Towards Principled Evaluations of Sparse Autoencoders for Interpretability and Control

Aleksandar Makelov
aleksandar.makelov@gmail.com

Georg Lange
mail@georglange.com

Neel Nanda
neelnanda27@gmail.com

Abstract

Disentangling model activations into meaningful features is a central problem in interpretability. However, the lack of ground-truth for these features in realistic scenarios makes the validation of recent approaches, such as sparse dictionary learning, elusive. To overcome this, we propose a framework to evaluate feature dictionaries in the context of specific tasks, by comparing them against *supervised* feature dictionaries. First, we demonstrate that supervised dictionaries achieve excellent approximation, control and interpretability of model computations on the task. Second, we use the supervised dictionaries to develop and contextualize evaluations of unsupervised dictionaries along the same three axes.

We apply this framework to the indirect object identification task (IOI) using GPT-2 Small, with sparse autoencoders (SAEs) trained on either the IOI or OpenWebText datasets. We find that these SAEs capture interpretable features for the IOI task, but they are not as successful as supervised features in controlling the model. Finally, we observe two qualitative phenomena in SAE training: feature occlusion (where a causally relevant concept is robustly overshadowed by even slightly higher-magnitude ones in the learned features), and feature over-splitting (where binary features split into many smaller features without clear interpretation). We hope that our framework will be a useful step towards more objective and grounded evaluations of sparse dictionary learning methods.

1 Introduction

While large language models (LLMs) have demonstrated impressive (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; OpenAI, 2023) results, the inner mechanisms behind their successes and failures largely remain a mystery (Olah, 2023). A central problem in this area is how to *disentangle* internal model representations into meaningful concepts, or *features*. If successful at scale, this research would deliver significant scientific and practical value, enabling enhanced model robustness, controllability, interpretability, and debugging (Gandelsman et al., 2023; Nanda et al., 2023; Marks et al., 2024).

A leading hypothesis for how LLMs represent and use features is the *linear representation hypothesis* (Mikolov et al., 2013b; Grand et al., 2018; Li et al., 2021; Abdou et al., 2021; Nanda et al., 2023), a strong version of which posits that individual activations of a model can be decomposed as sparse linear combinations of features that come from a large, shared *feature dictionary*. Recently, a series of works have proposed applying the (unsupervised) *sparse autoencoder* (SAE) framework to find such dictionaries (Olshausen & Field, 1997; Faruqui et al., 2015; Goh, 2016; Arora et al., 2018; Yun et al., 2021; Cunningham et al., 2023; Bricken et al., 2023).

While there are promising initial results (Bricken et al., 2023), this research area faces a key obstacle: we cannot directly evaluate the usefulness of features learned by an SAE, as we do not know the hypothetical 'true' features to begin with; indeed, finding them is why we use SAEs in the first place. This has two consequences: (1) the training objective of an SAE – balancing ℓ_2 reconstruction with an ℓ_1 penalty on feature coefficients – is only a substitute

for the true goal of finding meaningful features, and (2) the metrics used to evaluate SAEs are indirect and rely on proxies for the features, toy models, or non-trivial assumptions on SAE learning (Elhage et al., 2022b; Bricken et al., 2023; Sharkey et al., 2023). This holds back progress in the field, as there is no simple, objective and direct way to compare trained SAEs. In this paper, we take steps towards addressing this challenge. Our contributions are outlined as follows:

- We propose a principled method to compute sparse feature dictionaries to decompose language model activations on realistic tasks, using supervision from labels of concepts describing the prompts.
- We apply this method to the IOI task, and show these dictionaries exhibit three desirable properties in the task's context: (1) sufficiency/necessity of activation reconstructions, (2) sparse controllability of model behavior via feature editing, and (3) interpretability of the features consistent with their causal role[1].
- We use these feature dictionaries to design and contextualize evaluations of *unsupervised* feature dictionaries along the same three axes. Importantly, we aim to develop evaluations that are agnostic to whether or not unsupervised dictionaries use the same concepts as those in the supervised ones.
- We apply this methodology to feature dictionaries learned by SAEs on either the IOI dataset (task-specific SAEs) or the LLM's pre-training dataset (full-distribution SAEs). We find that both kinds of SAEs find interpretable features for the task, but task-specific SAEs are able to edit concepts by changing fewer features than full-distribution SAEs. Still, both kinds of SAEs fall short of supervised dictionaries.
- Finally, we dive deeper into some qualitative phenomena observed in task-specific SAEs: feature occlusion (whereby SAEs have a tendency to learn only the higher-magnitude one of two task-relevant concepts, even when the concepts have slightly different magnitudes), and feature over-splitting (whereby SAEs have a tendency to learn $\gg 2$ features without a clear interpretation for binary concepts). We reproduce both phenomena in simple toy models, suggesting they may generalize beyond our specific setting.

Our results underscore the need for more principled training and evaluation methods in this active area, and suggest that supervised feature dictionaries can be a valuable tool for automating aspects of this process.

2 Preliminaries

The linear representation hypothesis and sparse autoencoders. A central hypothesis in interpretability is the *linear representation hypothesis*. A strong variant of this hypothesis posits that model activations can be decomposed into meaningful features using a *sparse feature dictionary*: given a location in the model (e.g., an attention head output), there exists a set of vectors $\{\mathbf{u}_i\}_{i=1}^m$ such that each activation $\mathbf{a}$ at this location can be approximated as a sparse linear combination of the $\mathbf{u}_j$ with non-negative coefficients. In particular, recent work suggests that n-dimensional activations $\mathbf{a} \in \mathbb{R}^n$ may be best described by $m \gg n$ such features in *superposition* (Elhage et al., 2022a; Gurnee et al., 2023). Recently, SAEs have been proposed as a way to disentangle these features. Following the setup of Bricken et al. (2023) here and in the rest of this work, a sparse autoencoder (SAE) is an unsupervised model which learns to reconstruct activations $\mathbf{a} \in \mathbb{R}^n$ as a weighted sum of m features with non-negative weights. Specifically, the autoencoder computes a hidden representation

$$\mathbf{f} = \text{ReLU}\left(W_{enc}\left(\mathbf{a} - \mathbf{b}_{dec}\right) + \mathbf{b}_{enc}\right)$$

[1] As a by-product, our supervised feature dictionaries also demonstrate that activations for the task can be usefully disentangled in a way that adheres to the linear representation hypothesis and exhibits superposition. This is the first time such a disentanglement has been achieved in a realistic LLM task to the best of our knowledge.

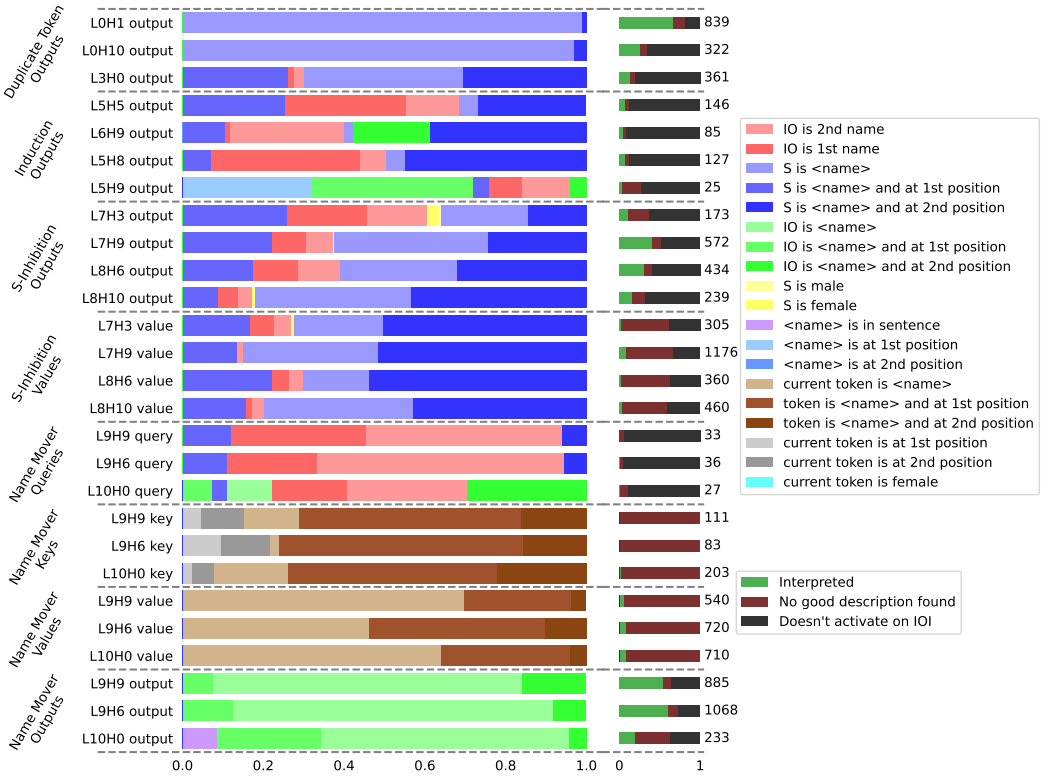

Figure 1: Interpreting the IOI features learned by SAEs trained on OpenWebText. For each node in the IOI circuit, we show the distribution of interpretations for the features which have any interpretation with F_1 score above a threshold. The numbers in the right column indicate the number of features with an assigned interpretation by our method, and the color bars show the overall distribution of the SAE features (conditioned on the feature not being dead on the SAE training distribution). See Section 5 for methodology; details on the interpretations considered are given in Appendix A.12.

and a reconstruction

$$\hat{\mathbf{a}} = W_{dec}\mathbf{f} + \mathbf{b}_{dec} = \sum_{j=1}^{m} \mathbf{f}_j (W_{dec})_{:,j} + \mathbf{b}_{dec} \tag{1}$$

where $W_{enc} \in \mathbb{R}^{m \times n}$, $W_{dec} \in \mathbb{R}^{n \times m}$, $\mathbf{b}_{dec} \in \mathbb{R}^n$, $\mathbf{b}_{enc} \in \mathbb{R}^m$ are learned parameters. The rows of W_{enc} are the *encoder directions*, and the columns of W_{dec} are the *decoder directions*. Similarly, $\mathbf{b}_{enc}$ is the encoder bias and $\mathbf{b}_{dec}$ is the decoder bias. The decoder directions determine the features we decompose the activations into, while the encoder directions compute the coefficients of these features for a given activation. The decoder directions are constrained to have unit norm: $\|(W_{dec})_{:,i}\|_2 = 1$. The training objective over examples $\{\mathbf{a}^{(k)}\}_{k=1}^N$ is the sum of the MSE between the activations $\mathbf{a}^{(k)}$ and their reconstructions $\hat{\mathbf{a}}^{(k)}$, and the ℓ_1 regularization term $\lambda \sum_{k=1}^N \left\|\mathbf{f}^{(k)}\right\|_1$, where λ is the ℓ_1 regularization coefficient.

The IOI task. In Wang et al. (2023), the authors analyze how the decoder-only transformer language model GPT-2 Small (Radford et al., 2019) performs the Indirect Object Identification (IOI) task. In this task, the model is required to complete sentences of the form 'When Mary and John went to the store, John gave a book to' (with the intended completion in this case being ' Mary'). We refer to the repeated name (John) as **S** (the subject) and the non-repeated name (Mary) as **IO** (the indirect object). For each choice of the **IO** and **S** names, there are two patterns the sentence can have: one where the **IO** name comes first (we call these 'ABB examples'), and one where it comes second (we call these 'BAB examples'). We refer to

this binary attribute as the **Pos** attribute (short for position). Additional details on the data distribution, model and task performance are given in Appendix A.4.

Wang et al. (2023) discover several classes of attention heads in GPT2-Small that collectively form the *IOI circuit* solving the IOI task. Specifically, Wang et al. (2023) argue that the circuit implements the algorithm:

1. detect the (i) position in the sentence and (ii) identity of the repeated name in the sentence (i.e., the **S** name). This information is computed and moved by *duplicate token/induction* and *S-Inhibition* heads;
2. based on the two signals (i) and (ii), exclude this name from the attention of the *name mover heads*, so that they copy the remaining name (i.e., the **IO** name) to the output.

We refer the reader to Appendix A.1 and Appendix Figure 5 for more details on the IOI circuit.

The logit difference metric. To discover the circuit, Wang et al. (2023) used the logit difference: the difference in log-probabilities assigned by the model to the **IO** and **S** names. This metric is more sensitive than accuracy, which makes possible the detection of individual model components with a consistent but non-pivotal role in the task. Accordingly, we also use the logit difference throughout this work to evaluate the causal effect of fine-grained model interventions.

3 Methods for Evaluating Feature Dictionaries

3.1 Overview and Motivation

In this section, we describe and motivate our methodology for evaluating sparse feature dictionaries in the context of a specific task an LLM can perform. Throughout, let $\mathcal{D}$ be a distribution over input prompts for the task.

Step 1: Parametrize inputs via (task-relevant) attributes. First, we choose attributes a_i : $\text{support}(\mathcal{D}) \to S_i$ taking values in some finite sets S_i. For example, in the IOI task we will focus on the attributes **IO**, **S**, and **Pos** described in Section 2, with **IO** and **S** taking values in the set of names in our dataset, and **Pos** taking values in the set $\{\text{ABB}, \text{BAB}\}$.

Step 2: Compute high-quality supervised dictionaries using the attributes. The second step is to learn and validate *supervised* dictionaries computed using the attribute labels $a_i(p), p \sim \mathcal{D}$ (methodology described in Section 4). In particular, not all attribute configurations will result in dictionaries with good approximation, control and interpretability for the task[2].

Step 3: Evaluate (unsupervised) dictionaries using the supervised dictionaries as a benchmark. Finally, we evaluate the usefulness of a given feature dictionary for the chosen task, using the supervised features as a reference point in various ways (described in the next subsections). The supervised dictionaries are a necessary part of this evaluation because they allow us to *contextualize* performance differences between feature dictionaries for the chosen task.

Why are supervised dictionaries a meaningful benchmark? On the one hand, supervised dictionaries' features are hard-coded to correspond to the 'human-salient' attributes a_i. So, our evaluations risk overlooking useful feature dictionaries that do not align with this human-chosen ontology. On the other hand, the saliency of the attributes is precisely what makes them desirable targets for controlling and interpreting the model; for example, in the

[2]To get high-quality supervised dictionaries, we should choose attributes that completely capture the task-relevant information in the prompt, and are moreover compatible with the intermediate states of the model's computation on the task. Validating that our attributes satisfy this is a non-trivial but key prerequisite for our evaluation to be meaningful. We do this for the IOI task in Section 4, where we show the supervised dictionaries score highly on all tests described in the current section.

IOI task it is natural to want to change e.g. the representation of the **IO** name in the model's computation. Thus, our evaluations aim to balance two opposing goals: establish properties useful from a human standpoint, while still being agnostic to the exact concepts represented in the unsupervised dictionary.

3.2 Test 1: Sufficiency and Necessity of Dictionary Reconstructions for the Task

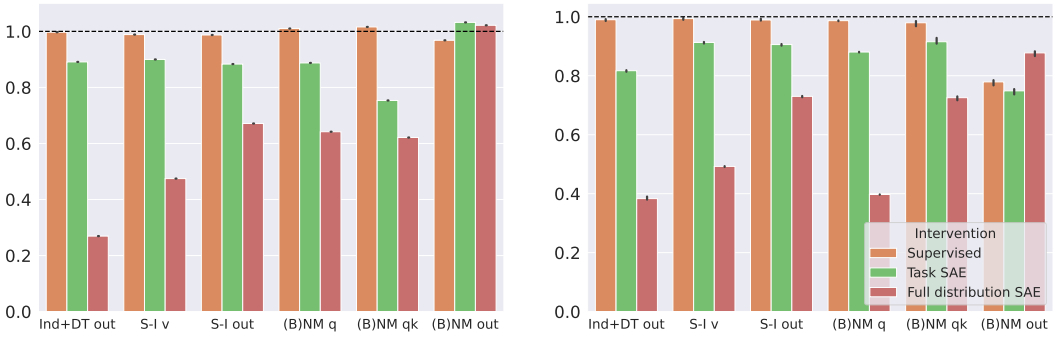

Figure 2: Sufficiency (left) and necessity (right) evaluations of reconstructions of cross-sections of the IOI circuit computed using supervised feature dictionaries, task- and full-distribution SAEs. **Left**: average logit difference when replacing activations in cross-sections of the IOI circuit with their reconstructions, normalized by the average logit difference over the data distribution in the absence of intervention (a y-axis value of 1 is best). **Right**: drop in logit difference when deleting reconstructions, normalized by the respective drop when performing mean-ablation, and linearly rescaled so that values close to 1 are best. See Appendix A.3 for details.

Our first test checks if activation reconstructions $\widehat{\mathbf{a}}$ *as a whole* are sufficient and necessary for the model to perform the task[3]. To evaluate **sufficiency**, we intervene by replacing internal activations $\mathbf{a}$ with their reconstructions $\widehat{\mathbf{a}}$, and measure the drop in performance on the task. To evaluate **necessity**, we intervene by replacing activations $\mathbf{a}$ with $\mathbb{E}_{\mathcal{D}}[\mathbf{a}] + (\mathbf{a} - \widehat{\mathbf{a}})$, and compare the resulting drop in performance to the effect of replacing activations with $\mathbb{E}_{\mathcal{D}}[\mathbf{a}]$ alone (also known as *mean ablation*; see Appendix A.2 for motivation) [4].

3.3 Test 2: Sparse Controllability of Attributes

With this test, we want to measure the degree to which the feature dictionary can be used to control the model's behavior on the task by editing intermediate representations of attribute values. To use a feature dictionary for editing, we write an activation as a linear combination of dictionary vectors, and try to remove/add dictionary elements to the combination in order to achieve the desired change. To evaluate such an intervention, we consider both (1) the number of features in the dictionary that need to be changed in order to achieve a given change in outputs (editing is trivial if e.g. activations are dense sums of random features and we change many of them), and (2) the geometric magnitude of the edit (editing is trivial if you are willing to throw away the entire activation vector and replace it with the target).

Being agnostic to feature interpretations. Importantly, we require this test to be agnostic to any human interpretation of the feature dictionary. This is valuable for several reasons: (1) even if our attributes are compatible with model computations, the feature dictionary may

[3]This test is analogous to the faithfulness/completeness test from Wang et al. (2023); the sufficiency test is also widely used in the literature to evaluate feature dictionaries.

[4]This test has very few assumptions: it is independent of the attributes we have chosen to describe inputs with, and of any interpretation assigned to the features. However, it only measures the quality of the dictionary as a whole, and not how effectively and sparsely it disentangles any hypothetical concepts in the activations. We include this test because poor performance indicates a fundamental failure of the feature dictionary relative to the task.

not be directly interpretable in terms of these attributes (see Appendix A.10 for hypothetical examples in the IOI task); (2) human-generated feature interpretations, even with meticulous methodology, may be subjective; (3) sparse control may be possible to a useful degree even if the features are inherently non-interpretable from a human perspective. Thus, this test should evaluate the degree to which the feature dictionary sparsely disentangles the attributes, but is independent of whether the features are interpretable in terms of the attributes (or at all) [5].

Implementation design space. To achieve independence of interpretations, we frame the problem as a combinatorial optimization problem over the feature dictionary: given an activation $\mathbf{a}_s$ for prompt p_s and a **counterfactual** activation $\mathbf{a}_t$ for a prompt p_t which differs from p_s only in the values of the attributes we wish to edit, we can optimize over subsets of features active in $\widehat{\mathbf{a}_s}$ to subtract from $\mathbf{a}_s$, and subsets of features active in $\widehat{\mathbf{a}_t}$ to add to $\mathbf{a}_s$. There are multiple optimization objectives possible, such as making the edited activation similar to $\mathbf{a}_t$ geometrically, or making model behavior on the edited activation similar to model behavior on $\mathbf{a}_t$. The optimization may be constrained by the number of features changed and/or the magnitude of edits [6]. Importantly, this optimization proceeds on a **per-prompt** basis, which means that it can select different features to edit depending on the prompt being edited. This means it will not disadvantage dictionaries that do not dedicate a uniform w.r.t. all prompts set of features to each attribute (which would be a major assumption to impose).

Evaluation. To measure how well an edit changed model behavior, we compare against the 'ground truth' change in behavior that would be achieved by intervening on the model to replace $\mathbf{a}_s$ with $\mathbf{a}_t$ directly. To contextualize the magnitude of the edit, we can compare the contribution of the changed features to the reconstruction against the analogous quantity for our supervised feature dictionary (see Appendix A.2 for details), as we do later in Section 4.

3.4 Test 3: Interpretability

With our final test, we want to assess the degree to which the feature dictionary can be interpreted in terms of the task-relevant attributes and the features are causally relevant for the model's behavior in a way consistent with their interpretations. While failing this test does not necessarily mean that the feature dictionary is not useful (e.g., it could still be useful for control, or interpretable with respect to different attributes), passing it would increase our confidence in the feature dictionary's utility and our understanding of the model's computation. More broadly, passing this test on a wide range of important tasks would represent a qualitative improvement over control alone in applications such as auditing, debugging and verification.

Assigning interpretability scores. Given a feature $\mathbf{u}_j$ from the dictionary, its *active set* $F \subset \text{supp}(\mathcal{D})$ is the subset of the support of $\mathcal{D}$ where $\mathbf{f}_j > 0$. Given a binary property of inputs $P : \mathcal{D} \to \{0, 1\}$, following Bricken et al. (2023), we say that P is a good interpretation of the feature if F has high precision and recall relative to P [7]. We combine the precision and recall metrics into a single number using the F_1 score.

For example, each attribute defines $|S_i|$ binary properties $\mathbf{1}_{a_i(p)=v}$ for $v \in S_i$ that we can use to try to interpret the feature dictionary. When considering a set of binary properties $\{P_j\}_{j \in J}$

[5] Note that, to meaningfully conclude disentanglement of attributes from this test, the task must be one where at least some of the activations represent multiple attributes simultaneously. This is true in the IOI task, where many circuit locations represent the **S** and **Pos** information simultaneously. We also find that one particular location, the queries of the L10H0 name mover head, represent all three attributes **IO**, **S** and **Pos**.

[6] There are many possible ways to instantiate such an optimization; in Section 5 we present one such way based on a greedy algorithm minimizing ℓ_2 distance in activation space. Further methodological notes, such as how we implement edits in multiple model components at once, are given in Appendix A.2.

[7] Clearly, the best interpretation of F according to these metrics alone is the indicator $\mathbf{1}_F$ itself; for P to be a useful interpretation from a human perspective, it must in addition be 'simple' enough, a property harder to formalize.

as possible interpretations for a feature, we pick the one with the highest F_1 score and assign this as the interpretation of the feature. Other interpretability measures are possible; for a discussion of the limitations of the F_1 score, see Appendix A.2.

Which interpretations to consider? As with controllability, we should not assume that the features correspond 1-to-1 with the attributes we have chosen. Thus, evaluating the F_1 scores only for the binary properties $\mathbf{1}_{a_i(p)=v}$ corresponding to a single value for a single attribute may be subjective. On the other hand, when the chosen attributes result in a good supervised dictionary (required by Step 2 of our method), the attribute values will be sufficient to deduce the causally important information in model activations. This suggests that causally-relevant features in an arbitrary feature dictionary will correspond to particular subsets of the cartesian product $\prod_i S_i$ of the values the attributes can take.

This motivates us to look for interpretations that are expressible as intersections and unions of the indicator sets $\mathbf{1}_{a_i(p)=v}$ of the chosen attributes. We use heuristics to navigate this search space. We present one possible implementation of this in Section 5 for the IOI task. As we will see, this choice, while possibly arbitrary/subjective, is to some extent validated empirically: we find that many features can be interpreted in this way for a significantly high F_1 score threshold. We further discover some interpretable structure in the ways attribute values group together in unions.

Causal evaluation of interpretations. Finally, while our previous interpretability methods look for correlations with properties of the input prompts, we also want to know if the interpretations we assign to features in the dictionary are consistent with their causal role in the model's computation. There are two increasingly demanding ways to check this that mirror our first and second tests (Subsections 3.2 and 3.3):

- **interpretation-aware sufficiency/necessity of reconstructions**: similar to our first test (Subsection 3.2), we can (1) subtract non-interpretable features from activations and see if the model is still able to perform the task; (2) subtract highly-interpretable features and see if the model's performance degrades to the same extent as with mean ablation. This evaluates whether our interpretability method as a whole flags the important features for the task;

- **interpretation-aware sparse controllability**: like our second test (Subsection 3.3), but explicitly using highly-interpretable features (with respect to a given attribute) to remove/add in order to edit an activation. This evaluates whether interpretations that specifically relate features to attributes find the attribute-relevant features.

4 Computing and Validating Supervised Feature Dictionaries

We first present our methods and results for computing *supervised* feature dictionaries, in which features correspond 1-to-1 with the possible values of attributes we have chosen. This is a key prerequisite for our evaluation of SAEs later on, as it (1) verifies that the attributes we have chosen are compatible with the model's internal states; (2) demonstrates the existence of high-quality sparse feature dictionaries for the task; and (3) provides an 'ideal' reference for SAE evaluation. Specifically, we will establish that feature dictionaries for all locations in the IOI circuit exist with the following properties:

- they approximate activations well using as few as 3 active features per activation;

- any of the **IO**, **S**, and **Pos** attributes can be edited precisely by replacing only 1 active feature with a different one;

- the features in the dictionary are by construction interpretable w.r.t. the attributes, and furthermore, interactions in the IOI circuit exhibit sparsity when decomposed into feature-to-feature interactions.

4.1 Computing Supervised Feature Dictionaries

Algorithms. Motivated by the linear representation hypothesis, we conjecture that given activations $\mathbf{a}(p) \in \mathbb{R}^d$ of a given model component (e.g., outputs of an attention head) for prompts $p \sim \mathcal{D}$, there exists a choice of attributes $\{a_i : \mathcal{D} \to S_i\}_{i \in I}$ such that

$$\mathbf{a}(p) \approx \mathbb{E}_{p \sim \mathcal{D}}[\mathbf{a}(p)] + \sum_{i \in I} \mathbf{u}_{a_i(\cdot)=v} := \widehat{\mathbf{a}} \tag{2}$$

where $\widehat{\mathbf{a}}$ is the *reconstruction* of $\mathbf{a}$, and $\mathbf{u}_{a_i(\cdot)=v} \in \mathbb{R}^d$ is a feature corresponding to the i-th attribute having value $v \in S_i$ [8].

Given a dataset of prompts $\{p_k\}_{k=1}^N$ with associated activations $\{\mathbf{a}(p_k)\}_{k=1}^N$, how should we compute 'good' values for the vectors $\mathbf{u}_{a_i(\cdot)=v}$? While we considered several ways to do so, our best method for the IOI task is simple: we average all activations for prompts for which $a_i(p) = v$. Formally,

$$\mathbf{u}_{a_i(\cdot)=v} := \frac{1}{|\{k : a_i(p_k) = v\}|} \sum_{k : a_i(p_k)=v} \mathbf{a}(p_k) - \overline{\mathbf{a}}$$

where $\overline{\mathbf{a}} = \frac{1}{N} \sum_{k=1}^N \mathbf{a}(p_k)$ is the empirical mean activation. We refer to these dictionaries as **mean feature dictionaries**. One can prove that, in the limit of infinite data, the mean features for an attribute not linearly detectable in the activations will converge to zero (Appendix A.5).

The main alternative method we considered was **MSE feature dictionaries**, which use a least-squares linear regression to predict the activations from the attribute values. We note that mean feature dictionaries work well in our setting because the attributes we choose in the IOI task are probabilistically independent in the IOI distribution; we recommend using MSE dictionaries in general (and see Appendix A.6 for more details on MSE dictionaries and comparisons to mean dictionaries).

Choosing attributes for the IOI task. Not every set of attributes will result in a good approximation of the model's internal activations; in fact, we find that the choice of attributes is a crucial modeling decision. Recall that, according to Wang et al. (2023), each IOI prompt p is described by three properties influencing how p is processed in the IOI circuit: the subject (**S**) and indirect object (**IO**) names, and their relative position (**Pos**). Motivated by this, we chose the attributes **S**, **IO**, and **Pos** to describe each prompt. We experimented with other choices of attributes, but did not find them to be more successful in our tests (see Appendix A.8 for details[9]). We emphasize that there are many other imaginable choices of attributes; see Appendix A.10 for further discussion.

4.2 Evaluation Results

Our evaluations are carried out at the main cross-sections of the IOI circuit. These cross-sections, as well as the information processing they perform and the expected effect of editing attributes in them, are described in detail in Appendix A.3. These considerations guide our choice of the attributes to edit in each given cross-section.

Sufficiency/necessity plots using the mean feature dictionaries are shown in Figure 2 (orange bars); we find that our supervised dictionaries are quite successful. For controllability, note that attribute editing can naturally be defined in closed form, because we have a 1-to-1 correspondence between attribute values and features by construction. We find that simple feature arithmetic works quite well. Formally, given prompt p with $a_i(p) = v$, we can

[8] This formulation is less expressive than SAE reconstructions, as it effectively requires a given feature to always appear with the same coefficient in reconstructions; we discuss this choice in Appendix A.3.

[9] In particular, we found that another set of attributes, though more expressive in principle, learns to approximate the features we get using the **S**, **IO** and **Pos** attributes through a change-of-variables-like transformation.

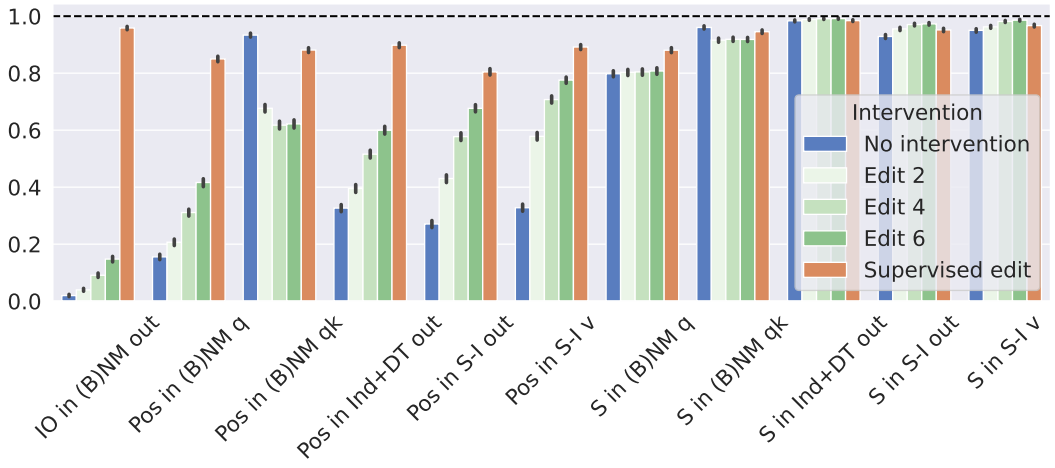

Figure 3: Accuracy when editing **IO**, **S** and **Pos** for circuit cross-sections using our supervised feature dictionaries and task-specific SAEs; the outcome in the absence of intervention is shown in blue for reference. When using task-specific SAEs, we edit either 2, 4 or 6 features (which means we in total add and/or remove up to that many features from activations). For comparison, supervised edits always involve removing 1 feature and adding 1 feature. Accuracy is measured as the proportion of examples for which the model's prediction agrees with the ground-truth prediction for the edit; see Section 5.1 and Appendix A.3 for details.

edit the attribute a_i to have value v' via $\mathbf{a}_{edited}(p) := \mathbf{a}(p) - \mathbf{u}_{a_i(\cdot)=v} + \mathbf{u}_{a_i(\cdot)=v'}$. Results for editing in cross-sections of the IOI circuit are shown in Figure 3 (orange bars), where we report the fraction of the time our intervention predicts the same token as the ground-truth intervention that substitutes counterfactual activations in the corresponding cross-section (a value of 1 is best). We observe that, when editing the **S** attribute, not performing any intervention often already mostly agrees with the ground-truth edit, effectively reducing the resolution of our evaluation results for this attribute. Otherwise, we find our supervised dictionaries to perform well, always achieving $> 80\%$ agreement with the ground-truth intervention.

Finally, the supervised feature dictionaries tautologically pass the interpretability test, as they were defined to have a single feature activating for each possible attribute value, achieving perfect F_1 scores. Accordingly, we performed a more demanding test of interpretability: decomposing internal model computations in terms of interactions between individual features. We find that pre-softmax attention scores and composition between heads can be decomposed in terms of feature-level interactions, such that many interactions are close to zero, and the few non-zeros correspond to those expected based on the high-level IOI circuit description from Wang et al. (2023); see Appendix A.7 for details.

5 Evaluating Task-Specific and Full-Distribution Sparse Autoencoders

5.1 Methodology

SAE training on the task-specific and full pretraining distributions. We trained SAEs on all IOI circuit locations, using activations from either the IOI dataset ('task SAEs') or OpenWebText distribution (Gokaslan & Cohen, 2019) ('full-distribution SAEs'). While performance varied strongly across circuit locations, most full-distribution SAEs had an ℓ_0-loss between 2 and 12 and a recovered loss fraction (against a mean ablation baseline) between 0.4 and 0.9 (both measured on OpenWebText). Similarly, most task-SAEs had an

ℓ_0-loss below 25 and a recovered logit difference fraction against mean ablation > 0.8 (both measured on the IOI dataset) [10]. Further details are given in Appendix A.11.

Sparse controllability implementation in the IOI task. We instantiate the sparse controllability test from Subsection 3.3 as follows. Suppose our SAE has a dictionary of decoder vectors $\{\mathbf{u}_j\}_{j=1}^m$, and the original and counterfactual activations $\mathbf{a}_s, \mathbf{a}_t$ have reconstructions respectively

$$\widehat{\mathbf{a}}_s = \sum_{i \in S} \alpha_i \mathbf{u}_i + \mathbf{b}_{dec}, \quad \widehat{\mathbf{a}}_t = \sum_{i \in T} \beta_i \mathbf{u}_i + \mathbf{b}_{dec}$$

for $S, T \subset \{1, \ldots, m\}$ and $\alpha_i, \beta_i > 0$. Consider the optimization problem

$$\min_{R \subset S, A \subset T, |R \cup A| \leq k} \left\| \mathbf{a}_s - \sum_{i \in R} \alpha_i \mathbf{u}_i + \sum_{i \in A} \beta_i \mathbf{u}_i - \mathbf{a}_t \right\|_2$$

In words, this problem asks for at most k features to remove (R) from and/or add (A) to the original activation to bring it as close as possible to the counterfactual activation, where the features to add are taken directly from the counterfactual one. In general, this problem has no polynomial-time solution in $k, |S|, |T|$ (the NP-hard problem SUBSETSUM reduces to it); instead, we use a greedy algorithm to find a solution.

Measuring the magnitude of edits. To measure the magnitude of the edit, we compare the contribution of the changed features to the reconstruction against the analogous quantity for our 'ideal' supervised feature dictionary. Namely, for each summand in the reconstruction we assign a measure of its contribution $\text{weight}(i) = (f_i \mathbf{u}_i)^\top (\widehat{\mathbf{a}} - \mathbf{b}_{dec}) \,/\, \|\widehat{\mathbf{a}} - \mathbf{b}_{dec}\|_2^2$ so that $\sum_{i=1}^k \text{weight}(i) = 1$[11]. Note that weights are additive in the features, so that the sum of weights of some subset of features is the weight for these features' total contribution to the reconstruction. We then measure the magnitude of an edit by the total weight of the features removed during the edit.

Interpretability implementation for the IOI task. We instantiate the interpretability test from Section 3 as follows. Starting with our primary attributes **IO**, **S**, and **Pos**, we consider intersections of each of **IO** and **S** with the **Pos** attribute. Then, for each resulting attribute that represents variation over the **IO** or **S** name, we also consider unions of up to 30 attribute values. Finally, we add some other attributes of interest, such as the gender commonly associated with names in our dataset. Details on all the possible interpretations we consdered are given in Appendix A.12. Additionally, we evaluate the causal role of highly interpretable features as described in Subsection 3.4:

- **sufficiency/necessity of interpretable features**: for sufficiency, we intervene by subtracting from activations only the features for which *no* interpretation has an F_1 score above some threshold. To test necessity, we instead subtract the features for which their chosen interpretation has an F_1 score above a threshold. This experiment is directly comparable with the sufficiency/necessity of reconstructions experiments, and uses the same baselines.
- **interpretability-aware sparse control**: we attempt to edit a given attribute by removing/adding only $\leq k$ features with highest F_1 score for the value of this attribute in the original/counterfactual prompt. This experiment is directly comparable to the (interpretation-agnostic) sparse control experiment and uses the same baseline.

Additional details are given in Appendix A.12.

[10] Importantly, we did not perform exhaustive hyperparameter tuning to train these SAEs, as our main goal was to evaluate the methodology and how it can distinguish between different classes of feature dictionaries, rather than to achieve state-of-the-art performance. Thus it is possible that significantly better performance could be achieved with more tuning. Indeed, it is our hope that the methods we present here will be useful for tuning SAEs in the future.

[11] While weights can in general take any real value, we find that in practice they are almost always approximately in $[0, 1]$; see Appendix A.16 for empirical details.

5.2 Results

Test 1: Sufficiency/necessity of reconstructions. The sufficiency/necessity plots for the task-specific and full-distribution SAEs are shown in Figure 2 (green and red). We find that the task-specific SAEs offer a significantly worse, but not catastrophically bad, approximation compared to the supervised feature dictionaries. Meanwhile, the full-distribution SAEs fare notably worse than the task SAEs.

Test 2: (Interpretation-agnostic) sparse controllability.

Task-specific SAEs. Results for sparse controllability using our task-specific SAEs are shown in Figure 3 for $k = 2, 4, 6$. We find that our SAEs can edit the **Pos** attribute and, to a small extent, the **IO** attribute, even though this requires changing more features compared to the supervised dictionaries. For the **S** attribute, the results are less clear, because the range of performance between 'no intervention' and the supervised edit is often within the margin of error; the exception is the queries of the name mover heads, where results indicate failure of the SAE features. Regarding the magnitude of edits, the most successful edits introduce higher-magnitude changes as measured by the weight (recall Subsection 5.1) than the corresponding supervised edits (results in Appendix Figure 20). On the positive side, the edits don't overwrite the SAE features entirely.

Full-distribution SAEs. Full-distribution SAEs require a significantly larger number of features to achieve a statistically significant level of control compared to task-specific ones (often 32 or more); results are shown in Appendix Figure 23. We also found that the magnitude of these edits surpsasses that of supervised edits significantly, often up to the point of throwing away total weight approaching 1.

A baseline: task-specific SAEs with decoder directions frozen at initialization. Do these results demonstrate *any* non-trivial controllability afforded by the SAE features? To check, we run the same controllability pipeline on task-specific SAEs which were trained with frozen at initialization decoder directions; thus, these SAEs are forced to approximate activations using sparse sums of random features. Results in Appendix Figure 22 show that our evaluation distinguishes between these two types of SAEs: the frozen-decoder SAEs perform much worse than the ordinary task-specific SAEs.

Test 3: Interpretability.

Correlational evaluation. Full-distribution SAEs must capture variation in activations across a large set of text, of which IOI-like prompts are only a small subset. Consistent with this, we found that only a subset of full-distribution SAE features activates on IOI prompts, with the number of features that fire on IOI prompts varying strongly between components. We scored the features that do fire on IOI prompts and found a significant amount of feature descriptions with high F_1-score. We summarize the number of high-F_1-score features per type in Figure 1. Remarkably, we find that the interpretable features in the full-distribution SAEs and the task-specific SAEs are qualitatively similar; corresponding task-specific results are given in Appendix Figures 24 (showing the most interpretable SAEs at each node of the IOI circuit) and 25 (showing the SAEs chosen to optimize the tradeoff between the ℓ_0 loss and the logit difference reconstruction).

In practice, we want to use feature explanations to get insight into the more general computation of a component. Thus, we investigated whether the features found are consistent with the previously established function of the heads from Wang et al. (2023). We found that this was true for all heads and that simply looking at the number of features with a given interpretation draws a clear picture; examples are provided in Appendix A.13.

Our results also suggests several details about the IOI circuit that weren't reported previously, which we summarize in Appendix A.13. We were also curious about how the detected features behave on arbitrary text of the model's training distribution. As creating a rigorous test for this is difficult, we report some anecdotal evidence in support of feature generalization in Appendix A.13.

Causal evaluation: sufficiency/necessity of interpretable features. Results here are encouraging: keeping/removing the features with F_1 score ≥ 0.6 often goes a long way towards preserv-

ing/degrading the model's performance on the task. Appendix Figures 28 and 29 show the results of these experiments for task SAEs.

Causal evaluation: sparse control via interpretable features. Here, results are also moderately encouraging. We find that, for the task-specific SAEs, editing using the high-F_1-score features w.r.t. a given attribute as a guide performs similarly to the interpretation-agnostic editing method. Results are provided in Appendix Figure 26. Similarly, for full-distribution SAEs, we again need to edit a high number of features to achieve a noticeable effect (results in Appendix Figure 27).

6 Qualitative Phenomena in SAE Learning

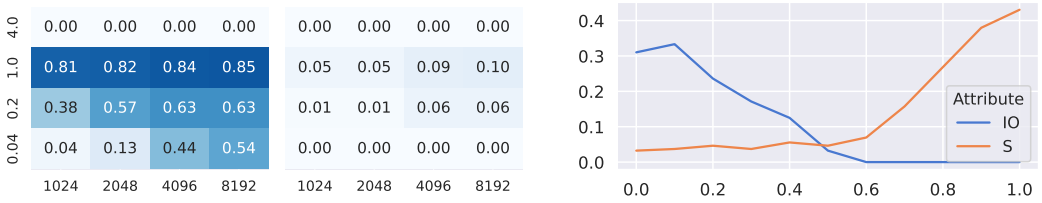

Figure 4: **Left**: Fraction of **IO** (left subplot) and **S** (right subplot) names in our dataset for which a feature with F_1 score ≥ 0.5 is found, as a function of dictionary size (x-axis) and effective ℓ_1 regularization coefficient (y-axis), over a wide hyperparameter sweep for the queries of L10H0. **Right**: fraction of **IO** and **S** names in our dataset for which a feature with F_1 score ≥ 0.5 is found, as a function of α (x-axis), the fraction of supervised **IO** features we subtract from the activations.

6.1 Feature Occlusion

Our experiments suggest that when two (causally relevant) attributes are represented in the same activation, but one attribute has overall higher magnitude, SAEs have a tendency to robustly learn more interpretable features for the attribute with higher magnitude. We observe this in the queries of the L10H0 name mover head at the END token in the IOI circuit, where the **IO** and **S** attributes are both represented and causally relevant. We find that our SAEs consistently find features with high F_1 score for individual **IO** names, but fail to find a significant number of features for individual **S** names. In Figure 4 (left), we show interpretability results from training SAEs over a wide grid of hyperparameters that confirm this observation; more details in Appendix A.14.

Hypothesis: feature magnitude is a driver of occlusion. We noticed that the supervised features for **IO** and **S** names in the L10H0 queries have significant difference in norm (see Appendix Figure 11 (left)). We then hypothesized that feature magnitude is a factor in this phenomenon. To check this, we surgically reduce the magnitude of **IO** features in the activations using our supervised feature dictionaries, and observe that the number of **S** features discovered in these modified activations monotonically increases as we remove larger fractions of the **IO** features (Figure 4 (right); see Appendix A.14 for methodology). We furthermore constructed a simple toy model based on i.i.d. isotropic random features that mimic the norms of supervised features in the L10H0 queries, and find that a similar phenomenon occurs for the distribution of features with high F_1 score (e.g. higher than 0.9; see Appendix A.14).

6.2 Feature Over-splitting

Our experiments suggest that SAEs have a tendency to split a single binary attribute into multiple features, even when the number of features available could in principle be spent on other attributes. Note that, while this behavior may be counter-intuitive from a human

standpoint, it does not necessarily mean that the SAE failed; it may be that the binary attribute is not part of an optimal sparse description of the model's internal states. We observe this phenomenon with the **Pos** attribute in the IOI task (again in the queries of the L10H0 name mover), which is robustly split into many (e.g. ≥ 30) features by our SAEs that activate for small, mostly non-overlapping subsets of examples sharing the same **Pos** value that have no clear semantic interpretation.

Is over-splitting a form of over-fitting? To investigate whether this is due to overfitting, we compared **Pos** features between (1) different random seeds for the same training dataset and (2) different training datasets. In both cases, we found that the **Pos** features discovered are similar above chance levels, suggesting that the over-splitting is not due solely to overfitting to randomness in the training algorithm and/or dataset.

Reproducing the over-splitting phenomenon in a simple toy model. On the other hand, we show empirical evidence that in a toy model where activations are a uniform mixture of two isotropic Gaussian random variables, an appropriately *randomly initialized* SAE with enough hidden features will achieve lower total loss than an ideal SAE with just two features corresponding to the two components of the mixture. Such a randomized construction exists for *any* ℓ_1 regularization coefficient, even in the limit of infinite training data. Details are given in Appendix A.15.

7 Related Work

Learning and evaluating SAE features. The SAE paradigm predates the recent surge of interest in LLMs. Early work in ML focused on the analysis of word embeddings (Mikolov et al., 2013a), with works such as Faruqui et al. (2015); Subramanian et al. (2017); Arora et al. (2018) finding sparse linear structure. Elhage et al. (2022a) proposed the use of sparse autoencoders to disentangle features in LLMs. Sharkey et al. (2023) used SAEs to learn an over-complete dictionary in a toy model and in a one-layer transformer, and follow-up work by Cunningham et al. (2023) applied this technique to residual stream activations[12] of a 6-layer transformer from the Pythia family (Biderman et al., 2023). Bricken et al. (2023) trained SAEs on the hidden MLP activations of a 1-layer language model, and performed several thorough evaluations of the resulting features. Similarly to us, Gould et al. (2023) trained SAEs on a *narrow* data distribution instead of internet-scale data. Tamkin et al. (2023) incorporated sparse feature dictionaries (without per-example weights for the features) into the model architecture itself, and fine-tuned the model on its pre-training distribution to learn the dictionaries. More recently, Kissane et al. (2024) used SAEs on attention layer outputs of GPT-2 Small and found learned features that are consistent with the IOI circuit from Wang et al. (2023).

Sparse autoencoders seek to (approximately) decompose activations into meaningful features, and are thus a stronger form of interpretability than linear probing for individual concepts Alain & Bengio (2016), or finding individual subspaces with causal effect (Geiger et al., 2023).

Throughout this line of work, the evaluation of learned SAE features has been a major challenge. The metrics used so far can be broadly categorized as follows:

- **indirect geometric measures**: Sharkey et al. (2023) proposed use of the mean maximum cosine similarity (MMCS) between two different SAEs' learned features to evaluate their quality. However, this metric relies on the assumption that convergence to the same set of features is equivalent to interpretability and having found the 'true' features.
- **auto-interpretability**: Bricken et al. (2023); Bills et al. (2023); Cunningham et al. (2023) used a frontier LLM to obtain natural-language descriptions of SAE features based on highly activating examples, and use the LLM to predict feature activations on unseen text; the prediction quality is then used as a measure of interpretability.

[12]We adopt the terminology of (Elhage et al., 2021) when discussing internal activations of transformer-based language models.

However, the use of maximum (or even stratified by activation value) activating examples has been criticized as potentially giving an illusory and subjective sense of interpretability (Bolukbasi et al., 2021).

- **manually crafted proxies for ground truth**: Bricken et al. (2023) manually formed hypotheses about a handful of SAE features and defined computational proxies for the ground truth features based on these hypotheses. This method may be less prone to blind spots than auto-interpretability, but still relies on the correctness of the computational proxy.
- **toy models**: Sharkey et al. (2023) used a toy model where ground-truth features are explicitly defined; however, it is unclear whether toy models miss crucial aspects of real LLMs. Similar objections apply to manually injecting ground-truth features into a real model.
- **direct logit attribution**: Bricken et al. (2023) additionally considered the direct effect of a feature on the next-token distribution of the model; this method is valuable because it tells us about the causal role of a feature, but it cannot detect its indirect effects via other features.

Beyond the evaluation challenges, there is debate about whether SAEs find computationally non-trivial, compositional features, or merely clusters of similar examples in the data Olah et al. (2024).

Mechanistic interpretability and circuit analysis. Mechanistic interpretability (MI) aims to reverse-engineer the internal workings of neural networks (Olah et al., 2020; Elhage et al., 2021). In particular, MI frames model computations as a collection of *circuits*: narrow, task-specific algorithms (Olah et al., 2020). So far, circuit analyses of LLMs have focused on the component level, mapping circuits to collections of components such as attention heads and MLP layers (Wang et al., 2023; Heimersheim & Janiak).

However, the linear representation hypothesis suggests that component activations can be broken down further into (sparse) linear combinations of meaningful feature vectors; thus, the eventual goal of MI is to give a precise, *subspace-level* understanding of the model's circuits. Initial steps in this direction have been taken using methods distinct from SAEs. Geiger et al. (2023) propose finding meaningful subspaces using an optimization-based method; Nanda et al. (2023) discover linear subspaces in emergent world-models on a toy task; Tigges et al. (2023) discover linear subspaces corresponding to sentiment in a LLM. However, while these works focus on finding individual subspaces representing specific concepts, the SAE paradigm is more ambitious, as it aims to fully decompose activations as a sum over meaningful features. This is a stronger property than identifying individual meaningful subspaces, and would accordingly provide a more exhaustive form of interpretability.

More broadly, MI has found applications in several downstream tasks: removing toxic behaviors from a model (Li et al., 2023b), changing factual knowledge encoded by models (Meng et al., 2022), improving the truthfulness of LLMs at inference time (Li et al., 2023a), studying the mechanics of gender bias in language models (Vig et al., 2020), and reducing spurious correlations by intervening on model internals (Gandelsman et al., 2023).

8 Limitations and Conclusion

We have taken steps towards more principled and objective evaluations of the usefulness of sparse feature dictionaries for disentangling LLM activations. In particular, we have demonstrated that:

- simple supervised methods can be used as a principled way to compute high-quality feature dictionaries in a task-specific context;
- these dictionaries can be used as 'skylines' to evaluate and contextualize the performance of unsupervised methods, such as SAEs.

Limitations. The central conceptual limitation of our work is that our method relies on supervision in the form of a potentially subjective choice of variables used to parametrize task-relevant information in model inputs. We mitigate this to some extent by requiring this parametrization to be *consistent* with the internal computations of the model, as quantified by our tests for approximation, control and interpretability of model computations on the task. However, in principle there could be many parametrizations that are just as consistent, but fundamentally different (recall the discussion in Appendix A.8 and A.10). Thus, we risk making the proverbial 'judging a fish by its ability to climb a tree' mistake. We have mitigated this problem further by devising evaluations that are, when possible, agnostic to the precise features in a dictionary, as long as they allow us to disentangle and control our chosen variables in a sparse manner.

Finally, we find it likely that features used by LLMs in tasks of practical interest will be quite complex from a human standpoint, and we believe it is useful to be able to assess the degree to which these features can be harnessed towards controlling model behavior along more top-down, concise and human-understandable concepts. Our evaluations provide such an assessment.

Our work is also limited in that we only consider a single task, and a single language model. We hope that in future work we will reduce these limitations.

Conclusion. Sparse dictionary learning, and SAEs in particular, represent an interesting and promising avenue for disentangling internal representations of LLMs. However, in order to measure progress in this area, it is imperative to have nuanced conceptual understanding of the goals and downstream applications of such disentangling, and benchmarks faithful to this understanding. We hope that our work will inspire further research in this direction, and that our methods will be useful for practitioners in the field.

References

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can language models encode perceptual structure without grounding? a case study in color. *arXiv preprint arXiv:2109.06129*, 2021.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *ArXiv*, abs/1610.01644, 2016. URL `https://api.semanticscholar.org/CorpusID:9794990`.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. URL `https://api.semanticscholar.org/CorpusID:8236317`.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *ArXiv*, abs/2306.03819, 2023. URL `https://api.semanticscholar.org/CorpusID:259088549`.

Stella Biderman, Hailey Schoelkopf, Quentin G. Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *ArXiv*, abs/2304.01373, 2023. URL `https://api.semanticscholar.org/CorpusID:257921893`.

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

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

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

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

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022b.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse overcomplete word vector representations. In *Annual Meeting of the Association for Computational Linguistics*, 2015. URL https://api.semanticscholar.org/CorpusID:9397697.

Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting clip's image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D Goodman. Finding alignments between interpretable causal variables and distributed neural representations. *arXiv preprint arXiv:2303.02536*, 2023.

Gabriel Goh. Decoding the representation of code in the brain: an fmri study of code review and expertise. 2016. URL https://gabgoh.github.io/ThoughtVectors/.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. *ArXiv*, abs/2312.09230, 2023. URL https://api.semanticscholar.org/CorpusID:266210012.

G Grand, I Blank, F Pereira, and E Fedorenko. Semantic projection: Recovering human knowledge of multiple, distinct object features from word embeddings. arxiv. *arXiv preprint arXiv:1802.01241*, 2018.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Stefan Heimersheim and Jett Janiak. A circuit for Python docstrings in a 4-layer attention-only transformer. URL https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only.

Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. Attention saes scale to gpt-2 small. Alignment Forum, 2024. URL https://www.alignmentforum.org/posts/FSTRedtjuHa4Gfdbr.

Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023a.

Maximilian Li, Xander Davies, and Max Nadeau. Circuit breaking: Removing model behaviors with targeted ablation. *arXiv preprint arXiv:2309.05973*, 2023b.

Tom Lieberum, Matthew Rahtz, János Kramár, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

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

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

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3111–3119, 2013a. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *North American Chapter of the Association for Computational Linguistics*, 2013b. URL https://api.semanticscholar.org/CorpusID:7478738.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

Christopher Olah. Interpretability dreams. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/interpretability-dreams/index.html.

Christopher Olah, Shan Carter, Adam Jermyn, Joshua Batson, Tom Henighan, Tom Conerly, Adly Templeton, Trenton Bricken, Jonathan Marcus, Brian Chen, and Nicholas L Turner. Circuits updates - january 2024. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/jan-update.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997. URL https://api.semanticscholar.org/CorpusID:14208692.

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

OpenAI. Gpt-4 technical report, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Lee Sharkey, Dan Braun, and Beren Millidge. Taking the temperature of transformer circuits. 2023. URL https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. Spine: Sparse interpretable neural embeddings. *ArXiv*, abs/1711.08792, 2017. URL https://api.semanticscholar.org/CorpusID:19143983.

Alex Tamkin, Mohammad Taufeeque, and Noah D Goodman. Codebook features: Sparse and discrete interpretability for neural networks. *arXiv preprint arXiv:2310.17230*, 2023.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

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=NpsVSN6o4ul.

Zeyu Yun, Yubei Chen, Bruno A. Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *Workshop on Knowledge Extraction and Integration for Deep Learning Architectures; Deep Learning Inside Out*, 2021. URL https://api.semanticscholar.org/CorpusID:232417301.

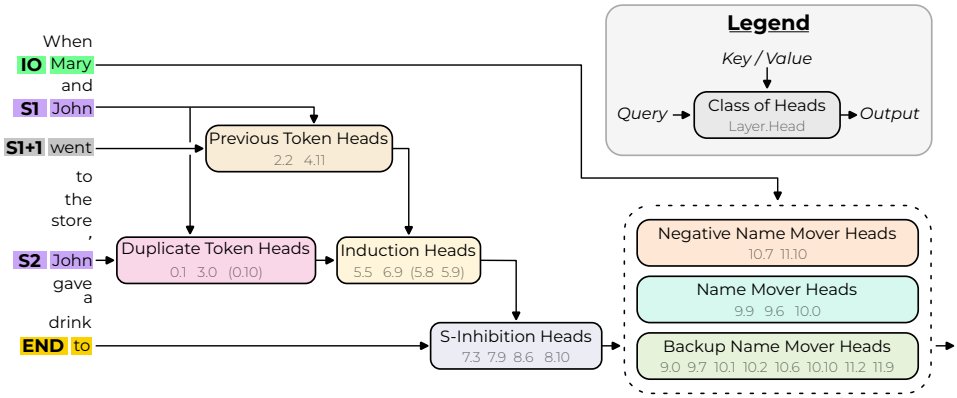

Figure 5: A reproduction of Figure 2 from Wang et al. (2023), showing the internal structure of the IOI circuit. Original caption: *The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.*

A Appendix

A.1 Additional details on the IOI circuit

Circuit structure. To refer to individual token positions within the sentence, we use the notation of Wang et al. (2023): IO denotes the position of the **IO** name, S1 and S2 denote respectively the positions of the first and second occurrences of the **S** name (with S1+1 being the token position after S1), and END denotes the last token in the sentence (at the word 'to').

Wang et al. (2023) suggest the model uses the algorithm 'Find the two names in the sentence, detect the repeated name, and predict the non-repeated name' to do this task. Specifically, they discover several classes of heads in the model, each of which performs a specific subtask of this overall algorithm. A simplified version of the circuit involves the following three classes of heads and proceeds as follows:

- **Duplicate token heads**: these heads detect the repeated name in the sentence (the **S** name) and output information about both its position and identity to the residual stream[13]
- **S-Inhibition heads**: these heads read the identity and position of the **S** name from the residual stream, and output a signal to the effect of 'do not attend to this position / this token identity' to the residual stream
- **Name Mover heads**: these are heads that attend to names in the sentence. Because the signal from the S-Inhibition heads effectively removes the **S** name from the attention of these heads, they read the identity of the **IO** name from the input prompt, and copy it to the last token position in the residual stream.

In reality, the circuit is more nuanced, with several other classes of heads participating: previous token heads, induction heads (Olsson et al., 2022), backup name mover heads, and negative name mover heads. In particular, the circuit exhibits *backup behavior* (McGrath et al., 2023) which poses challenges for interpretability methods that intervene only on single model components at a time. We refer the reader to Figure 5 for a schematic of the full circuit, and to Wang et al. (2023) for a more complete discussion.

[13] We follow the conventions of Elhage et al. (2021) when describing internals of transformer models. The residual stream at layer k is the sum of the output of all layers up to $k-1$, and is the input into layer k.

A.2 Additional details for Section 3

Why do we use mean ablation as a baseline for necessity of reconstructions? Using the mean instead of simply a zero value is intended to not take the model away from the task distribution. The information that is constant across the task distribution – such as grammar and syntax – will remain in the mean ablation, while task-specific information will be averaged out. If the residual $\mathbf{a} - \widehat{\mathbf{a}}$ is not task-relevant, we expect that the model will perform the task as well as with the mean ablation; conversely, if the reconstructions $\widehat{\mathbf{a}}$ leave out task-relevant information, we expect that the model will perform better with our intervention than with the mean ablation.

Editing in multiple model locations at once. We do this by pre-computing edited activations for each location, then patching all of them into the model's forward pass at once. This means that we are only editing computational cross-sections of the model; however, applying the edits sequentially as the model is run is prone to taking activations off-distribution, since an edit will propagate to downstream activations, where making the same edit again may be the wrong intervention; we have observed this in practice, in the interaction of the name mover heads in layers 9 and 10 in the IOI circuit.

Evaluating edits. Our edits are intended to only change the values of specific attributes, and leave all other information in the activations unchanged. A ground-truth baseline for the effect of an edit would be to take the model's activation on a *counterfactual* prompt: one which differs from the original prompt only in the value of the attribute being edited, in the precise way that the edit changes it. The existence of counterfactual prompts in the support of $\mathcal{D}$ is not guaranteed in general, but it holds in our IOI distribution.

Precision, recall, and the F_1-score. Given a set of examples S used for evaluation, a learned feature f active on a subset $F \subset S$ of examples, and a binary attribute of a prompt which is true for the subset $A \subset S$, we define $\text{recall}(F, A) = |A \cap F| \,/\, |A|$ and $\text{precision}(F, A) = |A \cap F| \,/\, |F|$. Following Bricken et al. (2023), we consider a feature meaningful for a given property if it has both high recall and high precision for that property, and we combine them into a single number using the F-score:

$$F_1(F, A) = \frac{2\,\text{precision}(F, A)\,\text{recall}(F, A)}{\text{precision}(F, A) + \text{recall}(F, A)}.$$

An F_1-score of α guarantees that both precision and recall are at least $\frac{\alpha}{2-\alpha}$. For example, when $\alpha = 0.8$ (the value we use in most evaluations), both precision and recall are at least $0.8/1.2 \approx 0.67$. Requiring a sufficiently high F_1 value is important in order to avoid labeling a trivial feature as meaningful for attributes where $|A|$ is large, because then a feature active for all examples can have a high F_1-score.

The F_1 score has some limitations in the context of our work:

- it does not take into account the magnitude of the feature activations; for instance, a feature that is active for all examples in S but only has high activation values on the examples in A may have a low F_1 score, even though it is in some sense highly informative for the attribute A.
- it is a very conservative metric, in that it requires both high precision and high recall to be high. For example, a feature with precision 0.5 but recall 0.02 will have an F_1 score of ~ 0.04, heavily skewed towards the lower of the two metrics, even though it is in some sense informative for the attribute A.

We hope to address these limitations in future work.

A.3 Additional details for Section 4

Why do we use fixed coefficients for our supervised activation reconstructions? Note that the formulation from Equation 2 is *less expressive* than the reconstruction provided by an SAE (Equation 1), as it requires the coefficients of decoder vectors to be fixed (similar to Tamkin et al. (2023), and unlike in the SAE, where they are computed from the input via a

linear function and a ReLU). This is a reasonable assumption in settings where we expect the relevant features to behave as binary, on/off switches (as opposed to having continuous degrees of activation). The IOI task is an example of such a setting, as we expect that there is no 'degree' to which a given name in the sentence is present or not, or to which a given name is repeated or not. See also the discussion of 'features as directions' vs 'features as points' in Tamkin et al. (2023).

Computing and evaluating supervised feature dictionaries. For each parametrization and each method to compute feature dictionaries, we use 20,000 prompts sampled from our IOI distribution (see Appendix A.4) to compute feature dictionaries for the query, key, value, and attention output (i.e., attention-weighted values) of the relevant token positions of all 26 heads identified in Wang et al. (2023) (recall Figure 5). We use another sample of 5,000 prompts to validate the quality of the feature dictionaries.

Cross-sections of the circuit. Based on the understanding of the IOI circuit from Wang et al. (2023), we identify several cross-sections of the computational graph of the IOI circuit where feature editing is expected to have effects meaningful for the task:

- *outputs of (backup) name mover heads at END (**(B)NM out**)*: these activations encode the **IO** name and write it to the END token of the residual stream. We expect that editing the **IO** name in these activations will directly affect the model's prediction, while editing other attributes will not have a significant effect.
- *queries+keys of (backup) name movers at END (**(B)NM qk**)*: the queries represent the **S** name and **Pos** information, but they are mainly used as *inhibitory* signals for the model, decreasing the attention to the **S** token[14]. The keys represent information about the **IO** and **S** names: in particular, the **S** information combines with the query to inhibit attention to the **S** token.

  We expect that editing the **S** and **Pos** attributes in *both* the keys and queries will not significantly hurt model performance, because as a result attention to the **S** token will again be inhibited. By contrast, it is unclear what editing the **IO** name is expected to do, since its role in the attention computation is not fully described in Wang et al. (2023).
- *outputs of S-Inhibition heads at END (**S-I out**), values of S-Inhibition heads at S2 (**S-I v**), and outputs of duplicate token and induction heads at S2 (**Ind+DT out**)*: these activations transmit the inhibitory signal to the name mover heads through the residual stream. We expect that editing **S** and **Pos** in these activations will lower the model's logit difference by disrupting the inhibitory signal, while editing **IO** will have no effect.

Evaluating necessity of feature reconstructions: When we intervene on the model by removing reconstructions from activations in cross-sections of the circuit, model performance on the IOI task (as measured by the logit difference) goes down from the clean value $\text{logitdiff}_{\textbf{clean}}$ to a lower value $\text{logitdiff}_{\textbf{intervention}}$. As we describe in the main text, the ground-truth intervention for removing the features from the activations is mean ablation of the corresponding cross-section, which also results in a lower value of the logit difference, $\text{logitdiff}_{\textbf{mean ablation}}$. We want to measure the degree to which $\text{logitdiff}_{\textbf{intervention}}$ approximates $\text{logitdiff}_{\textbf{mean ablation}}$, in a way that normalizes for different values of $\text{logitdiff}_{\textbf{mean ablation}}$ across cross-sections of the circuit. We use the following metric to do this:

$$\text{necessity score} = 1 - \frac{\left|\text{logitdiff}_{\textbf{mean ablation}} - \text{logitdiff}_{\textbf{intervention}}\right|}{\left|\text{logitdiff}_{\textbf{mean ablation}} - \text{logitdiff}_{\textbf{clean}}\right|}.$$

Evaluating accuracy of attribute edits. In our figures on attribute editing (e.g., Figure 3), we report the proportion of examples (in a test set not used to compute feature dictionaries) for which the model's prediction when intervening via a given edit equals the model prediction

[14]In addition, we later find that the queries of the L10H0 name mover head also represent the **IO** attribute, and serve an *inhibitory* role for it as well, decreasing the attention to the **IO** token.

when we intervene by the ground-truth editing intervention described in 3.3. This metric's ideal value is 1, and its worst value is zero. In many cases, simply not intervening on the model already achieves a nontrivial (and sometimes very high) value of this score; this is why we also report the value in the absence of intervention.

A.4 Dataset, Model and Evaluation Details for the IOI Task

We use GPT2-Small for the IOI task, with a dataset that spans 216 single-token names, 144 single-token objects and 75 single-token places, which are split 1 : 1 across a training and test set. Every example in the data distribution includes (i) an initial clause introducing the indirect object (**IO**, here 'Mary') and the subject (**S**, here 'John'), and (ii) a main clause that refers to the subject a second time. Beyond that, the dataset varies in the two names, the initial clause content, and the main clause content. Specifically, use three templates as shown below:

Then, [ ] and [ ] had a long and really crazy argument. Afterwards, [ ] said to
Then, [ ] and [ ] had lots of fun at the [place]. Afterwards, [ ] gave a [object] to
Then, [ ] and [ ] were working at the [place]. [ ] decided to give a [object] to

and we use the first two in training and the last in the test set. Thus, the test set relies on unseen templates, names, objects and places. We used fewer templates than the IOI paper Wang et al. (2023) in order to simplify tokenization (so that the token positions of our names always align), but our results also hold with shifted templates like in the IOI paper.

On the test partition of this dataset, GPT2-Small achieves an accuracy of $\approx 91\%$. The average difference of logits between the correct and incorrect name is ≈ 3.3, and the logit of the correct name is greater than that of the incorrect name in $\approx 99\%$ of examples. Note that, while the logit difference is closely related to the model's correctness, it being > 0 does not imply that the model makes the correct prediction, because there could be a third token with a greater logit than both names.

A.5 Properties of mean feature dictionaries

Mean feature dictionaries enjoy several convenient properties:

- The vectors $\mathbf{u}_{iv}$ for an attribute a_i do not depend on which other attributes $a_l \neq a_i$ we have chosen to describe the prompt p with.
- If an attribute i is not linearly represented in the activations, the mean code features $\mathbf{v}_{iv} \to 0$ in the limit of infinite data (see below). In particular, this also holds if the attribute is not represented *at all* in the activations.

This suggests that mean feature dictionaries are robust to the inclusion of irrelevant or non(-linearly)-represented attributes, which is a desirable property in real settings where we may not know the exact attributes present in each activation. However, mean feature dictionaries are *not* robust to the inclusion of redundant attributes, as the lack of interaction between the attributes means that redundant attributes cannot 'coordinate' to reduce the reconstruction error $\|\mathbf{a} - \widehat{\mathbf{a}}\|_2^2$.

A.5.1 Mean features are zero for non-linearly-represented attributes.

Suppose we have a random vector $\mathbf{x}$ for a k-way classification task with one-hot labels $\mathbf{z} \in \mathcal{Z} = \{\mathbf{z} \in \{0,1\}^k \text{ s.t. } \|\mathbf{z}\|_1 = 1\}$. In Section 3 of Belrose et al. (2023), it is shown that the following are equivalent:

- the expected cross-entropy loss of a linear predictor $\widehat{\mathbf{z}} = \mathbf{w}^\top \mathbf{x} + \mathbf{b}$ for $\mathbf{z}$ is minimized at a *constant* linear predictor. In other words, the optimal logistic regression classifier (in the limit of infinite data) is no better than the optimal constant predictor (which, at best, always predicts the majority class).

- the class-conditional mean vectors $\mathbb{E}[\mathbf{x}|\mathbf{z} = e_i]$ are all equal to the overall mean $\mathbb{E}[\mathbf{x}]$ of the data.

If we translate this to the context of mean feature dictionaries from Subsection 4, we have that logistic regression for the value of an attribute a_i will degenerate to the majority class predictor if and only if the mean feature dictionaries for all values of this attribute are zero. In the finite data regime, this gives us some theoretical grounds to expect that the mean feature dictionaries will be significantly away from zero if and only if the attribute's values can be non-trivially recovered by a (logistic) linear probe. As a special case, if an attribute is not represented in the data at all, we expect the mean feature dictionaries for this attribute to be zero.

A.6 Definition and Properties of MSE Feature Dictionaries

MSE feature dictionaries compute $\mathbf{u}_{a_i(\cdot)=v}$ by directly minimizing the ℓ_2 reconstruction error over the centered activations:

$$\{\mathbf{u}_{a_i(\cdot)=v}\}_{i\in I, v\in S_i} = \arg\min_{\mathbf{u}_{a_i(\cdot)=v}} \frac{1}{N}\sum_{k=1}^{N}\left\|(\mathbf{a}(p_k) - \overline{\mathbf{a}}) - \sum_{i\in I}\mathbf{u}_{a_i(\cdot)=a_i(p_k)}\right\|_2^2 \tag{3}$$

This objective is convex, and is equivalent to a least-squares regression problem; in fact, the optimal solutions take a form very similar to the mean feature dictionaries (see below). Furthermore, this objective closely mimics the SAE objective: here, the sparsity is hard-coded, leaving only the ℓ_2 objective.

We next discuss some properties of the MSE feature dictionaries. For brevity, in the remainder of this section we write $\mathbf{u}_{iv}$ instead of $\mathbf{u}_{a_i(\cdot)=v}$.

A.6.1 MSE feature dictionaries as a multivariate least-squares regression problem.

Let $S = \sum_{i=1}^{N_A}|S_i|$ be the total number of possible values for all attributes. For each attribute i, consider the characteristic matrix $C_i \in \mathbb{R}^{N\times S_i}$ of the dataset for this attribute, where

$$C_{kj} = \begin{cases} 1 & \text{if } a_i(p^{(k)}) = v_j \\ 0 & \text{otherwise} \end{cases}$$

for some ordering $(v_1, \ldots, v_{|S_i|})$ of the values in S_i, and let $C = [C_1 \quad C_2 \quad \cdots \quad C_{N_A}] \in \mathbb{R}^{N\times S}$ be the concatenation of all characteristic matrices. Also, let $A \in \mathbb{R}^{N\times d}$ be the matrix of activations with rows $\mathbf{a}^{(k)}$. Then the objective function for the MSE feature dictionaries can be written as the multivariate least-squares regression problem

$$\min_{U\in\mathbb{R}^{S\times d}} \frac{1}{N}\|A - CU\|_F^2$$

where the rows of U are the vectors $\mathbf{u}_{iv}$ across all i and $v \in S_i$, with the optimal solution given by

$$U^* = \left(C^\top C\right)^+ C^\top A \tag{4}$$

A.6.2 MSE feature dictionaries as averaging over examples.

Using the special structure of the objective, we can also derive some information about the optimal solutions $\mathbf{u}^*_{iv}$. Namely, at optimality we should not be able to decrease the value of the objective by changing a given $\mathbf{u}^*_{iv}$ away from its optimal value. The terms containing $\mathbf{u}^*_{iv}$ in the objective are

$$\begin{aligned}\frac{1}{N}\sum_{k\in P_{iv}}\left\|\mathbf{a}^{(k)} - \sum_{l\neq i}\mathbf{u}^*_{l v_l^{(k)}} - \mathbf{u}^*_{iv}\right\|_2^2 &= \frac{1}{N}\sum_{k\in P_{iv}}\left\|\left(\mathbf{a}^{(k)} - \sum_{l\neq i}\mathbf{u}^*_{l v_l^{(k)}}\right) - \mathbf{u}^*_{iv}\right\|_2^2 \\ &= \frac{1}{N}\sum_{k\in P_{iv}}\left\|\overline{\mathbf{a}}^{(k)} - \mathbf{u}^*_{iv}\right\|_2^2\end{aligned}$$

where recall that $P_{iv} = \{k \,|\, a_i(p^{(k)}) = v\}$, and $\overline{\mathbf{a}}^{(k)}$ is the residual of $\mathbf{a}^{(k)}$ after subtracting the reconstruction using all other attributes $l \neq i$. Since this value cannot be decreased by changing $\mathbf{u}^*_{iv}$, we have that it equals the minimizer of this term (holding $\overline{a}^{(k)}$ fixed). In other words, if we define

$$f(\mathbf{u}) = \frac{1}{N} \sum_{k \in P_{iv}} \left\| \overline{\mathbf{a}}^{(k)} - \mathbf{u} \right\|_2^2$$

we have that $\mathbf{u}^*_{iv} = \arg\min_{\mathbf{u}} f(\mathbf{u})$. Since f is a sum of convex functions, it is itself convex, and so the first-order optimality condition is also sufficient for optimality. We have

$$\nabla f(\mathbf{u}) \propto \sum_{k \in P_{iv}} \left(\overline{\mathbf{a}}^{(k)} - \mathbf{u} \right) \propto \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \overline{\mathbf{a}}^{(k)} - \mathbf{u}$$

and so

$$\mathbf{u}^*_{iv} = \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \overline{\mathbf{a}}^{(k)} \tag{5}$$

Note that this is very similar to the definition of mean feature dictionaries, but also importantly different, because the optimal $\mathbf{u}^*_{iv}$ depends on the optimal values of the feature dictionaries for the other attributes.

A.6.3 MSE feature dictionaries with independent attributes.

Finally, we can prove that, under certain conditions, attributes for which $\mathbb{E}[\mathbf{a}|a_i(p) = v_i] = \mathbb{E}[\mathbf{a}]$, i.e. the conditional mean of activations over values of the attribute is the same as the overall mean (assuming both means exist), will have (approximately) constant MSE feature dictionaries $\mathbf{u}_{iv} = \mathbf{u}_i \forall v$. This is a counterpart to the result from Appendix A.5 for MSE feature dictionaries:

Lemma A.1. *Suppose that all conditional means $\mathbb{E}_{p\sim\mathcal{D}}[\mathbf{a}|a_i(p) = v]$ exist for all $i, v \in S_i$. Let a_i be an attribute such its values appear independently from the values of all other attributes, i.e.*

$$\mathbb{P}_{p\sim\mathcal{D}}[a_i(p) = v_i, a_l(p) = v_l] = \mathbb{P}_{p\sim\mathcal{D}}[a_i(p) = v_i]\,\mathbb{P}_{p\sim\mathcal{D}}[a_l(p) = v_l] \quad \forall v_i \in S_i, v_l \in S_l, l \neq i$$

Then, in the limit of infinite training data, the conditional means $\mathbb{E}[\mathbf{a}|a_i(p) = v]$ are all equal to the overall mean $\mathbb{E}[\mathbf{a}]$ if and only if the optimal MSE feature dictionaries $\mathbf{u}^*_{iv}$ for this attribute are constant with respect to the value v of the attribute, i.e. $\mathbf{u}^*_{iv} = \mathbf{u}_i$ for all $v \in S_i$.

Proof. From Equation 5, we have

$$\begin{aligned}
\mathbf{u}^*_{iv} &= \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \overline{\mathbf{a}}^{(k)} = \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \left(\mathbf{a}^{(k)} - \sum_{l \neq i} \mathbf{u}^*_{l v_l^{(k)}} \right) \\
&= \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \mathbf{a}^{(k)} - \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \sum_{l \neq i} \mathbf{u}^*_{l v_l^{(k)}}
\end{aligned}$$

The first term converges to $\mathbb{E}[\mathbf{a}|a_i(p) = v]$. The second term is a sum of terms of the form

$$\frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \mathbf{u}^*_{l v_l^{(k)}} = \frac{1}{|P_{iv}|} \sum_{v_l \in S_l} \mathbf{u}^*_{l v_l} \left| \{k \text{ s.t. } a_i(p_k) = v, a_l(p_k) = v_l\} \right| \tag{6}$$

for $l \neq i$. Since we are assuming a_i is uncorrelated with a_l, in the limit of the size N of the dataset $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(N)}$ going to infinity, $|\{k \text{ s.t. } a_i(p_k) = v, a_l(p_k) = v_l\}|$ will approach $|P_{iv}|\,\mathbb{E}_{p\sim\mathcal{D}}\left[\mathbf{1}_{a_l(p)=v_l}\right]$. Moreover, note that in the closed-form solution $U^* = \left(C^\top C\right)^+ C^\top A = \left(\frac{C^\top C}{N}\right)^+ \frac{C^\top}{N} A$ from Equation 4, the matrix $\frac{1}{N} C^T C$ converges to some limit $\Sigma \in \mathbb{R}^{S \times S}$ as $N \to \infty$, and the matrix $\frac{1}{N} C^\top A$ similarly converges to some limit $M \in \mathbb{R}^{S \times d}$ by the assumption that all conditional means for all attributes exist. Thus, the optimal feature

dictionaries $\mathbf{u}^*_{iv}$ will also converge as $N \to \infty$. So we see that the sum in Equation 6 will converge to a value that is independent of the value v for the attribute a_i.

Thus, if the conditional means $\mathbb{E}[\mathbf{a}|a_i(p) = v]$ are all equal to the overall mean $\mathbb{E}[\mathbf{a}]$, we get that $\mathbf{u}^*_{iv}$ is independent of v; conversely, if $\mathbf{u}^*_{iv}$ is independent of v, we get that the conditional means are all equal to the overall mean. This completes the proof.

□

A.7 Feature-level mechanistic analyses for Section 4

Since each activation is approximated as the sum of several vectors from a finite set, it becomes possible to decompose the model's internal operations in terms of elementary interactions between the learned vectors themselves. In the current paper, we are particularly interested in attention heads, as they are the building blocks of the IOI circuit. We consider the following subspace-level analyses:

- **Attention scores**: The attention mechanism is considered to be a crucial reason for the success of LLMs (Vaswani et al., 2017), but a subspace-level understanding of it is mostly lacking (but see Lieberum et al. (2023)). How do the features in the keys and queries of attention heads combine to produce the attention scores? Which feature pairs are most important for the head's behavior?
- **Head composition**: If we are to understand a circuit on the subspace level, we need to develop a subspace-level account of how the outputs of one attention head compose with the queries, keys and values of a downstream head in the circuit. Each head adds its output to the residual stream, and downstream heads' query/key/value matrices read from the residual stream. We can thus examine the contribution, or *direct effect*, of a head's output to another head's queries/keys/values. We can decompose this direct effect in terms of the features of the source head to calculate contributions of each feature to the direct effect.

Implementation details for these analyses follow.

Attention scores. Given feature dictionary reconstructions for the keys and queries of an attention head at certain positions

$$\mathbf{k} \approx \sum_{i \in I} \mathbf{u}_{a_i(\cdot)=a_i(p)}, \quad \mathbf{q} \approx \sum_{i \in I} \mathbf{v}_{a_i(\cdot)=a_i(p)}$$

we can decompose the attention scores as a sum of pairwise dot products between the dictionary features

$$\mathbf{q}^T\mathbf{k}/\sqrt{d_{head}} \approx \sum_{i,j \in I} \mathbf{v}^T_{a_i(\cdot)=a_i(p)} \mathbf{u}_{a_j(\cdot)=a_j(p)} / \sqrt{d_{head}}$$

where d_{head} is the dimension of the attention head. This allows us to examine which feature combinations are most important for the head's attention according to the learned dictionaries. Variants of this decomposition can also be applied to e.g. the difference in attention scores at two different token positions.

Head composition. Following the terminology and results from Elhage et al. (2021), the residual stream $\mathbf{r}_{l,t}$ of a transformer at a given layer l and token position t is the sum of the input embedding and the outputs of all earlier MLP and attention layers at this position. The residual stream is in turn the input to the next attention layer; so, for example, we can write the query vector for the h-th head at layer l and token t as

$$\mathbf{q}_{l,t,h} = W^Q_{l,h}\,\text{LayerNorm}\,(\mathbf{r}_{l,t}) = W^Q_{l,h}\,\text{LayerNorm}\left(\bar{\mathbf{r}}_{l,t} + W^O_{l',h'}\mathbf{z}_{l',t,h'}\right)$$

where $\mathbf{z}_{l',t,h'}$ is the attention-weighted sum of values of the h'-th head at layer $l' < l$ and token t, $\bar{\mathbf{r}}_{l,t}$ is the remainder of the residual stream after removing the contribution of this head, and LayerNorm is the model's layer normalization operation (Ba et al., 2016) before the attention block in layer l. By treating the layer normalization as an approximately

linear operation (taking the scale from an average over the dataset[15]), we can derive an approximation of the *(counterfactual) direct effect* of the output of the h'-th head at layer l' and token t on the query vector of the h-th head at layer l and token t:

$$\mathbf{q}_{l,t,h} \approx W^Q_{l,h}\left(\gamma_l \odot \frac{\mathbf{r}_{l,t} - \mu_{l,t}}{\sqrt{\widehat{\sigma}_l^2 + \epsilon}} + \beta_l\right)$$

where γ_l, β_l are the learned scale and shift parameters of the LN operation, $\mu_{l,t}$ is the average of the vector $\mathbf{r}_{l,t}$ over its coordinates, and $\widehat{\sigma}_l$ is an average over the dataset of the standard deviation of the residual stream at this position. Alternatively, we can use the exact layernorm scale from the forward pass over a large sample to compute the statistics of the exact direct effect over observed data.

With either approach, we obtain a decomposition

$$\mathbf{q}_{l,t,h} \approx \sum_{l'<l,h'} \mathbf{u}_{t,(l',h')\to(l,h)} + \bar{r}_{l,t}$$

of direct contributions from the outputs of earlier heads at this position, plus some residual terms $\bar{r}_{l,t}$ (which are the contributions of all previous MLP layers and the input embedding to the query vector). We can then further decompose $\mathbf{u}_{t,(l',h')\to(l,h)}$ by replacing it with its reconstruction from our feature dictionary.

For either way to treat the layer normalization, we can use the learned feature dictionaries for the outputs, keys, queries and values of attention heads in a number of ways to decompose the direct effect further:

- **feature attribution**: fixing the head (l, h), we can vary the head (l', h') and break down the direct effects (projected on the query vector) by feature.
- **feature composition**: we can expand the direct effect's projection on the query vector as a sum of pairwise dot products between the dictionary features, similar to the attention decomposition.

Results. An interesting location to examine is the attention of the name mover heads from END to the IO and S1 positions, where (according to the analysis in Wang et al. (2023)) the signal from the S-Inhibition heads effectively removes the **S** name from the attention of these heads.

We show the results for the head L10H0 in Figure 6. Crucially, we observe that most interactions are tightly clustered around zero, which suggests that these feature dictionaries provide a sparse and interpretable account of the attention mechanism. The only significantly nonzero interactions are (1) between the **S** features in the query and the key at the S1 position; (2) between the **Pos** features in the query and the key at both positions; and (3) between the **IO** features in the query and the key at the IO position. The first two interactions are expected given the findings of Wang et al. (2023). More interesting is the third interaction, which is negative, suggesting that the L10H0 head inhibits both the **S** and **IO** name tokens, and effectively relies only on the **Pos** attribute to distinguish between the two names. Notably, this is in contrast with the other two name movers L9H6 and L9H9 (for which analogous plots are shown in Figure 14 and 15), where the inhibition of the **IO** attribute is absent. We found that using other methods to compute feature dictionaries result in less sparse and interpretable patterns.

We further investigated the queries of the L10H0 head, by looking at which features from upstream head outputs at the END token have a large direct effect on these queries. Following the methodology detailed in Appendix A.7, we plot the direct effect from the outputs of the S-Inhibition heads, as well as the two name mover heads L9H6 and L9H9 in layer 9 in Figure 7. We find that the S-Inhibition heads' **IO** features have no significant contribution to the queries, but the **IO** features from the two name mover heads in layer 9 have a significant

[15]This is justified by the empirical observation that the layer normalization scales across the dataset are well concentrated around their mean.

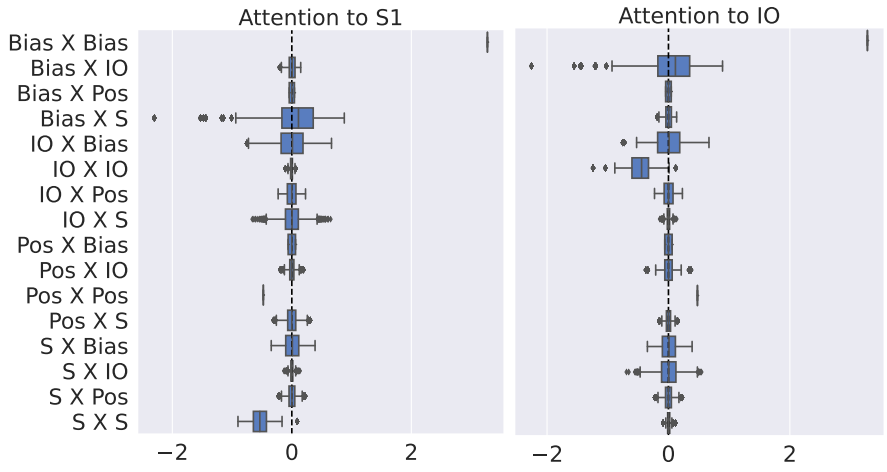

Figure 6: Decomposing the attention scores of the name mover head L10H0 from END to the S1 (left) and IO (right) positions. The y-axis ranges over the combinations of features from the query (first element) and the key (second element). The boxplots show the distribution of dot products between the corresponding feature vectors. The interaction between the bias terms (i.e., the means of the respective queries/keys) provides a sense of the scale of the effects.

direct effect (aligned with the overall centered query vector). This suggests that, having already computed a representation of the **IO** attribute, these heads transmit it to the next layer, where it gets picked up by the L10H0 head's query. Conversely, the S-Inhibition heads contribute significantly with their **Pos** and **S** features, whereas the name mover heads in layer 9 do not.

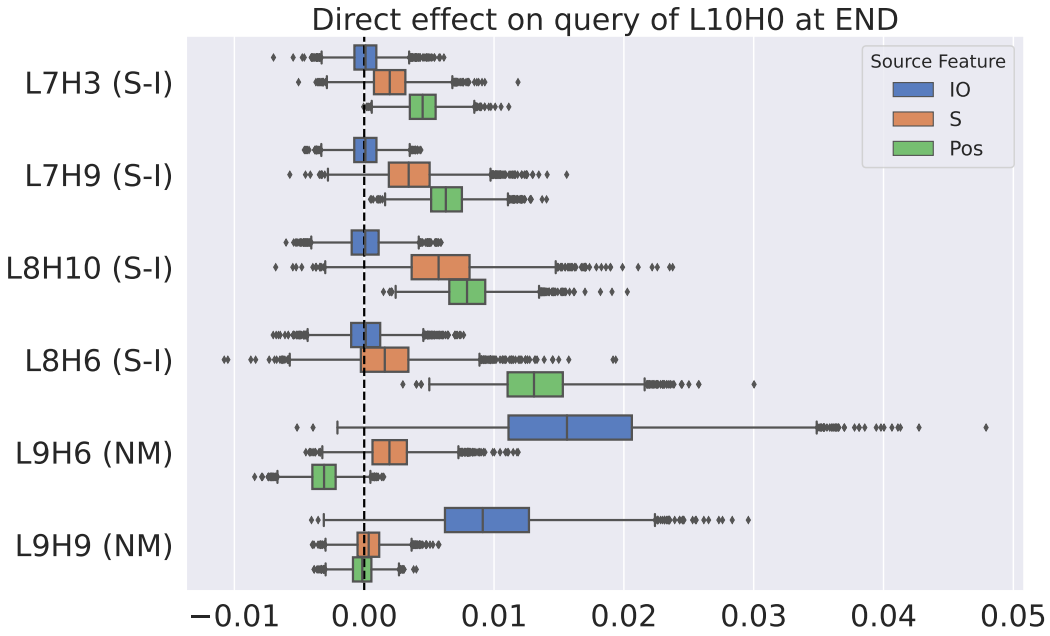

Figure 7: Direct effect of supervised features in the output of S-Inhibition heads, and Name Mover heads in layer 9, on the queries of the L10H0 name mover head at the END token.

A.8 Alternative parametrizations for the IOI task

We mostly experimented with two possible parametrizations of prompts via attributes:

- **independent parametrization**: we use the three independently varying attributes – **S**, **IO** and **Pos** – to describe each prompt. This is the parametrization used in the main text.
- **coupled parametrization**: we couple position with name, and use the two attributes (**S**, **Pos**) and (**IO**, **Pos**) to describe each prompt. This parametrization is more expressive than the independent one, as it allows for different features for the same name depending on whether it comes first or second in the sentence. At the same time, the coupled parametrization can express the independent one as a special case (Appendix A.9).

We find that these parametrizations arrive at highly similar activation reconstructions $\widehat{\mathbf{a}}$. In fact, we find an even stronger property: the coupled parametrization essentially simulates the features in the independent one; details are given in Appendix A.9.1.

Finally, we note that the fact that we find parametrizations that result in good approximation is not trivial. Not every 'natural-seeming' parametrization will lead to a good approximation of model behavior; we show an example of this with a 'names' parametrization in Appendix Figure 13, where we instead use an attribute for the 1st, 2nd and 3rd name in the sentence.

What about other parametrizations? Note that there are combinatorially many possible ways to pick attributes to disentangle the activations into, and a priori any specific choice is arbitrary. We justify our choice of parametrizations in several ways: (1) they pass our tests for model approximation, control and interpretability given later in this section; (2) they correspond to the internal states of the IOI circuit identified in Wang et al. (2023); (3) we experimented and/or considered other parametrizations, but found they either perform the same or worse on our tests. In Appendix A.10, we provide a more detailed discussion of different possible parametrizations in the IOI task and their relative strengths and weaknesses.

A.9 Comparing the coupled and independent parametrizations

A.9.1 The coupled parametrization captures the independent one

Idealized model. We first note that the coupled parametrization can express all reconstructions expressible by the independent parametrization. Suppose we have an IOI distribution using a set of available names S_{names}, and let $\mathbf{pos}_{ABB}, \mathbf{pos}_{BAB}, \{\mathbf{io}_a\}_{a\in S_{names}}, \{\mathbf{s}_a\}_{a\in S_{names}}$ be feature dictionaries for the independent parametrization at some model activation. Then, we can define the following feature dictionaries for the coupled parametrization:

$$\mathbf{io}_{a,ABB} = \mathbf{io}_a + \frac{1}{2}\mathbf{pos}_{ABB}, \quad \mathbf{io}_{a,BAB} = \mathbf{io}_a + \frac{1}{2}\mathbf{pos}_{BAB},$$
$$\mathbf{s}_{a,ABB} = \mathbf{s}_a + \frac{1}{2}\mathbf{pos}_{ABB}, \quad \mathbf{s}_{a,BAB} = \mathbf{s}_a + \frac{1}{2}\mathbf{pos}_{BAB}$$

Then for a prompt p of the form ABB (the BAB case is analogous), with the **IO** name being a and the **S** name being b, we have that the reconstruction of an activation $\mathbf{a}$ using the independent parametrization is

$$\widehat{\mathbf{a}}_{independent} = \mathbf{io}_a + \mathbf{s}_b + \mathbf{pos}_{ABB}$$

and the reconstruction using our coupled parametrization is

$$\begin{aligned}\widehat{\mathbf{a}}_{coupled} &= \mathbf{io}_{a,ABB} + \mathbf{s}_{b,ABB} = \mathbf{io}_a + \mathbf{s}_b + \frac{1}{2}\mathbf{pos}_{ABB} + \frac{1}{2}\mathbf{pos}_{ABB} \\ &= \mathbf{io}_a + \mathbf{s}_b + \mathbf{pos}_{ABB} = \widehat{\mathbf{a}}_{independent}\end{aligned}$$

Empirical evaluation. We evaluated whether this occurs empirically with the MSE features for the two parametrizations. First, we plot the fraction of variance explained by the reconstructions using the independent parametrization in the reconstructions using the coupled parametrization. We find very high agreement (Figure 8); results in the other direction are almost identical and are not shown here for brevity. Next, we check if the coupled parametrization essentially simulates the independent one as described analytically above. We do this by measuring the cosine similarity between the vector $\mathbf{pos}_{ABB} - \mathbf{pos}_{BAB}$ from the independent parametrization and vectors $\mathbf{io}_{a,ABB} - \mathbf{io}_{a,BAB}$ and $\mathbf{s}_{a,ABB} - \mathbf{s}_{a,BAB}$ from the coupled parametrization. In our idealized simulation of the independent parametrization using the coupled one, these values would be exactly 1 for all names. We find that in all circuit locations that represent both **IO** and **Pos**, the similarities w.r.t the **IO** differences are significant; similarly, in all circuit locations that represent both **S** and **Pos**, the similarities w.r.t the **S** differences are significant (Figure 9). For reference, in a space of this dimensionality (64), the expected magnitude of the cosine similarity between two random vectors is $1/8$.

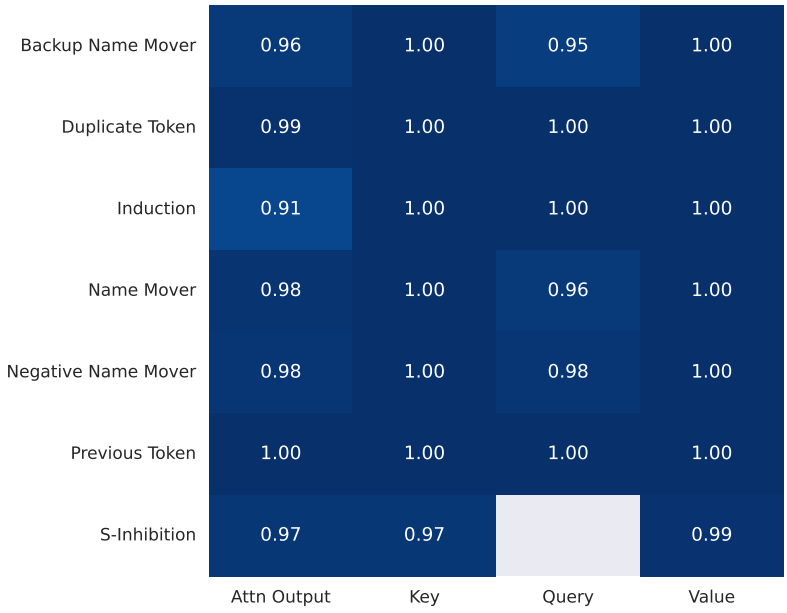

Figure 8: Variance explained by the reconstructions using the independent parametrization, with respect to the reconstructions using the coupled parametrization, averaged over combinations of class of heads in the circuit and activation types.

A.9.2 Other parametrizations expressible by the coupled parametrization.

Consider the parametrization with attributes (**S**, **Pos**) and **IO**. Again, let $\{\mathbf{io}_a\}_{a \in S_{names}}, \{\mathbf{s}_{a,ABB}\}_{a \in S_{names}}, \{\mathbf{s}_{a,BAB}\}_{a \in S_{names}}$ be feature dictionaries for this parametrization at some model activation. Then, we can define the following feature dictionaries for the coupled parametrization:

$$\mathbf{io}'_{a,ABB} = \mathbf{io}'_{a,BAB} = \mathbf{io}_a, \quad \mathbf{s}'_{a,ABB} = \mathbf{s}_{a,ABB}, \quad \mathbf{s}'_{a,BAB} = \mathbf{s}_{a,BAB}$$

Then for a prompt p of the form ABB (the BAB case is analogous), with the **IO** name being a and the **S** name being b, we have that the reconstruction of an activation $\mathbf{a}$ using the (**S**, **Pos**) + **IO** parametrization is

$$\widehat{\mathbf{a}}_{(\mathbf{S},\mathbf{Pos})+\mathbf{IO}} = \mathbf{io}_a + \mathbf{s}_{b,ABB} = \mathbf{io}'_{a,ABB} + \mathbf{s}'_{b,ABB} = \widehat{\mathbf{a}}_{coupled}$$

and so the coupled parametrization can express all reconstructions expressible by the (**S**, **Pos**) + **IO** parametrization. Analogously, it can express all reconstructions expressible by the (**IO**, **Pos**) + **S** parametrization.

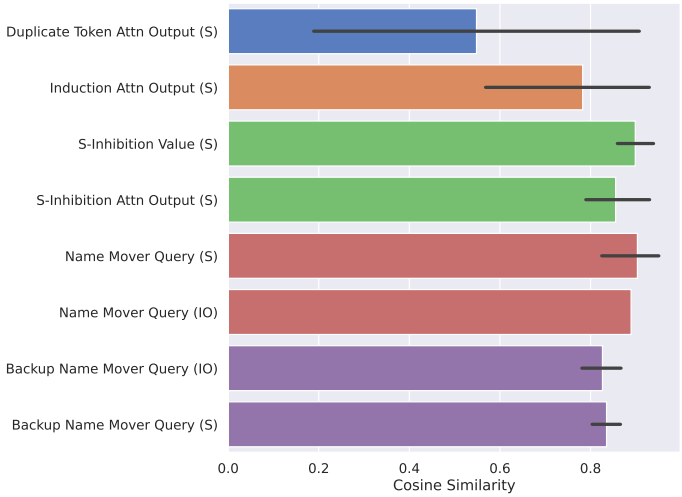

Figure 9: Cosine similarity between the vector $\mathbf{pos}_{ABB} - \mathbf{pos}_{BAB}$ from the independent parametrization and vectors $\mathbf{io}_{a,ABB} - \mathbf{io}_{a,BAB}$ and $\mathbf{s}_{a,ABB} - \mathbf{s}_{a,BAB}$ from the coupled parametrization, averaged over several classes of circuit locations. When evaluating similarity for the $\mathbf{io}_{a,ABB} - \mathbf{io}_{BAB}$ vectors, we only include locations where both the **IO** and **Pos** attributes are represented; and similarly for the **s**-vectors.

A.9.3 Editing methodology with the coupled parametrization.

For each activation, we may choose to edit one or several of the **IO**, **S** and **Pos** properties of the prompt. With the independent parametrization, this is straightforward, since the attributes match these properties. With the coupled parametrization, suppose we are given a prompt of the form ABB (the BAB case is analosous) with the **IO** name being a and the **S** name being b. Given an activation $\mathbf{a}$ with corresponding reconstruction under the coupled parametrization

$$\widehat{\mathbf{a}} = \mathbf{io}_{a,ABB} + \mathbf{s}_{b,BAB}$$

we can perform edits as follows:

- to change the **IO** name from a to a': $\mathbf{a}_{edit} = \mathbf{a} - \mathbf{io}_{a,ABB} + \mathbf{io}_{a',ABB}$
- to change the **S** name from b to b': $\mathbf{a}_{edit} = \mathbf{a} - \mathbf{s}_{b,ABB} + \mathbf{s}_{b',ABB}$
- to change the **Pos** property from ABB to BAB: $\mathbf{a}_{edit} = \mathbf{a} - \mathbf{io}_{a,ABB} - \mathbf{s}_{b,ABB} + \mathbf{io}_{a,BAB} + \mathbf{s}_{b,BAB}$

A.10 On Possible Feature Dictionaries for the IOI Task

In this section, we compare the properties of several *a priori* possible ways in which the activations of the IOI circuit could be disentangled via sparse feature dictionaries. The main goal is to illustrate that different feature dictionaries can have similar usefulness in terms of model control and interpretability, even if they fail natural tests that directly look for similar features in the two dictionaries. This motivates evaluations that are agnostic to the specific features in a dictionary, as long as the features can parsimoniously disentangle the model's internal computations.

The independent parametrization from the main text. It is worth explicitly describing the properties of the supervised feature decomposition we constructed in Section **??**, which uses the **IO**, **S** and **Pos** attributes to describe the prompts; it serves as an idealized example against which to compare other possible decompositions. In this decomposition, we can approximate an activation $\mathbf{a}$ for a prompt p where the **IO** name is a and the **S** name is b, with the **IO** name appearing first, as follows:

$$\mathbf{a} \approx \mathbf{io}_a + \mathbf{s}_b + \mathbf{pos}_{ABB}$$

where the vector $\mathbf{io}_a$ is the feature for the **IO** name a, the vector $\mathbf{s}_b$ is the feature for the **S** name b, and the vector $\mathbf{pos}_{ABB}$ is the feature for the **Pos** attribute when the **IO** name appears first in the sentence (and analogously for $\mathbf{pos}_{BAB}$).

Imagine now that we are given an 'unlabeled' feature dictionary that corresponds to this decomposition (i.e., we don't know which attribute each feature corresponds to). We want to evaluate the usefulness of this decomposition for model control and interpretability relative to the 'human-legible' attributes **IO**, **S** and **Pos**. This will be tautologically successful:

- the reconstructions are a faithful and complete representation of the model's internal computations;
- the dictionary can (by definition) express edits to the **IO**, **S** and **Pos** attributes very efficiently, as we only need to change a single feature vector to change the corresponding attribute's value.
- the features are fairly interpretable: we can understand the meaning of each feature in terms of the attribute it represents.
- moreover, using metrics such as recall and precision (following the evaluation methodology of Bricken et al. (2023)) will readily surface the features that are most important for each attribute.

Using per-gender vectors to describe names. Another possible decomposition is

$$\mathbf{a} \approx \mathbf{io}_a + \mathbf{io\_gender}_{g(a)} + \mathbf{s}_b + \mathbf{s\_gender}_{g(b)} + \mathbf{pos}_{ABB}$$

where $g : \mathbf{Names} \to \{M, F\}$ is some labeling function that roughly classifies names according to how they are typically gendered[16]. Here, we hypothesize that the model may use features of high norm that sort names into genders (which may be useful to the model for various reasons), and then add a small-norm per-name correction to obtain a name-specific representation. In particular we imagine that $\mathbf{io}_a + \mathbf{io\_gender}_{g(a)}$ in this representation would correspond to $\mathbf{io}_a$ in the supervised decomposition, and similarly for the **S** name.

- This decomposition is also fairly sparse and interpretable, and it can express edits almost as parsimoniously as the supervised decomposition (we only need to change *two* feature vectors to edit a name, and one to edit **Pos**).
- Moreover, metrics such as recall and precision will pick up on the per-name features;
- If we compare the prompts for which a feature activates for this dictionary and our 'independent parametrization' feature dictionary (discussed above), we will easily discover the per-name features that correspond to the **IO** and **S** names.
- **However**, if we instead directly use cosine similarity to the supervised features $\mathbf{io}_a, \mathbf{s}_b$ as a metric, we may be misled, because if the per-gender features have sufficiently higher norm than the per-name corrections, the cosine similarity may be low.

Features for small, overlapping subsets of names. Going further, we can imagine a decomposition where we have features that correspond to pairs of names, such that each name is in exactly two pairs (this can be achieved by partitioning all names into pairs along a cycle). We can express a name as a sum of the features for the two pairs it is in, with some superposition. Note that more sophisticated constructions with more features per name / more names per feature are possible by e.g. picking subsets at random or using expander graphs (Hoory et al., 2006), as they will 'spread out' the superposition more evenly.

- This decomposition is *somewhat* sparse and interpretable, and can likely be used for feature editing in a reasonable way, as long as the sets of features associated with each name are not too large. Even though we would need to change several feature vectors to edit a name, there should also be a fair amount of disentanglement so that we don't also need to throw away all the features active in an example to change a single attribute.

[16] We experimented with this decomposition in our supervised framework, but did not find it to confer additional benefits for the purposes of our tests.

- **However**, comparing our supervised decomposition against this one using geometric metrics such as cosine similarity may be misleading, because while a sum of a few feature vectors associated with the same name may point in the same direction as our supervised feature, any individual feature may not.
- **Furthermore**, it also has significantly reduced precision for the features, because each of the few features associated with a name will also activate for several other names. This can make directly looking for features whose activation patterns resemble the ones in our supervised decomposition misleading.

Our experiments suggest that both task-specific and full-distribution SAEs trained on IOI circuit activations learn a decomposition resembling this abstract construction more than any other decomposition discussed here.

Overfitting dictionaries. Finally, a worst-case decomposition would be to have a single feature for each possible set of values of the **S**, **IO** and **Pos** attributes.

- This decomposition is not interpretable, and it is not editable in any non-trivial way: to change a single attribute, the entire decomposition must be replaced;
- Features of this form will have maximum precision, but very low recall for the attributes.

A.11 Details for training Sparse Autoencoders

Task SAEs. We followed the methodology of Bricken et al. (2023), with the exception that our neuron re-initialization method is not as sophisticated as theirs: we simply re-initialize the encoder bias, encoder weights and decoder weights for the dead neurons every 500 training epochs.

Training SAEs on the IOI distribution alone allows us to do a more extensive hyperparameter search. Importantly, we normalized SAE inputs across attention heads so that activations have an ℓ_2 norm of 1 on average in order to make it easier for the same set of hyperparameters to work well across different heads. In our main experiments, we use SAEs that were trained with a $16\times$ hidden expansion factor, (effective) ℓ_1 regularization coefficient in $(0.01, 0.05, 0.1, 0.3)$, batch size of 1024, and learning rate of 0.001.

We evaluated two key test-set metrics across training epochs: the average number of active features per example (i.e. the average ℓ_0 norm of activations), and the fraction of the logit difference recovered when using the SAE reconstructions at the given model location instead of the original activations, scaled against a mean-ablation baseline (which is more stringent than the zero-ablation baseline employed in most other work). We chose the regularization coefficient and training checkpoint for each node that provided a good trade-off between these two metrics; in particular, for almost all nodes, we recover logit difference to within 20% with respect to the mean-ablation baseline, and there are < 25 active features per example. We provide the results of this sweep in Figure 16. While most SAEs seemed adequate, some still have poor approximation as measured by the reconstructed logit difference.

We use a training set of 20,000 examples and an evaluation set of 8,000 examples (for the purposes of automatic interpretability scoring, we need a large enough evaluation sample so that each property in the distribution appears a significant number of times). Since our training regime is significantly distinct from that of Bricken et al. (2023) (we use a much smaller dataset), we first experimented extensively with different hyperparameters, focusing on training SAEs on the queries of the name mover heads. We observed that the most important hyperparameters are the dictionary size and the effective ℓ_1 regularization coefficient. We found that the batch size did not influence the eventual quality of the learned features, only the speed of convergence, and that a learning rate of 10^{-3} (as in Bricken et al. (2023)) was a good choice throughout. The runs reported here used a dictionary size of 1024 (a $16\times$ increase over the dimensionality of attention head activations in GPT-2 Small), an effective ℓ_1 regularization between 0.05 and 0.3, and a batch size of 1024.

We normalized activations across the circuit to make it easier for the same range of hyperparameters to give good results, and ran a sweep over ℓ_1 regularization coefficients in $(0.01, 0.05, 0.1, 0.3)$.

Full-distribution SAEs. We trained full-distribution SAEs on every IOI component using OPENWEBTEXT as training data. We mostly followed the method outlined in Bricken et al. (2023). We added a standardization procedure to be able to train SAEs on components with different activation scales using the same l1-coefficient. Before training, we calculated the mean and the mean l2-norm over 10 million activations. These values were then frozen and used to standardize all activations as a preprocessing step and to rescale the SAE reconstructions to match the original scale as a post-processing step. We generated the training dataset by extracting a buffer of 10 million activation vectors from the shuffled OPENWEBTEXT dataset at a time with a maximal context window of 512 tokens. We then trained the SAEs for 250 million activation vectors and resampled dead neurons after 50000 steps (around 100 million activations) as outlined in Bricken et al. (2023). We used a batch size of 2048 and 8192 features per SAE. Post-training, we excluded dead and ultra-low frequency neurons that we define as neurons who activate less than once per million activations. The amount of dead neurons varies across SAEs between 20 and 90%. We plot the fraction of dead neurons versus ℓ_0 loss in Figure 18, and the loss recovered versus ℓ_0 loss in Figure 19.

We used an ℓ_1 coefficient of 0.006 initially for all SAEs, and retrained SAEs with a different ℓ_1 coefficient for crosssections whose SAE metrics were undesired (e.g. very low ℓ_0 / bad reconstruction or very high ℓ_0). For the name mover outputs, we used 0.025, and for S-Inhibition keys we used 0.005. The test ℓ_0 and loss-recovered metrics were calculated on 81920 unseen activation vectors.

We trained fewer SAEs on the full pre-training distribution compared to the IOI distribution, as the computational cost is higher. We observe that SAEs with a lower number of active features per example generally perform better for IOI-related tests, even if their other metrics (such as loss recovered on OPENWEBTEXT) are worse.

SAE training loss metrics. The most important loss metrics to track during SAE training are the ℓ_0 loss (measuring the average number of active features per activation) and the language model loss recovered when using the learned features to reconstruct the model's logits (Bricken et al., 2023). To turn the loss recovered into a meaningful quantity, it is rescaled against a baseline; both zero ablation and mean ablation have been used for this purpose in the literature (Bricken et al., 2023; Kissane et al., 2024). In this work, we used mean ablation, as it is a more strict test of the quality of approximation.

A.12 Additional notes on methodology for SAE interpretability

Interpretations considered. Let **Names** be the set of names in our IOI dataset. We consider the following binary predicates over prompts as possible interpretations for SAE features in the activations at the S2 and END tokens:

- **IO is 2nd name**: the **Pos** attribute having value corresponding to BAB-type prompts;
- **IO is 1st name**: the **Pos** attribute having value corresponding to ABB-type prompts;
- **S is <name>**: the **S** attribute has a certain value in **Names**;
- **S is <name> and at 1st position**: same as above, but also the **S** name is at 1st position in the prompt (i.e., this is a BAB-type prompt);
- **S is <name> and at 2nd position**: same as above, but for ABB-type prompts;
- **IO is <name>**: the **IO** attribute has a certain value in **Names**;
- **IO is <name> and at 1st position**: same as above, but also the **IO** name is at 1st position in the prompt (i.e., this is a ABB-type prompt);
- **IO is <name> and at 2nd position**: same as above, but for BAB-type prompts;

- **S is male**: the **S** name is labeled as a male name under our labeling of **Names** provided by GPT-4;
- **S is female**: same as above for female names;
- **<name> is in sentence**: a certain name in **Names**;
- **<name> is at 1st position**: same as above, but the name is the first name in the sentence;
- **<name> is at 2nd position**: same as above, but the name is the second name in the sentence;

The next several interpretations are only defined for the keys and values of the name mover heads. We collect together activations for the keys according to their role in the IOI circuit as opposed to absolute position: we group together all activations at the IO token position (these are the **IO** keys/values), even though they come from different absolute positions across IOI prompts, because in ABB prompts the **IO** name comes first, while in BAB prompts it comes second. The same applies for gathering the **S** keys/values.

The key/value activations described above have not yet 'seen' the repeated name in the sentence, so there is no meaningful concept of **IO** and **S** for them. Instead, the only potentially task-relevant information contained in them is about the name(s) seen so far in the sentence, and the position (1st name or 2nd name) where the activation is taken from. Accordingly, the applicable interpretations for features contained in these activations are different:

- **current token is <name>**: the token from which the activations are taken holds a certain value in **Names**.
- **token is <name> and at 1st position**: the activation was taken from a token with a certain value in **Names**, and in addition it comes from the first name in the sentence;
- **token is <name> and at 2nd position**: same as above, but activation is from the second name in the sentence;
- **current token is at 1st position**: the activation is from the first name in the sentence;
- **current token is at 2nd position**: same as above, but from second name;
- **current token is female**: the token the activation is from is female under our labeling of **Names**.

Unions of interpretations. In addition, for each type of predicate that has a free parameter in **Names**, we considered unions of up to 30 such predicates (recall that we have a total of 216 names in our dataset). We ordered the individual predicates according to their F_1 score, and chose the union of the first $k \leq 30$ predicates with the highest F_1 score as a possible interpretation.

Sufficiency/necessity of interpretable features. We take the interpretations of the features described above and their respective F_1 scores, and for each threshold $t \in [0, 1]$ over F_1 scores consider two interventions:

- **sufficiency**: we subtract from the respective activation all active features with F_1 score $< t$;
- **necessity**: we subtract from the respective activation all active features with F_1 score $\geq t$;

Interpretation-aware sparse control. The goal of this experiment is to evaluate the usefulness of our feature interpretations for editing the attributes **IO**, **S** and **Pos** we have chosen to describe prompts with. In particular, this is a different goal from our exploratory interpretability experiment, where we were concerned with assigning interpretations to each feature in a way agnostic to whether the features correspond 1-to-1 with the attributes.

Correspondingly, for this experiment we use the F_1 score with respect to each attribute as a guide for the potential relevance of a feature to this attribute. Specifically, recall from Subsection 3.4 that we can assign an F_1 score to each combination of a feature and a possible value of one of the attributes **IO**, **S** and **Pos**.

To perform editing, let our SAE have a dictionary of decoder vectors $\{\mathbf{u}_j\}_{j=1}^m$, and the original and counterfactual prompts p_s, p_t have respective activations $\mathbf{a}_s, \mathbf{a}_t$ with reconstructions

$$\widehat{\mathbf{a}}_s = \sum_{i \in S} \alpha_i \mathbf{u}_i + \mathbf{b}_{dec}, \quad \widehat{\mathbf{a}}_t = \sum_{i \in T} \beta_i \mathbf{u}_i + \mathbf{b}_{dec}$$

for $S, T \subset \{1, \ldots, m\}$ and $\alpha_i, \beta_i > 0$. Suppose the original and counterfactual prompts differ only in the value of the attribute a we wish to edit, with $a(p_s) = v_s, a(p_t) = v_t$. Using a test set, we estimate the F_1 score of each possible value v of the attribute a, and for each v we order the SAE features in decreasing order of the F_1 score, obtaining a list top-features $(v) = [j_1, j_2, \ldots]$ with $1 \leq j_k \leq m$.

Then, fixing some value of k, we compute the edited activation as

$$\mathbf{a}_{edited} = \mathbf{a}_s - \sum_{j \in (S \cap \text{top-features}(v_s))[:k]} \alpha_j \mathbf{u}_j + \sum_{j \in (T \cap \text{top-features}(v_t))[:k]} \beta_j \mathbf{u}_j.$$

Note that this is a somewhat *less expressive* intervention than our per-prompt agnostic feature editing, because here we require that there is some fixed ordering from which we choose features, instead of being free to pick features for each prompt independently.

A.13 Additional observations on feature interpretations

Interpretable features agree with head roles identified in the IOI circuit by Wang et al. (2023). For example, duplicate token heads attend to a previous occurrence of the previous token and write information about this to the residual stream. Consistent with this, we found that SAEs trained on them contain features that indicate the duplicated name, the subject in case of IOI. This information is then used by the induction heads that determine whether the subject is the first or second name. The subject position is subsequently copied to the END position by the S-Inhibition heads, where it will query the name movers to *not* attend to the subject, and to copy and predict the IO name. Indeed, we find features of the outputs of induction heads, in the outputs and values of the S-I heads, and in the queries of the name movers that inform about the position of the indirect object. Lastly, the name movers attend to the IO position and copy the name to predict it. As anticipated, the name mover values and outputs contain features that specify the concrete IO name. In summary, the type of features detected informs well about the function of the head on a certain task.

New insights from the feature dictionaries.

- The first layer DT-heads almost exclusively contain **S** features but the third layer duplicate token head also has positional features, suggesting more sophisticated text processing happening there.
- Induction head encompass different positional features. L6H9 features inform about what name is at the second position, while L5H5 and L5H8 activate when the **IO** is at the first position. L5H9 is comprised of different features including positional features that don't inform about the role (**IO** vs **S**) of the name.
- L7H3 is the only head that contains a significant amount of gender features.
- S-Inhibition heads include primarily features that are a combination of names and **S**, while the name mover queries seem to only contain **Pos** features.
- The keys and values of name movers both inform about the token at the current position, but there is an important difference in the type of features: while the values primarily contain features that contain the name (that later gets moved to the END position), the keys consist of positional features and combinations of position and name. This hints at an important mechanism where the name mover query contains positional information about the **S** name that gets matched with the corresponding key, effectively shifting the attention towards the IO token such that the value with the **IO** gets copied to the END position.

Editing interpretable features

While the feature descriptions generated through our automatic scoring predict well when a feature is active, it is still unclear whether they also have an interpretable causal role, i.e. whether activating or deactivating a certain feature leads to a change in output logits that would be expected from the feature's description. To test this, we propose two experiments to judge the faithfulness and completeness of our interpreted features that involve patching activations from a counterfactual prompt and calculating the effect on the model's output:

- **Estimating Faithfulness:** To estimate faithfulness, we construct SAE activations where we fix features with a test F1-score smaller than a threshold and patch activations of features with a high F1-score from the counterfactual prompt. We then calculate reconstructions of this new SAE activation vector, patch cross-sections, and record whether the model successfully predicts the correct counterfactual name.
- **Estimating Completeness:** To estimate completeness, we propose a similar experiment where we fix features with a high test F1-score and patch all remaining features. This intervention should not change the output logits if our features are complete.

We run this experiment on cross-sections of name mover outputs and repeat this experiment for different thresholds. We observe that for a threshold F1-score of 0.6, the SAE features are both faithful and complete to a high degree. We observe that the faithfulness metric significantly decreases for higher F1-scores of 0.7 and 0.8 but remarkably, we also observe that only fixing features with a very high F1-score of > 0.8 while patching all other features from the counterfactual prompt is sufficient to keep the model predicting the base prompt's output.

Generalization of features to the full distribution. We sample prompts from OpenWebText and visualize prompts that highly activate a feature. We do this for the name mover queries and outputs. For the outputs, we calculate the direct-feature-attribution (DFA) metric that calculates per position how large the contribution of its values to activating the feature is. Thus, it informs what position led to the feature being activated.

We find that features mostly generalize. For example, we investigated a feature that activated if the IO name starts with the letter "E" and we found that on full distribution, this feature fires at tokens preceeding words starting with "E". DFA suggests that previous tokens starting with "E" activate this feature, and calculating the unembed $W_{dec}[j]W_O W_U$, denoted with "Positive Logits" and "Negative Logits" in Figure 12 shows that activating the feature increases logits for words starting with "E". Together, this hints at a general name moving mechanisms to predict words that previously occurred in the context that drives in-context learning where the head's QK-circuit drives attention to the position of the word to predict, and the OV circuit copies the token from the previous to the current position to predict it.

A.14 Feature occlusion details

Training an exhaustive set of SAEs. Focusing on the L10H0 queries, we found that our SAEs consistently find a single feature for almost all **IO** names, but fail to find a significant number of features for individual **S** names.

To investigate further whether this is a result of poor hyperparameter choices, we trained SAEs on the queries of L10H0 over a wide grid of hyperparameters, so that performance deteriorates/plateaus at the edges of the grid. We pushed the dictionary size, training dataset size, ℓ_1 regularization coefficient and number of training epochs significantly beyond the values used in our other experiments. Specifically, we used a training set of 100,000 examples (this is more datapoints than all $\sim 93,000$ possible combinations of **S**, **IO** and **Pos** in our data); we trained dictionaries of up to $128 * 64 = 8192$ features (our supervised feature dictionaries contain ~ 500 features); we varied the effective ℓ_1 regularization coefficient across two orders of magnitude; and we trained for up to $6,000$ epochs.

Results on the number of **IO**/**S** features with F_1 score > 0.5 are reported in Figure 4 (left). We observe that across hyperparameters, we often find as many **IO** features as there are names in our dataset; however, the number of **S** features is consistently low, never exceeding 22.

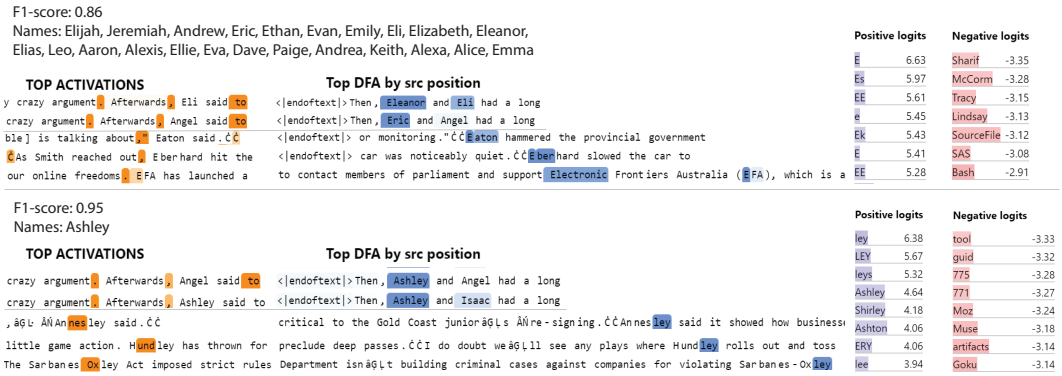

Figure 10: Two representative features discovered in the output name mover SAE L9H9 to illustrate the features behavior or webtext. Both features are IO name features, the upper one containing 23 names, the lower one only a single name. Left: Feature activation per position; Middle: Direct Feature Attribution (DFA) that tracks the position whose values contribute most to activating the given feature; Right: The output tokens whose logits get increased (positive) or decreased (negative) when the feature is active, calculated by $\mathbf{v}\mathbf{W}_O\mathbf{W}_U$ with $\mathbf{v}$ being the decoder weight vector of the feature of interest, $\mathbf{W}_O$ the output weight of the head and $\mathbf{W}_U$ the unembed

Magnitudes of the IO and S features. As a first step, we investigate the distribution of the norms of the features for **IO** and **S** across the names in the IOI dataset in the L10H0 queries in our supervised feature dictionaries from Section 4; results are shown in Figure 11 (left). We observe that the norms of the **IO** features are significantly (but not by much) higher than those of the **S** features.

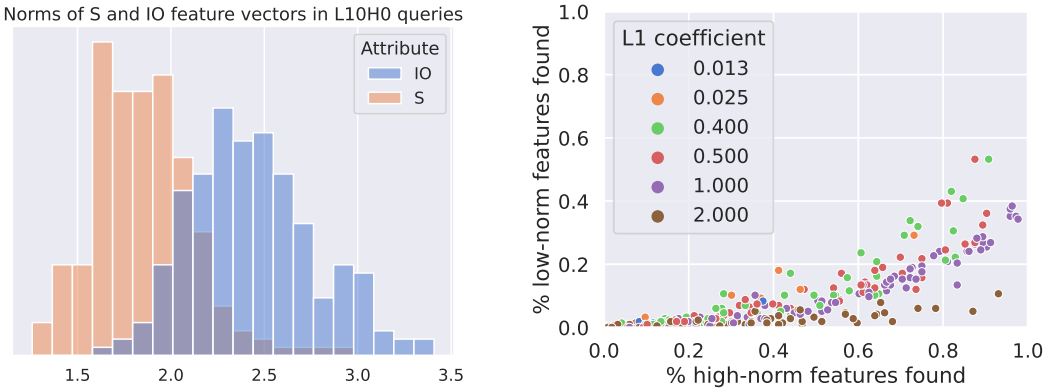

Figure 11: **Left**: distributions of the ℓ_2 norms of the feature vectors for the **IO** and **S** attributes from our supervised feature dictionary for the queries of the L10H0 name mover. **Right**: the results of the toy model experiment, where we investigate whether a disparity in feature magnitudes alone can lead to the occlusion phenomenon. The x-axis shows the fraction of high-magnitude ground-truth features for which we find an SAE feature with F_1 score > 0.9; the y-axis shows the same for the low-magnitude ground-truth features.

Surgically reducing the magnitude of IO features. To examine the role of feature magnitude in the occlusion phenomenon, we continuously reduce the magnitude of **IO** features. Namely, given our supervised feature dictionary with features $\{\mathbf{io}_a\}_{a\in\text{Names}}, \{\mathbf{s}_b\}_{b\in\text{Names}}, \{\mathbf{pos}_v\}_{v\in\{\text{ABB,BAB}\}}$ and an activation $\mathbf{a}(p)$ for a prompt p where the **IO** name is a, we construct a new activation

$$\mathbf{a}_\alpha(p) = \mathbf{a}(p) - \alpha\mathbf{io}_a$$

for $\alpha \in [0, 1]$. We find that, applying this intervention without any hyperparameter tuning (with a modest dictionary size of 1024, and ℓ_1 regularization coefficient of 0.2), increasing α from 0 to 1 gradually makes the number of **IO** features with F_1 score > 0.5 to decrease, while the number of **S** features with F_1 score > 0.5 increases; results are shown in Figure 4 (right).

Reproducing the occlusion phenomenon in a toy model. Finally, we wanted to know if a disparity in feature magnitudes alone could lead to the occlusion phenomenon. We constructed a simple toy model closely based on the empirical setup in the queries of the L10H0 head to test this hypothesis.

We form synthetic activations $\mathbf{a} = \mathbf{u}_i + \mathbf{v}_j$ for random pairs $i, j \in \{1, \ldots, |\text{Names}|\}$, where $\mathbf{u}_i, \mathbf{v}_j \in \mathbb{R}^{d_{head}}$ are sampled independently from a standard normal distribution centered at zero, and then $\mathbf{u}_i$ are rescaled so that their mean norm matches the mean norm of **IO** features in the L10H0 queries, and similarly for $\mathbf{v}_j$ and **S** features. We train SAEs on these activations over a wide grid of hyperparmeters: dictionary sizes in $(512, 1024, 2048)$, ℓ_1 regularization in $(0.0125, 0.025, 0.4, 0.5, 1.0, 2.0)$, batch size in $(256, 1024)$ and learning rate in $(0.001, 0.003, 0.0003)$. We trained for 1000 epochs, saving checkpoints in a geometric progression of epochs. Results for the number of high- and low-magnitude features with F_1 score ≥ 0.9 discovered are shown in Figure 11 (right); we observe that we easily find one SAE feature per each high-magnitude ground-truth feature, but it is more difficult to find an SAE feature for each low-magnitude feature.

However, we note that with lower F_1 thresholds, this effect is less pronounced and eventually disappears.

A.15 Feature over-splitting in a mixture of gaussians toy model

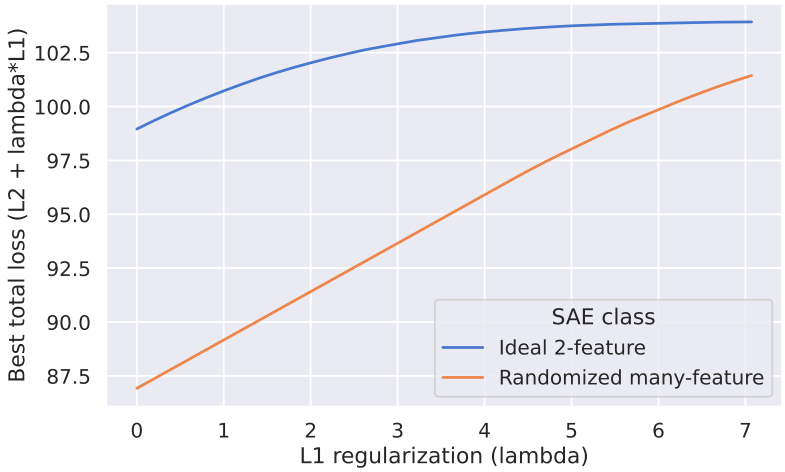

Figure 12: Main experiment for our toy model of feature oversplitting. The data distribution is a uniform mixture of two standard multivariate Gaussian random variables in 100 dimensions. **Blue**: the (approximate) best possible total loss (in the infinite data limit) achieved by a class of 'ideal' SAEs that use two features pointing towards the means of the two components of the mixture. **Orange**: an approximate upper bound on the best possible total loss achieved by an SAE with $m = 1,000$ hidden features (in the infinite data limit). The x-axis is the ℓ_1 regularization coefficient λ. The cutoff on the x-axis is chosen so that the idealized solution activates only for a vanishing fraction ($< 2\%$) of the examples in the mixture.

Our goal in this section is to demonstrate that there *exist* setups where an SAE with a large number of hidden features $m \gg 2$, when trained on a uniform mixture of two isotropic gaussian variables, will prefer a solution with $\gg 2$ features (as opposed to the 'ideal'

solution with only two features, one per component of the mixture), for *any* value of the ℓ_1 regularization coefficient λ and *any* amount of training data.

Setup. We consider a simple toy model where activations are distributed according to a uniform mixture $\mathcal{D}_{toy}$ of two isotropic gaussians $\pm\boldsymbol{\mu} + \mathcal{N}(0, \mathbf{I}_d)$ in $\mathbb{R}^d$ (i.e., we first flip a fair coin to determine the sign of $\boldsymbol{\mu}$, and then sample an extra additive term from $\mathcal{N}(0, \mathbf{I}_d)$). We sample an i.i.d. dataset from this mixture. We used $d = 100$, $\|\boldsymbol{\mu}\|_2 = 2$ in the experiments below; this choice guarantees that the two components of the mixture are separable with high ($> 95\%$) probability.

The 'ideal' solution with 2 features. We might hope that, with the right ℓ_1 regularization, an SAE trained on such a distribution will discover only two interpretable features, one for each component in the mixture, with the encoder/decoder vectors aligned with $\boldsymbol{\mu}$ and $-\boldsymbol{\mu}$ respectively, and each feature activating for (approximately) the examples in its corresponding component.

We find this is the case empirically when we train an SAE with only two hidden features on this toy distribution. Specifically, there is a range of λ values $1 \leq \lambda \leq 3$ where the SAE reliably approximately recovers this ideal solution, with the encoder bias controlling the trade-off between the two loss terms: when λ increases, the encoder bias changes so that fewer examples in each component of the mixture are activated (and the active examples have a lower ℓ_1 loss). Beyond this range, the SAE often fails to activate any feature on any examples ($\lambda > 3 + \varepsilon$), or activates both features on almost all examples ($\lambda < 1 - \varepsilon$).

Analyzing the ideal solution across ℓ_1 coefficients. To study the properties of the 'ideal' solution analytically, we make the following assumptions (borne out empirically with 2-feature SAEs) using only symmetries of the data distribution:

- the decoder vectors are $\pm\boldsymbol{\mu}$, normalized to have unit ℓ_2 norm (by symmetry of each component around its mean);
- the respective encoder vectors are $\pm k\boldsymbol{\mu}$ for some $k > 0$ (again by symmetry of each component around its mean);
- the decoder bias is zero (by symmetry of the mixture around zero);
- both encoder biases are set to $-\gamma$ for some $\gamma > 0$ (again by symmetry of the mixture around zero).

This leaves only two parameters to tune: the encoder bias γ and the encoder scale k. We can thus use the following strategy to analytically approximate the best loss of this class of solutions for a given λ:

- approximate the expected ℓ_1 and ℓ_2 losses over a fine grid of values for γ and k, for a large dataset of samples from the mixture;
- given a λ value, find the point in the grid that minimizes the total loss $\ell_2 + \lambda\ell_1$.

We implemented this using 10^5 samples, with a grid of 100 values for γ in $[0, 5]$ and 20 values for k in $[0, 2]$, over a grid of 100 values of λ in $[0, 20]$. We verify that the best values chosen for each λ are not on the edges of the grid; the resulting curve of best total loss values versus λ is shown in Figure **TODO** (blue), cut off at $\lambda \approx 7$, beyond which the selected SAE activates for $< 2\%$ of examples in the components of the mixture.

SAEs with $m \gg 2$ features prefer other solutions even with infinite data. Next, we want to show that with enough features, SAEs will prefer solutions different from the class of 2-feature solutions described above. How can we give an *empirical* argument that applies to *any* amount of training data and any λ? After all, a *trained* SAE is a function of the data it is trained on, so no experiment on datasets of bounded size can establish properties of SAEs trained on arbitrarily large datasets.

We get around this by defining a class of SAEs that is competitive with the class of ideal 2-feature SAEs *upfront*, independent of the training sample, by using a randomized construction that works w.h.p., and then estimating the expected loss of this SAE in the infinite

data limit empirically. To give evidence of our result for arbitrary λ, we consider a fine enough grid of λ values, and for each λ we construct an SAE that is competitive with the best ideal 2-feature SAE.

Our randomized SAE construction proceeds as follows:

- Sample m encoder vectors $W_{enc} \in \mathbb{R}^{m \times d}$ from $\beta * \mathcal{D}_{toy}$ (i.e. a version of $\mathcal{D}_{toy}$ scaled by β) where $\beta > 0$ is a hyperparameter that we will tune;
- Set the decoder vectors $W_{dec} \in \mathbb{R}^{d \times m}$ to be the same as the encoder vectors $W_{enc}^{\top}$, but normalized so that each column of W_{dec} has unit ℓ_2 norm;
- Set all encoder biases to $-\gamma$ (where $\gamma > 0$ is a hyperparameter that we will tune), and the decoder bias to $\mathbf{0} \in \mathbb{R}^d$.

We used $m = 1,000$, a grid of 100 values for β in $[0, 0.1]$ to search for the best β for each λ, and fixed $\gamma \approx 2.35$.

Empirical confirmation. Finally, we actually trained SAEs with many hidden features on $\mathcal{D}_{toy}$, and observed that these SAEs reliably learned solutions with many active features across λ values.

A.16 Additional details for Section 5

Feature weights are mostly in the interval $[0, 1]$. Recall that given a reconstruction $\widehat{\mathbf{a}} = \sum_i \mathbf{u}_i$, we defined the feature weight for the i-th feature as

$$\text{weight}\,(i) = \mathbf{u}_i^{\top} \widehat{\mathbf{a}} / \left\| \widehat{\mathbf{a}} \right\|_2^2 .$$

For our supervised feature dictionaries, we find that 10% of all weights are negative, and that the average value of all negative weights across all nodes in the IOI circuit and all three attributes is -0.037. Similarly, for our task-specific SAE feature dictionaries, even though 31% of all weights are negative, the average value of all negative weights is -0.002. The number and magnitude of weights higher than 1 are even smaller.

Causal evaluation using interpretability. While the feature descriptions generated through our automatic scoring predict well when a feature is active, it is still unclear whether they also have an interpretable causal role, i.e. whether activating or deactivating a certain feature leads to a change in output logits that would be expected from the feature's description. To test this, we propose two experiments to judge the sufficiency and necessity of our interpreted features that involve patching activations from a counterfactual prompt and calculating the effect on the model's output:

- **Estimating sufficiency:** To estimate sufficiency, we construct SAE activations where we fix features with a test F1-score smaller than a threshold and patch activations of features with a high F1-score from the counterfactual prompt. We then calculate reconstructions of this new SAE activation vector, patch cross-sections, and record whether the model successfully predicts the correct counterfactual name.
- **Estimating necessity:** To estimate necessity, we propose a similar experiment where we fix features with a high test F1-score and patch all remaining features. This intervention should not change the output logits if our features are complete.

We run this experiment on cross-sections of name mover outputs and repeat this experiment for different thresholds. We observe that for a threshold F1-score of 0.6, the SAE features are both faithful and complete to a high degree. We observe that the faithfulness metric significantly decreases for higher F1-scores of 0.7 and 0.8 but remarkably, we also observe that only fixing features with a very high F1-score of ¿0.8 while patching all other features from the counterfactual prompt is sufficient to keep the model predicting the base prompt's output.

A.17 Additional figures

Figure 13: Fraction of recovered logit difference for several different methods to compute feature dictionaries, across cross-sections of the circuit. For a definition of the 'names' parametrization, see Appendix A.10.

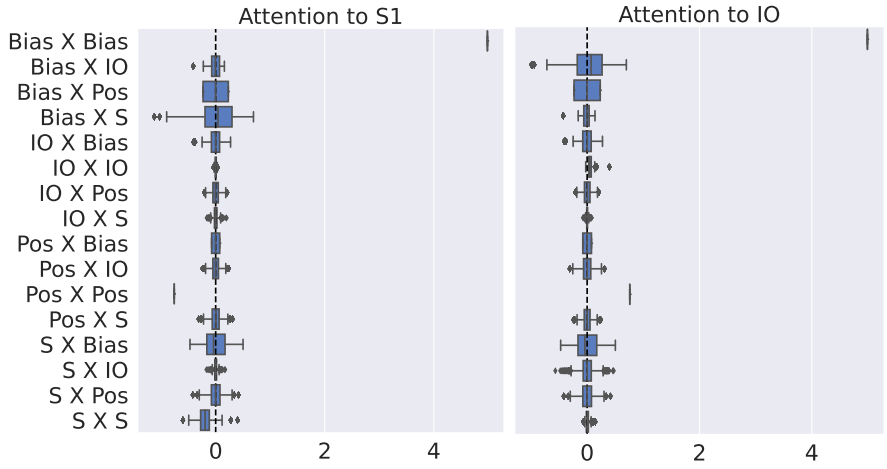

Figure 14: Attention score decomposition for the L9H6 name mover (see Figure 6 for explanation). Notice that, in contrast with L10H0 attention socres, there is no significant (inhibitory) interaction between the IO features in the query and key.

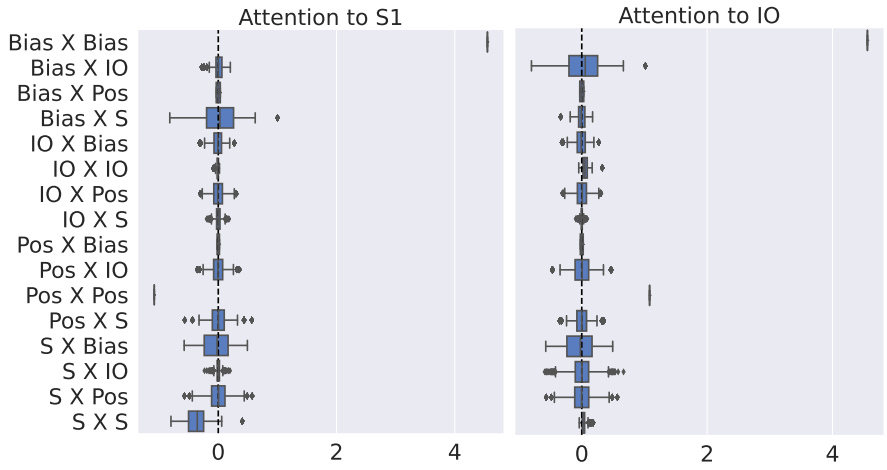

Figure 15: Attention score decomposition for the L9H9 name mover (see Figure 6 for explanation).

Figure 16: Metrics for our chosen task-specific SAEs for each relevant node in the IOI circuit. The x-axis shows the absolute value of the difference in logit differences between a clean run of the model, and a run where the activations at the given node are replaced by the SAE's reconstructions, normalized by the difference between the clean logit difference and the logit difference when the node is mean-ablated instead. The y-axis shows the average number of features active per prompt.

Figure 17: Counterpart to Figure 16 where the decoder vectors are frozen during SAE training.

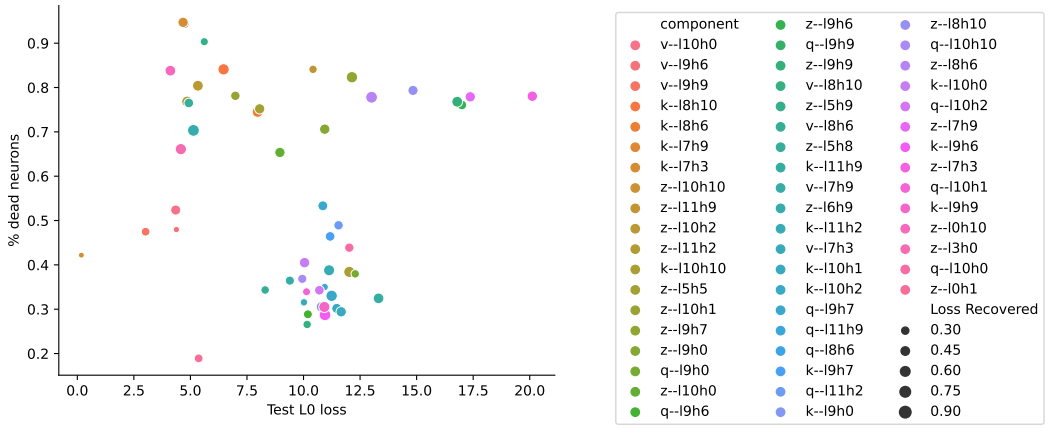

Figure 18: ℓ_0 loss versus fraction of dead neurons for our full-distribution SAEs.

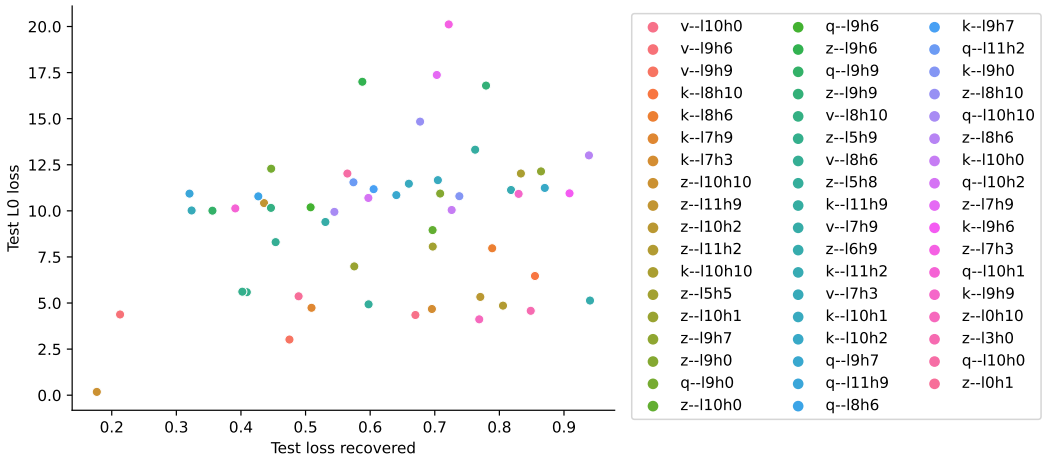

Figure 19: ℓ_0 loss versus loss recovered (against a mean ablation) for our full-distribution SAEs.

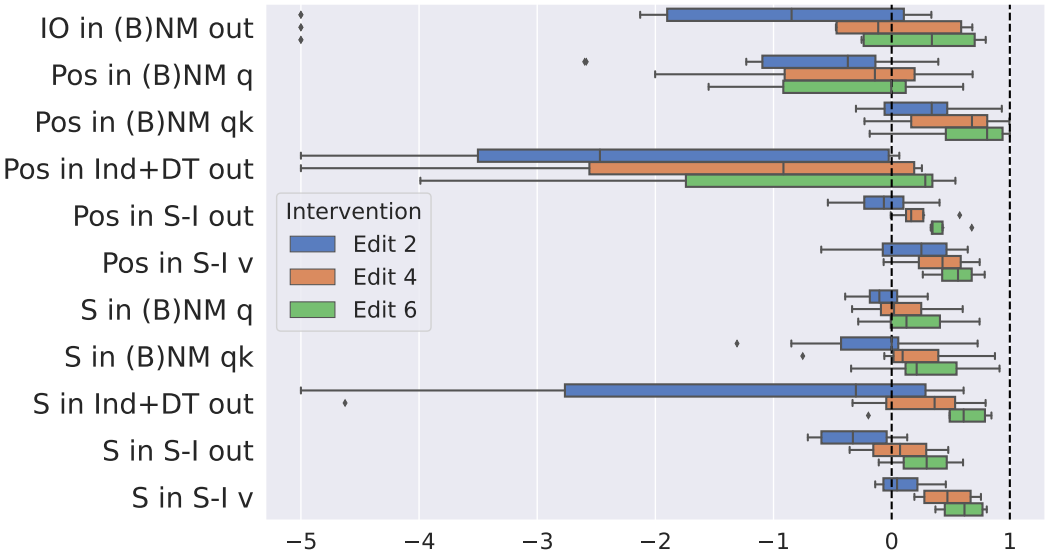

Figure 20: Distribution of the average feature weight removed by our interpretation-agnostic edits using SAE feature dictionaries, over all locations in a given cross-section, normalized by the corresponding weight for the supervised feature dictionaries. Weight removed values are transformed linearly so that a value of 0 indicates that the weight removed by the edit equals the weight removed by the corresponding 'ground truth' supervised edit; and a value of 1 indicates that the edit removed a total weight of 1, meaning that the edit essentially overwrites all SAE features present in the activation.

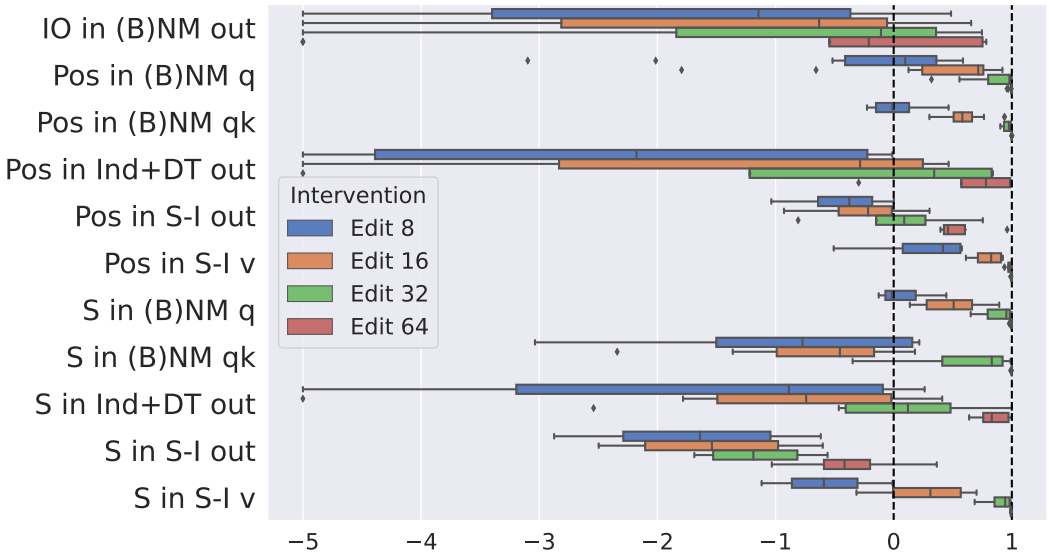

Figure 21: Counterpart of Figure 20 for full-distribution SAEs, when editing using features with high F1 score for the attribute

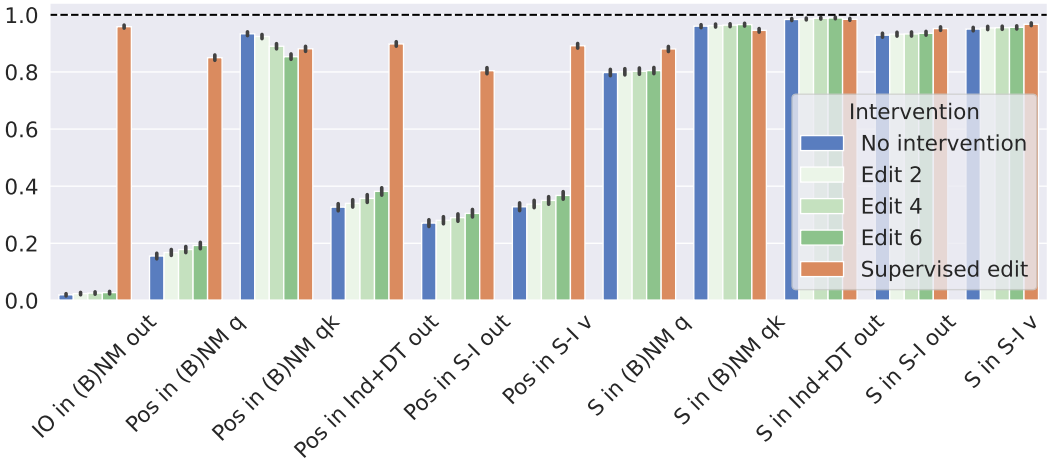

Figure 22: Counterpart to Figure 3, where the decoder vectors are frozen during SAE training.

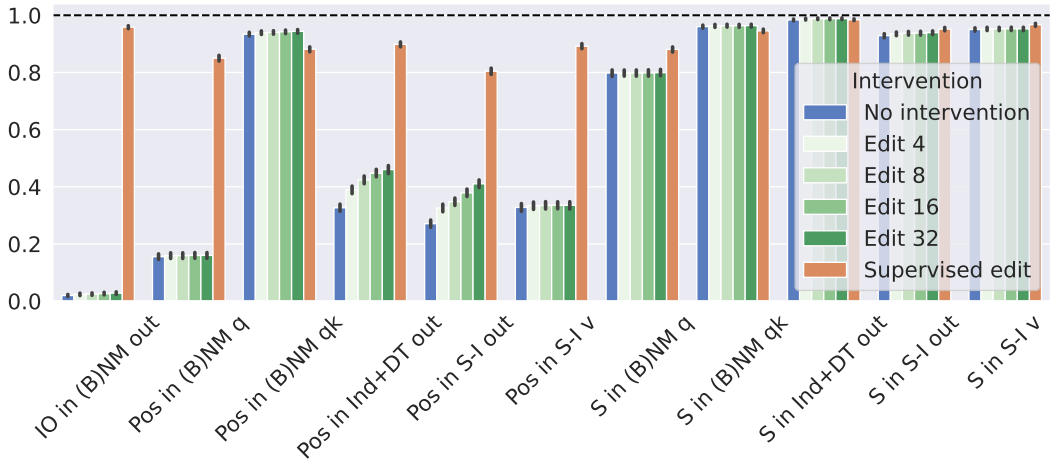

Figure 23: Counterpart to Figure 3, where we use full-distribution SAEs instead. Here, we need to change a much higher number of features in order to have a noticeable effect (and sometimes editing even 32 features fails)

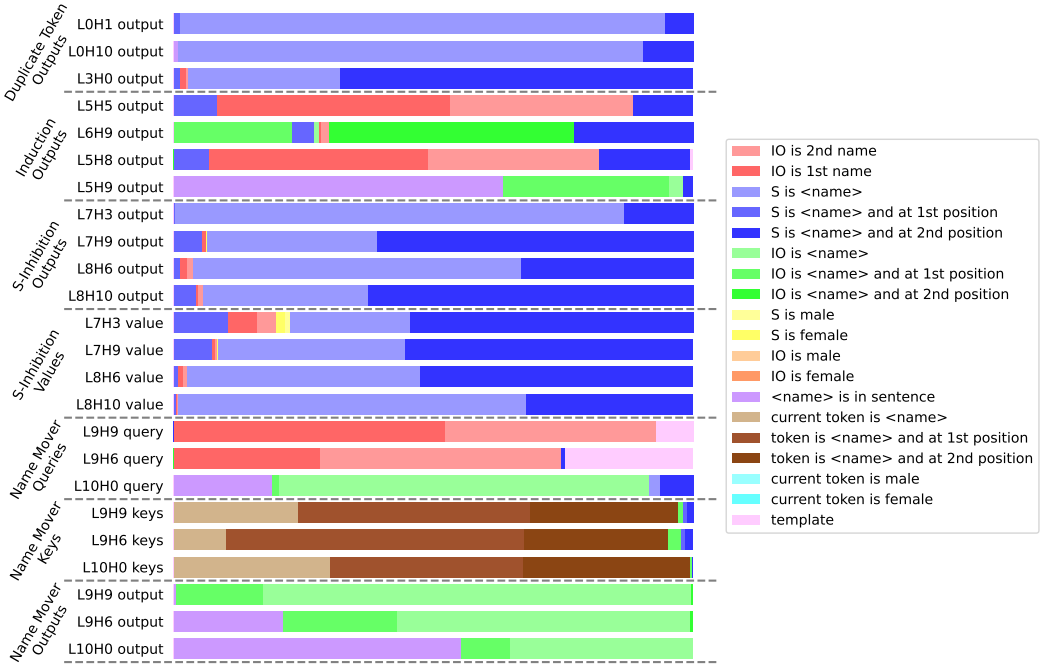

Figure 24: Interpreting the features learned by the task SAEs. For each node in the main IOI circuit (without backup/negative name movers), we show the distribution of the features which have an explanation with F_1 score above a threshold. The SAE chosen at each node is the one with the most interpretable features out of all SAEs trained on this node during our hyperparameter sweep.

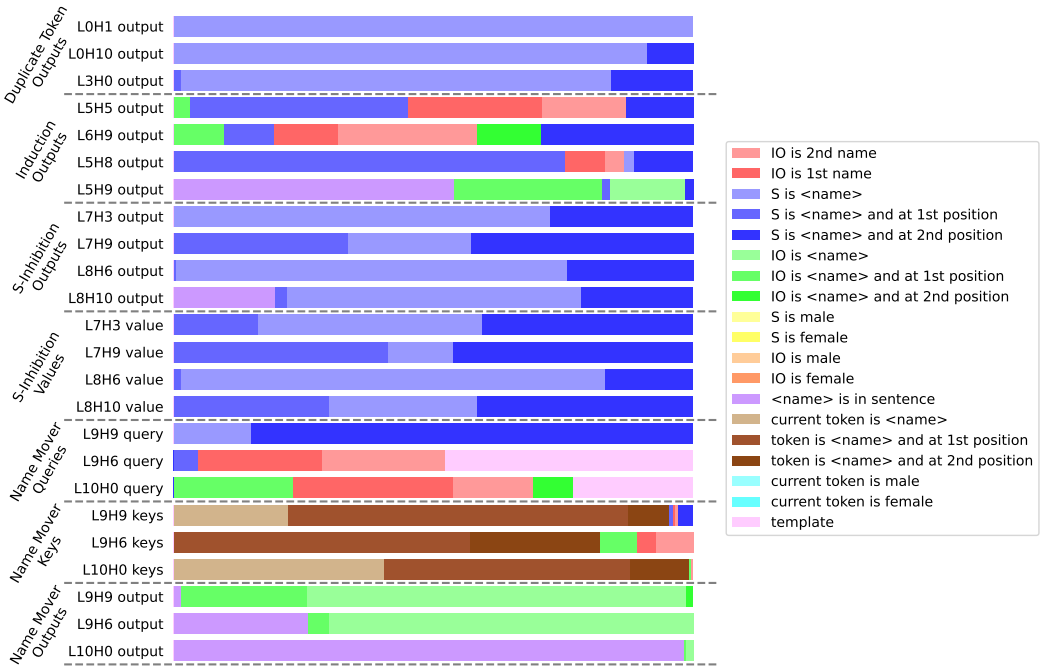

Figure 25: Interpreting the features learned by the task SAEs. This is a counterpart to Figure 24 for the SAEs chosen based on the ℓ_0 and logit difference recovered metrics.

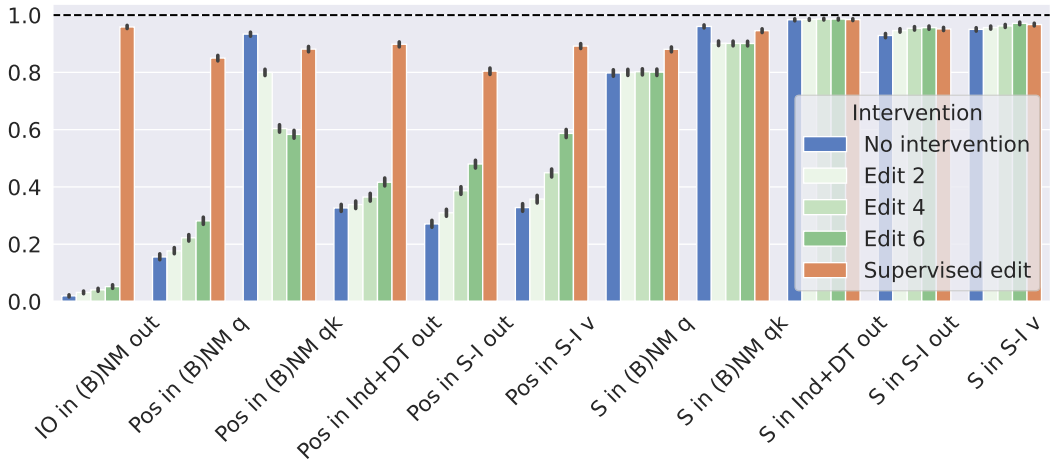

Figure 26: Interpretation-aware sparse control, using task SAE features with the highest F_1 score with respect to the given attribute.

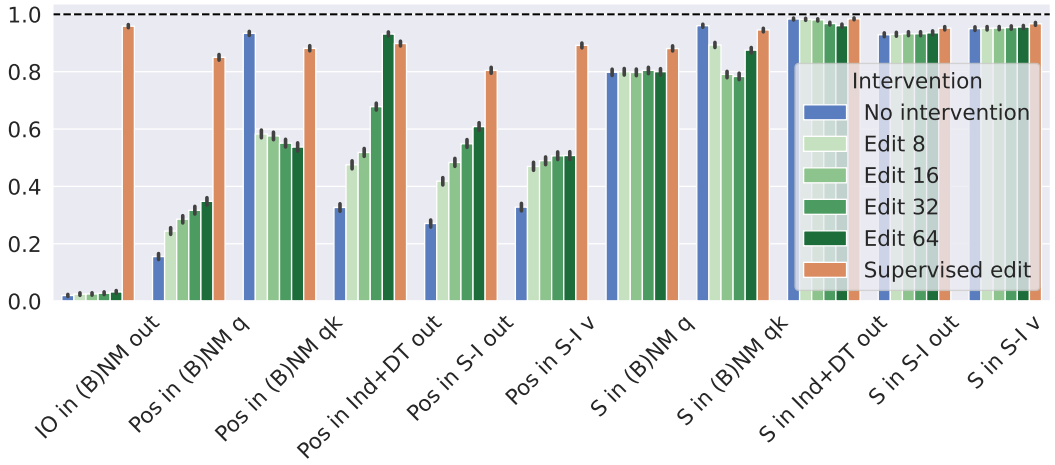

Figure 27: Counterpart of Figure 26 with full-distribution SAEs.

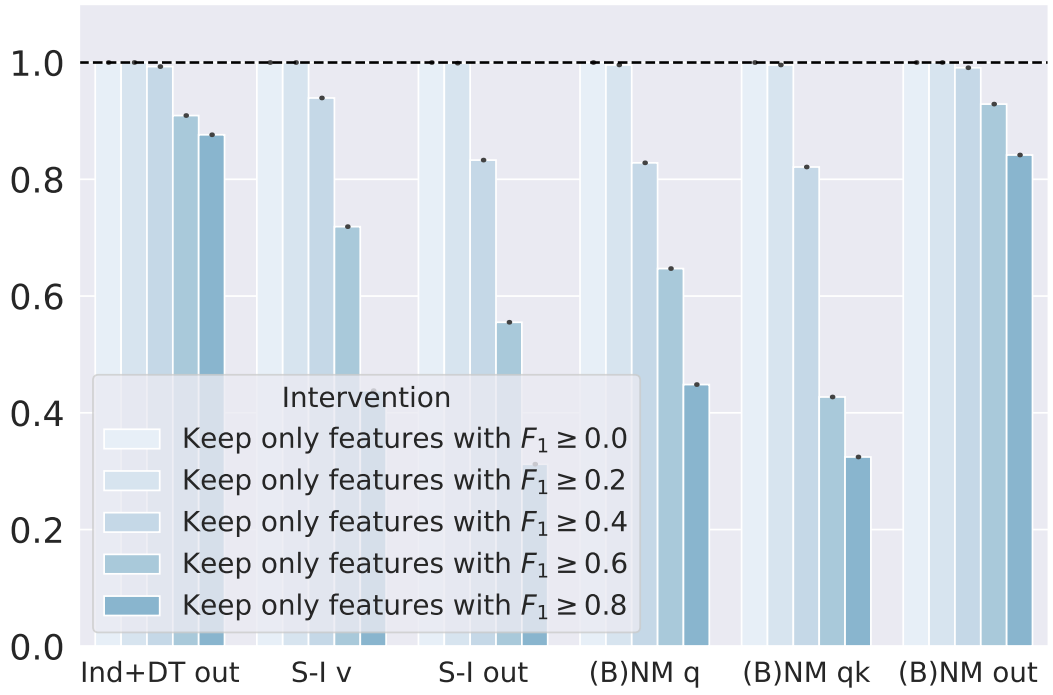

Figure 28: Measuring the **sufficiency** of interpretable features for task SAEs: effect of subtracting features with the lowest F_1 score from activations on logit difference. A value of 1 is best.

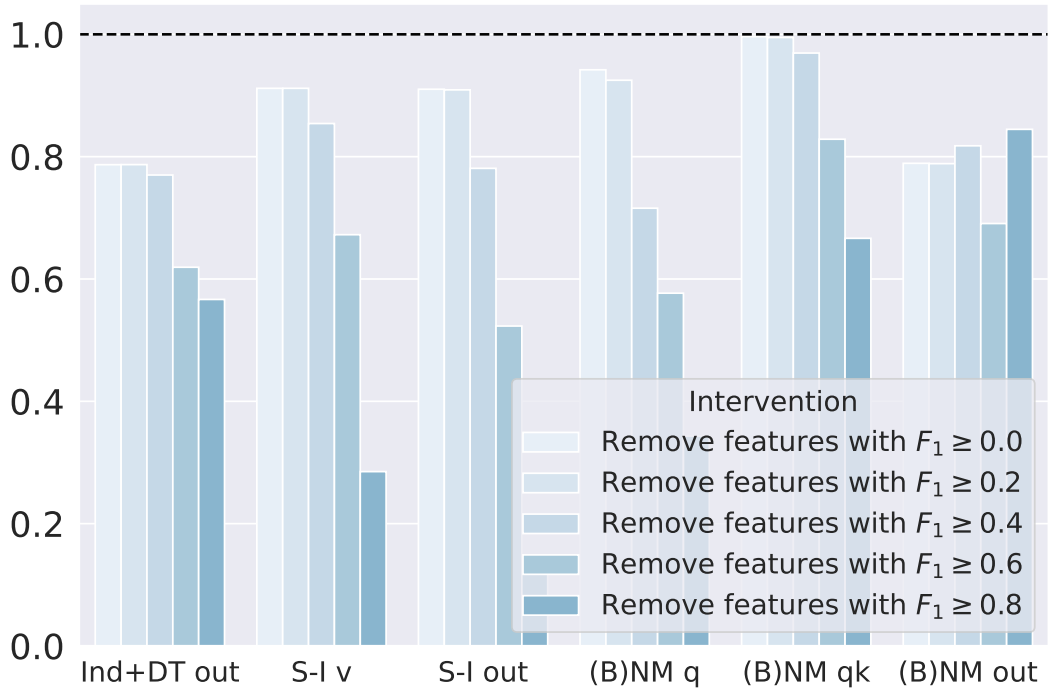

Figure 29: Measuring the **necessity** of interpretable features for task SAEs: effect of removing features with the highest F_1 score from activations on the logit difference. Values are rescaled linearly so that a value of 1 corresponds to perfect recovery of the logit difference achieved by mean ablation (i.e., ideal intervention removing all features). A value of 1 is best.