IPE: Isolating Path Effects for Improving Latent Circuit Identification

Nicolò Brunello[1*], Andrea Cerutti[1*], Andrea Sassella[1], Mark James Carman[1]

[1] DEIB, Politecnico di Milano, Via Giuseppe Ponzio, 34, 20133 Milan (MI), Italy
[*] Equal contribution
Correspondence: nicolo.brunello@polimi.it

Abstract

Understanding why large language models (LLMs) exhibit certain behaviors is the goal of mechanistic interpretability. One of the major tools employed by mechanistic interpretability is **circuit discovery**, i.e., identifying a subset of the model's components responsible for a given task. We present a novel circuit discovery technique called **IPE** (Isolating Path Effects) that, unlike traditional edge-centric approaches, aims to identify *entire computational paths* (from input embeddings to output logits) responsible for certain model behaviors. Our method modifies the messages passed between nodes along a given path in such a way as to either precisely remove the effects of the entire path (i.e., ablate it) or to replace the path's effects with those that would have been generated by a counterfactual input. IPE is different from current *path-patching* or *edge activation-patching* techniques since they are not ablating single paths, but rather a set of paths sharing certain edges, preventing more precise tracing of information flow. We apply our method to the well-known Indirect Object Identification (IOI) task, recovering the canonical circuit reported in prior work. On the MIB workshop leaderboard, we tested IOI and MCQA tasks on GPT2-small and Qwen2.5. For GPT2, path counterfactual replacement outperformed path ablation as expected and led to top-ranking results, while for Qwen, no significant differences were observed, indicating a need for larger experiments to distinguish the two approaches.

1 Introduction

Mechanistic interpretability seeks to reverse-engineer the internal computations of large language models (LLMs) to understand how specific behaviors arise (Sharkey et al., 2025). A central goal in this field is circuit discovery: identifying the sub-networks of model components (such as attention heads and MLPs) that are causally responsible for a given task (Nikankin et al., 2025; Zhang et al., 2024; Hanna et al., 2023; Cao et al., 2020). Recent efforts, including the Mechanistic Interpretability Benchmark (MIB) (Mueller et al., 2025), have formalized this task by proposing standardized settings and quantitative evaluation metrics for measuring the quality of discovered circuits.

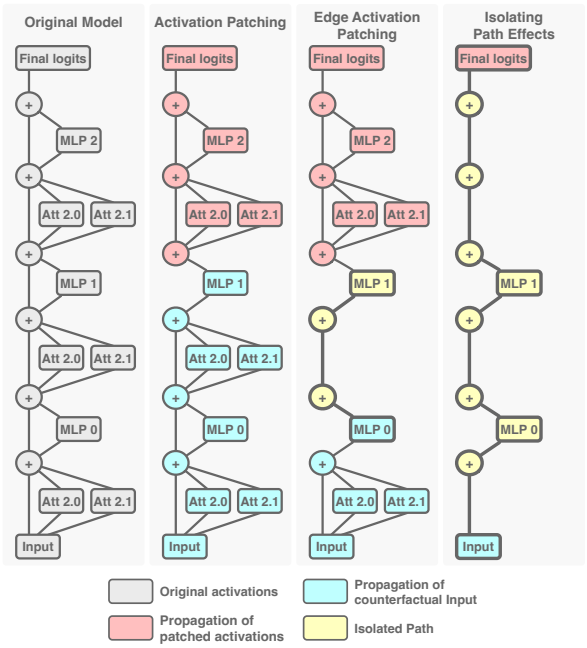

Figure 1: Theoretical differences in the identified paths with different methods. Given a toy model with three layers and two attention heads, we could apply *Activation Patching* to node `MLP 1` to check its importance. With *Edge activation patching* we assign a score to the edge `MLP 0` -> `MLP 1`, but still taking into account contributions from other edges (in red and blue). Finally, with *Isolating Path effects* we isolate the contribution of the path `Input` -> `MLP 0` -> `MLP 1` -> `Final Logits`.

Our contribution is a novel path-based circuit discovery technique, **IPE (Isolating Path Effects)**[1].

[1] Code at: https://github.com/nepp1d0/MIB-circuit-track-with-paths

Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 528–536
November 9, 2025

Rather than attempting to identify edges which are independently important within the network, IPE searches for **entire computational paths** that have significant effects on network predictions. Due to a **message-propagation patching technique**, IPE is able to evaluate precise single paths connecting the desired subcomponents, allowing for a more granular tracing of information throughout the network, as depicted in Figure 1. While evaluating a single edge (e.g. `MLP 0` -> `MLP 1`) standard Edge Activation Patching (EActP) (Mueller et al., 2025; Conmy et al., 2023) removes at the input of `MLP 1` the message passed from `MLP 0` and then patch the new activation of `MLP 1` in a forward pass, to get the final logits. IPE, instead, treats the logits as yet another subsequent node and subtracts again the new message generated by `MLP 1` from the last residual right before the logits, in order to not to contaminate the path score with contributions from other undesired edges.

Since the evaluation of all the paths in the network is prohibitive, due to the huge search space, we employed an efficient top-down search and carefully modified messages sent along these paths (from input to output) to estimate the exact contribution of the given path to model behavior.

This work was carried out in the context of the Circuit Localization track of the shared task associated with the MIB leaderboard. We first validated our results on the IOI task (Wang et al., 2023), ensuring to recover the core components of the known circuit. We tested IPE with path ablation and also path counterfactual replacement. As expected, counterfactual replacement showed better performance on the GPT-2 (Radford et al., 2019) testbed since the evaluation measure for the task makes use of counterfactuals. Meanwhile, for Qwen (Yang et al., 2024), we did not see significant improvement with counterfactuals over ablation, potentially due to insufficient search being performed given the higher number of layers in the network.

2 Background

Wang et al. (2023) introduced one of the first detailed mechanistic explanations for how GPT-2 solves the IOI (Indirect Object Identification) task. The authors combined multiple manual interpretability techniques to identify a sparse circuit of 26 attention heads grouped into 7 functional categories. They introduced a technique called **path-patching** to identify important edges in the graph, whereby they employ four forward passes of the model to measure the importance of one single edge. Building on this foundation, the ACDC technique by Conmy et al. (2023) automates the circuit discovery process by selecting important edges between components (e.g., attention heads, MLPs) using an iterative metric-driven pruning search strategy. The core technique consists of traversing the tree of components in reverse topological order in a top-down fashion and progressively applying **activation patching** at single edges. Each edge is evaluated by replacing the activation at the destination node with the one obtained from a corrupted (counterfactual) run and measuring how much this changes model output. By repeating this across many edges and selecting those with the highest importance, ACDC constructs a graph that approximates the causal structure of the circuit. Later work on **attribution patching** proposed a simple scalable alternative to ACDC, by applying a linear approximation to the patching operation, effectively estimating edge importance via gradients Syed et al. (2023); Sundararajan et al. (2017); Hanna et al. (2024); Marks et al. (2025). A related line of work is introduced by (Goldowsky-Dill et al., 2023), who explore the notion of *path patching* in a way that aligns with our theoretical definition of path contributions. While their approach does not provide a fully automated method for discovering circuits, it offers valuable theoretical foundations that motivate path-centric analyses. All of these methods operate in an **edge-centric** fashion, evaluating or ranking individual connections between components. They identify a circuit as the set of edges deemed most important under a chosen metric, often without enforcing path-level coherence or structural constraints. In contrast, our method takes a **path-centric** view leveraging the model's residual architecture to trace full computational flows and investigate path-level effects. Specifically, our algorithm provides an efficient mechanism for exclusively removing and quantifying the effects of the given path in a model.

3 Method Overview

Similar to previous works (Vig et al., 2020; Hanna et al., 2023; Nikankin et al., 2025), we define a path as a directed connected sequence of model components, but then impose a **structural constraint** that *all paths originate from the input embeddings*

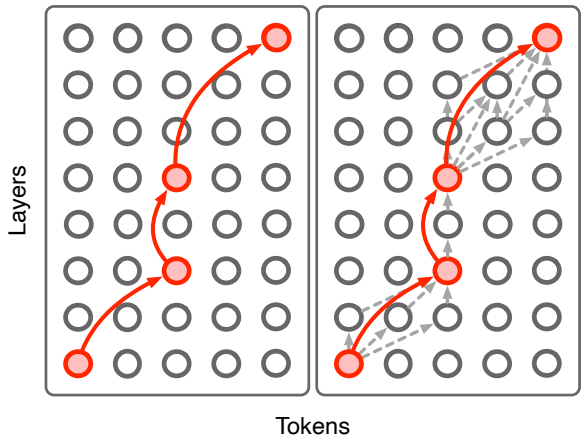

Figure 2: Consider a 3-step path connecting the input layer to the output layer for a particular model. If path importance is calculated by **propagating effects of removing entire paths**, the only information passed along the path itself will affect the distribution for the output token (on the left). If instead, path importance is computed by either **patching nodes** or **combining independent estimates of edge importance** (shown on the right), the final importance score will implicitly take into account contributions from many irrelevant paths shown in dashed gray arrows.

and terminate at the output layer (logits). We hypothesize that the Transformer's residual decoration mechanism naturally supports the emergence of human-interpretable circuits, where atomic features (single tokens' meanings) are injected at early layers and then refined over subsequent layers. By constraining our analysis to full paths from the input to the output, we aim to capture these end-to-end transformations directly, rather than reasoning over isolated edges or local interventions.

3.1 Path evaluation

To **evaluate the importance of a candidate circuit** prior works typically rely on ablations, such as zeroing out activations or applying counterfactual substitutions at selected components (Nikankin et al., 2025; Hanna et al., 2023). As illustrated in Figure 2, ablating a single edge introduces the effects from other earlier components of the model outside the identified circuit: since each node receives residual input from the entire previous model slice, its activation reflects not only the preceding edge under investigation but also all other components contributing to the residual stream. By contrast, ablating a complete path is possible by appropriately updating the intermediate messages passed along the causal chain, thereby isolating and removing the contribution of the given path while preventing the influence of other paths through the network.

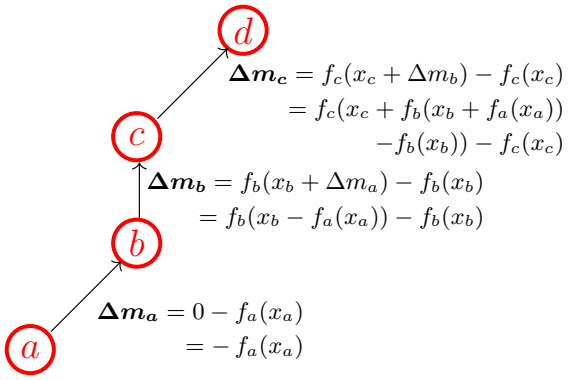

Figure 3: Direct calculation of the change in the propagated information along a 3-step path ($a \to b \to c \to d$) that results of isolating and removing that particular path from the model. Here x_a denotes the input message to node a (prior to path removal), $f_a()$ denotes the function computed by the model at that node, and Δm_a denotes the change in the output message from that node resulting from the removal of the path from the model. To compute the importance of the path, each Δm is **patched** as input to the next node in the path, starting from the lower node going up to the logits (last possible node in every path).

To **efficiently implement path ablation**, we design an iterative procedure that traverses the selected path from the input to the output calculating the change in the message (denoted Δm in Figure 3) applied to the input for each subsequent node in the path. Starting from the first edge in the path (i.e. source node is the input embedding) we remove the contribution of the *source node* from the *destination node*. We then perform a forward pass of the *destination node* using the modified input, producing an updated output. Now we treat this *destination node* as the **source node for the subsequent edge in the path** and repeat the procedure. This process continues iteratively and exclusively along each edge in the path, with each component receiving the appropriately updated output from the previous component, ensuring that the path is ablated in isolation from the rest of the model. The iteration terminates at the final node, after which the updated residual stream is used to compute the model's final logits. In the case of counterfactual injection, the process is the same except that the change in the message along the first edge (Δm_a in Figure 3) becomes $f(x'_a) - f(x_a)$, where x'_a is now counterfactual input to the node a.

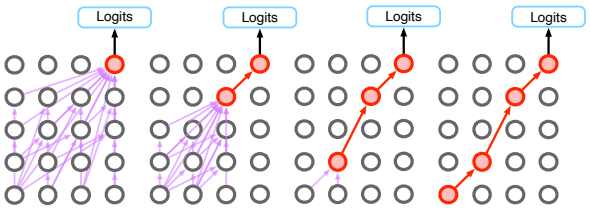

Figure 4: Iterative process of searching for paths. Starting from the logits, we explore all the possible paths one edge at time, approximating the contribution of the **complete path** (i.e., form an input token to the final logits) with the contribution of the **incomplete path** spanning from the output logits to last edge candidate. In purple we highlighted all the possible paths at each step of the search, all those paths may contribute differently, but we are assuming their contribution to be at most the one of the incomplete path (in red).

Algorithm 1: Backward Discovery (with minimum contribution threshold)

```
frontier ← [LOGIT Node]
completed ← [ ]
while frontier is not empty do
    path ← frontier.pop()
    candidates ← path.predecessors()
    foreach node in candidates do
        msg ← EVALUATEPATH(node, path)
        if METRIC(msg) ≥ threshold then
            if node is EMBED Node then
                add (node, path) to completed
            else
                add (node, path) to frontier
return completed
```

3.2 Search algorithm

Evaluating all possible paths from the input embeddings to the output logits is computationally intractable, as the number of such paths grows exponentially with the depth and width of the model. To address this, we employ a top-down search strategy (similar to (Conmy et al., 2023)) that incrementally constructs and evaluates candidate paths of increasing length, as outlined in Figure 4 and Algorithm 1 (please refer to Appendix B for further details). The search begins at the output node (i.e., the final logits) and recursively explores all possible paths up to length n, where n is typically set to be the number of network layers. A path of length 1 consists of a single edge that connects the input embedding directly to the output; a path of length 2 includes one intermediate component; and so on. At each step, only those partial paths whose current contribution, measured using a user-defined *scoring metric*, exceeds a predefined threshold are extended further. This approach leverages the empirical observation that longer paths tend to have diminishing influence on the model's final prediction due to the fading contribution of individual components in the residual stream. As a result, the search typically terminates well before reaching the maximum possible path length defined by the model's architecture, significantly reducing computational overhead while preserving high-quality circuit candidates.

4 Experimental Setup

This section describes the two main evaluation settings used in our study[2], i.e. the IOI task in GPT-2 small and the MIB leaderboard framework.

4.1 Comparison with Ground-Truth

We begin by evaluating our approach against the manually reverse-engineered IOI circuit reported in Wang et al. (2023). To quantify the importance of a path, we ablate it using the **logit difference** $logit(t|M) - logit(t|M_{\neg path})$, where t is the desired output token, M is the original model, and $M_{\neg path}$ denotes the *corrupted* model with the particular path removed. We run the search procedure with multiple threshold values on this metric. Lower thresholds result in the discovery of more paths, but also increase computational cost, (see Appendix A for more details).

4.2 MIB leaderboard

Using the threshold insights obtained from the previous experiment, we applied IPE with path ablation and IPE-CF with counterfactuals to the GPT-2 small and Qwen models for the IOI task, and to the Qwen model for the MCQA task, as defined in the MIB evaluation (Mueller et al., 2025). The metric used in the counterfactual setup is the *Indirect Effect* proposed by Stolfo et al. (2023). The motivation for using this metric is that, when combined with a counterfactual prompt, it enables the

[2]Experiments are conducted using NVIDIA A100 SXM4 and NVIDIA GeForce RTX 4090 GPUs.

algorithm to identify paths that not only contribute to increasing the logit of the correct token but also substantially reduce the logits of competing incorrect tokens, thereby favoring the correct prediction. In order to let our paths be evaluated by the MIB benchmark, we first give to each edge the score of the complete path it belongs to. Then, for edges appearing in different paths, we simply summed all their scores. This lets the final score for each edge be greater for edges appearing in multiple paths.

5 Results

Using an *indirect effect* threshold of 10^{-4}, we compared circuits identified by our method against the IOI ground-truth annotations. Our circuits retained 47.6% of the original ground-truth edges, while capturing 97.5% of the ground-truth nodes. Thus, our approach successfully recovers almost all the key nodes involved in the task, while the fraction of recovered edges is lower, due to the quadratic set of potential edges that need to be considered.

We evaluated our approach on the MIB leaderboard using both path ablation and path counterfactual strategies. A summary of the results is provided in Table 1, which shows that the use of a counterfactual substantially improves performance, with IPE-CF ranking among the top algorithms for GPT-2 IOI task, according to both the CPR (Circuit Performance Ratio) and CMD (Circuit-Model distance) metrics on the MIB leaderboard. Further analysis on the public test set confirms that the approach discovers a larger number of relevant edges during search. In contrast, the ablation variant achieves lower scores primarily because it identifies only task-relevant rather than counterfactual-relevant edges.

Isolating Path Effect (IPE) underperforms on larger models such as Qwen-2.5. We hypothesize that IPE identifies too few edges relative to the model's scale. Compared with GPT-2, Qwen-2.5 has substantially more layers, which expands the combinatorial space of candidate paths. Because IPE searches full paths, the same edge can be evaluated repeatedly across paths, while few new candidates are proposed. Consequently, achieving adequate coverage on larger models likely requires a more exhaustive search (e.g., lowering the edge-selection threshold or increasing the candidate budget to approach convergence).

6 Conclusion

We presented a new circuit discovery method for LLMs that focuses on scoring full computational paths rather than isolated edges, thereby preserving the global circuit structure, ignoring irrelevant paths, and offering greater interpretability. Our analysis shows that the set of recovered circuits depends critically on whether paths are simply removed from the model or replaced with a counterfactual input. These insights position path-based search as a promising direction for more faithful mechanistic interpretations of Transformer models.

7 Acknowledgements

This work was supported by the FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence).

References

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3243–3255. Association for Computational Linguistics.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching. *CoRR*, abs/2304.05969.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *CoRR*, abs/2403.17806.

Technique	IOI				MCQA	
	GPT-2		Qwen-2.5		Qwen-2.5	
	CPR	CMD	CPR	CMD	CPR	CMD
Random	0.25	0.75	0.28	0.72	0.27	0.73
EActP(CF)	2.30	0.02	1.21	0.49	0.85	0.36
EAP(CF)	1.20	0.03	0.26	0.15	0.85	0.07
EAP-IG-act. (CF)	1.82	0.03	1.63	0.01	0.77	0.05
NAP-IG (CF)	0.76	0.27	0.29	0.20	0.77	0.18
IFR	0.58	0.42	0.31	0.69	0.40	0.60
UGS	0.97	0.03	0.98	0.03	1.17	0.20
IPE	0.72	0.28	0.42	0.58	0.47	0.53
IPE (CF)	2.24	0.02	0.35	0.57	0.45	0.54

Table 1: Performance across tasks, models, and techniques on the MIB leaderboard.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, and 4 others. 2025. MIB: A mechanistic interpretability benchmark. *CoRR*, abs/2504.13151.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*. Accessed: 2024-11-15.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, and 10 others. 2025. Open problems in mechanistic interpretability. *CoRR*, abs/2501.16496.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7035–7052. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. *CoRR*, abs/2310.10348.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. 2024. Interpreting and improving large language models in arithmetic calculation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

A Relevance of the Retrieved Paths in gpt2-small

In this appendix, we provide some insight on how IPE behaves, highlighting some of the method's strengths and limitations. We focus on the version of IPE without counterfactual prompts, using our logit-difference metric and selecting only paths that positively contribute to the correct token's logit. We then use the manually reverse-engineered IOI circuit from Wang et al. (2023) (excluding negative Name Mover heads) as ground truth to evaluate the circuit.

A.1 Effect of contribution Threshold

As we can see from Figure 5 our algorithm find exponentially more node, edges and paths when the minimum contribution threshold is reduced. Therefore, the computational cost grows exponentially as the threshold decreases. This is due to an increasing number of relevant paths, to find, evaluate, and include in the circuit. Furthermore, as we reduce the minimum contribution threshold for each new edge ranked and included in the circuit an exponential number of paths need to be included in the search. These insight exposes the core weaknesses of our approach: to provide a score for all edges IPE must be run at a prohibitively low threshold, which makes it computationally expensive. This makes the current implementation of IPE unsuitable for much larger models but hints that any significant computational speedup of this algorithm is likely to results in more complete circuits.

A.2 Retrieval Performance Against Ground Truth

To assess the relevance of the circuit found by our algorithm beyond the metrics proposed in Mueller et al. (2025), we can compare it against a ground truth. In particular, given the nature of both our algorithm and the circuit proposed in the original paper Wang et al. (2023), we can evaluate the circuit in terms of the ability to retrieve the elements present in the ground truth circuit.

Despite differences in paradigm (path-centric) and metric definition, the recovered nodes align closely with the hand-annotated IOI circuit Figure 6. With almost all ground truth components included in at least one relevant path when the threshold is sufficiently low.

Furthermore, when adapting our algorithm to use **counterfactual ablation** and use a more relevant metric like indirect effect Stolfo et al. (2023), performance in terms of CPR and CMD drastically improves Table 1. On the other hand, the ability of the circuit to retrieve the ground truth improves

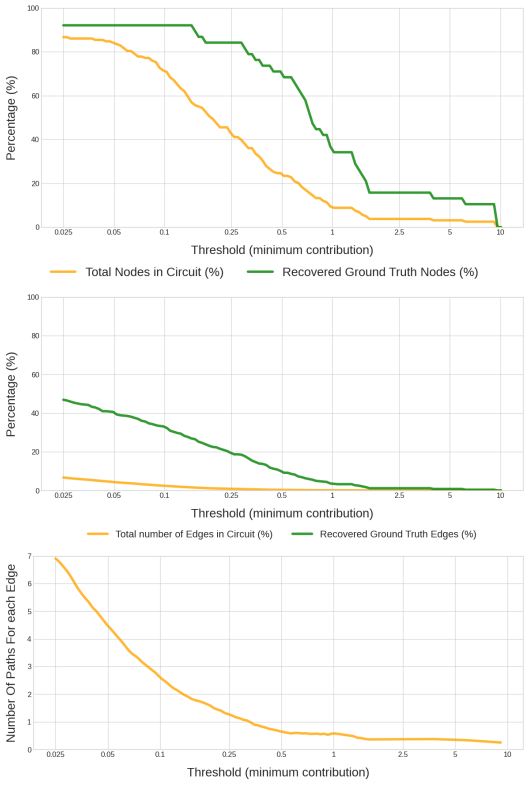

Figure 5: **(top)** Percentage of total nodes included in our circuit (yellow) and nodes from ground truth included in our circuit (green) as function of the minimum contribution threshold (log scaled). **(center)** Percentage of total edges included in our circuit (yellow) and edges from ground truth (green) included in our circuit as function of the minimum contribution threshold (log scaled). **(bottom)** average number of paths required for each edge in the circuit as a function of the minimum contribution threshold.

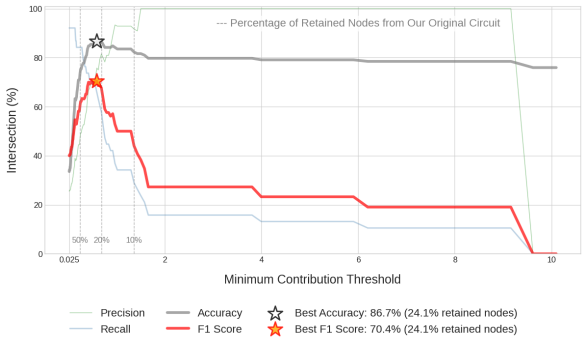

Figure 6: Performance of IPE with path ablation as a classifier of the ground truth nodes when varying the minimum path contribution threshold.

only slightly Figure 7

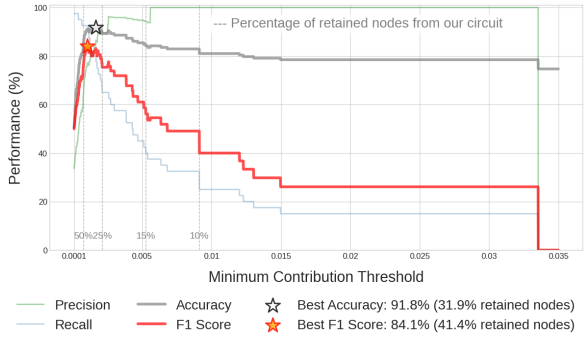

Figure 7: Performance of IPE with path counterfactual as a classifier of the ground truth nodes when varying the minimum path contribution threshold.

It is also interesting to note how the nodes retrieved using path ablation and the counterfactual have some differences. Particularly the union of the circuits obtained using the two methods, yields perfect recall. This highlights the importance of choosing the correct ablation type and metric.

B Search Algorithm Implementation

This appendix provides additional details on the implementation of our path-centric circuit discovery algorithm (Algorithm 1).

Our search for computational paths begins at the model's final residual and proceeds backward toward the input embeddings. We incrementally build and evaluate paths of increasing length, pruning branches that do not meet a minimum contribution threshold (to limit the unmanageable number of computational paths in a transformer model).

The algorithm has two core support functions $EvaluatePath$ and $Metric$.

B.1 EvaluatePath

The core of our method is the ability to isolate the effect on the final residual stream of a single path. This is orchestrated by the EvaluatePath function, which recursively computes the path's "message" using the method presented in figure Figure 3. This function takes as input a path, its starting node (n_0) and the initial Δm_0, the output is Δm_{path}, the difference in the path's contribution under the clean run and when the initial message is propagated through the path. Note that:

- When the Δm_0 is set to be $-out^{n_0}_{clean}$ the output will be the effect on the final residual of zero ablating the path.

Algorithm 2: EvaluatePath(node, path, Δm)

if *len(path) is 0* **then**
  return Δm
corr_msg = node.forward(node.input + Δm)
clean_msg = node.forward(node.input)
Δm = corr_msg - clean_msg
node = path.pop() *//pop current path start*
return EvaluatePath(node, path, Δm)

- When the Δm_0 is set to be $-out^{n_0}_{clean} + out^{n_0}_{cf}$, Δm_{path} is the effect on the final residual stream of substituting the path's contribution with its counterfactual value.

Where $-out^{n_0}_{clean}$ and $-out^{n_0}_{cf}$ are the output of the node n_0 respectively under clean and counterfactual runs.

As an example if we want to find the change obtained by zero ablating a path starting at the EMB node, we invoke $EvaluatePath(EMB, path, -EMB.forward(prompt_{clean})$. It is important to underline how this function does not require a complete forward pass on the model, only the nodes in the evaluated path are involved, drastically reducing the computational load required.

B.2 Metric

The Metric function is the one responsible for the actual attribution of scores to paths. This function adds Δm_{path} obtained from $EvaluatePath$ to the clean residual, obtaining the "corrupted" residual stream. Then, it assigns a score to the path by evaluating the difference between the distributions obtained from the clean and corrupt residual streams. In particular, we have used two different metrics: a custom logit difference and the indirect effect (Stolfo et al., 2023).

B.2.1 Target Logit Difference (%)

This metric measures a path's direct contribution to the logit of a target token, t. It is the change in the target logit, expressed as a percentage of the original logit's magnitude.

$$\text{LogitDifference\%} = 100 \times \frac{\mathcal{L}(\mathbf{r}_{\text{clean}}, t) - \mathcal{L}(\mathbf{r}_{\text{corr}}, t)}{|\mathcal{L}(\mathbf{r}_{\text{clean}}, t)|} \quad (1)$$

where $\mathcal{L}(\mathbf{r_{clean}}, t)$ is the logit of token t derived from residual $\mathbf{r_{clean}}$. Note that we have chosen to take the value as a percentage in order to make the threshold value more interpretable and generalizable across models.

B.2.2 Indirect Effect (IE)

The IE score is designed for counterfactual evaluation. It measures a path's ability to both increase the probability of the counterfactual target (t_{cf}) and decrease the probability of the original answer (t).

$$\text{IE} = \frac{1}{2}\left(\frac{P^*(t_{cf}) - P(t_{cf})}{P(t_{cf})} + \frac{P(t) - P^*(t)}{P^*(t)}\right) \tag{2}$$

where P denotes probabilities from a clean run and P^* from the corrupted run (when the path removal effect on the final residual is considered).

B.2.3 Search Direction

A key decision in our circuit discovery algorithm is the search direction. The $EvaluatePath$ function is "forward looking", as it propagates a message from the lower layers towards the top of the network. Therefore starting the search from the embedding and moving forward might seem more computationally efficient, as it allows for reusing the partial path message ($\Delta m_{partial}$) for the evaluation of all possible path expansion. Nonetheless we found this approach to be infeasible in practice.

A forward search requires calculating the contribution of an incomplete path, however, the $Metric$ attributes a score based on the effect on the final residual. An exact evaluation would require summing the effects of all downstream paths originating from the incomplete segment, which would require a complete forward pass on the model.

We also explored an approximated forward approach based on Attribution Patching (Syed et al., 2023). This method estimates the path's importance by measuring the alignment between the incomplete path's message delta ($\Delta m_{partial}$) and the gradient of the final metric with respect to the input of a candidate next component. However, this approximation performed poorly, especially in the crucial lower layers of the network where early search decisions have the largest impact. This poor performance in low layers is a known limitation of similar methods like Edge Attribution Patching.

Due to these limitations, we adopted the backward search direction, which allows for a path evaluation involving only the nodes in the path.

The mechanism chosen to guide the search space is a threshold-based Breadth-First Search. Nonetheless, we acknowledge that other valid alternatives exist, such as a Best-First Search that continues until N paths are found, or a Top-K Breadth-First Search that retains only the top K candidate expansions at each depth level. Based on limited empirical analysis, these methods appear to yield similar performances.