SPAR Summer 2024 Report: Circuit Phenomenology Using Sparse Autoencoders

David Udell [1] **Jackson Kaunismaa** [2] **Hardik Bhatnagar** [3] **Himadri Mandal** [4]

Abstract

Sparse autoencoders provide a means of projecting model activations into a more interpretable sparse vector space. With them, the field of mechanistic interpretability has taken to trying to understand the internals of large language models during training and inference. In particular, sparse autoencoder dimensions can be naturally assembled into *circuits* – directed graphs in which nodes are autoencoder dimensions and edges are their causal effects on each other. We looked at an unsupervised algorithm for recovering these circuits in prior work. In reimplementing that algorithm, we isolated a significant bug, with consequences for prior results. Since, we have built out two independent implementations of the circuit discovery algorithm for GPT-2-small (up from Pythia-70m) and are now continuing to work on tuning the graphing hyperparameters involved in graphing upsupervised forward passes.

1. Methods

We built upon prior work due to Marks et al. (2024), specifically focusing on their unsupervised circuit discovery algorithm using attribution patching (rather than integrated gradients, excluding any pre-clustering of forward passes using activation directions).

Naively, if you want to capture all the causal dynamics relating sparse autoencoder dimensions across layers, you'd want to compare unaltered forward passes to forward passes in which an autoencoder dimension is clamped to some value. That clamped forward-pass difference gives you the counterfactual contribution of that autoencoder dimension; the cost is that you do need that additional forward pass. So, naively, your time complexity for assembling a full causal graph between autoencoder dimensions for one forward pass goes as $\mathcal{O}(d_{\text{autoencoder}})$.

[1]udelldavidb@gmail.com [2]jackson.kaunismaa@mail.utoronto.ca [3]hrdk.bhatnagar@gmail.com [4]mandalhimadri06@gmail.com. Correspondence to: David Udell <udelldavidb@gmail.com>.

It's hard to get around this linear dependence on autoencoder width, when doing edge-level circuit discovery. Observe that while a *node* in a causal graph is an element in an activation vector (either in the neuron basis or in the autoencoder basis), an *edge* is an element in a *Jacobian*. What you're doing when you're doing edge-level circuit discovery is fundamentally "Jacobian shaped," so to speak.

One improvement is to start backwards, from the loss, rather than forwards, from the embedding. This way, right from the beginning, you can be sure to only look at causal effects that ultimately contributed to the loss that forward pass. If you're only plotting some subset of causal effects using a top-k or threshold value, you can take that subset directly from effects on the loss. Beginning at the embedding requires you to "speak the language" of activation magnitude differences instead, which aren't so immediately connected to the loss value; large magnitude changes in early layers need not connect to the model output and loss. Working backward from the loss, that is, lets you threshold efficiently. This is the approach taken in Marks et al. (2024).

The other significant algorithmic complication in Marks et al. (2024) is their correction for double-counting the contributions of residual connections. The double-counting occurs because dimensions at the layer N residual stream feed into both the layer $N+1$ attention sublayer *and* the layer $N+1$ residual stream (in the GPT-2 architecture). Asking about the effects of dimensions at the layer N residual stream on the layer $N+1$ residual stream – the residual connection edge – therefore end up counting the attention-to-MLP pathway too. You need to *first* calculate the edge contributions in the attention-to-MLP pathway, and subtract that from the residual-stream-N-to-residual-stream-$N+1$ pathway, to get accurate residual connection contributions. Naively, these threaten to themselves take $\mathcal{O}(d_{\text{autoencoder}})$, dragging up overall time complexity to $\mathcal{O}({d_{\text{autoencoder}}}^2)$. These corrections can in fact be done in a constant number of backwards passes, however, preserving the original linear time complexity. This extra complication and efficiently solving it accounts for much of the complexity of the unsupervised circuit discovery algorithm.

2. Results and Ongoing Work

A bug that we identified in the Marks et al. (2024) implementation was raised to the attention of the authors.

We now have two separate and independent reimplementations of the Marks et al. (2024) algorithm: here and here. Both are necessarily variations on the original since Pythia and GPT-2 differ in architecture (Pythia has parallel attention and MLP sublayers). As tuning graphing hyperparameters (as well as implementation more generally) has been a slow process, work on this will continue into the near future. Current blockers are ongoing manual hyperparameter search and some missing autoencoders in Neuronpedia, (Lin & Bloom, 2023).

Contributions

- David Udell: Team lead; reimplementing the circuit discovery algorithm; LaTeX
- Jackson Kaunismaa: Isolating and examining a serious bug in the Marks et al. (2024) codebase; reworking that codebase in a fork for GPT-2.
- Hardik Bhatnagar: Implementing the alternative integrated gradients circuit-discovery algorithm.
- Himadri Mandal: Work looking over the algorithmic details of Marks et al. (2024).

References

Lin, J. and Bloom, J. Neuronpedia: Interactive reference and tooling for analyzing neural networks with sparse autoencoders, 2023. URL https://www.neuronpedia.org. Software available from neuronpedia.org.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2024. URL https://arxiv.org/abs/2403.19647.