Token-Level Automatic Circuit Discovery in Language Models

Joseph Miller[1], **William Saunders**[2]

[1]FAR AI
[2]OpenAI
josephmiller101@gmail.com, williamrs@openai.com

Abstract

Mechanistic Interpretability research attempts to decipher black box models by identifying 'circuits' that implement particular capabilities. Previous work has introduced algorithms that automatically discover circuits in language models. These are 'component-level' circuits consisting of directed edges between attention heads and MLPs, affecting all token positions. We extend the automatic circuit discovery problem in transformers to consider 'token-level' circuits, where edges connect two components at a particular token position. This new problem asks us to discover not only which component interactions implement a behavior but also which tokens are used by each component. We extend several techniques to this problem and search for circuits that perform tasks studied in the literature. We compare the performance of the automatically discovered circuits and the manually discovered circuits from previous works. One of our new algorithms, 'Circuit Probing', which searches for a specific number of important edges via gradient descent, exceeds previous methods and finds circuits that outperform the human-discovered solutions on our benchmarks of faithfulness.

1 Introduction

Language models are rapidly progressing in capability (OpenAI 2023) and may pose extreme risks to society (Bostrom 2014), while our ability to understand and steer models remains primitive (Ngo, Chan, and Mindermann 2022). One hope of interpretability research is to open up these black boxes (Alishahi, Chrupała, and Linzen 2019) and enable greater control (Nanda 2022).

Mechanistic Interpretabilty tries to reverse engineer neural networks by studying small components and building up a low level understanding of the algorithms implemented in models (Olah 2022). Circuits are subgraphs of neural networks that perform a particular function (Olah et al. 2020). Identifying clean circuits that perform precise functions could be a useful step towards general understanding of neural networks. Automatic Circuit Discovery (ACDC) (Conmy et al. 2023) automated the process of finding circuits by performing causal interventions (Pearl 2009) for each edge. We build on this work by looking for more granular circuits in which edges only act between components at a particular token index, which we call token-level circuits.

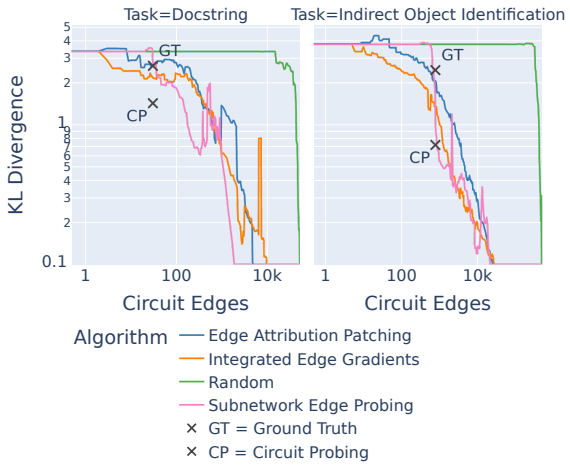

Figure 1: KL divergence from the model output for token-level circuits of different sizes on two natural language tasks. Circuits are tested with Tree Patching, so the x-axis shows the number of edges *not* being patched.

Our contributions are as follows:

1. We introduce the token-level circuit discovery problem.
2. We extend previous techniques to find token-level circuits. In particular, we introduce 'Circuit Probing', a variant of Subnetwork Probing (Cao, Sanh, and Rush 2021) that searches for circuits of a particular size using gradient descent.
3. We formalize a taxonomy of activation patching. We use 'Tree Patching' to compare token-level circuits, finding that 'Circuit Probing' outperforms other algorithms and human efforts.

2 Background

Any residual network can be rewritten as an equivalent network in which the output of each component is added directly to the input of each downstream component and the residual stream is removed. We refer to this as a factorized network. See Figure 2 for a visual illustration of this concept; Activation/Node Patching operates on the residual net-

work, whereas the other patching methods operate on the factorized network. Elhage et al. (2021) apply this to transformer models (Vaswani et al. 2017), in order to study the interactions of individual attention heads and MLPs.

2.1 Component-level Automatic Circuit Discovery

Conmy et al. (2023) present the automatic circuit discovery problem in language models. A circuit consists of a subset of the factorized network that implements some behavior or task, which is defined by a dataset of prompt-answer examples. These prompt-answer pairs are often accompanied by *corrupted* prompt-answer pairs that serve as useful counterfactual inputs that induce different answers from the model while retaining the structure of the input. This was inspired by previous work in mechanistic intepretability that found circuits manually (Wang et al. 2023; Heimersheim and Janiak 2023; Hanna, Liu, and Variengien 2023). Syed and Rager (2023) introduce Edge Attribution Patching based on Attribution Patching (Nanda 2023) as a fast approximation of ACDC. Surprisingly, they found that their method was in fact more accurate than ACDC.

3 Activation Patching Taxonomy

Activation patching (Vig et al. 2020), replaces activations in some component of a network with the equivalent activation on a different input. Several variants of activation patching have been used in mechanistic interpretability, with inconsistent terminology. We define a taxonomy for clarification (Figure 2).

Consider the computational graph of a factorized transformer; Node Patching, usually called simply Activation Patching, popularized by Meng et al. (2022), replaces the activations of nodes. Edge Patching, usually called Path Patching (Wang et al. 2023; Goldowsky-Dill et al. 2023) replaces the activations of edges. Circuit Patching (Wang et al. 2023), which has not previously been named, patches every edge *not* present in a circuit. And finally Tree Patching (Wang et al. 2023) (also not previously named) is equivalent to Circuit Patching, except it considers the computation graph at each token position separately. We name it Tree Patching because there will always be a single output node at the final token position and usually several input nodes from all of the relevant tokens. Circuit Patching and Tree Patching can be viewed as an approximate implementation of Causal Scrubbing (Chan et al. 2022) or Causal Abstraction (Geiger, Potts, and Icard 2023).

Note that we have not specified the granularity of components or the types of inputs being run. For example, we could run a corrupt input and perform neuron-level Circuit Patching with clean activations. Or we could run a clean input and Path Patch mean activations between attention heads. Tree Patching depends upon the particular architecture of a transformer, while the other types of patching are more general.

4 Related Work

We borrow the notation of previous authors to describe existing circuit discovery algorithms.

4.1 ACDC

ACDC (Conmy et al. 2023) uses Edge Patching to identify important nodes. Specifically, ACDC iterates through each edge in the network in reverse topological order and patches in a corrupt activation. If the overall output of the network changes less than some constant τ then the edge is deemed unimportant and discarded. Otherwise it is retained. The circuit identified is the set of edges that remain after ACDC has iterated through the entire graph.

We do not evaluate the ACDC algorithm in this work because it would be very slow to compute. ACDC has time complexity linear in the number of edges in the computational graph and each token position in a token-level circuit has the same set of edges as a component-level circuit. As ACDC is already far slower than all the other methods on the component-based circuit discovery problem and it is not the most accurate according to previous work (Syed and Rager 2023), we do not think it is necessary to include.

4.2 Edge Attribution Patching

Edge Attribution Patching (Syed and Rager 2023) computes the importance of each edge in a single forward/backward pass. For each component in the model, they find the gradient of $F(x_{clean})$, the logit difference between the correct and incorrect[1] tokens when the model is run on a clean input. Then for each edge, they multiply the gradient of the destination node by the difference in output of the source node on the clean and corrupt input:

$$(e_{corr} - e_{clean})^\top \frac{\partial F(x_{clean})}{\partial e_{clean}}$$

where e_{clean} and e_{corr} are the clean and corrupt edge activations (ie. source node outputs). This is a first order approximation of Edge Patching - so this method is a fast approximation of ACDC. But surprisingly, it outperforms all previous methods, including ACDC (Conmy et al. 2023). Note that this algorithm is a variation of Head Importance Scoring (Michel, Levy, and Neubig 2019), one of the baseline methods adapted by Conmy et al. (2023), using the difference in edge activations instead of the difference in node activations. See Appendix B, for another fomulation of Edge Attribution Patching.

In our experiments, we adapt Edge Attribution Patching to token-level circuit discovery by considering each token to have a separate set of edges and then running the algorithm as normal.

4.3 Subnetwork Probing

Conmy et al. (2023) apply Subnetwork Probing (Cao, Sanh, and Rush 2021) as the other baseline circuit discovery

[1]The incorrect token is usually defined as a plausible but wrong answer which could be output if the model cannot perform the task, but is merely following generic language heuristics to compute the output. Taking an example from the Indirect Object Identification task, given the prompt "When Jason and Charlie went to the school, Charlie gave a snack to", the incorrect token would be `" Charlie"`.

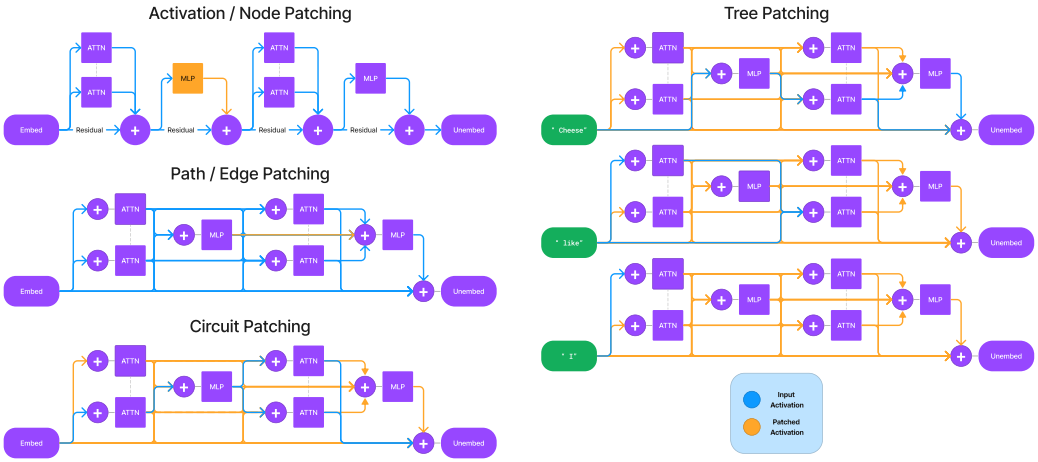

Figure 2: Activation Patching Taxonomy. Node patching (Vig et al. 2020) and Edge patching (Wang et al. 2023) replace the activations of nodes and edges respectively. Circuit Patching (Wang et al. 2023) patches the edges in the complement of the circuit. Tree Patching (Wang et al. 2023) applies Circuit Patching at each token position separately.

method in their experiments. For each component, Subnetwork Probing learns a value θ that parameterizes a stochastic mask $Z \in [0, 1]$. The mask is sampled from a HardConcrete distribution (Maddison, Mnih, and Teh 2016; Louizos, Welling, and Kingma 2017; Jang, Gu, and Poole 2016) once for each batch element in each forward pass. The training objective is to minimize KL divergence from the unmasked network while maximizing the number of masked components. After training, the masks are binarized and all edges between unmasked components are included in the circuit.

5 Methodology

We extend the problem of automatic circuit discover to consider 'token-level' circuits in which each edge connects two components at on the same token.

We introduce three new algorithms for token-level automatic circuit discovery, based on previous techniques.

5.1 Subnetwork Edge Probing

We introduce Subnetwork Edge Probing, a simple extension of Subnetwork Probing. Instead of learning a mask over components (nodes) in the network, we instead learn a mask over edges (while considering each token position to have a separate set of edges).

5.2 Circuit Probing

We introduce Circuit Probing, a variation of Subnetwork Edge Probing that searches for circuits of a particular size. Subnetwork Probing calculates the regularization term as the mean probability that each mask is non-zero, $\sigma(\theta - k)$, where θ is the mask parameter and k is a constant.

Circuit Probing alters this term to encourage a solution with a particular number of edges S. In each forward pass we sample the masks $\mathbf{Z}$ and calculate:

$$\frac{ReLU(|\mathbf{Z}| - S)}{S}$$

As with Subnetwork Probing we multiply the regularization term by a hyper-parameter λ.

5.3 Integrated Edge Gradients

We introduce Integrated Edge Gradients, based on the Integrated Gradients method (Sundararajan, Taly, and Yan 2017). For each edge, we find the integral over α of the gradient of the edge activation, where α parameterizes a linear interpolation between the clean and corrupt activations, e_{clean} and e_{corr}.

$$(e_{clean} - e_{corr})^\top \int_{\alpha=0}^{1} \frac{\partial L(e_{corr} + \alpha \times (e_{clean} - e_{corr}))}{\partial e} d\alpha$$

We approximate the integral using 50 samples of the gradient, evenly distributed between $\alpha = 0$ and $\alpha = 1$, again considering each token to have separate edges.

6 Recovering Ground Truth Circuits

Following Conmy et al. (2023) and Syed and Rager (2023) we test our circuit discovery algorithms using tasks for which circuits have previously been discovered through manual effort. Unfortunately, we cannot use the Greaterthan task (Hanna, Liu, and Variengien 2023) because the circuit described does not specify at which token positions each component is acting. However, we can use the Indirect Object Identification (IOI) task (Wang et al. 2023) and the Docstring task (Heimersheim and Janiak 2023) as we do have this information. The IOI circuit in GPT-2 performs the function of identifying the correct name to complete a sentence with an indirect object. For example: "When Jason and Charlie went to the school, Charlie gave a snack to". The Docstring circuit was identified in a small attention-only language model and performs the function of identifying the correct parameter name in a Python function docstring.

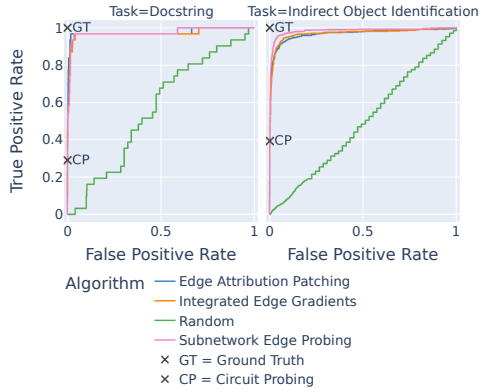

Figure 3: ROC curves for token-level circuits for two natural language tasks. The ground truth is taken from previous works that manually studied these tasks (Heimersheim and Janiak 2023; Wang et al. 2023).

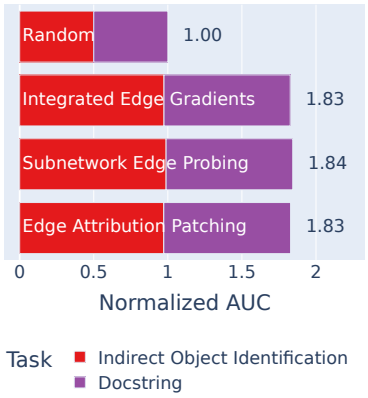

Figure 4: Normalized Area Under ROC curve for token-level circuits. Areas are normalized by dividing by the area under the random curves.

6.1 ROC Curves

We use our circuit discovery algorithms to obtain an attribution score for each edge in the computational graph and rank the edges accordingly. Again following Conmy et al. (2023), we plot the receiver operating characteristic (ROC) curve of true-positive rate against false-positive rate, using the manually discovered circuits as the ground truth.

Figure 3 shows the ROC curve for our algorithms along with a random baseline. Figure 4 shows the area under each curve (AUC). We normalize each area by the AUC of the random curve for the corresponding task. Then we divide by the number of tasks such that the random baseline has normalized area equal to 1.0.

At first glance our methods appear to do very well. In particular the AUCs are significantly higher than the corresponding curves for the component-level circuits found by Conmy et al. (2023) and Syed and Rager (2023). This suggests that, surprisingly, token-level circuit discovery may be an easier problem, even though there are far more edges to rank in importance.

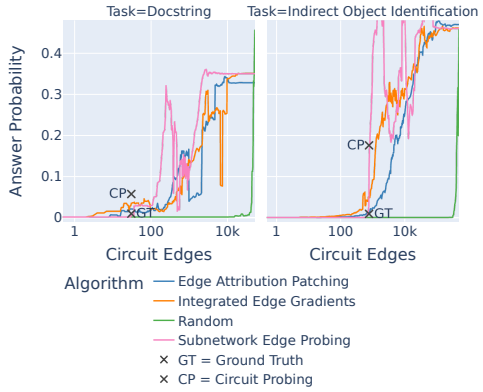

Figure 5: Probability of the correct answer for token-level circuits of different size on two natural language tasks. Circuits are tested with Tree Patching, so the x-axis shows the number of edges *not* being patched.

On the other hand, it is difficult to know how to compare the AUCs between the token-level task and the component-level task. There are far more possible edges in the token-level task, so the same False Positive Rate corresponds to a much greater number of incorrect edges. We see that Circuit Probing discovers circuits of the same size as the ground truth with less than half of the same edges. In Appendix A, we offer a visualization of the top 30 edges discovered by Edge Attribution Patching, and compare these to the manually discovered circuit (which itself consists of about 30 edges).

Ultimately, there is a limit to the extent that we can assess our algorithms using ROC curves. The manually discovered circuits are not really ground-truths because they are subject to human error and even according to metrics used by the researchers who discovered them, they do not fully explain the behavior of the models on their respective tasks.

Therefore we evaluate other metrics to measure the faithfulness of our circuits more thoroughly.

7 Testing our Circuits with Tree Patching

We use our circuit discovery algorithms to rank each edge in the factorized model and then perform Tree Patching on circuits of different sizes for all of our algorithms. First we measure KL divergence from the un-patched model run on a clean input. We plot this against the number of edges in the circuit in Figure 1. Circuit Probing and Integrated Edge Gradients achieve a lower KL divergence than the ground truth on both tasks.

Next we measure the probability of the correct answer. Figure 5 shows that again, Circuit Probing and Integrated Edge Gradients decisively outperform the human solution on both tasks.

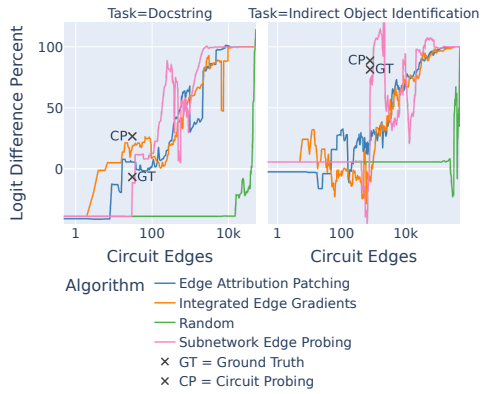

Figure 6: Percentage of the default logit difference between the correct and incorrect tokens recovered by token-level circuits of different size on two natural language tasks. Circuits are tested with Tree Patching, so the x-axis shows the number of edges *not* being patched.

7.1 Recovered Logit Difference

(Wang et al. 2023) evaluate their circuit for the Indirect Object Identification task using the average logit difference $F(C)$ between the correct (indirect object) and incorrect (subject) names when the nodes outside of the circuit are ablated to their mean value on the distribution. They use the percent of the logit difference on the un-patched model that is recovered by the circuit as their measure of faithfulness.

We test a very similar metric by finding the percentage of the logit difference that is recovered when we Circuit Patch the edges of our solutions. The results are shown in Figure 6. Integrated Edge Gradients performs poorly for the IOI task, but once again Circuit Probing finds a more faithful circuit than the ground truth in both instances.

7.2 Language Models are Surprisingly Linear

Prior to the experiment, we expected Integrated Edge Gradients to outperform Edge Attribution Patching. It seems advantageous to sample the gradient at multiple points between the clean and corrupt inputs, rather than using just the gradient at the clean input. However, the results indicate that both algorithms perform fairly similarly. This suggests that our model is quite linear between our clean and corrupt inputs.

8 Limitations

Our experiments are limited by the small number of tasks in language models that have been studied by mechanistic interpretability researchers. We require a complete token-level circuit as a ground truth, which requires considerable effort to find using manual techniques, and we are only aware of this being done for the two tasks discussed above.

8.1 Minimality and Completeness

Our metrics mostly measure the *faithfulness* of our circuits (ie. the degree to which they mimic the behavior of the full model on the given distribution). Two other desiderata for circuits could be *completeness* and *minimality* (Wang et al. 2023).

Minimality asks that we include no unnecessary edges in the circuit. While this can be approximated for most of our methods by studying the curves of edge counts against performance, our best method, Circuit Probing, aims at a particular circuit size, so cannot be used directly to discover the necessary size. We could run it several times for different sizes and compare the accuracy, but this would be very computationally expensive.

Completeness asks that we include all parts of the model that are relevant to the task, even those that inhibit performance or are inactive unless other heads fail to activate. We expect that KL divergence will capture some of this criteria because it penalises over-prediction of the answer tokens and therefore may reward the inclusion of inhibiting components. And because it requires the logits of all tokens to be close to the default output, KL divergence incentivizes all components to remain on distribution. However, we leave it to future work to rigorously establish the completeness of automatically discovered circuits.

9 Conclusion

We have introduced token-level circuit discovery as a step towards more precise mechanistic understanding of language models. We adapt and extend existing techniques to solve this challenge. We measure the faithfulness of our circuits using Tree Patching and find that Circuit Probing uncovers more accurate circuits for these tasks than all other algorithms and also exceeds the circuits found manually by human researchers.

10 Acknowledgments

Thanks to Arthur Conmy for his generous assistance in understanding and reproducing his work on Automatic Circuit Discovery. Thanks to Clement Neo, Adam Gleave, Steven Bills, Adrià Garriga-Alonso and Bilal Chughtai for their invaluable feedback and suggestions.

11 Impact Statement

This paper presents work whose goal is to advance the understanding of black box large language models, with the hope that this line of research can be used to reduce the risk of misaligned AI systems from subverting the goals of their creators.

References

Alishahi, A.; Chrupała, G.; and Linzen, T. 2019. Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. *Natural Language Engineering*, 25(4): 543–557.

Bostrom, N. 2014. *Superintelligence: Paths, Dangers, Strategies*. USA: Oxford University Press, Inc., 1st edition. ISBN 0199678111.

Cao, S.; Sanh, V.; and Rush, A. 2021. Low-Complexity Probing via Finding Subnetworks. In *Proceedings of the*

2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 960–966. Online: Association for Computational Linguistics.

Chan, L.; Garriga-Alonso, A.; Goldowsky-Dill, N.; Greenblatt, R.; Nitishinskaya, J.; Radhakrishnan, A.; Shlegeris, B.; and Thomas, N. 2022. Causal Scrubbing: a method for rigorously testing interpretability hypotheses [Redwood Research].

Conmy, A.; Mavor-Parker, A. N.; Lynch, A.; Heimersheim, S.; and Garriga-Alonso, A. 2023. Towards Automated Circuit Discovery for Mechanistic Interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*.

Geiger, A.; Potts, C.; and Icard, T. 2023. Causal abstraction for faithful model interpretation. *arXiv preprint arXiv:2301.04709*.

Goldowsky-Dill, N.; MacLeod, C.; Sato, L.; and Arora, A. 2023. Localizing Model Behavior with Path Patching. arXiv:2304.05969.

Hanna, M.; Liu, O.; and Variengien, A. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. arXiv:2305.00586.

Heimersheim, S.; and Janiak, J. 2023. A circuit for Python docstrings in a 4-layer attention-only transformer.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Louizos, C.; Welling, M.; and Kingma, D. P. 2017. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 14014–14024.

Nanda, N. 2022. A Longlist of Theories of Impact for Interpretability.

Nanda, N. 2023. Attribution Patching: Activation Patching At Industrial Scale.

Ngo, R.; Chan, L.; and Mindermann, S. 2022. The alignment problem from a deep learning perspective. *arXiv preprint arXiv:2209.00626*.

Olah, C. 2022. Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases. https://www.transformer-circuits.pub/2022/mech-interp-essay.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom In: An Introduction to Circuits. *Distill*.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Pearl, J. 2009. *Causality*. Cambridge University Press, 2 edition. ISBN 978-0-521-89560-6.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, 3319–3328. PMLR.

Syed, A.; and Rager, C. 2023. Attribution Patching Outperforms Automated Circuit Discovery. arXiv:2310.10348.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

Vig, J.; Gehrmann, S.; Belinkov, Y.; Qian, S.; Nevo, D.; Singer, Y.; and Shieber, S. 2020. Investigating gender bias in language models using causal mediation analysis. volume 33, 12388–12401.

Wang, K. R.; Variengien, A.; Conmy, A.; Shlegeris, B.; and Steinhardt, J. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations*.

A Qualitative Analysis of the Docstring Circuit

In Figure 7 we show the manually discovered Docstring circuit (Heimersheim and Janiak 2023). In Figure 8 we show the top edges by attribution score according to Edge Attribution Patching.

The automatically discovered circuit clearly succeeds at identifying the importance of attention heads 3.0 and 3.6. We see both the Key-Value inputs at token `C_def` and the outputs to the final logits at the `param3` token. We can also see some attributions at the correct token position for heads 0.5 and 1.4.

On the other hand, head 2.0 is missing, heads 0.2 and 0.4 have no attribution at the correct token position and head 1.4 is missing attribution at the `B_def` token.

B Another formulation of Edge Attribution Patching

Let $\alpha \in [0, 1]$ interpolate between the clean and corrupt edge activation e_{clean} and e_{corr}.

$$e_\alpha = e_{clean} + \alpha \times (e_{corr} - e_{clean})$$

Then

$$\frac{\partial F(e_\alpha)}{\partial \alpha} = \frac{\partial F(e_\alpha)}{\partial e_\alpha} \frac{\partial e_\alpha}{\partial \alpha}$$

$$= \frac{\partial F(e_\alpha)}{\partial e_\alpha} \frac{\partial [e_{clean} + \alpha \times (e_{corr} - e_{clean})]}{\partial \alpha}$$

$$= \frac{\partial F(e_\alpha)}{\partial e_\alpha} e_{corr} - e_{clean}$$

Set $\alpha = 0$, ie. $e_\alpha = e_{clean}$

$$= e_{corr} - e_{clean} \frac{\partial F(e_{clean})}{\partial e_{clean}}$$

Which is the definition of edge attribution patching.

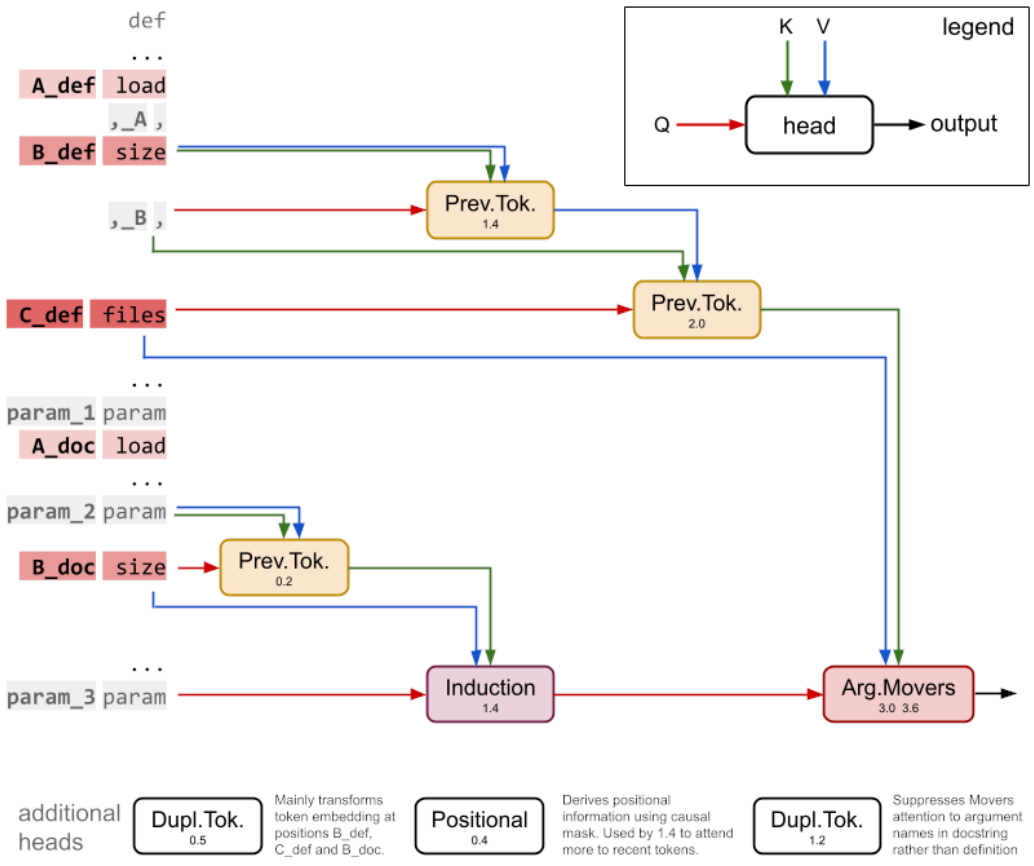

Figure 7: The manually discovered Docstring circuit. Figure credit: Heimersheim and Janiak (2023)

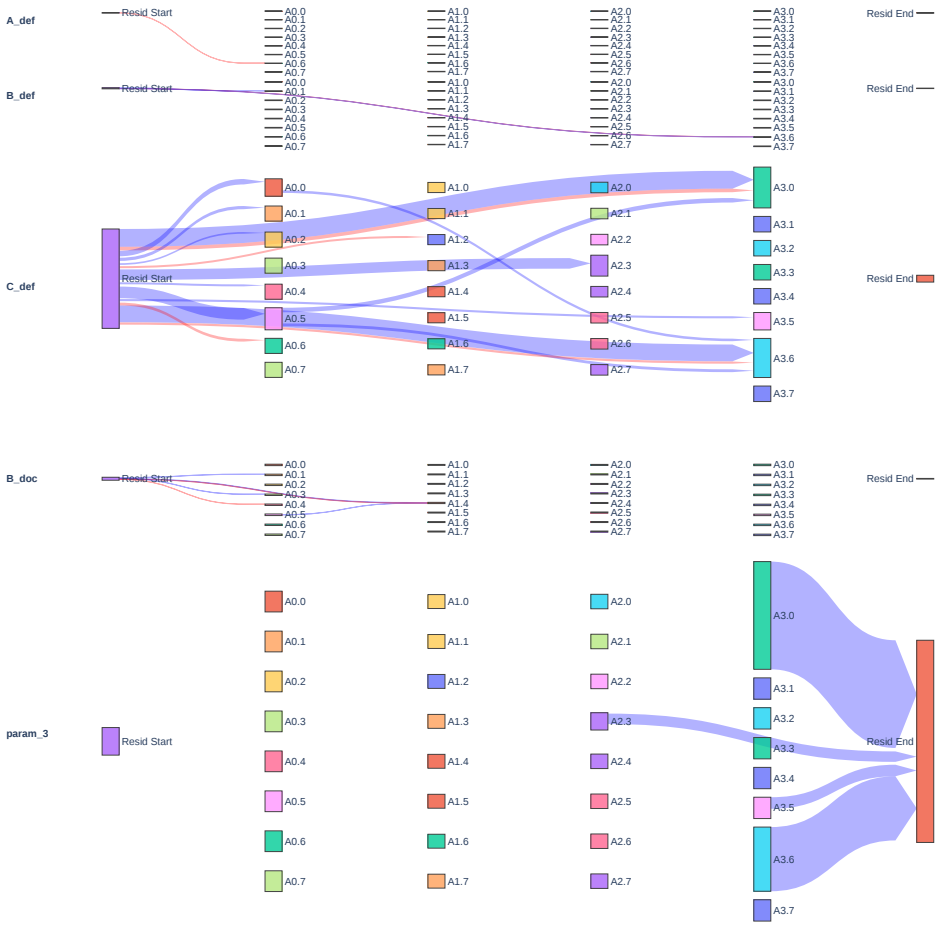

Figure 8: The top 30 most important edges for the Docstring task, according to Edge Attribution Patching. Blue edges represent positive contributions, while red edges are negative. Edge width corresponds to attribution score.