TOWARDS PRINCIPLED EVALUATIONS OF SPARSE AUTOENCODERS FOR INTERPRETABILITY AND CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

A major open problem in mechanistic interpretability is disentangling internal model activations into meaningful features, with recent work focusing on sparse autoencoders (SAEs) as a potential solution. However, verifying that an SAE has found the 'right' features in realistic settings has been difficult, as we don't know the (hypothetical) ground-truth features to begin with. In the absence of such ground truth, current evaluation metrics are indirect and rely on proxies, toy models, or other non-trivial assumptions.

To overcome this, we propose a new framework to evaluate SAEs: studying how pre-trained language models perform specific tasks, where model activations can be (supervisedly) disentangled in a principled way that allows precise control and interpretability. We develop a task-specific comparison of learned SAEs to our supervised feature decompositions that is *agnostic* to whether the SAE learned the same exact set of features as our supervised method. We instantiate this framework in the indirect object identification (IOI) task on GPT-2 Small, and report on both successes and failures of SAEs in this setting.

1 INTRODUCTION

Recently, language models (LLMs) have demonstrated impressive performance in increasingly general domains (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; OpenAI, 2023). However, a precise and scalable understanding of how their inner computations function and can be controlled – which is the goal of mechanistic interpretability (MI) – is still largely missing (Olah, 2023; Farquhar et al., 2023; Nanda et al., 2023b; Olah et al., 2024). A central hypothesis in MI is the *linear representation hypothesis* (Mikolov et al., 2013): the features LLMs represent and use are well-modeled by linear subspaces of component activations (Grand et al., 2018; Li et al., 2021; Abdou et al., 2021; Nanda et al., 2023a). Moreover, recent work suggests that n-dimensional activations may represent $m \gg n$ features in *superposition* (Elhage et al., 2022a; Gurnee et al., 2023). Motivated by these ideas, a series of works (Cunningham et al., 2023; Bricken et al., 2023) have recently proposed the *sparse autoencoder* (SAE) framework (Faruqui et al., 2015; Arora et al., 2018; Yun et al., 2021) to disentangle these superposed linear subspaces.

However, evaluating the success of SAEs has been a major challenge in the absence of 'ground-truth' features, with current methods relying on ad-hoc proxies, toy models, or indirect measures with extra assumptions (Elhage et al., 2022b; Bricken et al., 2023; Sharkey et al., 2023)[1]. As a step towards more principled and objective SAE evaluations, we propose to (1) find a high-quality feature decomposition using supervision in a task-specific setting, and (2) compare the learned SAE features against it for the purposes of reconstructing, controlling and interpreting the model in the task's context. Crucially, we observe that model activations may admit many different sparse feature decompositions, which may or may not allow for the same (or similar) degree of control and interpretability as our supervised decompositions. Thus, it is crucial that we evaluate the SAE's feature decomposition in a way that is agnostic to whether the SAE learned the same exact set of features as our supervised method.

Our contributions can be summarized as follows: (1) we propose principled methods to additively decompose activations into features in task-specific settings, using labels derived from input proper-

[1] We refer the reader to Appendix F for a more detailed discussion of these issues and other related work.

ties important to the model's computation on the task; (2) we implement and validate this framework on the indirect object identification (IOI) task on GPT-2 Small (Wang et al., 2023), where we show that our features allow us to precisely control and interpret the model's computation; (3) we evaluate how well SAEs (trained both on the IOI distribution and GPT2-Small's full pre-training distribution) allow us to control/interpret the model on the IOI task relative to the features we construct, and to what degree their features match our constructed features; (4) we report some interesting phenomena in SAE training, such as *feature occlusion*, where one salient feature is learned at the expense of others, and *feature oversplitting*, where a single feature is broken into multiple features.

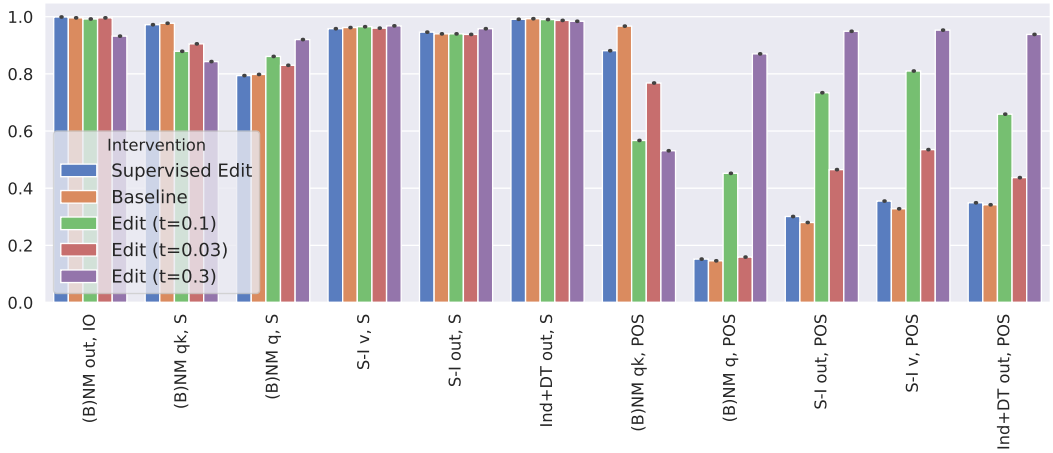

Figure 1: Edit accuracy at cross-sections of the IOI circuit, using our supervised feature decomposition (blue), a ground-truth baseline intervention (orange), and three methods for editing using task-specific SAEs (green, red and purple)

2 PRELIMINARIES

Sparse autoencoders. Following the setup of Bricken et al. (2023), a sparse autoencoder (SAE) is an unsupervised model which learns to reconstruct activations $\mathbf{a} \in \mathbb{R}^n$ as a weighted sum of $m \gg n$ learned *features* with non-negative weights. Specifically, the autoencoder computes a hidden representation $\mathbf{f} = \text{relu}\left(W_{enc}\left(\mathbf{a} - \mathbf{b}_{dec}\right) + \mathbf{b}_{enc}\right) \in \mathbb{R}^m$, and a reconstruction $\widehat{\mathbf{a}} = W_{dec}\mathbf{f} + \mathbf{b}_{dec} = \sum_{j=1}^{m} \mathbf{f}_j (W_{dec})_{:,j} + \mathbf{b}_{dec}$ 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 training objective is a sum of the MSE between reconstructions and original activations, with an ℓ_1 regularization term on the hidden representation $\mathbf{f}$, which encourages only a few features to be active for each given input.

The IOI task. Input prompts in the IOI task are sentences of the form 'When Mary and John went to the store, John gave a book to', where we refer to the repeated name (John) as **S** (the subject) and the non-repeated name (Mary) as **IO** (the indirect object); the correct completion is the **IO** name. The work of Wang et al. (2023) thoroughly analyzed how GPT2-Small performs this task, and identified a set of 26 attention heads in six classes that together form a circuit to solve the task (see Appendix Figure 3 and Appendix A.2). The main metric used to discover the IOI circuit was the **logit difference**: the difference in log-probabilities assigned by the model to the **IO** and **S** names. This metric surfaces contributions that are consistent and significant, but non-pivotal to the task, and we use it throughout this work to evaluate the causal effect of (fine-grained) model interventions.

Wang et al. (2023) present evidence that the model uses the algorithm 'Find the two names in the sentence, detect the repeated name, and predict the non-repeated name' to solve the task. In particular, they discover that the computation of the circuit is determined by the **IO** and **S** name tokens, as well as whether the **S** name comes first or second in the sentence, a binary attribute we refer to as **Pos** (short for 'position'). Of particular interest to us are several **computational cross-sections** of the circuit, where editing the **IO**, **S** and **Pos** features allows us to control the model's computation; we refer to Appendix A.3 for a detailed description.

3 SUPERVISED FEATURE DECOMPOSITIONS USING TASK-SPECIFIC LABELS

Setup. Suppose we have a (task-specific) distribution $\mathcal{D}$ over prompts $p \in \mathcal{P}$, such that each prompt can be described by *attributes*, which we model as functions $a_i : \mathcal{P} \to S_i$ taking values in finite sets S_i. Motivated by the linear representation hypothesis and the SAE framework, we can try to use these attributes to decompose a model activation $\mathbf{a} \in \mathbb{R}^n$ for a prompt p with $a_i(p) = v_i$ as a sum of feature vectors $\mathbf{a} \approx \sum_{i=1}^{N_A} \mathbf{u}_{iv_i} := \widehat{\mathbf{a}}$, where $\mathbf{u}_{iv} \in \mathbb{R}^n$ is a vector representing the attribute a_i having value $v \in S_i$ [2]. Given a set $\{\mathbf{a}^{(k)}\}_{k=1}^N$ of activations of some model component on the prompts $\{p^{(k)}\}_{k=1}^N$, with $\overline{\mathbf{a}}$ the average activation, we consider two ways to compute the vectors $\mathbf{u}_{iv}$:

MEAN CODES. We compute $\mathbf{u}_{iv}$ as the average of the centered activations $\mathbf{a}^{(k)} - \overline{\mathbf{a}}$ for the prompts where $a_i(p^{(k)}) = v$.

MSE CODES. We compute $\mathbf{u}_{iv}$ by minimizing the ℓ_2 reconstruction error $\frac{1}{N}\sum_{k=1}^N \left\| \mathbf{a}^{(k)} - \sum_{i=1}^{N_A} \mathbf{u}_{iv_i^{(k)}} \right\|_2^2$ over the dataset, where $a_i(p^{(k)}) = v_i^{(k)}$.

Both methods enjoy some desirable properties, as we discuss in Appendix B; in particular, no non-trivial logistic regression linear probe exists for an attribute a_i if and only if the mean codes for a_i are all zero, if and only if the MSE codes for a_i are constant across values in S_i (the last under the assumption that the values of a_i are probabilistically independent of the values of all other attributes). This suggests that both methods are robust to the inclusion of irrelevant or non-linearly represented attributes, which is convenient in practice, where we may not know the exact set of attributes that are relevant to a given model activation.

Implementation and evaluation in IOI. 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. In particular, the attributes should collectively give a complete description of the prompt's task-relevant properties and align with the model's internal 'ontology' of the computation (Geiger et al., 2023). We thus chose to use the indirect object (**IO**), subject (**S**) and position (**Pos**) as attributes; a discussion of alternative parametrizations we considered and their pros and cons is in Appendix C.1.

To validate the feature decomposition, we use several increasingly strict tests: (1) **faithfulness**: we intervene by replacing activations with their reconstructions, and check whether the model is still able to solve the IOI task; (2) **feature editing**: we intervene by adding/subtracting vectors corresponding to the **IO**, **S** and/or **Pos** attributes so as to change their values, and check if the model's behavior changes accordingly; (3) **subspace-level interpretability**: we investigate how the features compose to form the model's internal computation, such as attention scores and composition between attention heads in different layers. We refer the reader to Appendix C.2 for details on the methodology.

Results. We find that our feature decomposition results in a good approximation of the model's computation, both at the keys/values/queries/outputs of attention heads in the IOI circuit, and in terms of the logit difference when replacing activations of circuit cross-sections with their reconstructions (Appendix Figure 5). More interestingly, we find that our features allow us to precisely control the model's computation: we can edit any subset of the **IO**, **S** and **Pos** attributes via feature arithmetic, and the change in logit difference matches a ground-truth baseline (Figure 1; see Appendix C.3 for details). Finally, we use our features to gain mechanistic understanding of the model's computation. We find subspace-level expressions of the IOI circuit algorithm, such as the destructive interference between **S** features in the queries and keys of the name mover heads (Appendix Figure 6), and the communication of **Pos** and **S** information from the S-Inhibition heads to the name mover heads (Appendix Figure 7). We also observe and confirm new phenomena in the IOI circuit, such as a causally relevant **IO** feature in the queries of the L10H0 name mover head (Appendix D).

[2] Note that this decomposition is more restrictive than the one provided by a SAE, as the coefficients of the feature vectors are independent of the input example; this is a reasonable assumption in settings like the IOI task, where we expect the relevant features to behave as binary, on/off switches as oppposed to having continuous degrees of activation.

4 TRAINING AND EVALUATING SPARSE AUTOENCODERS IN THE IOI TASK

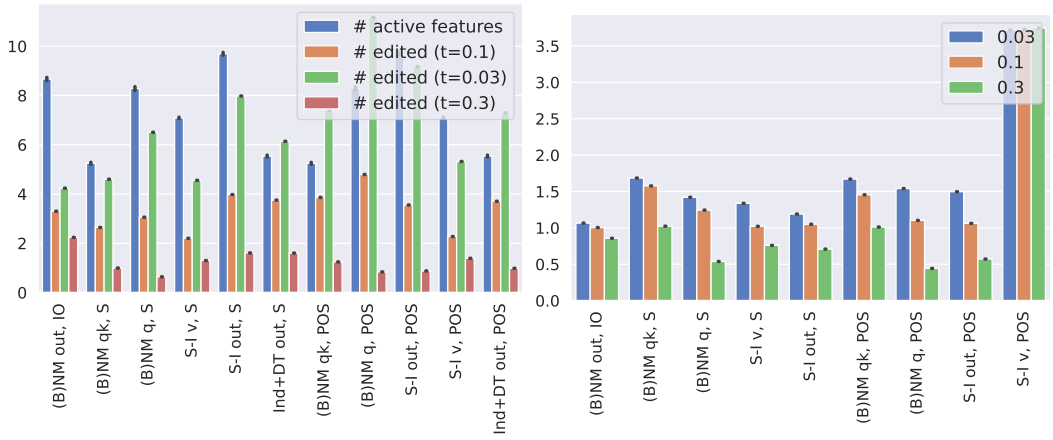

Figure 2: Measuring the success of editing using SAEs: number of total features edited in cross sections of the circuit compared to total active features (left); contribution of the edited features to the reconstruction, normalized by the same contribution using the supervised decomposition (right)

We trained SAEs on the IOI distribution (Appendix A.1), in all keys/queries/values/outputs of heads in the IOI circuit, using the same set of hyperparameters (details in Appendix E.2). We also trained SAEs on the pre-training distribution of GPT2-Small using the OpenWebText dataset (Gokaslan & Cohen, 2019) (details in Appendix E.4).

Evaluation methodology. Model activations may admit multiple sparse feature decompositions; in particular, even in a task as simple as IOI, there may exist many decompositions that allow for (approximately) the same level of control and interpretability as our supervised feature decomposition. Moreover, these decompositions may fail various tests that explicitly 'look for' our supervised features (see Appendix E.3 for illustrative examples).

Accordingly, we evaluate SAEs in a way that is agnostic to whether the SAE learned the same exact set of features as our supervised method. Given a reconstruction $\widehat{\mathbf{a}} = \sum_j \mathbf{v}_j + \mathbf{b}$ where $\mathbf{b}$ is a bias term, let $\mathbf{r} = \widehat{\mathbf{a}} - \mathbf{b} = \sum_j \mathbf{v}_j$. We define *feature weights* $w_j = \mathbf{v}_j^\top \mathbf{r} / \|\mathbf{r}\|_2^2$ which measure the contributions of each feature to the reconstruction, with $\sum_j w_j = 1$. Then, we use our SAEs to edit the **IO**, **S** and **Pos** attributes as follows. If we wish to edit the activation $\mathbf{a}$ of prompt p, let p' be the counterfactual prompt which differs from p only in the attributes we wish to edit, and $\mathbf{a}'$ its activation. Then we form the edited activation $\mathbf{a}_{edited}$ by subtracting the features in $\mathbf{a}$ where $w_j > t$ in p but not in p', and adding the features where $w_j > t$ in p' but not in p (for a threshold t, e.g. $t = 0.1$). We report the same metrics for this editing methodology as for our supervised feature decomposition. Moreover, we report the number of total active features, the number of features we change, and the total weight of the changed features normalized by the same weight when editing using the supervised decomposition. These metrics are necessary, because an edit which always changes all features, or more of the weight than necessary, suggests that the SAE has not learned a good disentanglement of the features.

Results. Results for feature editing for thresholds $t = 0.03, 0.1, 0.3$ are shown in Figure 1 (see Appendix Figure 8 for full-distribution-trained SAEs); we see that, overall, our task-specific SAEs are mostly able to edit the **IO** and **S** attributes, but struggle with the **Pos** attribute. Furthermore, we show the average number of total active features and edited features, as well as the (normalized) weight of edited features in 2. We observe that $t = 0.1$ provides a good balance between editing a few features and having reasonable editing accuracy.

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.

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.

Sebastian Farquhar, Vikrant Varma, Zachary Kenton, Johannes Gasteiger, Vladimir Mikulik, and Rohin Shah. Challenges with unsupervised llm knowledge discovery. *ArXiv*, abs/2312.10029, 2023. URL https://api.semanticscholar.org/CorpusID:266335236.

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.

Aaron Gokaslan and Vanya Cohen. Openwebtextcorpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus.

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.

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.

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, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

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

Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023b. URL https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall.

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.

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.

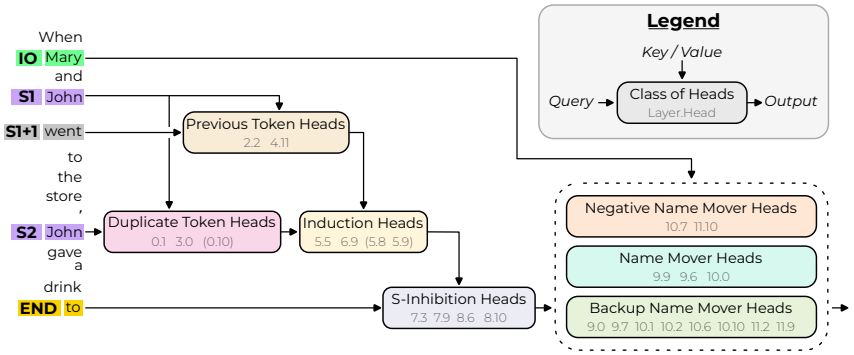

Figure 3: A reproduction of Figure 2 from Wang et al. (2023), showing the internal structure of the IOI circuit. Original caption: *The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.*

A IOI TASK AND DATASET DETAILS

A.1 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.2 CIRCUIT STRUCTURE

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[3]

[3]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.

- **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, and as 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 3 for a schematic of the full circuit, and to Wang et al. (2023) for a more complete discussion.

A.3 Computational Cross-Sections of the IOI 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[4]. The keys represent information about the **IO** and **S** names, and the position **Pos** of the **S** name: in particular, the **S** and **Pos** information combines with the query to inhibit attention to the S1 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.

B Mean Codes and MSE Codes

B.1 Overview

We always include a 'bias' attribute $a_1(p) = \perp$ for each prompt, with $S_1 = \{\perp\}$, to capture the mean activation of the component; this is analogous to the decoder bias in a SAE (Section 2).

Mean codes. Let $P_{iv} = \{k \mid a_i(p^{(k)}) = v\}$ be the set of prompts where the i-th attribute has value $v \in S_i$. We let the vector encoding $a_i(p) = v$ to simply be the average of the *centered* activations for the prompts in P_{iv}:

$$\mathbf{u}_{iv} := \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \left(\mathbf{a}^{(k)} - \overline{\mathbf{a}} \right), \tag{1}$$

where $\overline{\mathbf{a}} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{a}^{(k)}$ is the empirical mean activation. We also set $\mathbf{u}_{1\perp} = \overline{\mathbf{a}}$. Mean codes enjoy several convenient properties:

[4]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.

- 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 is not linearly represented, or not represented at all, in the activations, we expect the mean code to be ≈ 0 (see Belrose et al. (2023) and Appendix B.2).

This suggests that mean codes are somewhat 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 ground-truth attributes for each activation. However, mean codes 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 $\|\mathbf{a} - \widehat{\mathbf{a}}\|_2^2$.

MSE codes. Here, we instead find $\mathbf{u}_{iv}$ by directly minimizing the ℓ_2 reconstruction error over the dataset:

$$\min_{\mathbf{u}} \frac{1}{N} \sum_{k=1}^{N} \left\| \mathbf{a}^{(k)} - \sum_{i=1}^{N_A} \mathbf{u}_{iv_i^{(k)}} \right\|_2^2 \tag{2}$$

where $v_i^{(k)} = a_i(p^{(k)})$ is the value of the i-th attribute for the k-th prompt. 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 codes from Equation 1 (see Appendix B.3 for details). Furthermore, this objective closely mimics the SAE objective: here, the sparsity is hard-coded, leaving only the ℓ_2 objective.

Finally, as a counterpart to the theoretical results for mean codes, if the values of an attribute a_i are independent of the values of all other attributes, we can prove that the mean codes for a_i will all be zero if and only if the MSE codes for a_i are constant (though not necessarily zero); see Appendix Lemma B.1.

B.2 Mean codes capture 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 codes, 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 codes for all values of this attribute are zero. In the finite data regime, this gives us some theoretical grounds to expect that the mean codes will be nonzero 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 codes for this attribute to be zero.

B.3 Properties of MSE codes

MSE codes 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 codes 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

$$U^* = \left(C^\top C\right)^+ C^\top A \tag{3}$$

MSE codes 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

$$\mathbf{u}_{iv}^* = \frac{1}{|P_{iv}|} \sum_{k \in P_{iv}} \overline{\mathbf{a}}^{(k)} \tag{4}$$

Note that this is very similar to Equation 1, but also importantly different, because the optimal $\mathbf{u}_{iv}^*$ depends on the optimal values of the codes for the other attributes.

MSE codes 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 codes $\mathbf{u}_{iv} = \mathbf{u}_i \forall v$. This is a counterpart to the result from Appendix B.2 for MSE codes:

Lemma B.1. *Suppose that all conditional means $\mathbb{E}[\mathbf{a}|a_i(p) = v]$ 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.*

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

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 codes $\mathbf{u}_{iv}^*$ 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 4, 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

$$\frac{1}{|P_{iv}|}\sum_{k\in P_{iv}} \mathbf{u}^*_{lv_l^{(k)}} = \frac{1}{|P_{iv}|}\sum_{v_l\in S_l} \mathbf{u}^*_{lv_l} \left|\{k \text{ s.t. } a_i(p_k) = v, a_l(p_k) = v_l\}\right| \tag{5}$$

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 3, 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 codes $\mathbf{u}^*_{iv}$ will also converge as $N \to \infty$. So we see that the sum in Equation 5 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.

□

C Implementing Supervised Feature Decompositions in the IOI Task

C.1 Alternative Parametrizations

How we choose to parametrize the inputs to the model is a crucial assumption, as it will influence the way we interpret the model's internal computations, and the results of SAE training later on.

Recall that 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**). In addition, there is some random variation in the other words in the prompt that we expect only introduces noise in the IOI circuit (Appendix A.1). Motivated by this, we consider two possible parametrizations of prompts via attributes (we also add a bias attribute to both parametrizations, as explained earlier):

- **independent parametrization**: we use the three independently varying attributes – **S**, **IO** and **Pos** – to describe each prompt.
- **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 codes 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 **??**).

What about alternative parametrizations? A priori, there are as many possible parametrizations as there are ways (up to isomorphism) to pick (surjective) functions $a_i : \mathcal{P} \to S_i, i = 1, \ldots, k$ from the set of all possible prompts into finite sets S_i, such that the important properties of the prompt (**IO**, **S** and **Pos**) can be recovered from the values $(a_1(p), \ldots, a_{N_A}(p))$. Therefore any specific choice of parametrization is ad-hoc. Still, we believe that our parametrizations can be justified in two ways: first, they correspond to the intermediate states of the IOI circuit identified in Wang et al. (2023); second, we show through extensive tests that they both approximate the model's internal activations well, and can be used to steer the model in a predictable way.

We now further discuss some alternative parametrizations that we considered. First, note that a parametrization that groups **S** and **IO** together would be much less interesting. It would require a much larger dataset for learning the codes, because every possible combination of **S** and **IO** would need to be represented. Also, modulo random variation in the other words in the prompt, we would have a feature per every two examples in the distribution, which would make the resulting codes closer to being a clustering of similar examples than an interesting decomposition of the model's internal computation. Next, another reasonable parametrization would be to use an attribute for

each of the three names in the sentence in the order they appear, **Name1**, **Name2** and **Name3**. We refer to this parametrization as the 'names' parametrization. However, with this parametrization, we observe very poor reconstruction quality, as noted in Subsection **??** (also see Appendix Figure 4). Moreover, it can be shown that this parametrization is in a certain sense incompatible with both the independent and coupled parametrizations, as they cannot express the same set of prompt reconstructions (Appendix **??**). Other reasonable parametrizations would be to only group e.g. **S** and **Pos** and leave **IO** separate, or vice-versa. However, both of these are less expressive than the coupled parametrization (Appendix **??**).

Finally, it is possible that a different parametrization is best at different locations in the circuit; however, we do not explore this possibility in this work. We find that the simplest (independent) parametrization works well throughout the circuit, and we use it as our default parametrization in the main body of the paper.

C.2 Methodology for feature editing and mechanistic analysis

How can we make sure that a given set of sparse feature dictionaries captures the 'ground truth' features used by the model? Even in a setting as simple as the IOI task, this is a nuanced and multi-faceted question. We thus propose a strategy based on several tests of model control and interpretability:

- **Faithfulness**: when we intervene on the model by replacing internal activations $\mathbf{a}$ with their reconstructions $\widehat{\mathbf{a}}$ using the learned feature dictionaries, is model behavior preserved?
- **Counterfactual effect of editing individual features**: when we intervene on the model by editing given features, do we observe the expected changes in model behavior?
- **Subspace-level mechanistic analysis**: can we decompose the model's internal operations in terms of elementary interactions between the learned code vectors? Do these interactions align with the known circuit structure?

The first two tests are subspace-level refinements of the corresponding tests from Wang et al. (2023). We next describe in more detail the methodology for feature editing and mechanistic analysis in the context of our framework.

Editing internal model representations. A central goal of interpretability and control is the ability to do targeted, precise edits of internal model computations to steer the model in a certain way without otherwise degrading or affecting performance. Given a prompt p where $a_i(p) = v_i$, a hidden representation $\mathbf{a}$ for p, and some other value $v_i' \neq v_i$ for the i-th attribute, a straightforward way to change v_i to v_i' is to subtract the code vector for v_i and add the one for v_i', obtaining the edited representation

$$\mathbf{a}_{edit} = \mathbf{a} - \mathbf{u}_{iv_i} + \mathbf{u}_{iv_i'}.$$

This operation is simple, and the final result of multiple edits is independent of the order in which they are performed. We additionally explored two other ways to perform the part of the edit that removes the current attribute value v_i. Instead of subtracting $\mathbf{u}_{iv_i}$, we can subtract a *multiple* $\alpha\mathbf{u}_{iv_i}$ so that $\left(\mathbf{a} - \alpha\mathbf{u}_{iv_i}\right)^\top \mathbf{u}_{iv_i}$ is 0 (**zero ablation**) or the mean of $\mathbf{a}^\top \mathbf{u}_{iv_i}$ over the dataset (**mean ablation**).

Subspace-level mechanistic analysis. 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.

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)).

Given reconstructions for the keys and queries of an attention head at certain positions

$$\mathbf{k} \approx \sum_{i=1}^{N_A^{keys}} \mathbf{u}_{ik_i}, \quad \mathbf{q} \approx \sum_{i=1}^{N_A^{queries}} \mathbf{v}_{iq_i}$$

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

$$\mathbf{q}^T\mathbf{k} \approx \sum_{i=1}^{N_A^{queries}} \sum_{j=1}^{N_A^{keys}} \mathbf{v}_{iq_i}^\top \mathbf{u}_{jk_j}$$

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

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.

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

$$\mathbf{q}_{l,t,h} = W_{l,h}^Q \,\mathrm{LN}\left(\mathbf{r}_{l,t}\right) = W_{l,h}^Q \,\mathrm{LN}\left(\overline{\mathbf{r}}_{l,t} + W_{l',h'}^O \mathbf{z}_{l',t,h'}\right)$$

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, $\overline{\mathbf{r}}_{l,t}$ is the remainder of the residual stream after removing the contribution of this head, and LN 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[5]), 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:

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

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.

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.

C.3 METHODOLOGY FOR FEATURE EDITING ON THE IOI TASK

A key question is whether the learned codes are causally relevant to the model's computation. To test this, we intervene on activations using the codes and check if we observe the expected effect on the model's predictions.

Methodology. We consider edits of the **IO**, **S** and **Pos** properties, as well as subsets of them.

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

Implementation. To edit multiple locations at once when running the model on a prompt p, we pre-compute the edited representation of model activations (using the activations when the model is ran on p without any edits), and then plug all of them simultaneously into a new run of the model on p using activation patching (Vig et al., 2020). In particular, this means that we never observe the effects of an edit in a given location on the edits performed in its downstream locations. Roughly speaking, we are instead only witnessing the effects of editing computational cross-sections of the circuit. We also experimented with applying the edits dynamically as the model is run, but found that this led to worse performance in some cases, especially due to the interaction between the name mover heads in layers 9 and 10.

Baseline and evaluation. To evaluate the quality of the edits, we use activation patching from ground-truth activations that capture the same information that our edits are intended to introduce. Specifically, we use activations from *counterfactual prompts*: prompts which differ from the original prompt only in the value(s) of the attribute(s) being edited. When performing an edit to multiple model locations, we compare the resulting computation with activation-patching the same locations from a run of the model on the counterfactual prompt.

D Additional figures for Supervised Feature Decompositions

Figure 4: Fraction of recovered logit difference for all methods to compute codes, across cross-sections of the circuit.

E Additional details for SAE training and evaluation

E.1 Feature editing with SAEs

We define a feature-agnostic way to evaluate how well SAEs can edit the **S**, **IO** and **Pos** attributes. Given an SAE reconstruction $\mathbf{r} := \widehat{\mathbf{a}} - \mathbf{b}_{dec} = \sum_j f_j \mathbf{d}_j$, we can assign a weight $w_j = f_j \mathbf{d}_j^\top \mathbf{r} / \|\mathbf{r}\|_2^2$ to each feature representing its contribution to the reconstruction, with $\sum_j w_j = 1$. Given a pair of prompts p, p' with activations $\mathbf{a}, \mathbf{a}'$ such that p' differs from p only in the values of the attributes we want to edit, to edit the activation $\mathbf{a}$ of p, we subtract the terms for features with weight $> t$ in $\mathbf{a}$ but

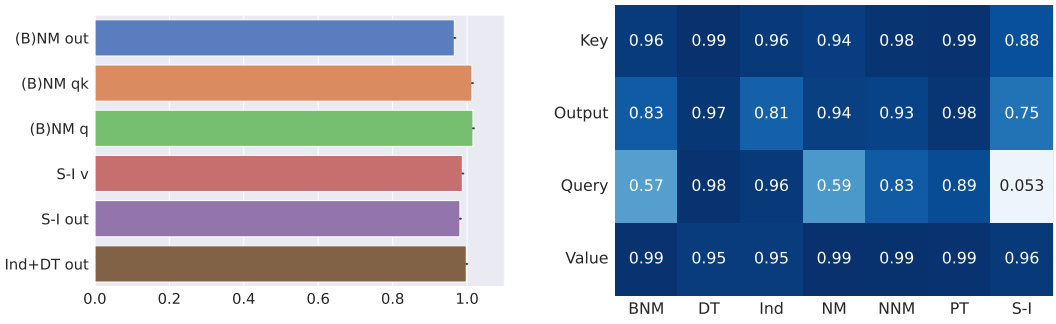

Figure 5: Measuring the approximation quality of the mean codes using the independent parametrization: fraction of (average) logit difference recovered when using the codes to reconstruct cross-sections of the circuit (left), and variance explained in each activation location, grouped by head class and attention component (right).

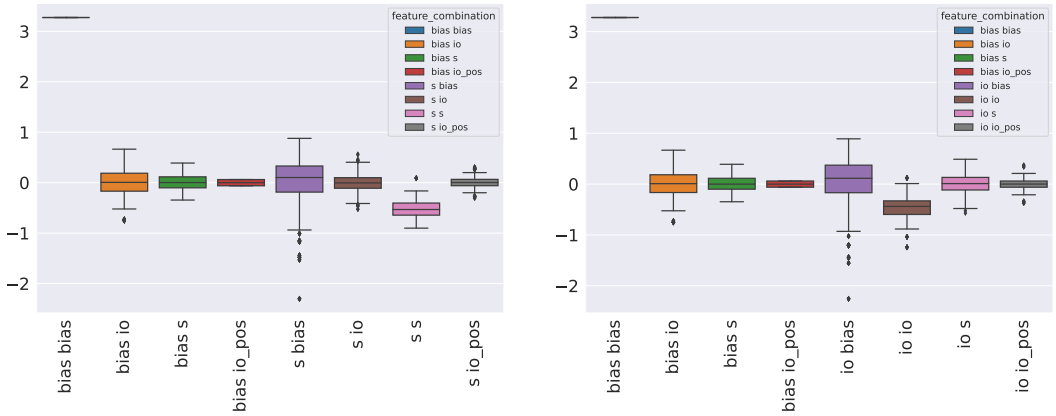

Figure 6: Decomposing the attention scores of the name mover head L10H0 from END to the S1 (left) and IO (right) positions.

not in $\mathbf{a}'$, and add the terms for features with weight $> t$ in $\mathbf{a}'$ but not in $\mathbf{a}$, where t is a threshold we can vary. We choose $t = 0.1$ for our experiments.

E.2 Task-specific SAE training and hyperparameters

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.

We use a training set of 20,000 examples and an evaluation set of 8,000 examples (for the purposes of autointerpretability, 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. A dictionary size of 512 (an $8\times$ increase over the dimensionality of attention head activations in GPT-2 Small), an effective ℓ_1 regularization of 1.0, and a batch size of 256 were good default choices. We keep the batch size deliberately small throughout

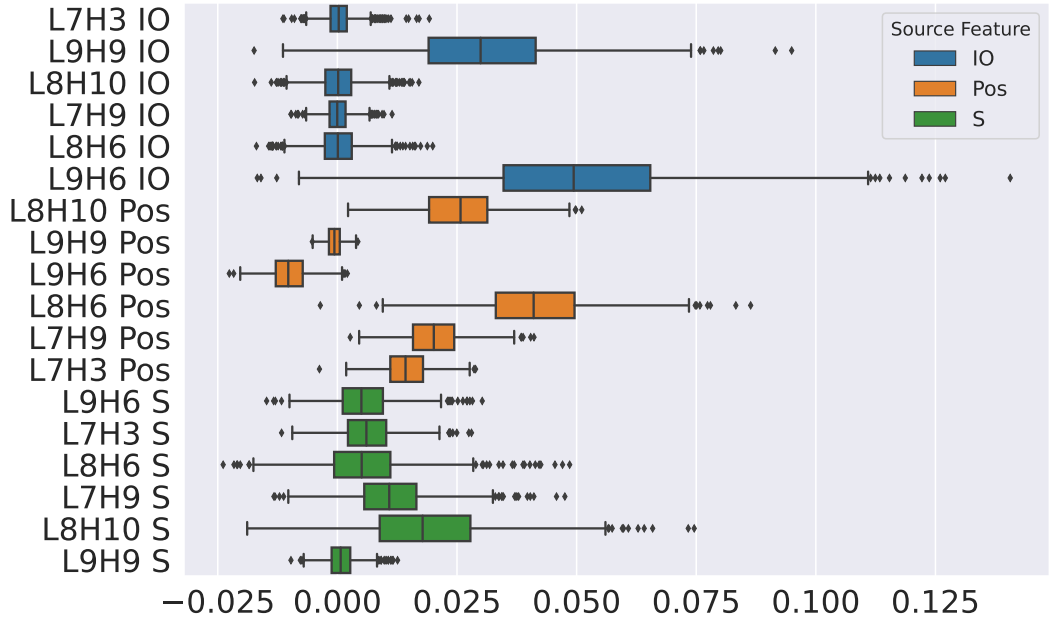

Figure 7: Decomposing the direct effect of the features in the output of s-inhibition heads, and name mover heads in layer 9, on the queries of the L10H0 name mover head at the end token (using mean codes with the independent parametrization).

hyperparameter searches to ensure data diversity in each batch, which may be important for good training (Bricken et al., 2023).

E.3 Examples of possible feature decompositions and their properties

The (default) supervised decomposition. It's worth first describing the properties of the supervised feature decomposition we constructed in Section 3, 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.

This decomposition has several desirable properties, in increasing order of usefulness for control and interpretability:

- As we have seen, the reconstructions are fairly faithful to the original activations and model computation;
- It can (unsurprisingly) 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.
- It is fairly interpretable: we can understand the meaning of each feature in terms of the attribute it represents.
- Moreover, using metrics such as sensitivity and specificity (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 would be

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

where $g : \mathbf{Names} \to \{M, F\}$ is some labeling function that roughly classifies names according to how they are typically gendered[6]. Here, we hypothesize that the model may have features of high norm that sort names into genders (which may be useful to the model for various reasons), and then add a small 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 sensitivity and specificity will pick up on the per-name features
- However, if we use the 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 will be low.

Using features for small 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 are possible by e.g. picking subsets at random or using expander graphs, and they may '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 change the features for other attributes.
- However, comparing our supervised decomposition against this one using 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 specificity 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 task-specific SAEs trained on the IOI circuit activations learn a decomposition resembling this abstract construction.

Memorizing decompositions. 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 very low sensitivity for the attributes;

E.4 Full-distribution SAE training and hyperparameters

We implemented the training of individual SAEs on all key components involved in the IOI task, including queries, keys, values, and head outputs. This process followed the methodology presented by Bricken et al. (2023), also incorporating their approach to resampling. As an additional preprocessing step, we sampled 10 million activations and computed their mean and norm. We subsequently standardized all activations with these metrics. This allowed us to compute the L1 and Mean Squared Error (MSE) losses using the standardized activations. Importantly, this preprocessing step does not modify the SAE function; rather, it decouples the L1 coefficient from the activation norm

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

(which widely varies between layers), facilitating consistent training across different layers with a unified L1 coefficient.

For training the SAEs, we extracted activations from the gpt2-small model, specifically selected from the OpenWebText dataset (Gokaslan & Cohen, 2019), which serves as a proxy for GPT-2's original training dataset. We sampled 10 million activations at a time from random contexts with a maximum sequence length of 512 and shuffled them before creating batches to ensure independence of activations within a batch. We then sampled batches of 2048 activations each for the training of the SAEs.

SAEs were trained for 250 million activations each, using an L1 coefficient of 0.006. To address non-responsive neurons — those that remained inactive over 5000 steps — we applied a resampling technique after the first 100 million activations as described by Bricken et al. (2023). The Adam optimizer was used, configured with a learning rate of 1e-3 and complemented by a linear learning rate warm-up over the first 3000 steps.

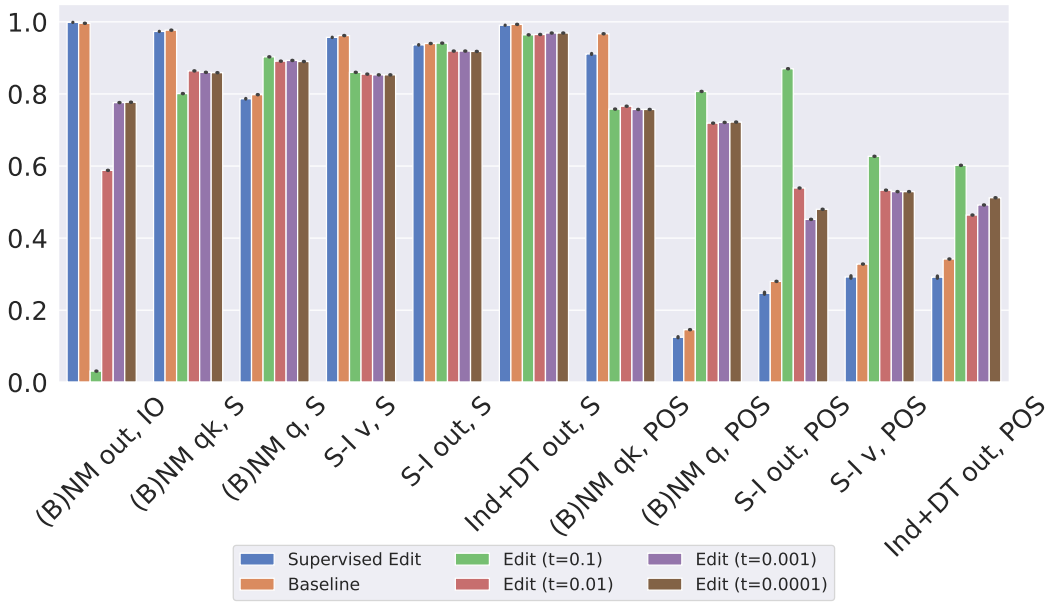

Figure 8: Edit accuracy at cross-sections of the IOI circuit, using our supervised feature decomposition (blue), a ground-truth baseline intervention (orange), and four methods for editing using full-distribution SAEs (green, red, purple, and brown)

F Related work

F.1 Learning and evaluating SAE features

The SAE paradigm predates the recent surge of interest in LLMs. Early work focused on the analysis of word embeddings (Mikolov et al., 2013), 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[7] 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

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

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).

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. 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).

F.2 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. (2023a) 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).