

Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models

Samuel Marks*
Northeastern University

Can Rager
Independent

Eric J. Michaud
MIT

Yonatan Belinkov
Technion – IIT

David Bau
Northeastern University

Aaron Mueller*
Northeastern University

Abstract

We introduce methods for discovering and applying **sparse feature circuits**. These are causally implicated subnetworks of human-interpretable features for explaining language model behaviors. Circuits identified in prior work consist of polysemantic and difficult-to-interpret units like attention heads or neurons, rendering them unsuitable for many downstream applications. In contrast, sparse feature circuits enable detailed understanding of unanticipated mechanisms. Because they are based on fine-grained units, sparse feature circuits are useful for downstream tasks: We introduce SHIFT, where we improve the generalization of a classifier by ablating features that a human judges to be task-irrelevant. Finally, we demonstrate an entirely unsupervised and scalable interpretability pipeline by discovering thousands of sparse feature circuits for automatically discovered model behaviors.

1 Introduction

The key challenge of interpretability research is to scalably explain the many unanticipated behaviors of neural networks (NNs). Much recent work explains NN behaviors in terms of coarse-grained model components, for example by implicating certain induction heads in in-context learning (Olsson et al., 2022) or MLP modules in factual recall (Meng et al., 2022; Geva et al., 2023; Nanda et al., 2023, *inter alia*). However, such components are generally polysemantic (Elhage et al., 2022) and hard to interpret, making it difficult to apply mechanistic insights to downstream applications. On the other hand, prior methods analyzing behaviors in terms of fine-grained units (Kim et al., 2018; Belinkov, 2022; Geiger et al., 2023; Zou et al., 2023) demand that researchers begin with the answer: they assume the existence of curated data that isolates the target behavior. Such approaches are not well-suited for discovering unanticipated mechanisms.

We propose to explain model behaviors using fine-grained components that play narrow, interpretable roles. Doing so requires us to address two challenges: First, we must identify the correct fine-grained unit of analysis, since obvious choices like neurons¹ are rarely interpretable, and units discovered via supervised methods require pre-existing hypotheses. Second, we must address the scalability problem posed by searching for causal circuits over a large number of fine-grained units.

We leverage recent progress in dictionary learning (Bricken et al., 2023; Cunningham et al., 2024) to tackle the first challenge. Namely, we train **sparse autoencoders** (SAEs) to identify directions in an LM’s latent space which represent human-interpretable features. Then, to solve the scalability challenge, we employ linear approximations (Sundararajan et al., 2017; Nanda, 2022; Syed et al., 2023) to efficiently identify SAE features which are most causally implicated in model behaviors, as well as connections between these features. The result is

*Correspondence to s.marks@northeastern.edu and aa.mueller@northeastern.edu.

¹We use “neuron” to refer to a basis-aligned direction in an LM’s latent space (not necessarily preceded by a nonlinearity).

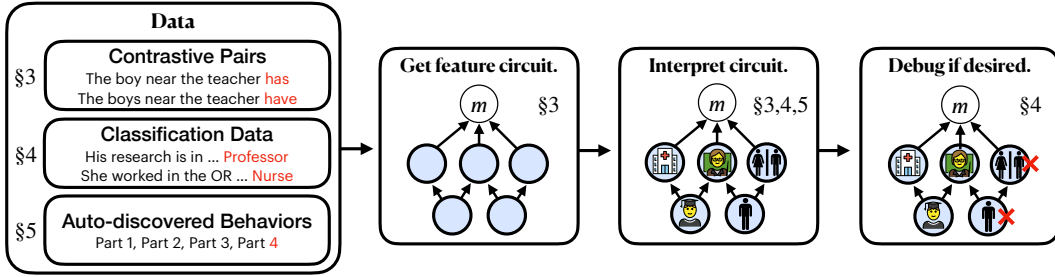


Figure 1: Overview. Given contrastive input pairs, classification data, or automatically discovered model behaviors, we discover circuits composed of human-interpretable sparse features to explain their underlying mechanisms. We then label each feature according to what it activates on or causes to happen. Finally, if desired, we can ablate spurious features out of the circuit to modify how the system generalizes.

a **sparse feature circuit** which explains how model behaviors arise via interactions among fine-grained human-interpretable units.

Sparse feature circuits can be productively used in downstream applications. We introduce a technique, Sparse Human-Interpretable Feature Trimming (SHIFT), which shifts the generalization of an LM classifier by surgically removing sensitivity to unintended signals—even without knowing what those signals are in advance. We demonstrate SHIFT by debiasing a classifier in a worst-case setting where an unintended signal (gender) is perfectly predictive of target labels (profession).

Finally, we demonstrate our method’s scalability by automatically discovering thousands of LM behaviors with the clustering approach of Michaud et al. (2023), and then automatically discovering feature circuits for these behaviors.

Our contributions are summarized as follows (Figure 1):

1. A scalable method to discover sparse **feature circuits**. We validate our method by evaluating feature circuits on a suite of subject-verb agreement tasks.
2. **SHIFT**, a technique for removing sensitivity to unintended signals without disambiguating data.
3. A *fully-unsupervised* automatic feature circuit discovery pipeline that identifies circuits for thousands of automatically discovered LM behaviors.

We release code, data and autoencoders at github.com/saprmrks/feature-circuits.

2 Formulation

Feature disentanglement with sparse autoencoders. A fundamental challenge in NN interpretability is that individual neurons are rarely interpretable (Elhage et al., 2022). Recently, Cunningham et al. (2024); Bricken et al. (2023) have shown that **sparse autoencoders** (SAEs) can be used to identify interpretable directions. We closely follow Bricken et al. (2023) to train SAEs for attention outputs,² MLP outputs, and residual stream activations for each layer of Pythia-70M (Biderman et al., 2023). Given an input activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ from one of these model components, the corresponding SAE computes a decomposition

$$\mathbf{x} = \hat{\mathbf{x}} + \epsilon(\mathbf{x}) = \sum_{i=1}^{d_{\text{SAE}}} f_i(\mathbf{x}) \mathbf{v}_i + \mathbf{b} + \epsilon(\mathbf{x}) \quad (1)$$

into an approximate reconstruction $\hat{\mathbf{x}}$ as a sparse sum of features \mathbf{v}_i and an SAE error term $\epsilon(\mathbf{x}) \in \mathbb{R}^{d_{\text{model}}}$. The *features* $\mathbf{v}_i \in \mathbb{R}^{d_{\text{model}}}$ are unit vectors, the *feature activations* $f_i(\mathbf{x}) \in \mathbb{R}$ are a sparse set of coefficients, $\mathbf{b} \in \mathbb{R}^{d_{\text{model}}}$ is a bias, and we take $d_{\text{SAE}} = 64 \cdot d_{\text{model}}$. The SAEs

²In other words, the output of the attention layer’s out projection.

are trained to minimize an L2 reconstruction error and an L1 regularization term which promotes sparsity. Details about our SAEs and their training can be found in Appendix B.

Scalably training better SAEs is an active area of research, and we develop our methods with rapid future progress in mind. However, at present SAE errors $\epsilon(\mathbf{x})$ account for a relatively significant 1–15% of the variance in \mathbf{x} . Our methods handle these SAE errors gracefully by incorporating them into our sparse feature circuits; this gives a principled decomposition of model behaviors into contributions from potentially-interpretable features and error components not yet captured by our SAEs. We additionally note that the main bottleneck to scaling our methods to larger models is the training of the SAEs themselves. As we expect increasing public availability of SAEs for large open-source models, we treat scaling SAEs themselves as out-of-scope and focus on the scalability of our core methods.

Attributing causal effects with linear approximations. Let m be a real-valued metric computed via a computation graph (e.g., a NN); let $\mathbf{a} \in \mathbb{R}^d$ represent a node in this graph. Following prior work (Vig et al., 2020; Finlayson et al., 2021), we quantify the importance of \mathbf{a} on a pair of inputs $(x_{\text{clean}}, x_{\text{patch}})$ via its *indirect effect* (IE; Pearl, 2001) on m :

$$\text{IE}(m; \mathbf{a}; x_{\text{clean}}, x_{\text{patch}}) = m\left(x_{\text{clean}} \mid \text{do}(\mathbf{a} = \mathbf{a}_{\text{patch}})\right) - m(x_{\text{clean}}). \quad (2)$$

Here $\mathbf{a}_{\text{patch}}$ is the value that \mathbf{a} takes in the computation of $m(x_{\text{patch}})$, and $m(x_{\text{clean}} \mid \text{do}(\mathbf{a} = \mathbf{a}_{\text{patch}}))$ denotes the value of m when computing $m(x_{\text{clean}})$ but *intervening* in the computation of m by manually setting \mathbf{a} to $\mathbf{a}_{\text{patch}}$. For example, given inputs $x_{\text{clean}} = \text{"The teacher"}$ and $x_{\text{patch}} = \text{"The teachers,"}$ we have metric $m(x) = \log P(\text{"are"}|x) - \log P(\text{"is"}|x)$ the log probability difference output by a LM. Then if \mathbf{a} is the activation of a particular neuron, a large value of $\text{IE}(m; \mathbf{a}; x_{\text{clean}}, x_{\text{patch}})$ indicates that the neuron is highly influential on the model’s decision to output “is” vs. “are” on this pair of inputs.

We often want to compute IEs for a very large number of model components \mathbf{a} , which cannot be done efficiently with (2). We thus employ linear approximations to (2) that can be computed for many \mathbf{a} in parallel. The simplest such approximation, *attribution patching* (Nanda, 2022; Syed et al., 2023; Kramár et al., 2024), employs a first-order Taylor expansion

$$\hat{\text{IE}}_{\text{atp}}(m; \mathbf{a}; x_{\text{clean}}, x_{\text{patch}}) = \nabla_{\mathbf{a}} m|_{\mathbf{a}=\mathbf{a}_{\text{clean}}} (\mathbf{a}_{\text{patch}} - \mathbf{a}_{\text{clean}}) \quad (3)$$

which estimates (2) for every \mathbf{a} in parallel using only two forward and one backward pass.

In practice, since we use small models, we can instead employ a more expensive but more accurate approximation based on integrated gradients (Sundararajan et al., 2017):

$$\hat{\text{IE}}_{\text{ig}}(m; \mathbf{a}; x_{\text{clean}}, x_{\text{patch}}) = \left(\sum_{\alpha} \nabla_{\mathbf{a}} m|_{\alpha \mathbf{a}_{\text{clean}} + (1-\alpha) \mathbf{a}_{\text{patch}}} \right) (\mathbf{a}_{\text{patch}} - \mathbf{a}_{\text{clean}}) \quad (4)$$

where the sum in (4) ranges over $N = 10$ equally-equally spaced $\alpha \in \{0, \frac{1}{N}, \dots, \frac{N-1}{N}\}$. This cannot be done in parallel for two nodes when one is downstream of another, but can be done in parallel for arbitrarily many nodes which do not depend on each other. Thus the additional cost of computing $\hat{\text{IE}}_{\text{ig}}$ over $\hat{\text{IE}}_{\text{atp}}$ scales linearly in N and the serial depth of m ’s computation graph.

The above discussion applies to the setting where we have a pair of clean and patch inputs, and we would like to understand that effect of patching a particular node from its clean to patch values. But in some settings (§4, 5), we have only a single input x . In this case, we instead approximate the indirect effect $\text{IE}(m; \mathbf{a}; x) = m(x \mid \text{do}(\mathbf{a} = \mathbf{0})) - m(x)$ of setting \mathbf{a} to $\mathbf{0}$. We get the modified formulas for $\hat{\text{IE}}(m; \mathbf{a}; x)$ from (3) and (4) by replacing \mathbf{a} with $\mathbf{0}$.

3 Sparse Feature Circuit Discovery

In this section, we introduce **sparse feature circuits**, which are computational sub-graphs that explain model behaviors in terms of SAE features and error terms. We first explain

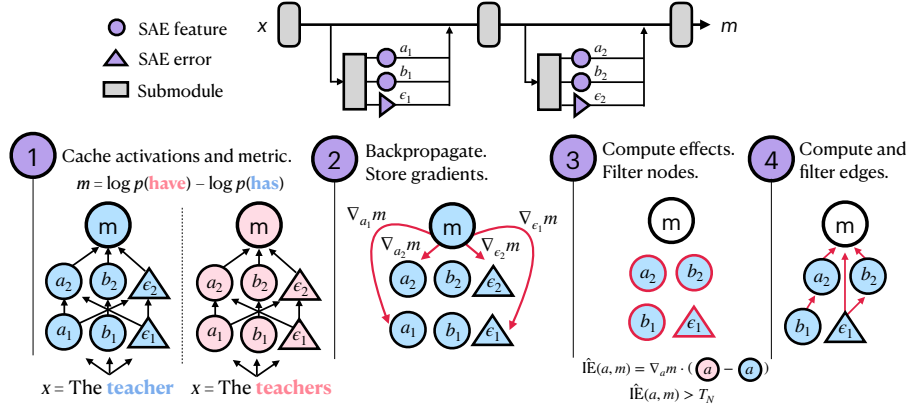


Figure 2: Overview of our method. We view our model as a computation graph that includes SAE features and errors. We cache activations (Step 1) and compute gradients (Step 2) for each node. We then compute approximate indirect effects with Eq. (3; shown) or (4) and filter according to a node threshold T_N (Step 3). We similarly compute and filter edges (Step 4); see App. A.1.

our circuit discovery algorithm (§3.1). Then, we analyze circuits found with our technique to show that they are both more interpretable and more concise than circuits consisting of neurons (§3.2). Finally, we perform a case study of the circuits we discovered for two linguistic tasks related to subject-verb agreement (§3.3).

3.1 Method

Suppose we are given an LM M , SAEs for various submodules of M (e.g., attention outputs, MLP outputs, and residual stream vectors, as in §2), a dataset \mathcal{D} consisting either of contrastive pairs $(x_{\text{clean}}, x_{\text{patch}})$ of inputs or of single inputs x , and a metric m depending on M 's output when processing data from \mathcal{D} . For example, Figure 2 shows the case where \mathcal{D} consists of pairs of inputs which differ in number, and m is the log probability difference between M outputting the verb form that is correct for the patch vs. clean input.

Viewing SAE features as part of the model. A key idea underpinning our method is that, by applying the decomposition (1) to various hidden states x in the LM, we can view the feature activations f_i and SAE errors ϵ as being part of the LM's computation. We can thus represent the model as a computation graph G where nodes correspond to feature activations or SAE errors at particular token positions.

Approximating the IE of each node. Let $\hat{\text{IE}}$ be one of $\hat{\text{IE}}_{\text{atp}}$ or $\hat{\text{IE}}_{\text{ig}}$ (see §2). Then for each node a in G and input $x \sim \mathcal{D}$, we compute $\hat{\text{IE}}(m; a; x)$. We apply some choice of node threshold T_N to select nodes with a large (absolute) IE.

Consistent with prior work (Nanda, 2022; Kramár et al., 2024), we find that $\hat{\text{IE}}_{\text{atp}}$ accurately estimates IEs for SAE features and SAE errors, with the exception of nodes in the layer 0 MLP and early residual stream layers, where $\hat{\text{IE}}_{\text{atp}}$ underestimates the true IE. We find that $\hat{\text{IE}}_{\text{ig}}$ significantly improves accuracy for these components, so we use it in our experiments below. See Appendix C for more information about linear approximation quality.

Approximating the IE of edges. Using an analogous linear approximation, we also compute the average IE of edges in the computation graph. Although the idea is simple, the mathematics are somewhat involved, so we relegate the details to App. A.1. After computing these IEs, we filter for edges with absolute IE exceeding some edge threshold T_E .

Aggregation across token positions and examples. For templatic data where tokens in matching positions play consistent roles (see §3.2, 3.3), we take the mean effect of nodes/edges across examples. For non-templatic data (§4, 5) we first sum the effects

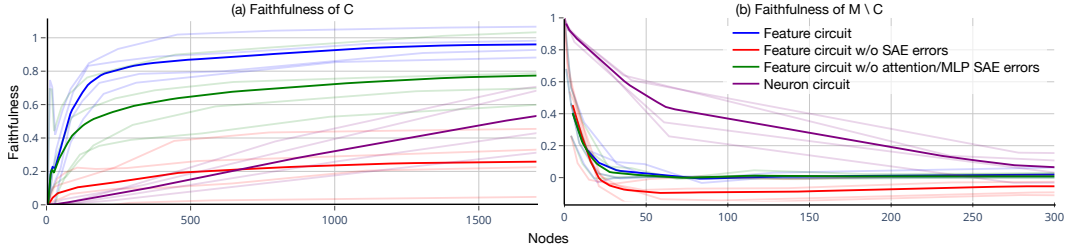


Figure 3: Faithfulness for circuits (a) and their complements (b), measured on held-out data. Faint lines correspond to the structures from Table 1, with the average in bold. The ideal faithfulness for circuits is 1, while the ideal score for their complements is 0.

of corresponding nodes/edges across token position before taking the example-wise mean. See App. A.2.

Practical considerations. Various practical difficulties arise for efficiently computing the gradients needed by our method. We solve using a combination of stop gradients, pass-through gradients, and tricks for efficient Jacobian-vector product computation; see App. A.3.

3.2 Discovering and Evaluating Sparse Feature Circuits for Subject–Verb Agreement

To evaluate our method, we discover sparse feature circuits (henceforth, feature circuits) for four variants of the subject-verb agreement task (Table 1). Specifically, we adapt data from Finlayson et al. (2021) to produce datasets consisting of contrastive pairs of inputs that differ only in the grammatical number of the subject; the model’s task is to choose the appropriate verb inflection. We use this data along with our SAEs from §2 to discover circuits for Pythia-70M.

We evaluate circuits for **interpretability**, **faithfulness**, and **completeness**. For each criterion, we compare to *neuron* circuits discovered by applying our methods with neurons in place of sparse features; in this setting, there are no error terms ϵ .

Interpretability. We asked human crowdworkers to rate the interpretability of random features, random neurons, features from our feature circuits, and neurons from our neuron circuits. Crowdworkers rate sparse features as significantly more interpretable than neurons, with features that participate in our circuits also being more interpretable than randomly sampled ones; see App. D.

Faithfulness. Given a circuit C and metric m , let $m(C)$ denote the average value of m over \mathcal{D} when running our model with all nodes outside of C mean-ablated, i.e., set to their average value over data from \mathcal{D} .³ We then measure faithfulness as $\frac{m(C) - m(\emptyset)}{m(M) - m(\emptyset)}$, where \emptyset denotes the empty circuit and M denotes the full model. Intuitively, this metric captures the proportion of the model’s performance our circuit explains, relative to mean ablating the full model (which represents the “prior” performance of the model when it is given information about the task, but not about specific inputs). We find that model components in early layers typically handle specific tokens; since the tokens appearing in our held-out evaluation set don’t necessarily align with those in our circuit discovery set, there isn’t *prima facie* reason for early circuit components to generalize. We thus measure the faithfulness of our circuits starting at layer 2.

We plot faithfulness for feature circuits and neuron circuits after sweeping over node thresholds (Figure 3a). We find that small feature circuits explain a large proportion of model behavior: the majority of performance is explained by less than 100 nodes. In contrast, around 1500 neurons are required to explain half the performance. However, as SAE errors are high-dimensional and coarse-grained, they cannot be fairly compared to neurons; we thus also plot the faithfulness of feature circuits with all SAE error nodes

³Following Wang et al. (2023), we ablate features by setting them to their mean *position-specific* values.

Structure	Example <i>clean</i> input	Example output
Simple	The parents	$p(\text{is}) - p(\text{are})$
Within RC	The athlete that the managers	$p(\text{likes}) - p(\text{like})$
Across RC	The athlete that the managers like	$p(\text{do}) - p(\text{does})$
Across PP	The secretaries near the cars	$p(\text{has}) - p(\text{have})$

Table 1: Example clean inputs x and outputs m for subject-verb agreement tasks.

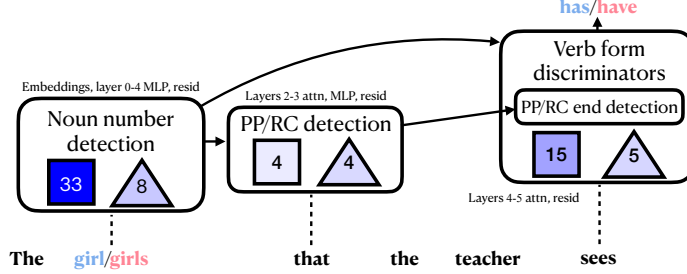


Figure 4: Summary of the circuit for agreement across RC (full circuit in appendix F.1). The model detects the number of the subject. Then, it detects the start of a PP/RC modifying the subject. Verb form discriminators promote particular verb inflections (singular or plural). Squares show number of feature nodes in the group and triangles show number of SAE error nodes, with the shading indicating the sum of $\hat{I}\hat{E}$ terms across nodes in the group.

removed, or with all attention and MLP error nodes removed. We find (unsurprisingly) that removing residual stream SAE errors severely disrupts the model and curtails its maximum performance. Removing MLP and attention errors is less disruptive.

Completeness. Are there parts of the model behavior that our circuit fails to capture? We measure this as the faithfulness of the circuit’s complement $M \setminus C$ (Figure 3b). We observe that we can eliminate the model’s task performance by ablating only a few nodes from our feature circuits, and that this is true even when we leave all SAE errors in place. In contrast, it takes hundreds of neurons to achieve the same effect.

3.3 Case study: Subject-verb agreement across a relative clause

We find that inspecting small feature circuits produced by our technique can provide insights into how Pythia-70M arrives at observed behaviors. To illustrate this, we present a case study of a small feature circuit for subject-verb agreement across a relative clause (RC).

To keep the number of nodes we need to annotate manageable, we set a relatively high node threshold of 0.1, resulting in a circuit with 69 nodes and faithfulness .19 (computed as in §3.2). We summarize this circuit in Figure 4; the full circuit (as well as small circuits for other subject-verb agreement tasks) can be found in App. F.1. We depict SAE features with rectangles and SAE errors with triangles. We generally noticed circuits discovered with qualitatively better SAEs attributed a smaller proportion of the total effect to SAE error nodes; this suggests that our circuit discovery technique could be adapted into a measure of SAE quality.

Our circuit depicts an interpretable algorithm wherein Pythia-70M selects appropriate verb forms via two pathways. The first pathway consists of MLP and embedding features which detect the number of the main subject and then generically promote matching verb forms. The second pathway begins the same, but moves the relevant number information to the end of the relative clause by using PP/RC boundary detectors.

We find significant overlap between this circuit and the circuit we discovered for agreement across a prepositional phrase, with Pythia-70M handling these syntactically distinct structures in a mostly uniform way. In accordance with Finlayson et al. (2021), we find less overlap with our circuits for simple agreement and within RC agreement (Appendix F.1).

Method	Accuracy		
	↑ Profession	↓ Gender	↑Worst group
Original	61.9	87.4	24.4
CBP	82.5	55.0	63.1
SHIFT	88.5	54.0	76.0
SHIFT + retrain	93.1	52.0	89.0
Neuron skyline	80.6	65.6	46.5
Feature skyline	88.5	54.0	62.9
Oracle	93.0	49.4	91.9

Table 2: Accuracies on balanced data for the ground-truth label (profession) and spurious label (gender). “Worst group accuracy” refers to whichever **profession** accuracy is lowest among male professors, male nurses, female professors, female nurses.

4 Application: Removing unintended signals from a classifier without disambiguating labels

NN classifiers often rely on unintended signals—e.g., spurious features. Nearly all prior work on this problem relies on access to *disambiguating labeled data* in which unintended signals are less predictive of labels than intended ones. However, some tasks have structural properties which disallow this assumption. For example, inputs for different classes might come from different data sources (Zech et al., 2018). Additionally, some have raised concerns (Ngo et al., 2024; Casper et al., 2023) that sophisticated LMs trained with human feedback (Christiano et al., 2023) in settings with easy-to-hard domain shift (Burns et al., 2023; Hase et al., 2024) will be misaligned because, in these settings, “overseer approval” and “desirable behavior” are equally predictive of training reward labels. More fundamentally, the problem with unintended signals is that they are *unintended*—not they are insufficiently predictive—and we would like our methods to reflect this.

We thus propose Spurious Human-interpretable Feature Trimming (SHIFT), where a human changes the generalization of a classifier by editing its feature circuit. We show that SHIFT removes sensitivity to unintended signals without access to disambiguating labeled data, or even without knowing what the signals are ahead of time.

Method. Suppose we are given labeled training data $\mathcal{D} = \{(x_i, y_i)\}$; an LM-based classifier C trained on \mathcal{D} ; and SAEs for various components of C . To perform SHIFT, we:

1. Apply the methods from §3 to compute a feature circuit that explains C ’s accuracy on inputs $(x, y) \sim \mathcal{D}$ (e.g., using metric $m = -\log C(y|x)$).
2. Manually inspect and evaluate for task-relevancy each feature in the circuit from Step 1.
3. Ablate from M features judged to be task-irrelevant to obtain a classifier C' .
4. (Optional) Further fine-tune C' on data from \mathcal{D} .

Step 3 removes the classifier’s dependence on unintended signals we can identify, but may disrupt performance for the intended signal. Step 4 can be used to restore some performance.

Experimental setup. We illustrate SHIFT using the Bias in Bios dataset (BiB; De-Arteaga et al., 2019). BiB consists of professional biographies, and the task is to classify an individual’s profession based on their biography. BiB also provides labels for a spurious feature: gender. We subsample BiB to produce two sets of labeled data:

- The **ambiguous set**, consisting of bios of male professors (labeled 0) and female nurses (labeled 1).
- The **balanced set**, consisting of an equal number of bios for male professors, male nurses, female professors, and female nurses. These data carry **profession labels** (the intended signal) and **gender labels** (the unintended signal).

The **ambiguous set** represents a worst-case scenario: the unintended signal is perfectly predictive of training labels. Given only access to the **ambiguous set**, our task is to produce a **profession** classifier which is accurate on the **balanced set**.

We train a linear classifier based on Pythia-70M using the **ambiguous set**; see App. E.1 for details. We apply SHIFT by first discovering a circuit using the zero-ablation variant described in §3.1; this circuit, shown in App.F.2, contains 67 features. We manually interpret each feature using an interface similar to that shown in App. D; namely, we inspect contexts and tokens from The Pile (Gao et al., 2020) that maximally activate the feature, as well as tokens whose log-probabilities are most affected by ablating the feature. We judge 55 of these features to be task-irrelevant (e.g., features that promote female-associated language in biographies of women, as in Figure 18; see App. G for more examples). Although this interpretability step uses additional unlabeled data, we emphasize that we never use additional *labeled* data (or even additional unlabeled inputs from the classification dataset).

Baselines and skylines. To contextualize the performance of SHIFT, we also implement:

- **SHIFT with neurons.** Perform SHIFT, but using neurons instead of SAE features.
- **Concept Bottleneck Probing (CBP)**, adapted from Yan et al. (2023) (originally for multimodal text/image models). CBP works by training a probe to classify inputs x given access only to a vector of affinities between the LM’s representation of x and various concept vectors. See App. E.2 for implementation details.
- **Feature skyline.** Instead of relying on human judgement to evaluate whether a feature should be ablated, we ablate the 55 features from our circuit that are most causally implicated in **spurious feature** accuracy on the **balanced set**.
- **Neuron skyline.** The same as the feature skyline, but using 55 neurons instead.
- **Oracle.** A classifier trained on **ground-truth labels** on the **balanced set**.

Results. We find (Table 2) that SHIFT near-completely removes our classifier’s dependence on gender information, with Step 3 of SHIFT providing the bulk of the improvement. We further find our judgements of which features are task-relevant to be highly informative: SHIFT without retraining matches the feature skyline.

Moreover, SHIFT critically relies on the use of SAE features. When applying SHIFT with neurons, essentially none of the neurons are interpretable, making it difficult to tell if they ought to be ablated; see Appendix G for examples. Because of this, we abandon the SHIFT with neurons baseline. Even using the balanced set to select neurons for removal (the neuron skyline) fails to match the performance of SHIFT.

5 Unsupervised Circuit Discovery at Scale

So far, we have relied on human-collected data to specify LM behaviors for analysis. However, LMs implement numerous interesting behaviors, many of which may be counterintuitive to humans. In this section, we adapt our techniques to produce a near fully-automated interpretability pipeline, starting from raw text data and ending with thousands of feature circuits for auto-discovered model behaviors.

We do this in two steps:

1. **Behavior discovery.** We implement variants of the quanta discovery approach from Michaud et al. (2023), which work by clustering contexts based on vectors derived from Pythia-70M activations, gradients or both. Implementation details can be found in App. H.
2. **Circuit discovery.** Given a cluster, we apply the zero-ablation variant of our technique from §3 using dataset $\mathcal{D} = \{(x_i, y_i)\}$, the set of contexts in the cluster together with the next token appearing in The Pile, and metric $m = -\log P(y_i|x_i)$.

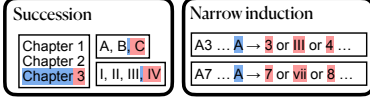
We present example clusters, as well as interesting features participating in their associated circuits (Figure 5). Full annotated circuits can be found in App. F.3, and an interface for exploring all of our clusters and (unlabeled) circuits can be found at `feature-circuits.xyz`.

Cluster 382: Incrementing sequences

var input = [1, 2, 3, 4, 5, 6, 7, 8]

Step 1. Download the latest CompsNY 3.49 Full
Step 2. Double click the Setup file and follow the prompts [...]
Step 3. After the main install closes, click OK [...]
Step 4

Example features involved:



Cluster 475: "to" as infinitive object

At issue, whether the defendant should be allowed to

British Prime Min David Cameron says in televised remarks he would like Britain to

Reader bloggers are asked to

Example features involved:

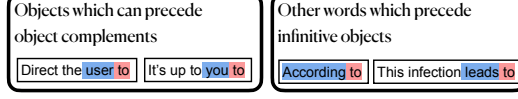


Figure 5: Example clusters and features which participate in their circuits. Features are active on tokens shaded blue and promote tokens shaded in red. (left) An example *narrow* induction feature recognizes the pattern $A3 \dots A$ and copies information from the 3 token. This composes with a succession feature to implement the prediction $A3 \dots A \rightarrow 4$. (right) One feature promotes “to” after words which can take infinitive objects. A separate feature activates on objects of verbs or prepositions and promotes “to” as an object complement.

While evaluating these clusters and circuits is an important open problem, we generally find that these clusters expose interesting LM behaviors, and that their respective feature circuits can provide useful insights on mechanisms of LM behavior. For instance, we automatically discover attention features implicated in succession and induction, two phenomena thoroughly studied in prior work at the attention head level using human-curated data (Olsson et al., 2022; Gould et al., 2023).

Feature circuits can also shed interesting light on their clusters. For example, while the clusters in Figure 5 seem at first to each represent a single mechanism, circuit-level analysis reveals in both cases a union of distinct mechanisms. For cluster 475, Pythia-70M determines whether “to [verb]” is an appropriate object in two distinct manners (see Figure 5 caption). And for cluster 382, the prediction of successors relies on general succession features, as well as multiple *narrow* induction features which recognize patterns like “ $A3 \dots A$ ”.

6 Related Work

Causal interpretability. Interpretability research has applied causal mediation analysis (Pearl, 2001; Robins & Greenland, 1992) to understand the mechanisms underlying particular model behaviors and their emergence (Yu et al., 2023; Geva et al., 2023; Hanna et al., 2023; Todd et al., 2024; Prakash et al., 2024; Chen et al., 2024, *inter alia*). This typically relies on counterfactual interventions (Lewis, 1973), such as activation patching or path patching on coarse-grained components (Conmy et al., 2023; Wang et al., 2023). Some techniques aim to, given a hypothesized causal graph, identify a matching causal mechanism in an LM (Geiger et al., 2021; 2022; 2023); in contrast, we aim here to discover causal mechanisms without starting from such hypotheses.

Robustness to spurious correlations. There is a large literature on mitigating robustness to spurious correlations, including techniques which rely on directly optimizing worst-group accuracy (Sagawa et al., 2020; Oren et al., 2019; Zhang et al., 2021; Sohoni et al., 2022; Nam et al., 2022), automatically or manually reweighting data between groups (Liu et al., 2021; Nam et al., 2020; Yaghoobzadeh et al., 2021; Utama et al., 2020; Creager et al., 2021; Idrissi et al., 2022; Orgad & Belinkov, 2023), training classifiers with more favorable inductive biases (Kirichenko et al., 2023; Zhang et al., 2022; Iskander et al., 2024), or editing out undesired concepts (Iskander et al., 2023; Belrose et al., 2023; Wang et al., 2020; Ravfogel et al., 2020; 2022a;b). All of these techniques rely on access to disambiguating labeled data in the sense of §4. Some techniques from a smaller literature focused on image or multimodal models apply without such data (Oikarinen et al., 2023; Yan et al., 2023). Our method here is inspired by the approach of Gandelsman et al. (2024) based on interpreting and ablating undesired attention heads in CLIP.

Feature disentanglement. In addition to the recent work SAE work of Cunningham et al. (2024); Bricken et al. (2023), other approaches to feature disentanglement include (Schmidhuber, 1992; Desjardins et al., 2012; Kim & Mnih, 2018; Chen et al., 2016; Makhzani & Frey, 2013; He et al., 2022; Peebles et al., 2020; Schneider & Vlachos, 2021; Burgess et al., 2018; Chen et al., 2018; Higgins et al., 2017, *inter alia*).

7 Conclusion

We have introduced a method for discovering circuits on sparse features. Using this method, we discover human-interpretable causal graphs for a subject–verb agreement task, a classifier, and thousands of general token prediction tasks where no clear right or wrong answer exists. We can edit the set of features that models have access to by ablating sparse features that humans deem spurious; we find that this is significantly more effective than a neuron-based ablation method which has an unfair advantage.

Acknowledgments

We thank Buck Schlegeris, Ryan Greenblatt, and Neel Nanda for discussion of ideas upstream to the experiments in §4. We thank Logan Riggs and Jannik Brinkmann for help training SAEs. We also thank Josh Engels and Max Tegmark for discussions about clustering and sparse projections related to §5. S.M. is supported by an Open Philanthropy alignment grant. C.R. is supported by Manifund Reagents. E.J.M is supported by the NSF Graduate Research Fellowship Program (Grant No. 2141064). Y.B. is supported by the Israel Science Foundation (Grant No. 448/20) and an Azrieli Foundation Early Career Faculty Fellowship. Y.B. and D.B. are supported by a joint Open Philanthropy alignment grant. A.M. is supported by a Zuckerman postdoctoral fellowship.

Limitations

The success of our technique relies on access to SAEs for a given model. Training such SAEs currently requires a large (but one-time) upfront compute cost, which poses an obstacle to running our methods. Additionally, model components not captured by the SAEs will remain uninterpretable after applying our method.

Much of our evaluation is qualitative. While we have quantitative evidence that feature circuits are useful for improving generalization without additional data (§4), evaluating dictionaries and circuits without downstream tasks is challenging.

References

- Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.
- Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=awIpKpwTwF>.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shrivanshu Purohit, USVSN Sai Prashanth, Edward Raff, 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.
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, 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>.
- Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β -vae, 2018.
- Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023.
- Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, J  r  my Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Rapha  l Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krashennnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biy  k, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023.
- Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=M05PiKHELW>.
- Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders, 2018. URL <https://openreview.net/forum?id=BJdMRoCIf>.
- Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, pp. 2180–2188, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.
- Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.
- Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adri   Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Elliot Creager, Joern-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2189–2200. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/creager21a.html>.
- Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=F76bwRSLeK>.
- Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 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*, FAT* ’19, pp. 120–128, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361255. doi: 10.1145/3287560.3287572. URL <https://doi.org/10.1145/3287560.3287572>.

- Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. *Computing Research Repository*, arXiv:1210.5474, 2012.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.
- Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1828–1843, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.144. URL <https://aclanthology.org/2021.acl-long.144>.
- Yossi Gandelsman, Alexei A. Efros, and Jacob Steinhardt. Interpreting CLIP’s image representation via text-based decomposition. *Computing Research Repository*, arXiv:2310.05916, 2024.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling, 2020.
- Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9574–9586. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f5c422f4d49a5a807eda27434231040-Paper.pdf.
- Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. 162:7324–7338, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/geiger22a.html>.
- Atticus Geiger, Chris Potts, and Thomas Icard. Causal abstraction for faithful model interpretation. *Computing Research Repository*, arXiv:2301.04709, 2023.
- Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL <https://aclanthology.org/2023.emnlp-main.751>.
- Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. *Computing Research Repository*, arXiv:2312.09230, 2023.
- Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=p4PckNQR8k>.
- Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegrefe. The unreasonable effectiveness of easy training data for hard tasks, 2024.
- T. He, Z. Li, Y. Gong, Y. Yao, X. Nie, and Y. Yin. Exploring linear feature disentanglement for neural networks. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, Los Alamitos, CA, USA, jul 2022. IEEE Computer Society. doi: 10.1109/ICME52920.2022.9859978. URL <https://doi.ieeecomputersociety.org/10.1109/ICME52920.2022.9859978>.

- Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=Sy2fzU9gl>.
- Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In Bernhard Schölkopf, Caroline Uhler, and Kun Zhang (eds.), *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pp. 336–351. PMLR, 11–13 Apr 2022. URL <https://proceedings.mlr.press/v177/idrissi22a.html>.
- Shadi Iskander, Kira Radinsky, and Yonatan Belinkov. Shielded representations: Protecting sensitive attributes through iterative gradient-based projection. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 5961–5977, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.369. URL <https://aclanthology.org/2023.findings-acl.369>.
- Shadi Iskander, Kira Radinsky, and Yonatan Belinkov. Leveraging prototypical representations for mitigating social bias without demographic information. *Computing Research Repository*, 2403.09516, 2024.
- Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.
- Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2649–2658. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/kim18b.html>.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL <https://api.semanticscholar.org/CorpusID:6628106>.
- Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *Computing Research Repository*, arXiv:2204.02937, 2023.
- János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. AtP*: An efficient and scalable method for localizing llm behaviour to components, 2024.
- David K. Lewis. *Counterfactuals*. Blackwell, Malden, Mass., 1973.
- Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6781–6792. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/liu21f.html>.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL <https://api.semanticscholar.org/CorpusID:53592270>.
- Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *Computing Research Repository*, abs/1312.5663, 2013. URL <https://api.semanticscholar.org/CorpusID:14850799>.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022. arXiv:2202.05262.

- Eric J Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=3tbTw2ga8K>.
- Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: Training debiased classifier from biased classifier. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Junhyun Nam, Jaehyung Kim, Jaeho Lee, and Jinwoo Shin. Spread spurious attribute: Improving worst-group accuracy with spurious attribute estimation, 2022.
- Neel Nanda. Attribution patching: Activation patching at industrial scale, 2022. URL <https://www.neelnanda.io/mechanistic-interpretability/attribution-patching>.
- Neel Nanda. Open source replication & commentary on Anthropic’s dictionary learning paper, 2023. URL <https://www.lesswrong.com/posts/fKuugaxt2XLtKAskk/open-source-replication-and-commentary-on-anthropic-s>.
- Neel Nanda, Senthooran Rajamanoharan, János Kramár, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, 2023. URL <https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall>.
- Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective. *Computing Research Repository*, arXiv:2209.00626, 2024.
- Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=FlCg47MNvBA>.
- 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. <https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html>.
- Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust language modeling. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4227–4237, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1432. URL <https://aclanthology.org/D19-1432>.
- Hadas Orgad and Yonatan Belinkov. BLIND: Bias removal with no demographics. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8801–8821, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.490. URL <https://aclanthology.org/2023.acl-long.490>.
- Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, UAI’01*, pp. 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

- Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2402.14811.
- Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7237–7256, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.647. URL <https://aclanthology.org/2020.acl-main.647>.
- Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18400–18421. PMLR, 17–23 Jul 2022a. URL <https://proceedings.mlr.press/v162/ravfogel22a.html>.
- Shauli Ravfogel, Francisco Vargas, Yoav Goldberg, and Ryan Cotterell. Adversarial concept erasure in kernel space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6034–6055, Abu Dhabi, United Arab Emirates, December 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.405. URL <https://aclanthology.org/2022.emnlp-main.405>.
- James M. Robins and Sander Greenland. Identifiability and exchangeability for direct and indirect effects. *Epidemiology*, 3(2):143–155, 1992. ISSN 10443983. URL <http://www.jstor.org/stable/3702894>.
- Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=ryxGuJrFvS>.
- Jürgen Schmidhuber. Learning Factorial Codes by Predictability Minimization. *Neural Computation*, 4(6):863–879, 11 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.6.863. URL <https://doi.org/10.1162/neco.1992.4.6.863>.
- Johannes Schneider and Michalis Vlachos. Explaining neural networks by decoding layer activations, 2021.
- Nimit S. Sohoni, Maziar Sanjabi, Nicolas Ballas, Aditya Grover, Shaoliang Nie, Hamed Firooz, and Christopher Ré. BARACK: Partially supervised group robustness with guarantees, 2022.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, pp. 3319–3328. JMLR.org, 2017.
- Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *NeurIPS Workshop on Attributing Model Behavior at Scale*, 2023. URL <https://openreview.net/forum?id=tiLbFR4bJW>.
- Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024.
- Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. Towards debiasing NLU models from unknown biases. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7597–7610, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.613. URL <https://aclanthology.org/2020.emnlp-main.613>.

- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.
- Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=NpsVSN6o4u1>.
- Tianlu Wang, Xi Victoria Lin, Nazneen Fatema Rajani, Bryan McCann, Vicente Ordonez, and Caiming Xiong. Double-hard debias: Tailoring word embeddings for gender bias mitigation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5443–5453, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.484. URL <https://aclanthology.org/2020.acl-main.484>.
- Yadollah Yaghoobzadeh, Soroush Mehri, Remi Tachet des Combes, T. J. Hazen, and Alessandro Sordani. Increasing robustness to spurious correlations using forgettable examples. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 3319–3332, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.291. URL <https://aclanthology.org/2021.eacl-main.291>.
- An Yan, Yu Wang, Yiwu Zhong, Zexue He, Petros Karypis, Zihan Wang, Chengyu Dong, Amilcare Gentili, Chun-Nan Hsu, Jingbo Shang, and Julian McAuley. Robust and interpretable medical image classifiers via concept bottleneck models, 2023.
- Qinan Yu, Jack Merullo, and Ellie Pavlick. Characterizing mechanisms for factual recall in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9924–9959, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.615. URL <https://aclanthology.org/2023.emnlp-main.615>.
- John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric Karl Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine*, 15(11): e1002683, November 2018. ISSN 1549-1676. doi: 10.1371/journal.pmed.1002683. URL <http://dx.doi.org/10.1371/journal.pmed.1002683>.
- Jingzhao Zhang, Aditya Krishna Menon, Andreas Veit, Srinadh Bhojanapalli, Sanjiv Kumar, and Suvrit Sra. Coping with label shift via distributionally robust optimisation. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=BtZhsSGNRNi>.
- Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Re. Correct-N-Contrast: A contrastive approach for improving robustness to spurious correlations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26484–26516. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/zhang22z.html>.
- Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. *Computing Research Repository*, arXiv:2310.01405, 2023.

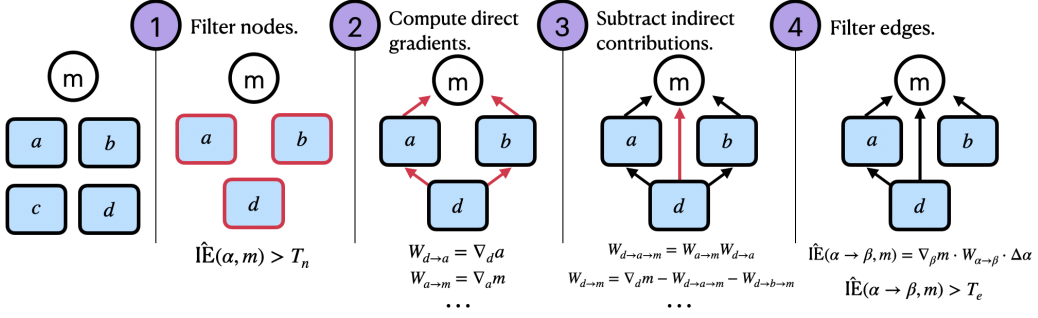


Figure 6: Computing edge weights. We compute all gradients between adjacent components by backpropagating from the activation of a downstream feature activation. Then, when computing contributions to non-adjacent nodes, we correct for indirect contributions by subtracting off a product of Jacobians. Finally, we compute estimated indirect effects via specific paths by multiplying gradients w.r.t. m by the source node’s activation difference. We only keep edges with an estimated effect above edge threshold T_E (a hyperparameter).

A Methodological Details

A.1 Computing Edge Weights

Let e be an edge between an upstream node \mathbf{u} and downstream node \mathbf{d} , corresponding to an upstream activation \mathbf{a}^u and downstream activation \mathbf{a}^d . When estimating the causal importance of e , we consider two cases:

- (a) e connects a layer ℓ residual stream component and a layer ℓ attention or MLP component, or e connects a layer ℓ attention or MLP component and a layer $\ell + 1$ residual stream component;⁴
- (b) e connects a layer ℓ residual stream component and a layer $\ell + 1$ residual stream component.

In case (a), the indirect effect of e corresponds to the result of intervening to set $\mathbf{d} = \mathbf{d}(x_{\text{clean}} | \text{do}(\mathbf{u} = \mathbf{u}_{\text{patch}}))$, but not intervening on any other downstream node on which \mathbf{a}^u has a *direct* effect. As with nodes, we use a linear approximation:

$$\begin{aligned} \hat{\text{IE}}(m; e; x_{\text{clean}}, x_{\text{patch}}) &= \nabla_{\mathbf{d}} m|_{\mathbf{d}_{\text{clean}}} \nabla_{\mathbf{u}} \mathbf{d}|_{\mathbf{u}_{\text{clean}}} (\mathbf{u}_{\text{patch}} - \mathbf{u}_{\text{clean}}) \\ &\approx m(x_{\text{clean}} | \text{do}(\mathbf{d} = \mathbf{d}(x_{\text{clean}} | \text{do}(\mathbf{u} = \mathbf{u}_{\text{patch}})))) \end{aligned} \quad (5)$$

If \mathbf{d} is an SAE error, then the naive approach to computing this expression involves performing d_{model} backwards passes; fortunately we can still compute the product in a single backwards pass as explained in §A.3.

In case (b), we would like to give e a score that represents the effect that \mathbf{a}^u has on m via \mathbf{a}^d , but which is not already explained by compositions of edges of type (a) via intermediate activations \mathbf{a}^m . As is shown in Figure 6, for each intermediate activation \mathbf{a}^m , we can correct for the nontransitive effect through \mathbf{a}^m by subtracting off a product of Jacobians. In general, we set

$$\hat{\text{IE}}(m; e; x_{\text{clean}}, x_{\text{patch}}) = \nabla_{\mathbf{d}} m|_{\mathbf{d}_{\text{clean}}} \left(\nabla_{\mathbf{u}} \mathbf{d}|_{\mathbf{u}_{\text{clean}}} - \sum_{\mathbf{m}} \nabla_{\mathbf{m}} \mathbf{d}|_{\mathbf{m}_{\text{clean}}} \nabla_{\mathbf{u}} \mathbf{m}|_{\mathbf{u}_{\text{clean}}} \right) (\mathbf{u}_{\text{patch}} - \mathbf{u}_{\text{clean}}) \quad (6)$$

where the sum is taken over intermediate nodes \mathbf{m} between \mathbf{u} and \mathbf{d} .

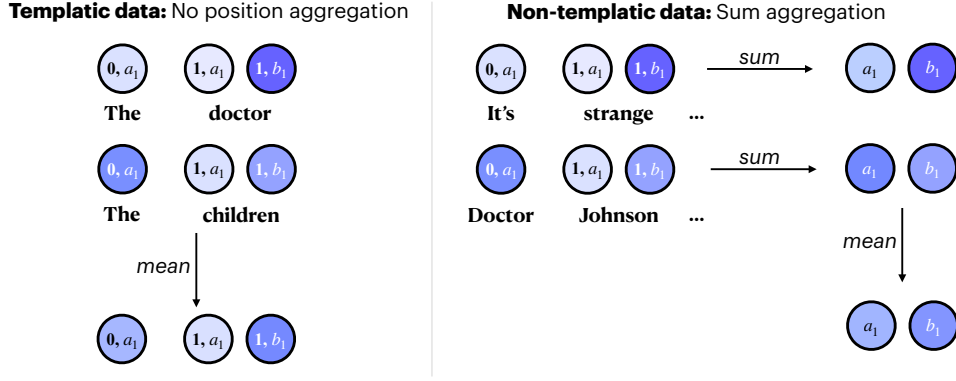


Figure 7: Aggregation of node/edge effects across examples (and sometimes, across token positions). Each feature is labeled as “token position, feature index.” If we have templatic data, we preserve token position information, and treat the same features in different token positions as different features. If we have more general non-templatic data, we first sum across positions, and then take the example-wise mean of the position-aggregated effects.

A.2 Aggregating across token positions and examples

Figure 7 summarizes how we aggregate effects across examples (and optionally across token positions). For templatic data where tokens in matching positions play consistent roles (see §3.2, 3.3), we take the mean effect of nodes/edges across examples. In this case, we treat the same feature (or neuron) in different token positions as different nodes altogether in the circuit, each with their own separate effects on target metric m .

For non-templatic data (§4, 5), we first sum the effects of corresponding nodes/edges across token positions before taking the example-wise mean. This means that each feature appears in the circuit once, representing its effects at all token positions in an input.

A.3 Practical considerations

Here we review a number of tricks that we use to compute the quantities defined above efficiently. The backbone of our approach is to, given an activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ of some submodule for which we have an SAE, use the SAE to compute the quantities $f_i(\mathbf{x})$ and $\epsilon(\mathbf{x})$ in (1), and then intervene in our model’s forward pass to set

$$\mathbf{x} \leftarrow \sum_i f_i(\mathbf{x}) \mathbf{v}_i + \mathbf{b} + \epsilon(\mathbf{x}). \quad (7)$$

Even though \mathbf{x} was already numerically equal to the right-hand side of (7), after the intervention the computation graph will incorporate the variables $f_i(\mathbf{x})$ and $\epsilon(\mathbf{x})$. Thus, when we use Pytorch’s autograd algorithm to perform backpropagation of downstream quantities, we will automatically compute gradients for these variables.

An alternative approach for computing gradients (which we do not use) is to simply run the model without interventions, use backpropagation to compute all gradients $\nabla_{\mathbf{x}} m$, and use the formulas

$$\nabla_{f_i} m = \nabla_{\mathbf{x}} m \cdot \mathbf{v}_i, \quad \nabla_{\epsilon} m = \nabla_{\mathbf{x}} m$$

which follow from the chain rule when m is any function of \mathbf{x} .

Stop gradients on SAE errors to compute SAE feature gradients. The natural way to compute the SAE error $\epsilon(\mathbf{x})$ is by first using the SAE to compute $\hat{\mathbf{x}}$ and then setting $\epsilon(\mathbf{x}) = \mathbf{x} - \hat{\mathbf{x}}$. However, if we take this approach, then after applying the intervention (7) we would have

$$\nabla_{f_i} m = \nabla_{\mathbf{x}} m \nabla_{f_i} \mathbf{x}^d = \nabla_{\mathbf{x}} m \nabla_{f_i} (\hat{\mathbf{x}} + \mathbf{x}^u - \hat{\mathbf{x}}) = 0$$

⁴As Pythia models employ parallel attention, the layer ℓ attention components have no effect on the layer ℓ MLP components.

where \mathbf{x}^d the copy of \mathbf{x} which is downstream of f_i in the computation graph, and \mathbf{x}^u is the copy which is upstream of f_i . To fix this, we apply a stop gradient to $\epsilon(\mathbf{x})$ so that $\mathbf{x}^d = \hat{\mathbf{x}} + \text{stopgrad}(\mathbf{x}^u - \hat{\mathbf{x}})$.

Pass-through gradients. Although the stop gradient from above solves the problem of vanishing gradients for the f_i , it interferes with the backpropagation of gradients to further upstream nodes. In order to restore exact gradient computation, we implement a pass-through gradient on the computation of our dictionary. That is, in the notation above, we intervene in the *backwards* pass of our model to set

$$\nabla_{\mathbf{x}^u} m \leftarrow \nabla_{\mathbf{x}^d} m.$$

Jacobian-vector products. Done naively, computing the quantities in (5) and (6) when \mathbf{d} or $\mathbf{m} \in \mathbb{R}^{d_{\text{model}}}$ are SAE errors would take $O(d_{\text{model}})$ backwards passes. Fortunately, one can use the following trick: when A is a constant $1 \times n$ matrix, $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{y} = \mathbf{y}(\mathbf{x}) \in \mathbb{R}^n$ is a function of \mathbf{x} , we have

$$A \nabla_{\mathbf{x}} \mathbf{y} = \nabla_{\mathbf{x}} (A \mathbf{y})$$

where the right-hand side is a $1 \times m$ Jacobian which can be computed with a single backward pass. Thus we can compute (5) with only two backwards passes by first computing $\nabla_{\mathbf{d}} m|_{\mathbf{d}_{\text{clean}}}$ and then computing $\nabla_{\mathbf{u}} \left(\nabla_{\mathbf{d}} m|_{\mathbf{d}_{\text{clean}}} \right)$ with another backwards pass, where the second $\nabla_{\mathbf{d}} m|_{\mathbf{d}_{\text{clean}}}$ is treated as a constant (e.g., by detaching it in Pytorch). A similar trick can be applied for (6).

B Details on Sparse Autoencoders

B.1 Architecture

Following Bricken et al. (2023), our SAEs are one-layer MLPs with a tied pre-encoder bias. In more detail, our SAEs have parameters

$$W_E \in \mathbb{R}^{d_{\text{SAE}} \times d_{\text{model}}}, W_D \in \mathbb{R}^{d_{\text{model}} \times d_{\text{SAE}}}, \quad b_E \in \mathbb{R}^{d_{\text{SAE}}}, b_D \in \mathbb{R}^{d_{\text{model}}}$$

where the columns of W_D are constrained to be unit vectors. Given an input activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$, we compute the sparse features activations via

$$\mathbf{f} = [f_1(\mathbf{x}) \quad \dots \quad f_{d_{\text{SAE}}}(\mathbf{x})] = W_E(\mathbf{x} - \mathbf{b}_D) + \mathbf{b}_E$$

and reconstructions via

$$\hat{\mathbf{x}} = W_D \mathbf{f} + \mathbf{b}_D.$$

The feature vectors $\mathbf{v}_i \in \mathbb{R}^{d_{\text{model}}}$ are the columns of W_D .

B.2 Training

Fix a specific choice of activation in Pythia-70M, e.g. MLP output, attention output, or residual stream in a particular layer. Following Cunningham et al. (2024); Bricken et al. (2023) we train an SAE for this activation by sampling random text from The Pile (Gao et al., 2020) (specifically the first 128 tokens of random documents), extracting the values \mathbf{x} for this activation over every token, and then training our SAE to minimize a loss function

$$\mathcal{L} = \mathcal{L}_{\text{reconstruction}} + \lambda \mathcal{L}_{\text{sparsity}} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \lambda \|\mathbf{f}\|_1$$

consisting of a L2 reconstruction loss and a L1 regularization term to promote sparsity. This loss is optimized using a variant of Adam (Kingma & Ba, 2014) adapted to ensure that the columns of W_D are unit vectors (see Bricken et al. (2023) or our code for details). We use $\lambda = 0.1$ and a learning rate of 10^{-4} .

Following Nanda (2023), we cache activations from 10000 tokens in a buffer and randomly sample batches of size 2^{14} for training our SAE. When the buffer is half-depleted, we

replenish it with fresh tokens from The Pile. We train for 120000 steps, resulting in a total of about 2 billion training tokens.

A major obstacle in training SAEs is *dead features*, that is, neurons in the middle layer of the SAE which never or rarely activate. We mitigate this by, every 25000 training steps, reinitializing features which have not activated in the previous 12500 steps using the same reinitialization procedure described in Bricken et al. (2023).

Finally, we use a linear learning rate warmup of 1000 steps at the start of training and after every time that neurons are resampled.

B.3 Evaluation

Here we report on various easy-to-quantify metrics of SAE quality. Note that these metrics leave out important qualitative properties of these SAEs, such as the interpretability of their features (App. D). Our metrics are:

- **Variance explained**, as measured by $1 - \frac{\text{Var}(\mathbf{x} - \hat{\mathbf{x}})}{\text{Var}(\mathbf{x})}$.
- Average **L1**, and **L0** norms of \mathbf{f} .
- **Percentage of features alive** as measured by features which activate at least once on a batch of 512 tokens.
- **Cross entropy (CE) difference** and **percentage of CE recovered**. The CE difference is the difference between the model’s original CE loss and the model’s CE loss when intervening to set \mathbf{x} to the reconstruction $\hat{\mathbf{x}}$. We obtain percentage of CE recovered by dividing this difference by the difference between the original CE loss and the CE loss when zero-ablating \mathbf{x} . These CE losses are computed averaged over a batch of 128 contexts of length 128.

These metrics are shown in Tables 3–6. Note that we index residual stream activations to be the layer which *outputs* the activation (so the layer 0 residual stream is *not* the embeddings, and the layer 5 residual stream is the output of the final layer, immediately preceding the final decoder).

% Variance Explained	L1	L0	% Alive	CE Diff	% CE Recovered
96	1	3	36	0.17	98

Table 3: Embedding SAE evaluation.

Layer	% Variance Explained	L1	L0	% Alive	CE Diff	% CE Recovered
Attn 0	92%	8	128	17%	0.02	99%
Attn 1	87%	9	127	17%	0.03	94%
Attn 2	90%	19	215	12%	0.05	93%
Attn 3	89%	12	169	13%	0.03	93%
Attn 4	83%	8	132	14%	0.01	95%
Attn 5	89%	11	144	20%	0.02	93%

Table 4: Attention SAE evaluation by layer.

C Quality of Linear Approximations of Indirect Effects

Figure 8 shows the quality of our linear approximations for indirect effects. Prior work (Nanda, 2022; Kramár et al., 2024) investigated attribution patching accuracy for IEs of coarse-grained model components (queries, keys, and values for attention heads, residual stream vectors, and MLP outputs) and MLP neurons. Working with SAE features and errors,

Layer	% Variance Explained	L1	L0	% Alive	CE Diff	% CE Recovered
MLP 0	97%	5	5	40%	0.10	99%
MLP 1	85%	8	69	44%	0.06	95%
MLP 2	99%	12	88	31%	0.11	88%
MLP 3	88%	20	160	25%	0.12	94%
MLP 4	92%	20	100	29%	0.14	90%
MLP 5	96%	31	102	35%	0.15	97%

Table 5: MLP SAE evaluation by layer.

Layer	% Variance Explained	L1	L0	% Alive	CE Diff	% CE Recovered
Resid 0	92%	11	59	41%	0.24	97%
Resid 1	85%	13	54	38%	0.45	95%
Resid 2	96%	24	108	27%	0.55	94%
Resid 3	96%	23	68	22%	0.58	95%
Resid 4	88%	23	61	27%	0.48	95%
Resid 5	90%	35	72	45%	0.55	92%

Table 6: Residual (Resid) SAE evaluation by layer.

our results echo previous findings: attribution patching is generally quite good, but sometimes underestimates the true IEs. Notable exceptions are the layer 0 MLP and the residual stream in early layers. We also find that our integrated gradients-based approximation significantly improves approximation quality.

D Human Interpretability Ratings for Sparse Features

We asked human crowdworkers to rate the interpretability of random features, random neurons, features from our feature circuits, and neurons from our neuron circuits on a 0–100 scale (Table 7). Crowdworkers rate sparse features as significantly more interpretable than neurons, with features that participate in our circuits also being more interpretable than randomly sampled features.

See Figures 9 and 10 for examples of the human annotator interface. Humans were presented with the tokens on which the feature activated most strongly, followed by the tokens whose probabilities were most affected in Pythia-70M when the feature was ablated. This is followed by a series of example contexts in which the feature activated on some subset of tokens, where feature activations are shown in varying shades of blue (darker shades indicate higher activations). On the same page below the contexts, we ask annotators to

Activation type	Interpretability
Dense (random)	32.6
Dense (agreement)	30.2
Dense (BiB)	36.0
Sparse (random)	52.8
Sparse (agreement)	62.3
Sparse (BiB)	81.5

Table 7: Human interpretability ratings for dense (neuron) vs. sparse (autoencoder) features. We present mean interpretability scores across features on a 0–100 scale. We show scores for features that were either uniformly sampled (random), the top 30 by $\hat{I}E$ from the subject-verb agreement across RC task (agreement; §3.3), or the top 30 by $\hat{I}E$ for the Bias in Bios task (BiB; §4).

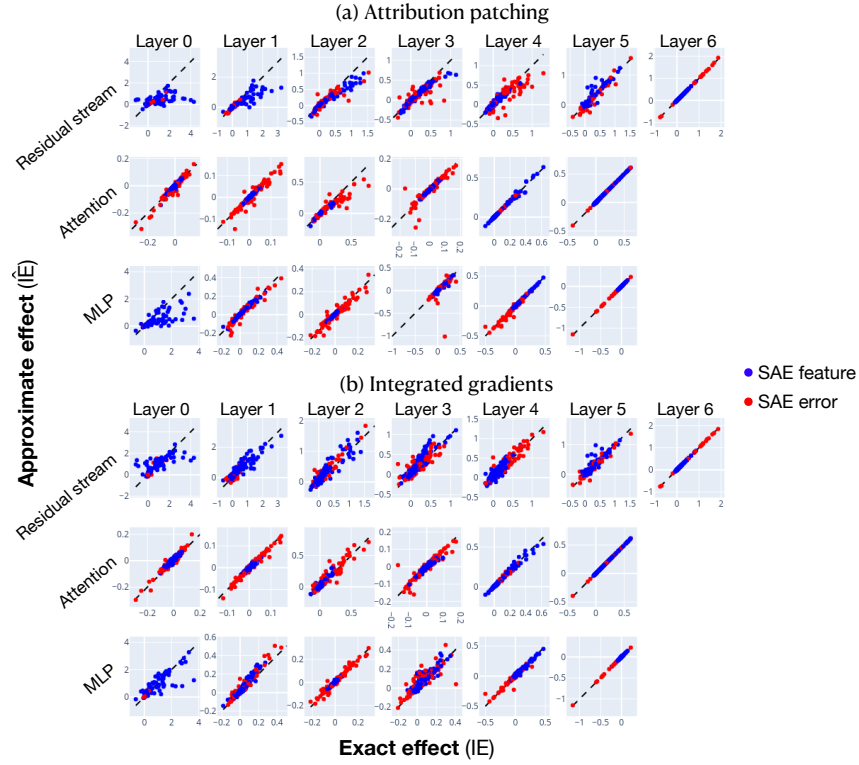


Figure 8: Approximate IEs (y -axis) and exact IEs (x -axis) using attribution patching (a; top) or integrated gradients (b; bottom). Each point corresponds to an SAE feature or SAE error at one token position of one input. Data were collected from 30 inputs from our across RC dataset.

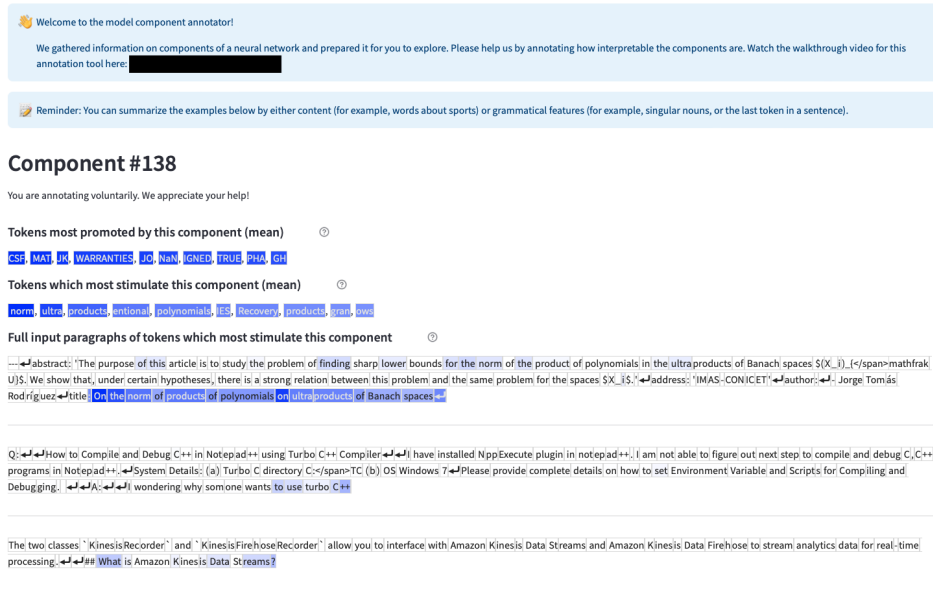


Figure 9: The human annotation interface used to obtain the interpretability ratings in Table 7. Here, we show the instructions, top-activating tokens, the token probabilities that were most affected when ablating the feature, and example contexts with feature activation values.

```

// Copyright (c) 2016 Dropbox, Inc. All rights reserved.
// Auto-generated by Stone, do not modify.
import <Foundation/Foundation.h>
import
DBSerializableProtocol.h
class DBTEAMPOLICIES

```

Milestones to recovery: preliminary validation of a framework to promote recovery and map progress through the medium secure inpatient pathway. Forensic mental health care in the UK has undergone a rapid expansion since the late 1990s. In medium secure units (MSUs), there is growing emphasis on developing care pathways without much theoretical underpinning. We developed a concept of 'Milestones to Recovery'

Component annotation

Concise component annotation: Summarize what the component is about in 1-5 words.

Interpretability score: How coherent are the examples shown above? Does the component consistently activate on (or promote) the same concept or rather various distinct concepts?

Please select 100 % (single common theme across contexts)

Semantic complexity score: How broad is the concept the token fires on? Does the component activate on (or promote) diverse tokens of a general theme or simply the same token all over again?

Please select streamlitApp 100 % (diverse tokens in broad context)

(Optional) Further notes on the component

(Required) Your user name

Please enter your user name to submit.

You annotated 0 components in this session. We need -3 more paid annotations to reach our goal. Thanks for contributing 🙌

Figure 10: The human annotation interface used to obtain the interpretability ratings in Table 7. Here, we show the rating interface on the same page as the content in Fig. 9, below the example contexts. Humans were asked to write a textual description of each feature, assign a 0–100 interpretability rating, and assign a 0–100 semantic complexity rating to each feature.

write a textual description of the feature, and rate both its interpretability and its semantic complexity on 0–100 scales.

Crowdworkers were recruited from the ARENA Slack channel, whose members are machine learning researchers interested in AI alignment and safety. The selection of annotators certainly influenced our results; a truly random sample of human annotators would likely display higher variance when annotating features.

One common error pattern we notice is that annotators often label features according to semantic groupings (e.g., “text about politics,” and do not pay attention to syntactic context (e.g., “plural nouns”). Future work could address this design bias by testing variants of the instructions.

E Implementation Details for Classifier Experiments

E.1 Classifier training

Here we describe how we train a classifier on Pythia-70M for the Bias in Bios (BiB) task of §4. To mimic a realistic application setting, we search over hyperparameters to train high-performing *baseline* and *oracle* classifiers (using the **ambiguous** and **balanced** datasets, respectively). Hyperparameters were **not** selected for strong SHIFT performance.

The inputs to our classifier are residual stream activations from the end of the *penultimate* layer of Pythia-70M.⁵ We mean-pool over (non-padding) tokens from the context. In preliminary experiments, we obtained slightly worse baseline and oracle performance when instead extracting representations over only the final token. We also obtained slightly worse performance when training the classifier on activations from the final layer of Pythia-70M.

We then fit a linear probe to these representations with logistic regression using the AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of 0.01 for a single epoch. When retraining after performing SHIFT, we tune *only* this linear probe—not the full model.

⁵The implication of this is that we freeze the LM and only update the parameters of the classifier.

For unclear reasons, we were unable to fit a probe with greater-than-chance accuracy when applying logistic regression to representations extracted from the final layer; this is why we used penultimate layer representations above.

E.2 Implementation for Concept Bottleneck Probing

Our implementation for Concept Bottleneck Probing (CBP) is adapted from (Yan et al., 2023). It works as follows:

1. First, we collect a number of keywords related to the intended prediction task. We use $N = 20$ keywords: nurse, healthcare, hospital, patient, medical, clinic, triage, medication, emergency, surgery, professor, academia, research, university, tenure, faculty, dissertation, sabbatical, publication, and grant.
2. We obtain *concept vectors* $\mathbf{c}_1, \dots, \mathbf{c}_N$ for each keyword by extracting Pythia-70M’s penultimate layer representation over the final token of each keyword, and then subtracting off the mean concept vector. (Without this normalization, we found that concept vectors have very high pairwise cosine similarities.)
3. Given an input with representation \mathbf{x} (obtained via the mean-pooling procedure in App. E.1), we obtain a concept bottleneck representation $\mathbf{z} \in \mathbb{R}^N$ by taking the cosine similarity with each \mathbf{c}_i .
4. Finally, we train a linear probe with logistic regression on the concept bottleneck representations \mathbf{z} , as in App. E.1.

We decided to normalize concept vectors but not input representations because it resulted in stronger performance.

F Feature Circuits

F.1 Subject–Verb Agreement

Here, we present the full agreement circuits for various syntactic agreement structures, where we annotate all features and do *not* collapse similar features into the same nodes. All circuits here are discovered with $T_N = 0.1$ and $T_E = 0.01$ unless otherwise noted. In each circuit, sparse features are shown in rectangles, whereas causally relevant error terms not yet captured by our SAEs are shown in triangles. Nodes shaded in darker colors have stronger effects on the target metric m . Blue nodes and edges are those which have positive indirect effects (i.e., are useful for performing the task correctly), whereas red nodes and edges are those which have counterproductive effects on m (i.e., cause the model to consistently predict incorrect answers).

First, we present agreement across a relative clause (Figure 11). The model appears to detect the subject’s grammatical number at the subject position. One position later, features detect the presence of relative pronouns (the start of the distractor clause). Finally, at the last token of the relative clause, the attention moves the subject information to the last position, where it assists in predicting the correct verb inflection.

The circuit for agreement across a prepositional phrase (Figure 12) looks remarkably similar to agreement across a relative clause; these two circuits share over 85% of their features, and many of the same features are used for detecting both prepositions and relative clauses.

For simple agreement (Figure 13, discovered with $T_N = 0.2$ and $T_E = 0.02$), many of the same features that were implicated in noun number detection and verb number prediction in the previous circuits also appear here. The model detects the subject’s number at the subject position in early layers. In later layers, these noun number detectors become inputs to verb number promoters, which activate on anything predictive of particular verb inflections.

Finally, the circuit for agreement within a relative clause (Figure 13, discovered with $T_N = 0.2$ and $T_E = 0.02$) appears to have the same structure as that for simple agreement: subject number detectors in early layers, followed by verb number promoters in later layers.

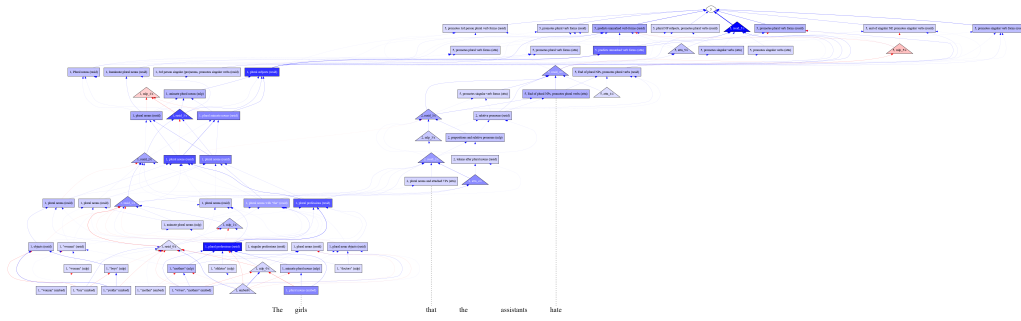


Figure 11: The full annotated feature circuit for agreement across a relative clause. The model detects the subject's number at the subject position. Other features detect relative pronouns (the start of the distractor clause). Finally, at the last token of the RC, the attention moves the subject information to the last position, where it assists in predicting the correct verb inflection.

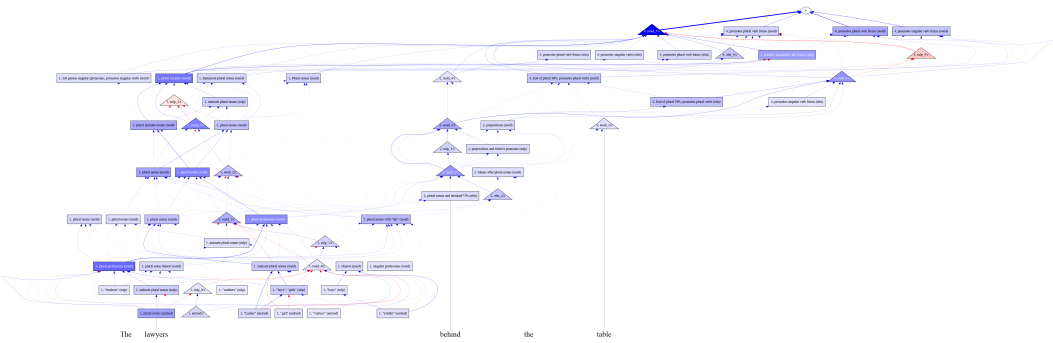


Figure 12: The full annotated feature circuit for agreement across a prepositional phrase. The model detects the subject's number at the subject position. Other features detect prepositional phrases (the start of the distractor clause). Finally, at the last token of the RC, the attention moves the subject information to the last position, where it assists in predicting the correct verb inflection.

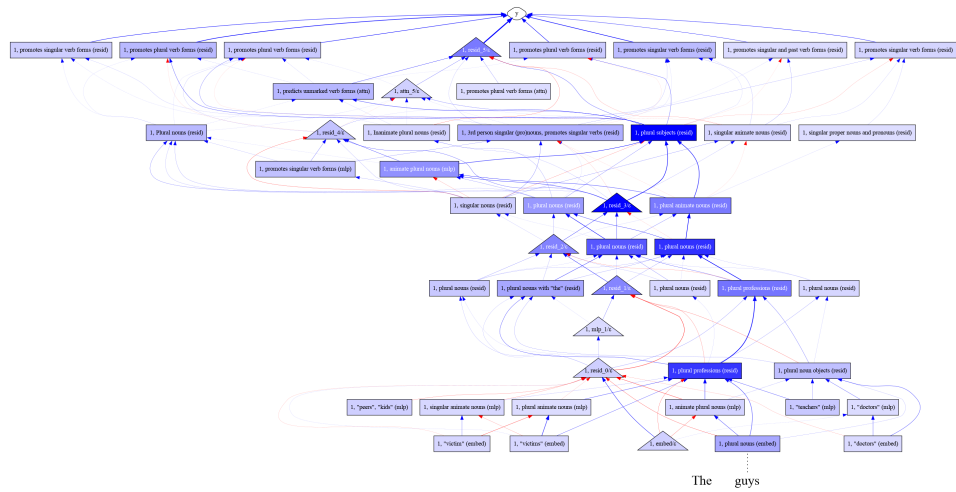


Figure 13: The full annotated feature circuit for simple agreement. The model detects the subject's number at the subject position in early layers. In later layers, these are inputs to features which activate on anything predictive of particular verb inflections.

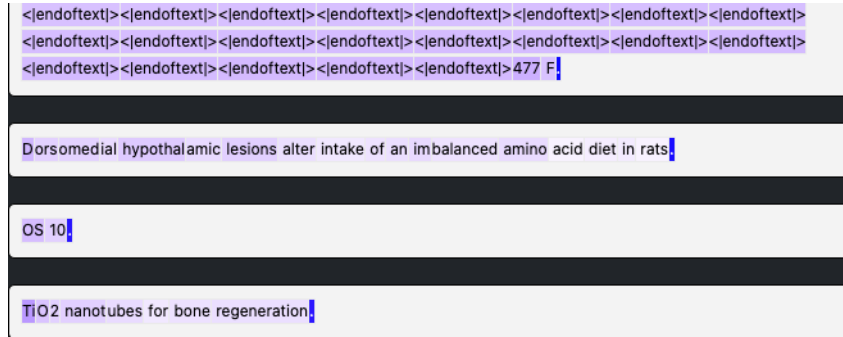


Figure 20: An example neuron from the Bias in Bios task. This appears to activate on beginnings and ends of sentences, but also more strongly on any token in a sentence that contains capital letters or numbers. We cannot deduce whether this would contribute more to gender or profession names.

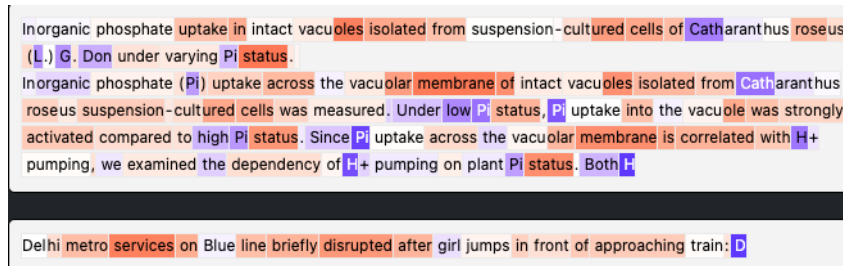


Figure 21: An example neuron from the Bias in Bios task. This activates positively on tokens starting with capital letters, but negatively on many other tokens (whose unifying theme we cannot deduce).

neuron skyline, where we allow the skyline an unfair advantage by simply ablating neurons which have positive effects on gender-based probabilities given the balanced set.

H Discovering LM Behaviors with Clustering

In this section, we describe our unsupervised method for discovering language model behaviors. More specifically, following Michaud et al. (2023), we cluster contexts from The Pile according to the Pythia-70M’s internal state during inference. In this section, we describe our clustering pipeline and methods.

H.1 Filtering Tokens

We must first locate (context, answer) pairs for which an LM correctly predicts the answer token from the context. We select The Pile (Gao et al. (2020)) as a general text corpus and filter to pairs on which Pythia-70M confidently and correctly predicts the answer token, with cross-entropy lower than 0.1 or 0.3 nats, depending on the experiment. The model consistently achieves low loss on tokens which involve “induction” (Olsson et al., 2022)—i.e., tokens which are part of a subsequence which occurred earlier in the context. We exclude induction samples by filtering out samples in which the bigram (final context token, answer token) occurred earlier in the context.

H.2 Caching Model-internal Information

We find behaviors by clustering samples according to information about the LM’s internals when run on that sample. We find clusters of samples where the model employs similar

mechanisms for next-token prediction. We experiment with various inputs to the clustering algorithm:

- **Dense Activations:** We take activations (residual stream vectors, attention block outputs, or MLP post-activations) from a given context and concatenate them. To obtain a vector whose length is independent of the context length, we can either use the activations at the last N context positions before the answer token, or aggregate (sum) across the sequence dimension. We experiment with both variants.
- **Sparse Activations:** Rather than dense model activations, we can use the activations of SAE features. We concatenate and aggregate these in the same manner as for dense activations.
- **Dense Component Indirect Effects:** We approximate the indirect effect of all features on the correct prediction using 2 without a contrastive pair—namely, by setting $\mathbf{a}_{\text{patch}} = 0$. The negative log-probability of the answer token $m = -\log p(\text{answer})$ serves as our metric for the correct prediction of the next token. The computation of linear effects requires saving both (1) activations and (2) gradients w.r.t m at the final N positions for each context in the dataset. We optionally aggregate by summing over all positions.
- **Sparse Indirect Effects:** Similarly, we can compute the linear effects of *sparse* activations on the correct prediction.
- **Gradient w.r.t. model parameters:** As in Michaud et al. (2023), we also experiment with using gradients of the loss w.r.t. model parameters, but with some modifications. We describe this method in more detail in §H.3 below.

H.3 Hyperparameters and Implementation Details

We apply either spectral clustering or k -means clustering. For spectral clustering, given either activations or effects x_i for sample i , we compute a matrix of pairwise cosine similarities $C_{ij} = x_i \cdot x_j / (||x_i|| ||x_j||)$ between all pairs of samples. Before performing spectral clustering, we normalize all elements of C to be in $[0, 1]$ by converting the cosine similarities to angular similarities: $\hat{C}_{ij} = 1 - \arccos(C_{ij}) / \pi$.

We use the `scikit-learn` (Pedregosa et al., 2011) spectral clustering implementation with k -means. For all inputs except gradients w.r.t. model parameters, we used spectral clustering across 8192 samples. We chose k (the number of total clusters) to maximize the number of clusters implicated in more than one input context.

We also experimented with using gradients w.r.t. model parameters as inputs, as in Michaud et al. (2023). Here, we scale up our approach to 100,000 samples. It is intractable to perform spectral clustering given 100,000 samples, so we instead use k -means clustering. Rather than clustering the gradients themselves (which are high-dimensional), we cluster sparse random projections of the gradients down to 30,000 dimensions. When projecting, we use a matrix with entries $\{-1, 0, 1\}$. When sampling the entries of this matrix, sample a nonzero value with probability $32/30000$, and if nonzero, sample -1 or 1 with equal probability. For a sparse projection matrix with dimensions $\mathbb{R}^{n \times 30000}$, there will on average be $32 \cdot n$ nonzero entries, where n is the number of parameters in the model.⁶

⁶We only consider gradients w.r.t. non-embedding and non-layernorm parameters.