg d$:

$$\boldsymbol{z} = \sigma(\boldsymbol{W}_{\text{enc}}\boldsymbol{h} + \boldsymbol{b}_{\text{enc}}), \quad \hat{\boldsymbol{h}} = \boldsymbol{W}_{\text{dec}}\boldsymbol{z} + \boldsymbol{b}_{\text{dec}},$$

where $\boldsymbol{W}_{\text{enc}}, \boldsymbol{b}_{\text{enc}}$ are encoder parameters and $\boldsymbol{W}_{\text{dec}}, \boldsymbol{b}_{\text{dec}}$ are decoder parameters; $\sigma(\cdot)$ is a nonlinear activation function. The SAEs are usually trained by minimizing the reconstruction error between model activations $\boldsymbol{h}$ and reconstructed activations $\hat{\boldsymbol{h}}$ subject to a sparsity regularizer:

$$\mathcal{L}_{\text{SAE}} := \|\boldsymbol{h} - \hat{\boldsymbol{h}}\|_2^2 + \alpha\mathcal{L}_{\text{reg}}(\boldsymbol{z}).$$

Recent work on SAEs explores various methods to achieve a better trade-off between fidelity and sparsity. **However, our focus is to uncover the structural dependencies among learned sparse SAE features, rather than to develop new SAE training methods.** We directly employ pre-trained SAEs on LLMs of various sizes. Details can be found in Appendix D.1.

3.3.2. Layer-wise Sparse Feature Circuit Discovery

We follow our formulation on neurons and still assume that structural relations follow the computation order of the underlying model neurons. To be specific, consider two model neurons of dimension d and their activations at locations i and j, denoted by $\boldsymbol{h}_i, \boldsymbol{h}_j \in \mathbb{R}^d$, where computation at i precedes computation at j. We obtain the corresponding SAE features $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{R}^D$ using trained SAEs:

$$\begin{aligned}\boldsymbol{z}_i &= \sigma(\widehat{\boldsymbol{W}}_{\text{enc},i}\boldsymbol{h}_i + \widehat{\boldsymbol{b}}_{\text{enc},i}),\\ \boldsymbol{z}_j &= \sigma(\widehat{\boldsymbol{W}}_{\text{enc},j}\boldsymbol{h}_j + \widehat{\boldsymbol{b}}_{\text{enc},j}).\end{aligned}$$

If a dependency exists between variables in $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, we constrain its direction to be from i to j. Given M observations of $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, we obtain input data $\boldsymbol{Z}_i \in \mathbb{R}^{D\times M}$ and $\boldsymbol{Z}_j \in \mathbb{R}^{D\times M}$. We estimate these relations by solving:

$$\widehat{A}_{i,j} = \arg\min_{A_{i,j}} \|\boldsymbol{Z}_j - A_{i,j}^\top \boldsymbol{Z}_i\|_F^2 + \lambda\|A_{i,j}\|_1, \quad (4)$$

where $A_{i,j} \in \mathbb{R}^{D\times D}$. This procedure is repeated for every pair (i,j) where i precedes j in the computation order. **In particular, learning $A_{i,j}$ for all transformer block outputs in consecutive layers provides insight into how semantic concepts are transferred and propagated, and how they evolve, across the model.** The computational cost of the learning problem in Eq. (4) is $\mathcal{O}(\frac{MD^2}{\sqrt{\epsilon}})$.

We also incorporate the downstream prediction target into circuit discovery to enable explanation of the model's predictive behavior. We formulate the following optimization problem to learn a model for predicting the downstream target y using SAE features $\boldsymbol{z}_i$, derived from model neuron activations at location i:

$$\widehat{A}_{i,y} = \arg\min_{A_{i,y}} \mathcal{L}_{\text{pred}}(y, A_{i,y}^\top \boldsymbol{Z}_i) + \lambda\|A_{i,y}\|_1, \quad (5)$$

where $A_{i,y} \in \mathbb{R}^D$ and $\mathcal{L}_{\text{pred}}(\cdot,\cdot)$ denotes the prediction loss[2]. In practice, with the learned $A_{i,y}$ and interpretable sparse features $\boldsymbol{z}$, we can not only explain the model's predictive behavior, but also rectify the prediction model to mitigate spurious or biased behavior. The computational cost of Eq. (5) is $\mathcal{O}(\frac{MD}{\sqrt{\epsilon}})$.

4. Experiments

We evaluate `CircuitLasso` on both model neurons (Section 4.1) and sparse autoencoder features (Section 4.2). Our central claim is *efficiency at parity of accuracy*: `CircuitLasso` recovers circuits whose structural accuracy matches state-of-the-art intervention-based methods, at a fraction of the computational cost. We then apply `CircuitLasso` to a setting whose inner workings have not previously been studied in mechanistic interpretability, and show that the recovered SAE-feature circuits yield interpretable insights. Finally, we apply our learned circuits to a benchmark domain-generalization (DG) task, achieving comparable, and in some cases slightly better, performance at substantially lower cost.

[2]Mean squared error for regression tasks and cross-entropy loss for classification tasks.

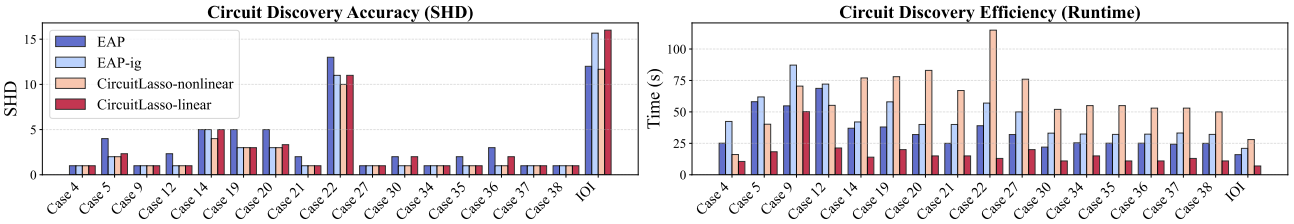


Figure 2. **Circuit discovery accuracy and efficiency on InterpBench.** We compare our `CircuitLasso` against the SOTA circuit learning baselines (EAP, EAP-ig) across 16 synthetic cases and the real IOI task. **Left:** SHD between recovered and ground-truth circuits (lower is better); **Right:** wall-clock runtime in seconds (lower is better).

4.1. Circuit Discovery on Model Neurons

Setup. We evaluate `CircuitLasso` on INTERPBENCH (Gupta et al., 2024), a collection of 86 semi-synthetic transformers with known ground-truth circuits, against the state-of-the-art Edge Attribution Patching (EAP) (Syed et al., 2024) and EAP with integrated gradients (EAP-ig) (Hanna et al., 2024). Following the protocol of Gupta et al. (2024), we evaluate on the *16 main synthetic cases* and the *real Indirect Object Identification (IOI) case*. Accuracy is measured by Structural Hamming Distance (SHD) and efficiency by runtime in seconds, averaged over three trials on a single NVIDIA A100. The full learning procedure is in Appendix C.2.

Results (Figure 2). *(i) Comparable accuracy at substantially lower cost.* Across the 17 tasks, `CircuitLasso`-linear attains a mean SHD of 3.16, statistically indistinguishable from EAP-ig (2.98) and below EAP (3.61), at a mean runtime of 16.3 s per case, 3.0 times faster than EAP-ig (49.1 s) and 2.1 times faster than EAP (33.7 s). In short, `CircuitLasso` recovers circuits as accurately as the strongest intervention-based baseline at a fraction of the cost, directly supporting our "efficiency at parity of accuracy" claim. *(ii) The nonlinear variant brings diminishing returns.* `CircuitLasso`-nonlinear achieves the lowest mean SHD (2.84), only marginally below `CircuitLasso`-linear, but at 3.7 times the runtime, even slower than EAP-ig in most cases. This is consistent with the intuition that the extra nonlinear capacity is wasted when the dependency structure is already well-captured by a linear edge-importance score.

4.2. Circuit Discovery on SAE Features

We adapt `CircuitLasso` to learn structural circuits on SAE features (as described in Section 3.3.2), revealing model behaviors in terms of human-interpretable concepts (Section 4.2.1). Following Marks et al. (2025), we also leverage insights from learned circuits for downstream domain generalization (Section 4.2.2).

4.2.1. INTERPRETABILITY CASE STUDIES

Data and Model. We demonstrate our approach on the **Co**rpus of **L**inguistic **A**cceptability (CoLA) task (Warstadt et al., 2019) from the GLUE benchmark (Wang et al., 2018), aiming to reveal the inner workings of GPT-2 small (Radford et al., 2019) through interpretable features derived from OpenAI's pre-trained sparse autoencoders (Gao et al., 2025) for GPT-2 small. The CoLA dataset is, to our knowledge, *new* to mechanistic interpretability studies. It contains 10,657 sentences from 23 linguistics publications, annotated for grammaticality by the original authors. We conduct our interpretability experiments on the 8,551 training sentences in the public release.

Sparse Feature Interpretation within Learned Circuits. We extract GPT-2 small's neuron activations and the corresponding SAE features on the M training sentences (Appendix C.3).

We construct circuits at two granularities, controlled by how the importance score is computed. For the dataset-level interpretability case study reported below (Figure 3 and Appendix D.4), we rank features at the prediction layer by the *dataset-level coefficient* $|A_{L,y}|$ obtained from the regression fit jointly on all M training prompts, yielding a single circuit that summarizes the model's behavior across the dataset. When the goal is to explain a single decision, we instead use the prompt-specific Hadamard product $\boldsymbol{s} = |A_{L,y}| \odot |\boldsymbol{z}_L| \in \mathbb{R}^D$ for the prompt of interest, which preserves the learned linear coefficients but reweights them by that prompt's activation pattern. Prompt averaging is appropriate when the target object of study is a population-level dependency, e.g., the mechanism by which the model generally distinguishes grammatical from ungrammatical sentences, while the prompt-specific score is appropriate when the target is per-instance mechanism. All circuits shown in the main paper use prompt averaging; per-prompt analyses are illustrated in Appendix C.3.

For each important feature, we label its semantic concept using two complementary procedures: a *multi-prompt* approach that aggregates tokens which strongly activate the feature across many prompts, and a *single-prompt* approach

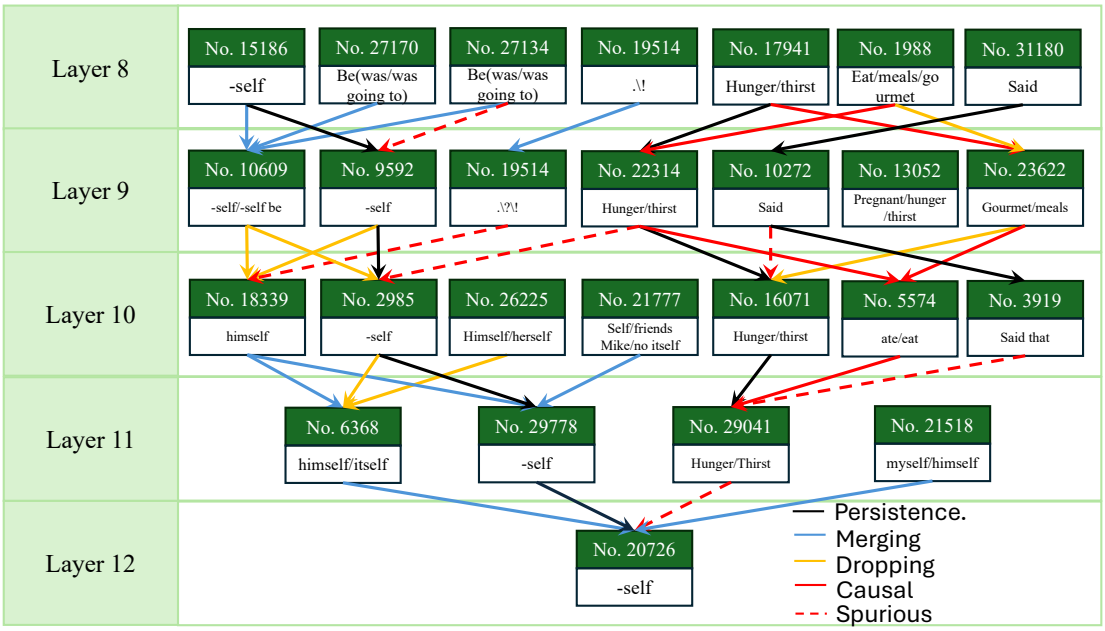


Figure 3. The learned circuits over SAE features on GPT-2 small model. Different colors represent different types of edges in the computation graph. The figure illustrates one example of a valid circuit path traced through the learned adjacency matrices: starting from four interpretable final-layer features ($z_{12,20726}$, $z_{12,776}$, $z_{12,19322}$, $z_{12,3092}$ labeled "-self," "hunger/thirst," "tired/weary," and "ending punctuation"), we recurse backward through $A_{i-1,i}$ to surface their most influential parents at each preceding layer. We do not impose a fixed cap on the number of features per layer or a global coefficient threshold; rather, the figure shows a representative traversal whose roots and parents are chosen for human-interpretable illustration, and an additional traversal starting from a different root is reported in Appendix D.4.

that perturbs targeted tokens in a single prompt and verifies that the activation changes accordingly. We then trace each important feature backward through earlier layers via the learned adjacency $A_{i-1,i}$ and label its parents with the same procedures, yielding tree-shaped circuit paths across transformer blocks. Full procedure, worked examples (e.g., feature $z_{12,20726}$ for "-self," $z_{12,776}$ for "thirst/hunger"), and per-feature tables are deferred to Tables 3, 4, 5, and 6 in the same section of Appendix.

Figure 3 presents such a tree-shaped circuit, consisting of sparse features with human-interpretable meanings across 5 layers. An additional example is provided in Appendix D.4. From the circuits in Figure 3, we make the following observations:

Persistence. Certain semantic concepts persist along circuit paths across multiple layers, particularly in the later layers. For example, the concept of "-self" is present in the 20726th feature of layer 12, the 6368th feature of layer 11, the 2985th feature of layer 10, the 9592th feature of layer 9, and the 15186th feature of layer 8. We highlight in black the circuit paths that capture persistence relations between consecutive layers.

Merging and Dropping. We also observe that sparse features in later layers can merge semantic concepts from multiple parent features in the preceding layer, or disregard (i.e., drop) certain concepts contributed by those parent features. For instance, the 10609th feature of layer 9 merges concepts from both the 15186th and 27170th features of layer 8. In contrast, the 6368th feature of layer 11 retains only the concept "himself" and disregards all other forms of "-self" from the 2985th feature of layer 10. We highlight circuit paths representing propagation in blue and decomposition in orange.

Cause-Effect and Spurious Correlations. Our circuits can capture causal relations between features that encode cause–effect semantic concepts. For example, the 22314th feature of layer 9 represents the concept of "hunger/thirst," which can be considered a cause of the action "ate/eat," encoded in the 5574th feature of layer 10. However, our assumption that causal orientations align with the computation order results in some circuit paths appearing anti-causal. From a human perspective, one typically feels hungry before taking actions such as "eat food/meals/gourmet," yet our circuit includes a path from the 1988th feature of layer 8 to the 22314th feature of layer 9, which implies the reverse. Moreover, our circuits also capture spurious correlations. For example, the 20726th feature of layer 12, which represents "-self," is spuriously correlated with the "hunger/thirst" concept encoded in the 29041th feature of layer 11. Such correlations are likely in-

Table 1. Runtime and numbers of selected features for SHIFT versus our `CircuitLasso` method. The runtime does not include manual interpretation time.

Method	Pythia-70M		Gemma-2-2b		Gemma-2-9b	
	# of features	Runtime (s) ↓	# of features	Runtime (s) ↓	# of features	Runtime (s) ↓
SHIFT	49	257.6	65	371.2	71	908.4
SHIFT-retrain		356.3		476.8		1056.0
`CircuitLasso`	41	36.5	55	47.2	59	107.4
`CircuitLasso`-retrained		61.9 (17.37%↑)		72.5 (15.20%↑)		125.2 (11.98% ↑)

Table 2. Prediction accuracy (%) on the Bias-in-Bios dataset across three LLMs and debiasing methods. For *Prof.* (profession-prediction accuracy on the balanced test set) and *Worst* (accuracy on the worst-performing demographic subgroup), higher is better. For *Gender* (how predictable gender is from the predictor's representation), values close to 50% are better, indicating that the predictor does not rely on the spurious gender signal; values much above 50% indicate residual gender leakage, while values much below 50% indicate over-correction. ORACLE is trained on the balanced set and shown as a non-comparable upper bound; bold marks the best non-ORACLE entry per column.

Method	Pythia-70M			Gemma-2-2B			Gemma-2-9B		
	Profession (↑)	Gender	Worst group(↑)	Profession(↑)	Gender	Worst group(↑)	Profession(↑)	Gender	Worst group(↑)
ORIGINAL	61.9	87.4	24.4	69.6	79.5	4.1	70.8	78.2	23.4
CBP	83.3	60.1	67.7	90.1	**50.2**	86.8	94.7	**50.0**	91.5
ORACLE	93.0	49.4	91.4	**95.1**	**50.2**	91.7	95.7	**50.0**	90.5
LINEAR PROBING	84.0	58.0	70.0	90.5	51.5	87.5	95.0	50.5	91.0
SHIFT	88.5	54.0	76.0	72.8	51.6	43.7	77.1	52.8	67.9
SHIFT-retrain	93.1	52.0	**89.0**	94.2	52.4	92.4	96.0	51.3	90.3
`CircuitLasso`	90.5	**50.1**	75.8	77.5	50.7	50.5	81.5	50.3	69.8
`CircuitLasso`-retrain	**94.2**	50.6	88.7	**95.1**	52.8	**92.9**	**96.9**	50.5	**91.5**

troduced by biases in the training data, such as the frequent co-occurrence of these two semantic concepts within the same sentence. By analyzing these circuit paths, we can infer the nature of dataset biases and potentially mitigate them through targeted model editing. We next show how such insights from a learned circuit can be leveraged to improve downstream domain generalization in Section 4.2.2.

Faithfulness and Completeness. We evaluate the learned SAE-feature circuits on CoLA using the faithfulness and completeness metrics of Marks et al. (2025). As Figure 4 shows, our circuits match the intervention-based SHIFT baseline under standard node ablation, while requiring no per-edge interventions. Because our sparse-regression framework learns explicit edge coefficients, we additionally introduce *edge ablation*, a capability that SHIFT does not support, and confirm that a small subset of essential edges (together with their connected SAE features) governs the model's prediction behavior. Full methodology and per-quartile results are in Appendix D.5.

4.2.2. Downstream Domain Generalization: A Utility Demonstration

We treat this section as a *utility demonstration* of the insights from our learned circuits, not as a multi-layer circuit learning experiment. Our central claim remains efficiency improvements and any accuracy advantage is interpreted as a side benefit of operating on disentangled SAE features, not as a contribution per se.

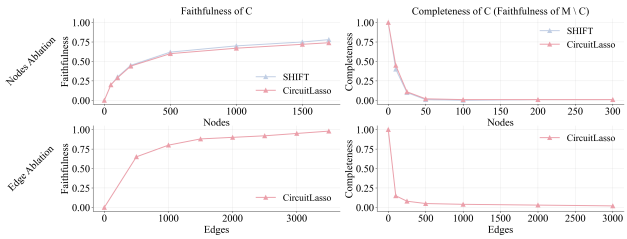


Figure 4. Faithfulness and completeness scores for the learned circuits on the CoLA training set, under node ablation (top) and edge ablation (bottom). Ideal faithfulness is 1, ideal completeness is 0.

Setup. We evaluate on the Bias-in-Bios dataset (BiB) (De-Arteaga et al., 2019) on Pythia-70M (Biderman et al., 2023), Gemma-2-2B and Gemma-2-9B (Team et al., 2024), using the pre-trained SAEs from Marks et al. (2025) for Pythia-70M and Lieberum et al. (2024) for Gemma-2-2B and Gemma-2-9B. The task is to predict profession from a biography under a gender-spurious training set and evaluate on a balanced test set in which profession and gender are independent. We compare against the following baselines: ORIGINAL, ORACLE, CBP (Yan et al., 2023), SHIFT and SHIFT-retrain (Marks et al., 2025), and LINEAR PROB-

ING (Gurnee et al., 2023).

Method. We rank SAE features at a single layer by $|\widehat{A}_{i,y}|$, manually identify gender-correlated features, zero them, and feed the result either directly into the trained classifier (`CircuitLasso`) or into a freshly retrained classifier (`CircuitLasso`-retrain). Full setup, gender-leakage, and worst-group columns are in Appendix D.6.

Results. Table 2 reports profession accuracy across the three LLMs. `CircuitLasso` and `CircuitLasso`-retrain achieve performance comparable to, and in some cases slightly better than, the strongest non-ORACLE baseline; we attribute the small accuracy advantage to finer-grained manipulation of disentangled SAE features rather than to a circuit learning effect. The central efficiency claim is in Table 1: `CircuitLasso` matches the strongest baseline at substantially lower cost, with the gap widening as model size grows.

5. Conclusion and Discussion

We present `CircuitLasso`, a sparse-regression surrogate for circuit discovery in LLMs that replaces the costly intervention-based pipeline of prior work with a Lasso-based, observational alternative. On the INTERPBENCH benchmark, `CircuitLasso` achieves *efficiency at parity of accuracy*: it matches the structural-recovery quality of EAP and EAP-ig at a fraction of the runtime, and scales cleanly to the high-dimensional SAE feature spaces where intervention-based methods become prohibitive. Operating on monosemantic SAE features further yields tree-shaped circuit paths whose nodes carry human-interpretable semantic labels, and a utility demonstration on Bias-in-Bios shows that insights from the learned circuits support domain generalization at substantially lower cost than the strongest non-ORACLE baseline.

Future work. Three directions stand out. *(i) Beyond a first-order surrogate.* Our linear formulation recovers the dependency skeleton accurately, but edge weights are not exact causal effects of the underlying nonlinear computation; characterizing regimes in which the linear coefficients become quantitatively faithful is an open question. *(ii) Architectures with within-layer feedback.* The block-triangular acyclicity we use is supplied by the transformer's feed-forward order; extending `CircuitLasso` to architectures with within-layer feedback would broaden its applicability. *(iii) Beyond approximate causal sufficiency.* The residual stream and SAE reconstruction error mean causal sufficiency holds only approximately; combining `CircuitLasso` with richer residual-stream models is a natural next step.

Acknowledgements

N.Y. and Y.Y. acknowledge support by the National Institutes of Health under award 1R01GM157589-01 and the AFOSR Grant No. FA9550-22-1-0197. N.Y. is also supported by IBM through the IBM–Rensselaer Future of Computing Research Collaboration.

References

Ameisen, E., Lindsey, J., Pearce, A., Gurnee, W., Turner, N. L., Chen, B., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. Circuit tracing: Revealing computational graphs in language models. Transformer Circuits Thread, Anthropic. `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`, 2025.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., et al. Towards monosemanticity: Decomposing language models with dictionary learning. `https://transformer-circuits.pub/2023/monosemantic-features`, Oct 2023. Accessed: 2025-09-03.

Bussmann, B., Leask, P., and Nanda, N. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Cao, S., Sanh, V., and Rush, A. M. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 960–966, 2021.

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

De-Arteaga, M., Romanov, A., Wallach, H., Chayes, J., Borgs, C., Chouldechova, A., Geyik, S., Kenthapadi, K., and Kalai, A. T. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 120–128, 2019.

Dunefsky, J., Chlenski, P., and Nanda, N. Transcoders find interpretable llm feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Gao, L., Dupre la Tour, T., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Geiger, A., Ibeling, D., Zur, A., Chaudhary, M., Chauhan, S., Huang, J., Arora, A., Wu, Z., Goodman, N., Potts, C., et al. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, 2023.

Gupta, R., Arcuschin Moreno, I., Kwa, T., and Garriga-Alonso, A. Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. *Advances in Neural Information Processing Systems*, 37: 92922–92951, 2024.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Hanna, M., Pezzelle, S., and Belinkov, Y. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *arXiv preprint arXiv:2403.17806*, 2024.

Jafari, F. R., Eberle, O., Khakzar, A., and Nanda, N. Relp: Faithful and efficient circuit discovery in language models via relevance patching. *arXiv preprint arXiv:2508.21258*, 2025.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Laptev, D., Balagansky, N., Aksenov, Y., and Gavrilov, D. Analyze feature flow to enhance interpretation and steering in language models. In *Forty-second International Conference on Machine Learning*, 2025.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R., and Nanda, N. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, 2024.

Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the biology of a large language model. Transformer Circuits Thread, Anthropic. `https://transformer-circuits.pub/2025/attribution-graphs/biology.html`, 2025.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Nanda, N. Attribution patching: Activation patching at industrial scale. *URL: https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*, 15: 19, 2023.

Nanda, N., Rajamanoharan, S., Kramar, J., and Shah, R. Fact finding: Attempting to reverse-engineer factual recall on the neuron level. In *AI Alignment Forum*, pp. 6, 2023.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Park, G. Identifiability of additive noise models using conditional variances. *Journal of Machine Learning Research*, 21(75):1–34, 2020.

Pearl, J. Causality:models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 2000.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053, 2014.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Reisach, A., Seiler, C., and Weichwald, S. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.

Syed, A., Rager, C., and Conmy, A. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416, 2024.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Sakenis, S., Huang, J., Singer, Y., and Shieber, S. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Yan, A., Wang, Y., Zhong, Y., He, Z., Karypis, P., Wang, Z., Dong, C., Gentili, A., Hsu, C.-N., Shang, J., et al. Robust and interpretable medical image classifiers via concept bottleneck models. *arXiv preprint arXiv:2310.03182*, 2023.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A. Theoretical Proofs

A.1. Proof of Proposition 3.1

Proof. The block upper-triangular structure of A decouples Eq. (3) into L independent multi-output Lasso subproblems, one per output neural activations. Specifically, the ℓ-th subproblem predicts the d-dimensional ℓ-th neuron activations from the $p_\ell = (\ell - 1)\, d$ coordinates of all preceding (earlier-indexed) layer blocks; in particular $p_1 = 0$, since the first layer has no preceding blocks.

Per-iteration cost. Let $\boldsymbol{h}_\ell \in \mathbb{R}^{d \times M}$ denote the ℓ-th neuron activation block and $\boldsymbol{h}_{<\ell} \in \mathbb{R}^{p_\ell \times M}$ denote the stacked activations of all preceding blocks. Each FISTA iteration on the ℓ-th subproblem evaluates the gradient of the quadratic data-fit term, dominated by the matrix product $\boldsymbol{h}_{<\ell}\, \boldsymbol{h}_{<\ell}^\top A_\ell$ with $A_\ell \in \mathbb{R}^{p_\ell \times d}$, at cost $\mathcal{O}(p_\ell\, d\, M)$. The subsequent soft-thresholding (proximal) step is $\mathcal{O}(p_\ell\, d)$ and is dominated. Summing across the L subproblems,

$$\sum_{\ell=1}^{L} p_\ell\, d \;=\; d^2 \sum_{\ell=1}^{L} (\ell - 1) \;=\; \frac{d^2\, L(L-1)}{2},$$

so a single outer iteration across all L subproblems costs $\mathcal{O}\left(\frac{M\, L(L-1)\, d^2}{2}\right)$.

Iteration complexity. The objective of each subproblem is the sum of a smooth quadratic with Lipschitz-continuous gradient (with constant $L_f^{(\ell)} = \lambda_{\max}\big(\boldsymbol{h}_{<\ell}\, \boldsymbol{h}_{<\ell}^\top\big)$) and a non-smooth ℓ_1 regularizer. FISTA on this composite objective satisfies the accelerated rate (Beck & Teboulle, 2009)

$$F(A_\ell^{(k)}) - F(\widehat{A}_\ell) \;\le\; \frac{2\, L_f^{(\ell)}\, \|A_\ell^{(0)} - \widehat{A}_\ell\|_F^2}{(k+1)^2},$$

matching the optimal rate of Nesterov's accelerated gradient method on smooth convex problems. Reaching ϵ-suboptimality therefore requires only $\mathcal{O}(1/\sqrt{\epsilon})$ outer iterations.

Combining the per-iteration and iteration-complexity bounds gives the total cost $\mathcal{O}\left(\frac{M\, L(L-1)\, d^2}{2\sqrt{\epsilon}}\right)$. □

A.2. Proof for Proposition 3.2

Proof. Let d be the model hidden dimension, n_{token} the sequence length, h the number of attention heads (so the head dimension is $k = d/h$), and f the feedforward expansion factor. Per transformer block, the dominant FLOP costs come from the Q/K/V projections ($\approx 6\, n_{\text{token}}\, d^2$), the $QK^\top$ products across all heads ($\approx 2\, n_{\text{token}}^2\, d$), the attention-weighted V across heads ($\approx 2\, n_{\text{token}}^2\, d$), the output projection ($\approx 2\, n_{\text{token}}\, d^2$), and the two-layer feed-forward network of widths $d \to fd \to d$ ($\approx 4f\, n_{\text{token}}\, d^2$). Summing, one forward pass per block costs $\mathcal{O}\big(n_{\text{token}}\, d^2(8 + 4f) + 4\, n_{\text{token}}^2\, d\big)$. A backward pass costs approximately twice the corresponding forward pass, $\mathcal{O}\big(n_{\text{token}}\, d^2(16 + 8f) + 8\, n_{\text{token}}^2\, d\big)$.

Shared activation-collection cost. Both `CircuitLasso` and EAP-ig must perform one forward pass per observation in order to obtain the per-location activations on which their respective methods operate. For M observations across S blocks, this shared cost is $\mathcal{O}\big(4MS\, n_{\text{token}}\, d^2(2 + f) + 4MS\, n_{\text{token}}^2\, d\big)$, and we subtract it from both sides of the comparison below.

EAP-ig additional cost. Beyond activation collection, EAP-ig requires one further forward pass and one backward pass per observation in order to compute the attribution-patching gradients. Across S blocks and M observations this is $\mathcal{O}\big(12MS\, n_{\text{token}}\, d^2(2 + f) + 12MS\, n_{\text{token}}^2\, d\big)$. The subsequent linear attribution-patching step over L component locations contributes an additional $\mathcal{O}(MLd)$, which is dominated by the forward/backward terms. EAP-ig's total additional cost is therefore

$$\mathcal{O}\big(12MS\, n_{\text{token}}\, d^2(2 + f) + 12MS\, n_{\text{token}}^2\, d + MLd\big)\,.$$

`CircuitLasso` additional cost. Beyond activation collection, `CircuitLasso` runs FISTA on the block upper-triangular Lasso problem of Eq. (3), with total cost $\mathcal{O}\left(\frac{ML(L-1)\, d^2}{2\sqrt{\epsilon}}\right)$ by Proposition 3.1.

Efficiency conditions. For `CircuitLasso` to have strictly lower additional cost than EAP-ig, the dominant EAP-ig term must exceed the FISTA cost. We consider the two regimes.

- *Case 1:* $d \gg n_{token}$. The dominant EAP-ig term is $12MS\, n_{\text{token}}\, d^2(2+f)$. Requiring this to exceed the FISTA cost gives

$$\begin{aligned} 12MS\, n_{\text{token}}\, d^2(2+f) \;&>\; \frac{ML(L-1)\, d^2}{2\sqrt{\epsilon}} \\ \implies n_{\text{token}} \;&>\; \frac{L(L-1)}{24\, S(2+f)\sqrt{\epsilon}}. \end{aligned}$$

- *Case 2:* $n_{token} \gg d$. The dominant EAP-ig term is $12MS\, n_{\text{token}}^2\, d$. Requiring this to exceed the FISTA cost gives

$$\begin{aligned} 12MS\, n_{\text{token}}^2\, d \;&>\; \frac{ML(L-1)\, d^2}{2\sqrt{\epsilon}} \\ \implies n_{\text{token}}^2 \;&>\; \frac{L(L-1)\, d}{24\, S\, \sqrt{\epsilon}}, \\ \implies n_{\text{token}} \;&>\; \sqrt{\frac{L\,(L-1)\, d}{24\, S\, \sqrt{\epsilon}}}. \end{aligned}$$

Combining the two regimes gives the stated conditions. □

B. Discussion of Assumptions

Treating Eq. (1) as a strict linear structural equation model (SEM) would, following the continuous causal discovery literature (Peters et al., 2014; Zheng et al., 2018; Park, 2020), require causal sufficiency, a noise model that is either non-Gaussian or Gaussian with equal variance, and a correctly specified linear functional form. All three of these are at best approximations in transformer LLMs, and we explicitly do not appeal to identifiability theorems that depend on them. Instead, we use Eq. (1) as a sparse regression surrogate whose purpose is to recover the dependency skeleton over the modeled components, which features influence which others, at a fraction of the cost of intervention-based circuit discovery. We outline below how each assumption is violated in our setting and why the skeleton-level conclusions of our method are nonetheless meaningful.

On Causal Sufficiency. The residual stream of a transformer carries every preceding layer's contribution forward in parallel with the attention and MLP outputs we explicitly model, and the SAE reconstruction error introduces additional latent variation that is not captured by the recovered sparse features. Causal sufficiency therefore holds only approximately: activations at one location are not generated solely from the parents we include in A. We do not claim to recover a complete causal graph of the LLM. Rather, the dependency skeleton estimated by Eq. (2) is a sparse summary of the linear-projection structure among the modeled components, with unmodeled contributions absorbed into the residual term. The ℓ_1 penalty filters out weak dependencies regardless of whether they originate from omitted parents or from genuine but small effects, so the recovered skeleton remains a useful and conservative map of the components included in the model.

On the Noise Model. LLM activations are deterministic functions of the input prompt: the same prompt, evaluated twice, produces identical activations. The residual ε in Eq. (1) is therefore best understood as *linearization error*, the deviation between the LLM's true nonlinear computation at each modeled component and the best linear projection of that component onto its predecessors over the observed prompts, rather than exogenous stochastic noise. In particular, we do not assume that ε is independent of the parents, and we do not invoke noise distribution conditions (non-Gaussianity, equal variance) that the deterministic generative process would not satisfy in any case. The linear regression in Eq. (2) is justified as a *projection* of the activation matrix onto the sparse linear hypothesis class, not as inference of an underlying stochastic SEM.

Linearity vs. Nonlinearity. The transformer's true computational graph is globally nonlinear: attention, LayerNorm, and the activation functions inside MLP and SAE encoders all but guarantee that exact functional relationships between layer activations are not linear. We adopt the linear formulation as a deliberate first-order surrogate. Two observations justify its use for skeleton recovery. First, where a true nonzero dependency exists in the nonlinear computation, the corresponding linear regression coefficient is generically nonzero, Lasso under moderate model misspecification preserves support recovery up to constants, and edge *presence* (not exact edge weight) is what we use downstream. Second, we

introduce the nonlinear (kernel/MLP) extension of `CircuitLasso`: The linear formulation can be naturally extended to model nonlinear dependencies by representing the relationships between a variable and its parents through kernel functions or multi-layer perceptrons. The recovered topological skeleton is essentially the same as in the linear case, at substantially higher computational cost. We therefore treat the linear surrogate as the operational definition of the method. One can always employ the nonlinear formulation when finer-grained edge attribution is needed. Edge weights in $\widehat{A}$ should not be interpreted as exact causal effects of the underlying nonlinear computation; their role is to rank dependencies for selection, not to quantify them.

On Acyclicity. Reisach et al. (2021) showed that continuous formulation of causal discovery methods, such as NOTEARS (Zheng et al., 2018), can succeed on synthetic benchmarks for the wrong reason, by exploiting marginal variances to recover the topological ordering, and that simply sorting variables by variance often matches their structural recovery. This critique does *not* apply to our setting because the topological ordering is supplied externally, by the LLM's feed-forward computation order, rather than learned from the data. Concretely, our acyclicity constraint is implemented as a fixed block triangular mask on A that respects the architectural layer order and the within-layer attention-then-MLP order; the optimizer never has access to a degree of freedom that could be co-opted by variance-based reordering. The Lasso step in Eq. (3) only selects which entries within the legal triangular pattern are nonzero. We therefore inherit none of the NOTEARS pathologies that the Reisach critique targets, and the acyclicity of the recovered circuit is a property of the architecture, not of an identifiability argument.

C. Experiment Details

C.1. `CircuitLasso` Optimization with FISTA

All instances of the `CircuitLasso` optimization in Eqs. (3), (4), and (5) decompose into multi-output Lasso subproblems of the form $\min_A \frac{1}{2}\|\boldsymbol{X} - A^\top \boldsymbol{X}\|_F^2 + \lambda\|A\|_1$. We solve each subproblem with FISTA (Beck & Teboulle, 2009) with backtracking line search, warm-started across both the neuron activations dimension and the regularization path. Predictor columns are standardized to unit ℓ_2 norm before solving, and coefficients are rescaled afterward. We terminate when the relative objective decrease falls below 10^{-6}. Our solver is implemented in PyTorch on a single NVIDIA A100 GPU.

C.2. `CircuitLasso` on Neurons

We begin by collecting activations from the model neurons. To ensure fairness, we use the same set of input prompts as the baselines. For example, in each case, EAP and EAP-ig employ two sets of data inputs: a clean run with M input prompts, each with n_{token} tokens, i.e., an $M \times n_{\text{token}}$ array of tokens $\boldsymbol{T}_{\text{clean}}$, and a corrupted run of the same dimensionality, an $M \times n_{\text{token}}$ array $\boldsymbol{T}_{\text{corrupted}}$. Our approach combines these two runs into a single dataset, an array $\boldsymbol{T} = (\boldsymbol{T}_{\text{clean}}, \boldsymbol{T}_{\text{corrupted}})$ of $2M \times n_{\text{token}}$ tokens, which is then used to generate neuron activations at various locations in the LLM. Given $\boldsymbol{T}$ and a pre-trained model, we obtain neuron activations at a location i with shape $d \times 2M \times n_{\text{token}}$ and average over tokens to produce activations $\boldsymbol{H}_i \in \mathbb{R}^{d \times 2M}$. Repeating this process across all L locations and sorting them according to the computation order yields $\tilde{\boldsymbol{H}} \in \mathbb{R}^{Ld \times 2M}$. Substituting $\tilde{\boldsymbol{H}}$, the collected data matrix with $N = Ld$ dimensions and $2M$ observations into Eq. (3), we aim to learn a weighted adjacency matrix $A \in \mathbb{R}^{N \times N}$ that encodes the causal relations between neuron locations. Finally, we infer the causal circuit.

C.3. `CircuitLasso` on Sparse Autoencoder Features

We extract GPT-2 small's neuron activations on the M training sentences. We select the final outputs from each transformer block (layer) as our locations of interest. Given a prompt with n_{token} tokens, we obtain transformer outputs at the ith layer with shape $d \times n_{\text{token}}$ and the corresponding sparse autoencoder features with shape $D \times n_{\text{token}}$. We then collect sparse autoencoder features for all M prompts and average across tokens to produce sparse feature activations $\boldsymbol{Z}_i \in \mathbb{R}^{D \times M}$. Repeating this across all L layers yields our dataset $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_L\}$. In our setting, $L = 12$, $d = 768$, $D = 32{,}768$, and $M = 8{,}551$. For the CoLA task, the prediction target y indicates whether a sentence is linguistically acceptable. Substituting $\{\boldsymbol{Z}_i\}_{i=1}^{L}$ and y into Eq. (4) and Eq. (5), we learn weighted adjacency matrices between consecutive layers, $\{A_{i,i+1}\}_{i=1}^{L-1}$, and the weighted adjacency matrix between the final layer sparse autoencoder features $\boldsymbol{z}_L$ and the prediction target y, i.e., $A_{L,y}$.

We expand on the procedure summarised in Section 4.2.1.

Identifying essential features. Starting with the final-layer adjacency $A_{L,y}$, we select features in $\boldsymbol{z}_L \in \mathbb{R}^D$ that are important for predicting y. The measure of importance can be either the absolute coefficients $|A_{L,y}|$, or, when prompt-specific analysis is desired, the Hadamard product $\boldsymbol{s} = |A_{L,y}| \odot |\boldsymbol{z}_L| \in \mathbb{R}^D$ between the absolute coefficients and the absolute activations of $\boldsymbol{z}_L$ for the chosen prompt.

Multi-prompt approach. Given an important feature $z_{L,k}$, we collect multiple prompts together with the tokens that strongly activate the feature, then infer the encoded semantic concept by inspecting the collected tokens. For example, words ending in "-self" consistently activate $z_{12,20726}$, suggesting that this feature captures the presence of such words. Tables 3, 4, and 5 provide additional examples.

Prompts	Values of $z_{12,20726}$	Per-token values
Kiss himself .	0.3051	(0, 0, 0, 1.5254 , 0)
This movie just watches itself .	0.1861	(0, 0, 0, 0, 0, 1.3027 , 0)
This window just opens itself .	0.1759	(0, 0, 0, 0, 0, 1.2311 , 0)
This list includes my name on itself .	0.1697	(0, 0, 0, 0, 0, 0, 0, 1.5277 , 0)
This silver polishes itself .	0.1692	(0, 0, 0, 0, 0, 1.1844 , 0)
He said that himself was hungry.	0.1689	(0, 0, 0, 0, 1.3509 , 0, 0, 0)
Every picture of itself arrived.	0.1665	(0, 0, 0, 0, 1.1652 , 0, 0)
Bill understands Mary and himself .	0.1638	(0, 0, 0, 0, 0, 1.1467 , 0)
Myself saw me.	0.1602	(0, 0, 0.8009 , 0, 0)

Table 3. Multi-prompt approach for identifying semantic concepts for sparse feature $z_{12,20726}$.

Prompts	Per-token values of $z_{12,776}$
Hun ger fainted Sharon.	(0, 0.9980 , 3.3986 , 0, 0, 0, 0)
Many people were dying of thirst .	(0, 0, 0, 0, 0, 0, 2.0910 , 0)
One people was dying of thirst .	(0, 0, 0, 0, 0, 0, 1.8140 , 0)
John whined that he was hungry .	(0, 0, 0, 0, 0, 0, 0, 0, 1.9004 , 0)
Many soldiers have claimed bottled water satisfies thirst best.	(0, 0, 0, 0, 0, 0, 0, 0, 1.9243 , 0, 0)

Table 4. Multi-prompt approach for identifying semantic concepts for sparse feature $z_{12,776}$.

Prompts	Per-token values of $z_{12,19322}$
The teacher became tired of the students.	(0, 0, 0, 0, 2.7252 , 1.0383 , 0, 0, 0)
The president looked weary .	(0, 0, 0, 0, 2.1888 , 0)
Genie intoned that she was tired .	(0, 0, 0, 0, 0, 0, 0, 0, 2.6658)
John placed him busy .	(0, 0, 0, 0, 1.5510 , 0)
Visiting relatives can be boring .	(0, 0, 0, 0, 0, 0, 1.8287 , 0)

Table 5. Multi-prompt approach for identifying semantic concepts for sparse feature $z_{12,19322}$.

Single-prompt approach. To validate the plausibility of the inferred semantic concept, we choose a single prompt, systematically perturb one or more tokens consistent with that concept, and observe the change in $z_{L,k}$. If the perturbation deactivates the feature, the concept is considered reasonable. For example, in the prompt "He said that himself was hungry," the word "himself" (the fifth token) activates $z_{12,20726}$ to 1.3509; replacing it with "him," "he," or other alternatives that lack the "-self" suffix drops the activation to 0. Table 6 provides an additional example.

Backward tracing across layers. Applying the above to the final layer yields semantic labels for the most important features ($z_{12,20726}$ for "-self," $z_{12,3092}$ for "ending punctuation," $z_{12,776}$ for "thirst/hunger," $z_{12,19322}$ for "tired/weary," etc.). For each important final-layer feature $z_{L,k}$, we trace its most influential parents in layer $L-1$ using the learned adjacency

Prompts	Per-token values of $z_{12,3092}$
He said that himself was hungry .	(0, 0, 0, 0, 0, 0, 0, 6.0688)
He said that himself was hungry yet	(0, 0, 0, 0, 0, 0, 0, 0)
He said that himself was hungry ,	(0, 0, 0, 0, 0, 0, 0, 0)
He said that himself was hungry ?	(0, 0, 0, 0, 0, 0, 0, 3.4722)
He said that himself was hungry !	(0, 0, 0, 0, 0, 0, 0, 3.7391)

Table 6. Single-prompt approach for identifying semantic concepts for sparse feature $z_{12,3092}$.

$A_{L-1,L}$; as in the final layer, we may instead use the prompt-specific importance $\boldsymbol{s} = |\partial z_{L,k}/\partial \boldsymbol{z}_{L-1}| \odot |\boldsymbol{z}_{L-1}| \in \mathbb{R}^D$. The top parents (e.g. $z_{11,6368}$, $z_{11,29778}$, $z_{11,29041}$, $z_{11,21518}$) are then labeled by the same multi- and single-prompt procedures. Repeating this across consecutive layer pairs yields tree-shaped circuit paths spanning the transformer blocks.

C.4. `CircuitLasso` for Domain Generalization

We select SAE features from a specific location of a pre-trained LLM, such as the transformer output at layer 22 in Gemma-2-2B. For each prompt, we average the D-dimensional SAE features at this location over tokens and collect them across all M prompts in the training data, producing $\boldsymbol{Z}_s \in \mathbb{R}^{D\times M}$. Substituting $\boldsymbol{Z}_s$ and the target observations $y = (y_1, y_2, \cdots, y_M) \in \mathbb{R}^M$ into Eq. (5), we estimate the weighted adjacency matrix $A_{s,y} \in \mathbb{R}^D$ and identify important features with large absolute coefficients. This process is equivalent to training a sparse linear classifier on the SAE features, which we later exploit for profession prediction. From $A_{s,y}$, we further interpret the semantics of the important features using our proposed *multi-prompt* and *single-prompt* approaches (Section 4.2.1) and manually identify spurious features associated with gender. Unlike the SHIFT method of Marks et al. (2025), which *decodes* SAE features into neuron activations after ablation, we ablate spurious features by setting their values to zero and *directly feed* the resulting SAE feature values into our trained linear classifier. In addition, similar to SHIFT, we also investigate retraining the linear classifier on the ablated SAE features.

D. Interpretability on Sparse Features Circuits

D.1. Preliminary of Sparse Autoencoders

Cunningham et al. (2023) uses a ReLU activation with L_1 sparsity regularization. Subsequent work explores alternative activation functions $\sigma(\cdot)$ to extract desired SAE features. Rajamanoharan et al. (2024) introduces a threshold to determine the minimum pre-activation for feature activation, while Gao et al. (2025) and Bussmann et al. (2024) enforce sparsity by selecting the top K features. Dunefsky et al. (2024) proposes Transcoders, which are similar to SAEs, but focusing on training interpretable approximations of MLPs. In this work, we employ the following pre-trained LLMs and SAEs:

- The open-source GPT-2 small SAEs (Gao et al., 2025) for all sublayers of the open-weights GPT-2 small model (Radford et al., 2019). These SAEs use a ReLU-linear encoder with $D = 32768$ and L_1 sparsity regularization.
- The open-source Pythia-70M SAEs (Marks et al., 2025) for all sublayers of the open-weight Pythia-70M (Biderman et al., 2023). These SAEs use a ReLU-linear encoder with $D = 64 \times d$ and L_1 sparsity regularization.
- The open-source Gemma Scope SAEs (Lieberum et al., 2024) for all sublayers of the open-weights Gemma-2-2B, Gemma-2-9B models (Team et al., 2024). These SAEs use the JumpReLU linear encoder and set $D = 8 \times d$.

D.2. Ablation Study of Sparsity Constraint

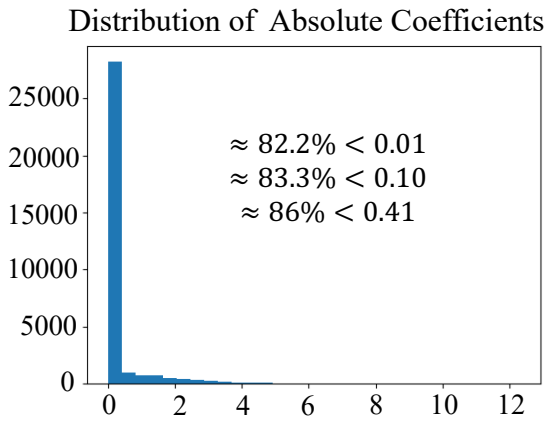
Table 7 presents an ablation study on the sparsity coefficient λ for circuit discovery between the last layer sparse features and prediction target. Without regularization ($\lambda = 0$), the model achieves the highest training accuracy (99.36%) but suffers severe overfitting, as reflected in a large performance drop on the test set (71.10%). Introducing a small sparsity constraint ($\lambda = 10^{-5}$) improves test accuracy to 72.77%, the best among all settings, indicating enhanced generalization. Larger values of λ further enforce sparsity but lead to higher training loss and a notable decline in both training and test accuracy, suggesting that excessive sparsity harms the model's capacity to capture meaningful circuit structure. We therefore select $\lambda = 10^{-5}$ as the optimal setting, as it achieves the best test accuracy while preserving a sparse, interpretable circuit structure.

λ	Training Set			Test Set	
	Pred. Loss	L1 Loss	Pred. Acc. (%)	Pred. Loss	Pred. Acc. (%)
0	0.0612	-	**99.36**	0.9977	71.10
10^{-5}	0.1741	9257.6	95.70	0.6684	**72.77**
5×10^{-5}	0.3498	1898.7	85.44	0.5566	72.48
10^{-4}	0.6143	0.5	70.44	0.6283	69.14
5×10^{-4}	0.5535	45.3	72.14	0.5566	70.28

Table 7. Ablation study of sparsity constraint coefficient λ for circuit discovery between last layer $\boldsymbol{z}_L$ and prediction target y.

D.3. Statistics of Learned Circuit Weighted Adjacency Matrices

Figure 5 examines the coefficient distribution $|A_{L,y}|$ under the best setting ($\lambda = 10^{-5}$). We observe that most coefficient values are extremely small, with 82.2% below 0.01, 83.3% below 0.10, 86% below 0.41, suggesting that only a small subset of features contribute substantially to the prediction. The top 5 essential features clearly dominate the distribution, highlighting the effectiveness of the sparsity constraint in filtering out irrelevant features and isolating semantically interpretable ones.



Top 5 Sparse Features Indices

Indices	Absolute Coefficients
20726	12.3138
3908	10.8914
3261	10.6946
4628	10.5880
1168	10.4870

Figure 5. The distribution of $|A_{L,y}|$ and the selected top 5 essential features.

D.4. More Examples of Sparse Feature Circuits

D.5. Faithfulness and Completeness

We expand on the summary in Section 4.2.1.

Setup. To more comprehensively evaluate the quality of our learned SAE-feature circuits on CoLA, we use the *faithfulness* and *completeness* metrics, following the standard protocol of Marks et al. (2025). Let the learned circuit be C, and define the model output as $m = p(Y = \text{grammatically correct}) - p(Y = \text{grammatically wrong})$.

Node ablation (standard, comparable to SHIFT). We first apply the standard feature-ablation protocol of Marks et al. (2025) and compare against the intervention-based circuit learning method SHIFT. To ensure fairness, we exclude SAE reconstruction errors and attention/MLP SAEs from the SHIFT setting. For `CircuitLasso`, we focus only on SAE features within the learned circuit and still ablate features in the original LLM. The top two plots of Figure 4 show the node-ablation results: our learned circuit achieves performance comparable to SHIFT, consistent with the finding of Marks et al. (2025) that relatively small feature circuits can explain a substantial portion of a model's behavior.

Edge ablation (novel, not supported by SHIFT). Because our sparse-regression framework explicitly learns edge coefficients $\widehat{A}_{i,j}$, each representing the direct influence of a parent SAE feature on a child node, we can ablate a specific edge by setting its coefficient to zero, thereby removing that dependence within our linear surrogate $\widehat{m} = \widehat{A}_{L,y}^\top \boldsymbol{z}_L$.[3] The

[3]We do not apply edge ablation to SHIFT, which does not provide edge-level correlations between SAE features and the output.

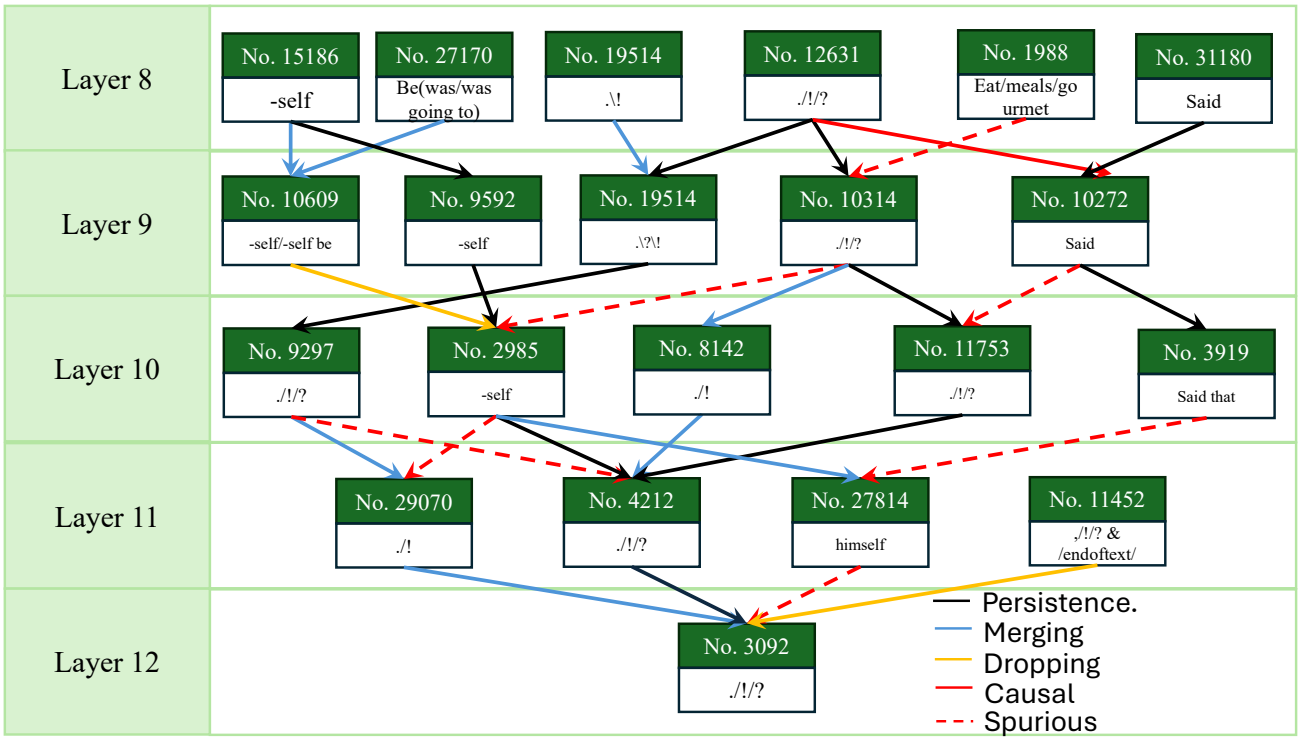


Figure 6. The learned circuit over SAE features on GPT-2 small model. Starting with feature No. 3092 in layer 12.

bottom two plots of Figure 4 report these results. The conclusions mirror those of node ablation: a small subset of essential edges, together with their corresponding SAE features, governs the model's prediction behavior.

D.6. Bias-in-Bios: Full Setup and Results

We expand on the summary in Section 4.2.2.

Data and task. The Bias-in-Bios dataset (BiB) (De-Arteaga et al., 2019) consists of professional biographies with the task of classifying an individual's profession. The dataset encodes a spurious semantic concept, gender, based on which two subsets are constructed: an *ambiguous* set, where profession and gender are strongly correlated (e.g., all professors are assumed to be male, while nurses are assumed to be female), and a *balanced* set, where profession and gender are independent (equal numbers of male professors, male nurses, female professors, and female nurses). The goal is to produce a profession classifier that performs accurately on the balanced set but is not biased by the spurious gender signal in the ambiguous set. While Marks et al. (2025) demonstrate their approach on Pythia-70M and Gemma-2-2B, we extend their evaluation to Gemma-2-9B.

Baselines. We compare against: (i) ORIGINAL, a predictor trained directly on the ambiguous set; (ii) ORACLE, a predictor trained on the balanced set, included as a (non-comparable) upper bound; (iii) CBP, concept bottleneck probing (Yan et al., 2023); (iv) SHIFT, spurious human-interpretable feature trimming (Marks et al., 2025); (v) SHIFT-retrain, the variant in which the linear classifier is retrained on the ablated features; and (vi) a *linear probing* (Gurnee et al., 2023) trained directly on the same SAE features with zero ablation to eliminate the irrelevant features. For SHIFT, we use the SAE-feature variant with manual inspection and exclude the neuron-level and unsupervised variants due to their consistently inferior performance reported in the original paper.

`CircuitLasso` for domain generalization. We select SAE features from a single layer of each LLM (e.g., the transformer output at layer 22 of Gemma-2-2B). For each prompt we average the D-dimensional SAE features over tokens, producing $\boldsymbol{Z}_s \in \mathbb{R}^{D\times M}$. Substituting $\boldsymbol{Z}_s$ and the target $\boldsymbol{y} \in \mathbb{R}^M$ into Eq. (5) estimates the weighted importance vector $\widehat{A}_{s,y} \in \mathbb{R}^D$, which is equivalent to training a sparse linear classifier on the SAE features. We rank features by $|\widehat{A}_{s,y}|$, manually identify gender-correlated features via the multi-prompt and single-prompt procedures, set their values to zero,

and feed the result either directly into the trained classifier (`CircuitLasso`) or into a freshly retrained linear classifier (`CircuitLasso`-retrain).

Full results. Table 2 reports profession accuracy, gender leakage (closer to 50 is better), and worst-group accuracy across the three LLMs. `CircuitLasso` and `CircuitLasso`-retrain achieve profession accuracies comparable to, and in some cases slightly better than, SHIFT and SHIFT-retrain; gender-leakage values stay near the 50% balanced target; and worst-group accuracies match or exceed the strongest non-ORACLE baseline. Efficiency results (Table 1) further underscore the strengths of `CircuitLasso`: it requires fewer features and substantially less runtime than SHIFT, particularly for large models.