Sparse Autoencoders Match Supervised Features for Model Steering on the IOI Task

Aleksandar Makelov [1]

Abstract

Sparse autoencoders (SAEs) have attracted attention as a way towards unsupervised disentangling of hidden LLM activations into meaningful features. However, evaluations of SAE architectures and training algorithms have so far been indirect due to the difficulty – both conceptual and technical – of obtaining 'ground truth' features to compare against. To overcome this, recent work (Makelov et al., 2024) has proposed a suite of SAE evaluations that compare SAE features against feature dictionaries learned with supervision for a specific model capability. However, the evaluations were implemented in a mostly exploratory way, and did not optimize for eliciting best SAE performance across different SAE variants.

In this work, we improve upon this by running a systematic and thorough study of using SAEs for steering on the IOI task, comparing several recently proposed SAE variants: 'vanilla' SAEs (Bricken et al., 2023), gated SAEs (Rajamanoharan et al., 2024) and topK SAEs (Gao et al., 2024). We find that, even by employing a simple and cheap heuristic for choosing good SAEs for editing, we are able to greatly improve upon the results of prior work, and demonstrate that SAE features are able to perform on par with supervised feature dictionaries. Further, we find that topK SAEs and gated SAEs generally outperform other variants on this test, and topK SAEs can almost match supervised features in terms of edit quality.

1. Introduction and Related Work

The *linear representation hypothesis* (Mikolov et al., 2013; Grand et al., 2018; Li et al., 2021; Abdou et al., 2021; Nanda et al., 2023) is a central organizing principle of mechanistic interpretability research. One version of this hypothesis is that LLM activations can be decomposed into sparse linear combinations of features from a large, shared *feature dictionary*. Recently, a series of works has 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; Rajamanoharan et al., 2024; Templeton et al., 2024).

While initial results are promising, they rely on qualitative and/or indirect evaluation of the learned features such as proxies for the 'true' features, non-trivial assumptions about SAE learning or success in toy models (Elhage et al., 2022; Bricken et al., 2023; Sharkey et al., 2023). As a step towards more objective SAE evaluations, recently Makelov et al. (2024) proposed to use sparse feature dictionaries learned with *supervision* in the context of a given model capability (specifically, the IOI task (Wang et al., 2023)) as a 'skyline' for achievable SAE performance w.r.t. this capability. They developed several evaluations that (1) confirm the supervised features provide a high-quality decomposition of model computations w.r.t the capability and (2) use these supervised features to contextualize SAE results, for SAEs trained on distributions of either capability-specific or internet text.

In this paper, we focus on, and methodologically improve upon, one of the evaluations proposed by Makelov et al. (2024): *sparse controllability*. It evaluates the degree to which we can add/remove a small number of SAE features in order to change the intermediate 'variables' represented by the supervised dictionaries developed by Makelov et al. (2024) for the IOI task[1]. Our main contributions are:

- Most importantly, we sweep over hyperparameters to find, for each location in the IOI circuit, the best SAE within each variant for editing that location. *This allows us to probe the limits of SAEs for controlling the model*, and reveals that some SAE variants, such as topK SAEs (Gao et al., 2024), can perform on par with

[1]Independent. Correspondence to: Aleksandar Makelov <aleksandar.makelov@gmail.com>.

[1]While not explicitly recognized by the authors, the sparse control evaluation is essentially a comparison between SAEs and a special case of another popular approach for LLM control, *steering vectors* (Rimsky et al., 2023; Zou et al., 2023).

supervised features for control in the IOI task.

- We improve upon the SAE training methodology of Makelov et al. (2024) by comparing several SAE variants and using various training tactics suggested in the literature.
- We investigate the correlation between various proxy metrics of SAE quality used in the literature (such as loss recovered and ℓ_0 loss) and success in controlling the model. We also evaluate the similarity of the features learned by the best SAEs for editing across the three variants to the supervised dictionaries.

2. Preliminaries

Vanilla SAEs. We begin by describing 'vanilla' SAEs, following Bricken et al. (2023). A 'vanilla SAE' for the purposes of this paper 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

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

and a reconstruction

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

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*. The decoder directions are constrained to have unit norm: $\left\|(W_{dec})_{:,i}\right\|_2 = 1$. The training objective over examples $\{\mathbf{a}^{(k)}\}_k$ is $\sum_k \left(\left\|\mathbf{a}^{(k)} - \widehat{\mathbf{a}}^{(k)}\right\|_2^2 + \lambda \left\|\mathbf{f}^{(k)}\right\|_1\right)$ where λ is the ℓ_1 regularization coefficient.

Gated SAEs. Proposed recently by Rajamanoharan et al. (2024), gated SAEs replace the encoder part of an SAE with two parallel paths: one determines which features to activate, and the other computes the magnitudes of these features. This is done in order to avoid the shrinkage of reconstructions inherent to the ℓ_1 regularization in vanilla SAEs; we refer the reader to Rajamanoharan et al. (2024) for more details on the architecture and loss function.

topK SAEs. Proposed recently by Gao et al. (2024), topK SAEs replace the ℓ_1 regularization in vanilla SAEs with a topK activation function: $\mathbf{f} = \mathrm{TopK}\left(W_{enc}(\mathbf{a} - \mathbf{b}_{dec})\right)$, where TopK zeroes out all but the k highest entries of its input; the reconstruction and loss are then computed as for vanilla SAEs.

The promise of topK SAEs is that they can overcome the shrinkage problem by dropping the ℓ_1 penalty, and at the same time be easy to tune, because the sparsity is directly controlled by the hyperparameter k.

The IOI task, and activation editing with supervised feature dictionaries. The IOI task (Wang et al., 2023) is a simple 'algorithmic' language task, where an LLM is prompted with 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'). Wang et al. (2023) discovered that the model uses a circuit of attention heads to solve this task, with several classes of heads performing specific subtasks of the overall algorithm (Appendix Figure 4). See Appendix A for more details on the IOI task and the precise dataset we use for it.

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). We will think of **IO**, **S** and **Pos** as the 'prompt attributes' in the IOI task: functions which take a prompt p as input, and output the value of the corresponding attribute of p.

It is shown in Makelov et al. (2024) that the activation $\mathbf{a}(p)$ of a prompt p at a given site (e.g., output of some attention head) in the IOI circuit can be well-approximated using a supervised feature dictionary which has features $\{\mathbf{a}_{\text{IO}=v}\}$, $\{\mathbf{a}_{\text{Pos}=v}\}$ and $\{\mathbf{a}_{\text{S}=v}\}$, one for each possible value[2] v of each of the three prompt attributes **IO**, **Pos** and **S**, so that

$$\mathbf{a}(p) \approx \overline{\mathbf{a}} + \mathbf{v}_{\text{IO=IO(p)}} + \mathbf{v}_{\text{Pos=Pos(p)}} + \mathbf{v}_{\text{S=S(p)}}$$

where $\overline{\mathbf{a}} = \mathbb{E}_{p'\sim\mathcal{D}}\left[\mathbf{a}(p')\right]$ is the expected activation over the entire IOI distribution. Specifically, a good choice for the supervised dictionaries is to simply take the average activation over the training set for each value of each attribute (and subtract the mean activation):

$$\mathbf{v}_{\text{IO}=v} = \mathbb{E}_{p\sim\mathcal{D}}\left[\mathbf{a}(p) | \mathbf{IO}\,(p) = v\right] - \overline{\mathbf{a}}.$$

They further show that 'activation arithmetic' (Mikolov et al., 2013) can be used to edit the model's internal representations of *individual* attributes by adding or removing features from the supervised dictionaries. For example, to change the model's internal representation of **IO** from 'Mary' to 'Alice', we can form the edited activation

$$\mathbf{a}_{edited} = \mathbf{a} - \mathbf{v}_{\text{IO=Mary}} + \mathbf{v}_{\text{IO=Alice}}.$$

[2]In the IOI dataset used in this work, there are 216 possible values for each of **IO** and **S**, and 2 possible values for **Pos**.

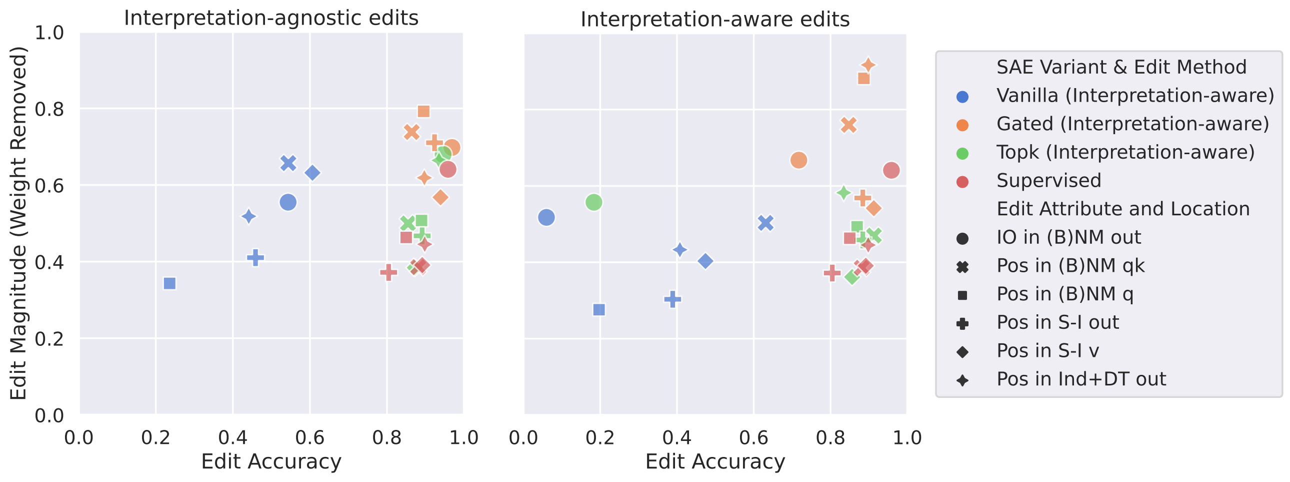

Figure 1. Main results from our study, showing the trade-offs between edit accuracy and edit magnitude (better is down and to the right). **Left**: results in the interpretation-agnostic editing regime. **Right**: results in the interpretation-aware editing regime. Each color represents an SAE variant, and each shape a cross-section of the IOI circuit (see Appendix A for more details on cross-sections). See Section 3.2 for more details on the two editing regimes.

3. Methods

3.1. SAE Training Methodology

The most important changes we make compared to the training methodology of Makelov et al. (2024) are various training tactics suggested in the literature, such as resampling dead neurons followed by a warmup of the learning rate (for vanilla and gated SAEs only), and a cooldown of the learning rate at the end of training. Details are given in Appendix C.

3.2. Sparse Controllability Evaluation

We faithfully follow the methodology of Makelov et al. (2024) for the sparse controllability evaluation. Specifically, we edit the **IO** and **Pos** attributes in cross-sections of the IOI circuit[3]. We use both the interpretation-agnostic and interpretation-aware editing methods (described in more detail in section 5.1. of Makelov et al. (2024)), as explained below. Furthermore, to do edits, we restrict ourselves to only subtracting a single SAE feature from the original activation, and adding a single SAE feature from the counterfactual activation. This is a fair comparison, since it matches the expressive power of the supervised dictionaries, which have exactly one feature per attribute. This is a departure from the methodology of Makelov et al. (2024), who varied the number of exchanged features beyond one. We focus on this regime because we found that, when we pick good SAEs, this is sufficient to achieve good results.

[3]We chose not to edit **S**, as it was shown in Makelov et al. (2024) that that the effect of ground-truth edits to **S** is very small, making evaluation noisy.

Counterfactual prompts. Suppose we're given an original activation $\mathbf{a}_s$ we want to edit, and a *counterfactual* activation $\mathbf{a}_t$ that was computed using a prompt capturing the edit we want to make. For example, suppose the original prompt is 'When Mary and John went to the store, John gave a book to', and we want to change the **IO** attribute from 'Mary' to 'Alice'. Then the counterfactual prompt would be 'When Alice and John went to the store, John gave a book to'.

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 $S, T \subset \{1,\dots,m\}$ and $\alpha_i, \beta_i > 0$.

Choosing the features for an edit. To obtain a fair comparison between SAEs and supervised dictionaries, we subtract a single feature active for the original activation $\mathbf{a}_s$, and add a single feature active for the counterfactual activation $\mathbf{a}_t$. The question is how to pick these features, which we describe next.

Interpretation-agnostic editing. The goal of this method is to measure the usefulness of a sparse feature dictionary for editing *independent of any human interpretation of the features*; this aims to achieve a more unbiased and objective assessment, and is motivated at length in Makelov et al. (2024). Here, we cast the editing problem as a combinatorial optimization over features to subtract/add in feature arithmetic. Specifically, consider the optimization problem

$$\min_{i \in S, j \in T} \|(\mathbf{a}_s - \alpha_i \mathbf{u}_i + \beta_j \mathbf{u}_j) - \mathbf{a}_t\|_2$$

In words, this problem asks for a feature to remove (i) from the original activation and a feature to add (j) from the counterfactual one to bring it as close as possible to the counterfactual activation, where the features to add are taken directly from the counterfactual one. We solve the problem by greedily (w.r.t. the objective) picking the first of the two features to exchange, and then the other.

Interpretation-aware editing. The goal of this method is to measure the usefulness of a sparse feature dictionary for editing *from a human-interpretable perspective*. Specifically, we look at the features active in the sparse reconstructions of $\mathbf{a}_s$ and $\mathbf{a}_t$, and remove/add features based on their F_1 score w.r.t. the value of the attribute being edited. See Appendix B for more details on how the F_1 score is computed and some of the subtleties of this method.

On the differences between the two editing regimes. It is important to understand how the two editing regimes differ. The interpretation-agnostic regime is more unrestricted, because it allows more flexibility in the choice of which features to add/remove per a pair of original and counterfactual activations. By contrast, the interpretation-aware regime requires going over the active features in a pre-determined order that is uniform over the dataset, as given by decreasing F_1 score for the original value of the attribute being edited (when choosing the feature to subtract) and the counterfactual value (when choosing the feature to add). This makes the interpretation-aware regime more constrained, but also more approachable and useful from a human perspective.

Evaluation metrics. Both of these methods are evaluated along two main axes (same as in Makelov et al. (2024)), explained below.

Edit accuracy. This metric measures agreement of next-token predictions against a 'ground-truth' edit. Specifically, given an edit (e.g., change the **IO** attribute), accuracy is the fraction of prompts for which performing the edit leads to the same next-token prediction as when we instead replace activations by the activations of counterfactual prompts corresponding to the edit.

Edit magnitude. This metric measures the 'damage' to the representation done by the edit. It is important to measure this, because editing would be trivial if one can simply remove all features from the reconstruction, and add back all features from the counterfactual activation. Specifically, given a reconstruction of the original activation

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

we measure the magnitude of editing out the feature $\alpha_i \mathbf{u}_i$ via the weight of the feature, defined as

$$\text{weight}\,(i) = \frac{\alpha_i \mathbf{u}_i^\top \left(\widehat{a}_s - \mathbf{b}_{dec}\right)}{\|\widehat{a}_s - \mathbf{b}_{dec}\|_2^2}.$$

The weight measures the contribution of the feature to the reconstruction, and the weights of all active features sum to 1: $\sum_{i \in S} \text{weight}\,(i) = 1$. While technically weights can be negative or greater than 1, it was observed in Makelov et al. (2024) that this rarely happens in practice. Finally, note that we only report the weight of the feature removed during the edit; in general, the weight of the feature added shows simliar trends.

Choosing SAEs to use for editing. In a major departure from the methodology of Makelov et al. (2024), we do not pick the SAEs to use for editing each circuit location in an ad-hoc way based on the ℓ_0 and logit difference recovered metrics. Instead, we directly choose SAEs best for editing.

How do we determine which of several SAEs for a circuit location is best for editing? Ideally, we would perform edits using each SAE, and evaluate similarity of the final model output to the output when activation patching counterfactual activations at the same attention head. Due to time constraints, we instead used geometric measures to evaluate how close to the target counterfactual activation $\mathbf{a}_t$ a given edited activation $\mathbf{a}_{edited}$ is. Specifically, we simply pick the SAE that minimizes the distance $\|\mathbf{a}_t - \mathbf{a}_{edited}\|_2$. We also experimented with using the attribution vector $\nabla_{\mathbf{a}_t}$ Loss to weight the distance between $\mathbf{a}_t$ and $\mathbf{a}_{edited}$, but found that this did not change results substantially.

We choose the available SAEs for each variant as follows. First, we take the last-epoch SAEs from each training run (we also tried optimizing the checkpoint epoch, but found this not to change results substantially), and then:

- for vanilla and gated SAEs (which use the ℓ_1 penalty), for each given location in the IOI circuit (e.g., the output of a given attention head), and for each attribute we wish to edit (**IO** and **Pos**), we pick the best ℓ_1 coefficient out of the four values $\lambda \in (0.5, 1.0, 2.5, 5.0)$ according to the above distance metric.
- for topK SAEs, where sparsity is hard-coded by the hyperparameter k, we **directly pick the SAE where** $k = 3$ for each location in the IOI circuit. This **puts topK SAEs at a relative disadvantage**, because we use our high-level understanding of the task and the supervised dictionaries for it to set $k = 3$ always (instead of optimizing over k across the four values we used $k \in (3, 6, 12, 24)$ for each circuit location). Still, we found that topK SAEs achieve very competitive (and almost always superior) results despite this limitation.

As mentioned, we always report results using the last epoch

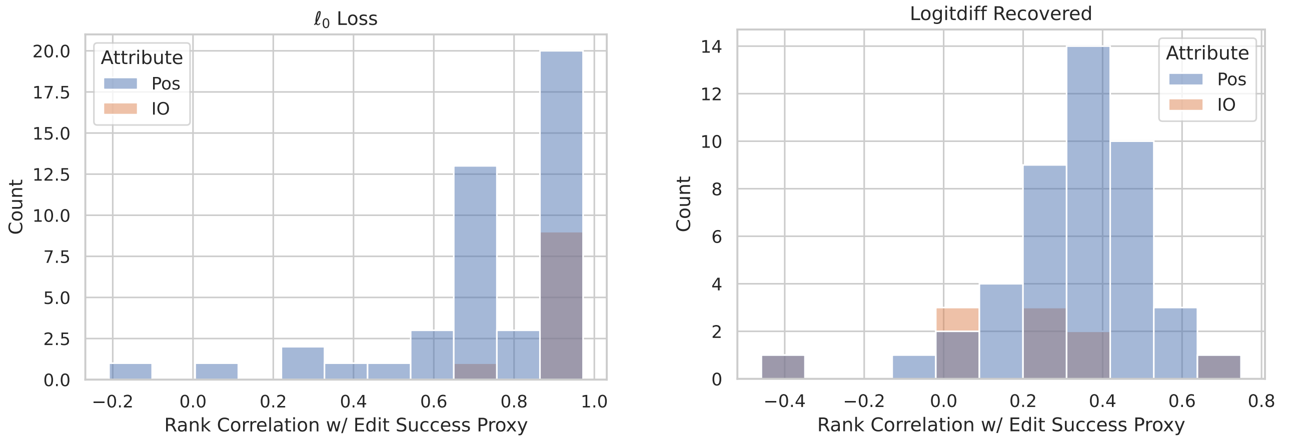

Figure 2. Distributions of rank correlations between our proxy for edit success (distance of the edited activation from the counterfactual activation) and two proxy metrics for SAE quality: ℓ_0 loss, i.e. the average number of active features per example (**left**) and the recovered fraction of the logit difference (**right**). Each datapoint represents a combination of a circuit location and a choice of ℓ_1 regularization. This plot only shows results for gated SAEs.

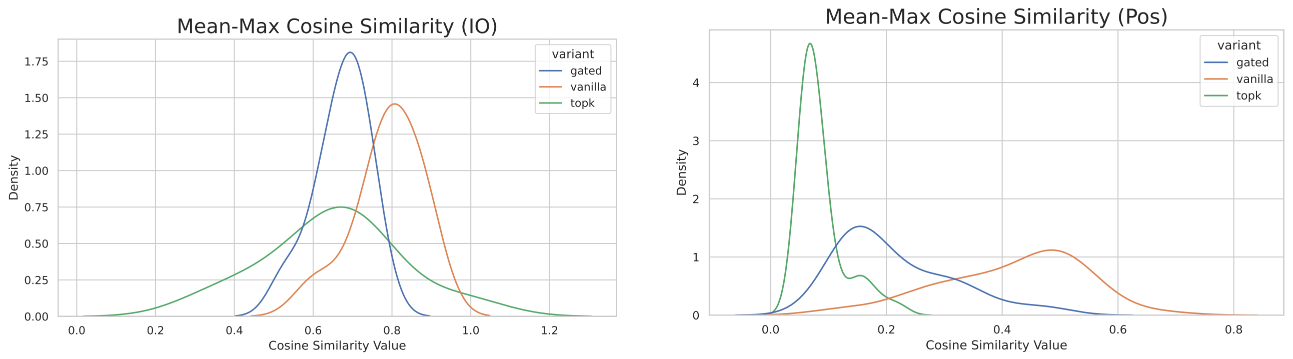

Figure 3. Distributions of the mean-max cosine similarity between the features in the best SAEs for editing across the three variants (vanilla, gated, topK) for the interpretation-agnostic editing regime. Here the datapoints are locations in the IOI circuit (smoothed with a kernel density estimate; values above 1 are artifacts of the smoothing).

of training for each SAE. We also tried optimizing over epochs, but found this not to improve results substantially.

4. Results

4.1. Main Editing Results

The main results of our study are shown in Figure 1. We find that, thanks to our careful choice of SAEs at each circuit node, we are able to greatly outperform the editing results from the original work Makelov et al. (2024):

- Remarkably, we find that **SAE features are able to perform on par with supervised features** for control in the IOI task, an important result for the field of interpretability.
- Specifically, topK SAEs almost match supervised features in the interpretation-agnostic editing regime, and also in the interpretation-aware regime, except for when editing the **IO** attribute (where they fail quite catastrophically, but gated SAEs do not).
- We further observe that topK and gated SAEs are far superior to vanilla SAEs, though gated SAEs fall a bit behind topK SAEs.

4.2. Some Additional Investigations

We were further motivated by two natural questions.

How do the traditional proxy metrics of SAE quality (such as loss recovered and ℓ_0 loss) correlate with success in controlling the model? We show rank correlation results for gated SAEs in Figure 2. We observe very good correlation with sparsity, and a moderate correlation with logit difference recovery. Both plots are created using the interpretation-agnostic editing regime.

How similar are the SAE features of the best SAEs to the supervised features in a direct geometric sense? We can measure this by evaluating the mean maximum cosine similarity between the set of SAE features for our best chosen SAEs, and the set of supervised features at each circuit location. Results are in Figure 3. We find that the cosine similarity is above chance (we work in 64 dimensions, so the chance similarity is $1/8$) for most variants and features, but also not very high. Interestingly, cosine similarity seems inversely correlated with edit success, which suggests that the best SAEs for editing may use diverse features across prompts.

5. Conclusion, Limitations, and Future Work

There are several possible improvements that would strengthen the signficance of this work. First, it would be interesting to investigate SAEs trained on the full pretraining distribution of GPT2-Small, and see how they compare to the IOI-specific SAEs we considered here. Second, a more direct evaluation when choosing which SAEs to use for editing, based on model behavior instead of geometric measures, would be an interesting addition. Finally, a much more ambitious avenue would be to investigate the use of SAEs for control in more complex tasks.

In conclusion, we have shown that SAEs can be used to control the model in the IOI task, and that some SAE variants can perform on par with supervised features for this task, which is an intriguing and perhaps unexpected success for unsupervised interpretability methods. It is exciting to see if these results can be extended to more natural and complex tasks.

Acknowledgements

The author would like to thank Georg Lange and Neel Nanda for valuable discussions informing the perspectives in the closely related work Makelov et al. (2024). The work in this paper was supported by the Long Term Future Fund. Two open source libraries were used to support the experiments for the paper: `transformerlens` (Nanda & Bloom, 2022)[4] and `mandala` (Makelov, 2024)[5].

References

Abdou, M., Kulmizev, A., Hershcovich, D., Frank, S., Pavlick, E., and Søgaard, A. Can language models encode perceptual structure without grounding? a case study in color. *arXiv preprint arXiv:2109.06129*, 2021.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Conerly, T., Templeton, A., Bricken, T., Marcus, J., and Henighan, T. Update on how we train saes. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/april-update/index.html#training-saes.

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

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

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

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., and Smith, N. A. Sparse overcomplete word vector representations. In *Annual Meeting of the Association for Computational Linguistics*, 2015. URL https://api.semanticscholar.org/CorpusID:9397697.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[4]https://github.com/TransformerLensOrg/TransformerLens

[5]https://github.com/amakelov/mandala

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

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

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

Makelov, A. Mandala: Persistent & queriable memoization for easy and powerful scientific data management. In *Proceedings of the 23rd Python in Science Conference*, 2024. To appear.

Makelov, A., Lange, G., and Nanda, N. Towards principled evaluations of sparse autoencoders for interpretability and control. *arXiv preprint arXiv:2405.08366*, 2024.

McGrath, T., Rahtz, M., Kramar, J., Mikulik, V., and Legg, S. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

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

Nanda, N. and Bloom, J. Transformerlens. https://github.com/neelnanda-io/TransformerLens, 2022.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

Olshausen, B. A. and Field, D. J. 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.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. M. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Sharkey, L., Braun, D., and Millidge, B. Taking the temperature of transformer circuits. 2023. URL https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-su

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. 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.

Yun, Z., Chen, Y., Olshausen, B. A., and LeCun, Y. 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.

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

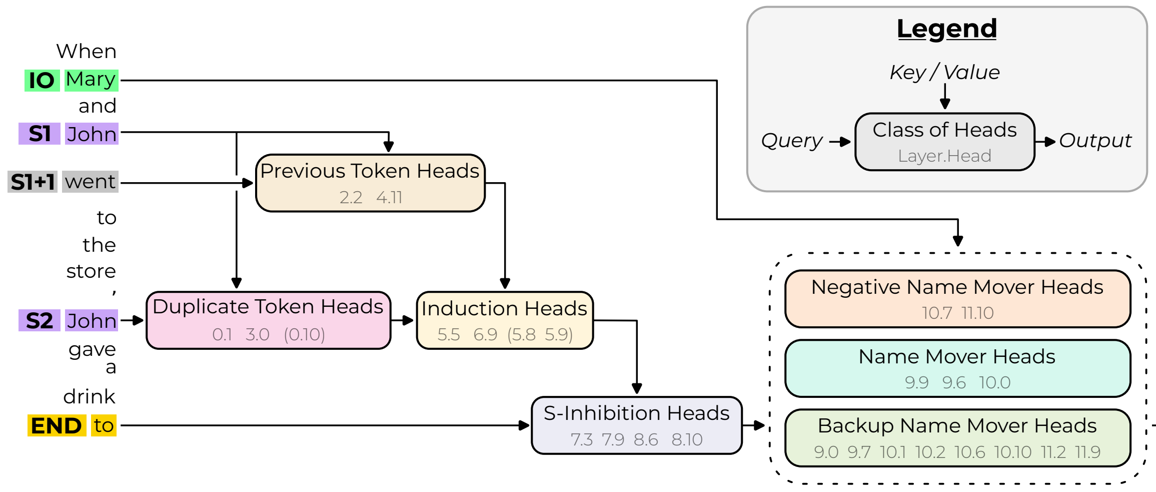

Figure 4. A reproduction of Figure 2 from Wang et al. (2023), showing the internal structure of the IOI circuit. Original caption: *The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.*

A. Additional Details on the IOI Task

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. 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[6]

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

B. 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 $\mathrm{recall}(F, A) = |A \cap F| \,/\, |A|$ and $\mathrm{precision}(F, A) = |A \cap F| \,/\, |F|$. 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:

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

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.

C. SAE training methodology

All SAEs are trained on the same IOI training set as in Makelov et al. (2024), consisting of 2×10^4 examples (see also Appendix A.1 for details on the dataset). Due to time constraints, we did not train sufficiently many SAEs on OpenWebText to be able to make a meaningful comparison.

Training schedule. We improve upon the ‘bare-bones’ SAE training methodology of Makelov et al. (2024) by incorporating several training tactics from recent literature. For all SAE variants considered, we used the same (small) learning rate of 3×10^{-4}, trained for 2000 epochs in total, and applied resampling followed by a learning rate warmup over 100 epochs (roughly following Rajamanoharan et al. (2024)) at epochs 501 and 1001. Our resampling methodology closely follows that of Bricken et al. (2023). In addition, we decay the learning rate linearly to zero over the last 25% of training (following Conerly et al. (2024)). For topK SAEs, we initialize the encoder to the transpose of the decoder, as suggested by Gao et al.

[6]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.

(2024); however, we do not use the auxiliary loss term suggested in section 2.4 ('Preventing dead latents') from that work (which is the only difference of this paper from the implementation of Gao et al. (2024)).

We checkpoint all models at 14 epochs: $(1, 2, 4, 8, 16, 32, 64, 128, 500, 750, 1000, 1250, 1500, 2000)$. This checkpoint schedule is chosen to ensure that we have a dense enough sampling of the early stages of training, while also capturing the state of the model right before resampling, and after the learning rate warmup that is done post-resampling is sufficiently in the past.

Preprocessing. We normalize all IOI circuit activations prior to passing them through our SAEs, following the scaling methodology in Conerly et al. (2024), so that they on average have ℓ_2 norm of $\sqrt{d_{head}}$. This helps us share hyperparameters across sites of the circuit, and reduces the range of hyperparameters to search over.

Hyperparameters. We sweep over values $\lambda \in (0.5, 1.0, 2.5, 5.0)$ for the ℓ_1 regularization penalty for vanilla and gated SAEs, and over values $k \in (3, 6, 12, 24)$ for topK SAEs. This is a reasonable range: we found that the highest value of λ leads to about 3-4 active features per example on average; the supervised dictionaries have 3. Conversely, the lowest value of λ lead to very good ℓ_2 loss and recover close to 100% of the logit difference, but have too many (about 20-30) active features per example on average. The range for k is chosen to include values equal and close to our expectation for the 'true' number of necessary features (3), while still allowing significantly more features to be active.