
Rethinking Circuit Completeness in Language Models: AND, OR, and ADDER Gates

Hang Chen

School of Computer Science and Technology
Xi'an Jiaotong University
albert2123@stu.xjtu.edu.cn

Jiaying Zhu

School of Computer Science and Engineering
The Chinese University of Hong Kong
jyzhu24@cse.cuhk.edu.hk

Xinyu Yang*

School of Computer Science and Technology
Xi'an Jiaotong University
yxyphd@mail.xjtu.edu.cn

Wenya Wang†

School of Computer Science and Engineering
Nanyang Technological University
wangwy@ntu.edu.sg

Abstract

Circuit discovery has gradually become one of the prominent methods for mechanistic interpretability, and research on circuit **completeness** has also garnered increasing attention. Methods of circuit discovery that do not guarantee completeness not only result in circuits that are not fixed across different runs but also cause key mechanisms to be omitted. The nature of incompleteness arises from the presence of **OR gates** within the circuit, which are often only partially detected in standard circuit discovery methods. To this end, we systematically introduce three types of logic gates: AND, OR, and ADDER gates, and decompose the circuit into combinations of these logical gates. Through the concept of these gates, we derive the minimum requirements necessary to achieve faithfulness and completeness. Furthermore, we propose a framework that combines noising-based and denoising-based interventions, which can be easily integrated into existing circuit discovery methods without significantly increasing computational complexity. This framework is capable of fully identifying the logic gates and distinguishing them within the circuit. In addition to the extensive experimental validation of the framework's ability to restore the faithfulness, completeness, and sparsity of circuits, using this framework, we uncover fundamental properties of the three logic gates, such as their proportions and contributions to the output, and explore how they behave among the functionalities of language models.

1 Introduction

As an intervention-based approach to mechanistic interpretability, circuit discovery allows for the extraction of subgraphs from the computational graph of a language model that play a significant role in task performance, referred to as circuits [1, 2, 3, 4]. Several key studies have supported its development [5, 6], such as those focusing on ensuring that circuits faithfully reflect the model's outputs [2, 7], enabling efficient circuit extraction [8], and addressing scalability challenges for models with extremely large parameters and corpora [9, 10, 11].

As the concept of circuits evolves, recent attention has increasingly focused on the **completeness** of circuits in addition to **faithfulness**. For example, completeness has been redefined such that

*Corresponding author

†Corresponding author

when a circuit is removed from the computational graph, the performance of the task should degrade significantly [12, 13]. Nevertheless, current circuit discovery methods have been found to lack completeness [9]. Moreover, theory analysis [14] indicates that incomplete circuits lead to two potential pitfalls: non-transitivity and preemption, which prevent the recovery of the key mechanisms underlying the circuit. Finally, incompleteness results in variability in circuit discovery outcomes, making the circuit appear more like an arithmetic solution to obtain the output rather than representing a closed-form solution with interpretability [15].

Incompleteness largely arises from the presence of **OR gates** [16, 2]. For instance, consider a model M that employs two identical and disjoint serial circuit paths, C_1 and C_2 , which operate in parallel and whose outputs are subsequently combined via an OR operation. In this case, identifying either path is sufficient to achieve faithfulness, and removing the other path is a preferable choice for promoting sparsity. However, restoring completeness by discovering OR gates remains a challenge. The simplest approach, which involves repeated interventions on combinations of components [14], could theoretically uncover the complete OR gate; however, this makes circuit discovery an NP problem (We explained it in Appendix E.1). Additionally, while denoising-based intervention methods can rapidly restore OR gates [17], they lead to a more severe loss of faithfulness. Furthermore, these methods fail to isolate the OR gates from the final circuit, resulting in a lack of logical interpretability.

To this end, we introduce the concept of logic gates, where any circuit can be decomposed into AND, OR, and ADDER gates, and propose a systematic framework to uncover and separate all the gates, and then to explain their correspondence to faithfulness or completeness with the sparsity constraint. Our specific contributions are as follows:

1. **We systematically introduce three types of logic gates that compose a circuit: AND, OR, and ADDER gates.** Through these gates, we are able to infer the minimum requirements for a circuit to achieve faithfulness and completeness, as well as assess the capability of noising-based and denoising-based interventions in restoring these gates. Based on these corollaries, we analyze three types of prevailing circuit discovery methods, named greedy search [2, 18, 11], linear estimation [8, 19], and differentiable mask [9, 12, 10], by evaluating their ability to recover the three logic gates and their faithfulness and completeness. Moreover, we conduct experiments to provide empirical evidence supporting these theoretical conclusions.
2. **We propose a framework capable of fully discovering the three logic gates**, which can be easily extended to current circuit discovery methods with constant-time complexity. Our framework combines noising-based and denoising-based interventions, ensuring both the faithfulness and completeness of the circuit, and enabling the separation of AND, OR, and ADDER gates from the final circuit. Extensive experimental results demonstrate that our framework achieves promising faithfulness and completeness. Additionally, to ensure consistency in the granularity of noising-based and denoising-based interventions, we introduce a misalignment score for AND and OR gates to measure whether the scales of the two intervention strategies are aligned when combined.
3. **We explore the characteristics of AND, OR, and ADDER gates in a circuit**, including their proportions and contributions to the output, building upon our proposed logic gates and recovery framework. Furthermore, we examine the relationship between logic gates and the functionality of language models. Experimental results show that OR gates typically link multiple backup paths for the same function, while AND gates often connect paths for different necessary functions.

2 Preliminaries

2.1 Circuit Discovery

In Transformer decoder-based language models, the forward pass is typically conceptualized as a **computational graph** $\mathcal{G}$, where the nodes represent components (such as attention heads, MLPs, or even more granular elements like the query, key, and value matrices) and an edge $i \rightarrow j$ denotes a connection where the output of component i serves as input to component j . Circuit discovery seeks to identify a subgraph (circuit) $\mathcal{C} \subset \mathcal{G}$ that captures the task-relevant behavior of the model [3].

The process used to prune and obtain the circuit $\mathcal{C}$ is referred to as **intervention** (also known as **knockout**, **ablation**) [17, 20, 21, 22]. For a given task $\mathcal{T}$, each sample x is referred to as **clean text**, and the corresponding forward pass yields the **clean activation** x_i at each component i . A perturbed version of the input, denoted $\tilde{x}$, is called **corrupted text**, producing a corresponding **corrupted**

activation [23, 17]. The corrupted activation $\tilde{x}_i$ depends on the specific ablation method used. For example, ZERO ABLATION sets $\tilde{x}_i = 0$, while NOISE ABLATION draws $\tilde{x}_i$ from a predefined noise distribution. A widely used method, INTERCHANGE ABLATION, defines $\tilde{x}_i$ as the activation resulting from an input text that has been minimally perturbed to produce a different task label [10].

The intervention is divided into two strategies: **noising-based intervention** (hereafter referred to as **Ns**) and **denoising-based intervention** (hereafter referred to as **Dn**) [24]. The **Ns** first runs the clean text in the computational graph. Then, corrupted activations replace each clean activation to observe the change in the final output y . If replaced (also known as removed or pruned) activations lead to a significant change in output, they are considered to make an important contribution to the task $\mathcal{T}$ and should be retained in the circuit $\mathcal{C}$ [17]. Let $p_{\mathcal{G}}(y|x)$ denote the model’s original output, $p_{\mathcal{C}}(y|x, \tilde{x})$ represent the circuit’s output after intervention. Specifically, if an edge $j \rightarrow i$ is retained within $\mathcal{C}$, the activation of component i keeps the clean one (x_i). Conversely, it is replaced by the corrupted one ($\tilde{x}_i$). Let s denote the requirement of sparsity, and D represent the distance used to quantify the difference between the two outputs. **Ns** has the following objective:

$$\arg \min_{\mathcal{C}} \mathbb{E}_{(x, \tilde{x}) \in \mathcal{T}} [D(p_{\mathcal{G}}(y|x) || p_{\mathcal{C}}(y|x, \tilde{x}))], \quad s.t. \quad 1 - |\mathcal{C}|/|\mathcal{G}| \geq s \quad (1)$$

Equation 1 indicates that the circuit is a subgraph that most closely approximates the functionality of the computational graph, where the components and edges have the most significant effect on the output. Similarly, the **Dn** first performs the corrupted run in the computational graph, and then replaces the corrupted activations with the clean activations. Those activations that lead to significant changes in the output ($\tilde{y}$) consist of the circuits. **Dn** thus has the following objective:

$$\arg \min_{\mathcal{C}} \mathbb{E}_{(x, \tilde{x}) \in \mathcal{T}} [D(p_{\mathcal{G}}(\tilde{y}|\tilde{x}) || p_{\mathcal{C}}(\tilde{y}|\tilde{x}, x))], \quad s.t. \quad 1 - |\mathcal{C}|/|\mathcal{G}| \geq s \quad (2)$$

Most of the related work on circuit discovery follows the **Ns** strategy. We categorize these works into three types: (1) **Greedy search** [2, 18, 11], which iteratively examines each edge (or node) through intervention to obtain a greedy solution for the circuit. (2) **Linear estimation** [8, 19], where the contribution of each edge is approximated by a gradient measure obtainable in a single backward pass. This approach ranks the importance of each edge to approximate the circuit. (3) **Differentiable masks** [9, 12, 10], where a learnable mask is assigned to each edge (or node), treating circuit discovery as an optimization problem to derive the optimal circuit.

2.2 Circuit Evaluation

Circuit evaluation is primarily defined by three aspects: **faithfulness**, **completeness**, and **sparsity**.

Faithfulness refers to the circuit’s ability to perform task $\mathcal{T}$ in isolation, which is defined as the difference between the circuit’s output and the model’s original output [16, 9, 17]. This is represented in Equations 1 as $\mathbb{E}_{(x, \tilde{x}) \in \mathcal{T}} [D(p_{\mathcal{G}}(y|x) || p_{\mathcal{C}}(y|x, \tilde{x}))]$ (simplified as $D(\mathcal{G} || \mathcal{C})$). Method ACDC [16] measures faithfulness by computing the average difference in the unnormalized output logits between the correct token and an incorrect option. Recently, work [2, 17, 25] proposes that KL divergence provides a better measure of the distribution over the vocabulary, while other work [9, 15] suggests that task accuracy can avoid the overemphasis on irrelevant vocabulary in the KL divergence. In this paper, we measure faithfulness using both KL divergence and task accuracy as metrics.

Completeness refers to whether the circuit includes all the important paths that have an effect on the output. The work [16] first introduces the concept of circuit completeness, stating that $\mathcal{C}$ and $\mathcal{G}$ should ensure similar outputs even under any knockout. Therefore, the incompleteness score is defined as the difference $D(\mathcal{C} \setminus \mathcal{K} || \mathcal{G} \setminus \mathcal{K})$ for any subcircuit $\mathcal{K} \subset \mathcal{C}$. Existing work [9, 15] proposes that insufficient sampling of $\mathcal{K}$ may lead to unreliable approximations (we show the practical results in Appendix A), and thus recommends evaluating completeness by assessing the performance after the circuit’s removal from the computational graph on the task $\mathcal{T}$, i.e., $D(\mathcal{G} \setminus \mathcal{C} || \mathcal{G})$ [13, 12]. In this paper, we also adopt it to evaluate completeness.

Sparsity refers to that the circuit should be as small as possible. Currently, many works [10, 9, 15] recommend measuring sparsity using the ratio $|\mathcal{C}|/|\mathcal{G}|$, which represents the proportion of edges in the circuit relative to those in the computational graph. In fact, higher sparsity tends to result in lower faithfulness, meaning that the circuit always reflects some trade-off between sparsity and faithfulness.

3 Circuit Logic

3.1 Logical Gates

Recent studies [16, 2, 17] have increasingly observed that within a circuit, certain subcircuits influence the output according to logical relationships resembling AND or even OR operations. Building on these findings, we systematically introduce three fundamental circuit logic types: the **AND** gate, **OR** gate, and **ADDER** gate.

Definition 1. We assume a common paradigm in which a receiver node B , which is connected by more than 1 sender node $A_1, A_2, \dots$. For any edge $A_i \rightarrow B$, we use binary values ‘0’ and ‘1’ to represent the activation state of a node. Specifically, $A_i = 0$ indicates that node A_i is removed, ablated, or deactivated, whereas $A_i = 1$ indicates that node A_i is retained and active. When the sender nodes are ablated, the effect of node B on the output exhibits three distinct patterns, which are as follows:

AND: All sender nodes satisfy an AND logical relationship with the receiver node, i.e., $B = A_1 \wedge A_2 \wedge \dots$. In this case, node B exerts a significant effect on the output only if all of its sender nodes are retained. If even a single sender node is ablated, the effect of B on the output is nearly eliminated.

OR gate: All sender nodes satisfy an OR logical relationship with the receiver node, i.e., $B = A_1 \vee A_2 \vee \dots$. In this case, node B always exerts a significant effect on the output if one or more of its sender nodes are retained. Only if all sender nodes are ablated, the effect of B on the output is nearly eliminated.

ADDER gate: all sender nodes satisfy an ADDER logical relationship with the receiver node, i.e., $B = A_1 + A_2 + \dots$. In this case, node B exhibits its maximal effect on the output only when all of its sender nodes are retained. If any single sender node is ablated, the effect of B on the output is substantially diminished; when all sender nodes are ablated, B ’s effect on the output is reduced to zero. Accordingly, we define the state of B as taking values $0, 1, 2, \dots$, where the total number of distinct states equals the number of sender nodes.

Theoretical analyses support the view that N_s is capable of recovering a complete AND gate but fails to recover a complete OR gate, whereas D_n demonstrates the opposite pattern [17]. This asymmetry is straightforward to interpret. The N_s procedure corresponds to the transition from a clean activation state (state = 1) to a corrupted activation state (state = 0). Since all gates can be regarded as being initialized with activation states equal to 1, any transition to state = 0 induces a significant change in the effect of AND and ADDER gates on the output. Consequently, N_s can reliably identify AND and ADDER gates. Conversely, the D_n procedure corresponds to initialization with activation states equal to 0. In this case, any transition to state = 1 produces a significant change in the effect of OR and ADDER gates on the output. Moreover, we design a toy model to explain ADD, OR, ADDER gates in Appendix D.

Therefore, we denote the circuit constructed under the N_s strategy as C_{N_s} , and the one constructed under the D_n strategy as C_{D_n} . Based on the above set-theoretic relationships between C_{N_s} and C_{D_n} , we extract subsets of edges corresponding to AND, OR, and ADDER gates as follows:

- AND gate (C_{AND}): edges that are present in C_{N_s} but absent from C_{D_n} .
- OR gate (C_{OR}): edges that are present in C_{D_n} but absent from C_{N_s} .
- ADDER gate (C_{ADDER}): edges that are shared between C_{N_s} and C_{D_n} .

We conduct an ablation on these edges: for each gate, we randomly remove either one or two edges on the same receiver node and measure the resulting change in the KL divergence of the output. This procedure is repeated 30 times for each receiver node, and the distributions of ΔKL values are summarized via box plots, as shown in Figure 1 (Detailed results are shown in Appendix E.2). We selected the computational graph of GPT2-small as $\mathcal{G}$, and Indirect Object Inference (IOI) [16] as the test task. For the baseline methods, we chose ACDC [2] to represent the greedy search method, EAP [8] to represent the linear estimation method, and EdgePruning [10] to represent the differentiable mask method. For details regarding the implementation of these strategies within each baseline, we refer the reader to Appendix E.

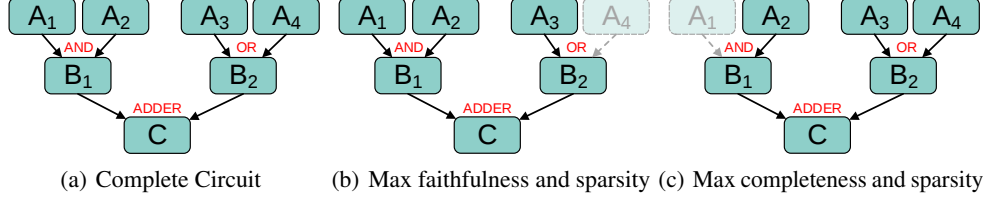


Figure 2: Presentation of a toy model designed to elucidate the logical relationships among faithfulness, completeness, and sparsity. Suppose that $A_1 \wedge A_2 = B_1$, $A_3 \vee A_4 = B_2$, and $B_1 + B_2 = C$, and among the three only C is connected to the output. When optimizing for faithfulness and sparsity alone, it is possible to remove one edge from the OR gate (either $A_3 \rightarrow B_2$ or $A_4 \rightarrow B_2$), thereby ensuring the minimal number of edges. Similarly, when optimizing for completeness and sparsity, one edge from the AND gate (either $A_1 \rightarrow B_1$ or $A_2 \rightarrow B_1$) can be eliminated for sparsity.

In Figure 1, for **AND** gates, the ΔKL values resulting from removing one versus two edges are similar, consistent with the conclusion that the disruption of any single edge in AND gates renders the gate ineffective. For **OR** gates, removing a single edge has little effect on ΔKL , supporting the idea that the OR gates remains functional as long as at least one edge in OR gates remains intact. In contrast, for **ADDER** gates, removing two edges leads to a significantly larger increase in ΔKL compared to removing one, indicating that the edges contribute independently to the gate’s function.

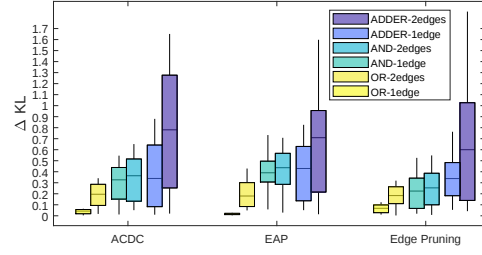


Figure 1: ΔKL in removing 1 and 2 edges of AND, OR, ADDER gates.

3.2 How to Use Logical Gates to Interpret Faithfulness and Completeness

These logical gates reveal some interesting phenomena as shown in Figure 2. For optimal faithfulness and sparsity, the circuit only needs to include **one** edge from each OR gate. For optimal completeness and sparsity, the circuit only needs to include **one** edge from each AND gate. Based on the definitions in Section 2, we can draw the following corollary regarding these properties:

Corollary 1. *The minimal edge subset that satisfies optimal faithfulness consists of **all** edges from the AND gates, **all** edges from the ADDER gates, and any **one** edge from each OR gate. The minimal edge subset that satisfies optimal completeness consists of **all** edges from the OR gates, **all** edges from the ADDER gates, and any **one** edge from each AND gate (The proofs are shown in Appendix C).*

Corollary 1 provides an explanation for why existing circuit discovery algorithms predominantly adopt the Ns strategy. The reason is that Ns is able to recover complete AND and ADDER gates, while its recovery of OR gates remains incomplete. This tradeoff corresponds precisely to the optimal balance between faithfulness and sparsity. Conversely, the Dn strategy is not adopted because it fails to recover complete AND gates, thereby severely compromising the faithfulness of the resulting circuit.

Therefore, we further evaluate the performance of existing work in terms of faithfulness and completeness. Table 1 presents the specific results for the three types of circuit discovery methods mentioned in Section 2.

Among these, methods based on greedy search and differentiable masks can identify partial OR gates, whereas methods based on linear estimation are unable to detect any edges of OR gates. Similarly, methods from Dn exhibit a similar pattern. While they can completely identify OR and ADDER gates, methods based on greedy search and differentiable masks can detect partial AND gates, while methods based on linear estimation fail to identify any. In Appendix D, we explain why greedy search and differentiable mask methods are able to identify some edges, whereas linear estimation completely fails to do so. Moreover, inspired by [2], we design a simple one-layer transformer toy model to implement the basic AND, OR, and ADDER gates, and validate the performance of these circuit discovery methods corresponding to the conclusion from Table 1.

Table 1: Capabilities and performances of three types of circuit discovery methods in recovering logical gates, faithfulness, and completeness. The symbol $\checkmark$ represents the ability to fully satisfy the corresponding requirement, $\times$ indicates the complete inability to satisfy the corresponding requirement, and $\bigcirc$ denotes the ability to partially satisfy the corresponding requirement.

Strategy	Method	AND	OR	ADDER	Faithfulness	Completeness
Ns	greedy search [2, 18, 11]	$\checkmark$	$\bigcirc$	$\checkmark$	$\checkmark$	$\times$
	linear estimation [8, 19]	$\checkmark$	$\times$	$\checkmark$	$\times$	$\times$
	differentiable mask [9, 12, 10]	$\checkmark$	$\bigcirc$	$\checkmark$	$\checkmark$	$\times$
Dn	greedy search [2, 18, 11]	$\bigcirc$	$\checkmark$	$\checkmark$	$\times$	$\checkmark$
	linear estimation [8, 19]	$\times$	$\checkmark$	$\checkmark$	$\times$	$\times$
	differentiable mask [9, 12, 10]	$\bigcirc$	$\checkmark$	$\checkmark$	$\times$	$\checkmark$

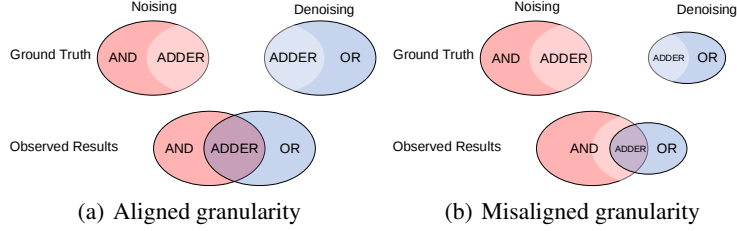


Figure 3: A Venn diagram for $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$. In the case of granularity alignment, the intersection correctly separates the AND, OR, and ADDER gates (left figure). However, in the case of misalignment, it results in some ADDER gates being incorrectly classified as AND (or OR) gates (right figure).

3.3 Granularity Alignment between $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$

The intersection operation mentioned above raises concerns about the **granularity alignment** between $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$. As illustrated in Figure 3, if the number of edges in $\mathcal{C}_{Ns}$ significantly exceeds that in $\mathcal{C}_{Dn}$, some edges identified as the AND gates could belong to the true type of the ADDER gates. This misalignment in granularity can also occur when $\mathcal{C}_{Dn}$ is considerably larger. Therefore, we propose two metrics (refer to Appendix F) to assess the degree of misalignment between $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$ when performing intersection.

We report the misalignment of $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$ at different scales in Appendix F. The results indicate that when the number of edges in the Dn circuit is approximately equal to that in the Ns circuit, both the misalignment score and its standard deviation reach an acceptable level. Therefore, throughout this paper, we assume that the optimal alignment occurs when Ns and Dn contain an **equal number of edges** and conduct experiments based on this assumption by scaling the number of edges identified by Ns and Dn strategies in a similar range.

4 Discovering Logically Sound Circuit

4.1 Optimization for Logically Sound Circuit

Existing baseline methods are capable of recovering only complete AND and ADDER structures, as demonstrated in Table 1. Therefore, the recovery of complete OR gates remains a challenge. Several approaches can be considered to address this problem, such as introducing additional combinations of interventions or varying the order of the intervention to identify different surviving edges of the OR gate, or incorporating a completeness score, such as $D(\mathcal{G} \setminus \mathcal{C} \parallel \mathcal{G})$, into the circuit discovery process. However, these approaches come with significant drawbacks. Expanding the space of intervention combinations renders circuit discovery an NP problem. Meanwhile, the inclusion of completeness scores is incompatible with non-differentiable optimization strategies such as greedy search, and it also fails to effectively split the three logic gate types in the recovered circuits.

Therefore, we propose a combined **Ns+Dn** approach to recover logically complete gates. This method is compatible with a wide range of circuit discovery algorithms, introduces minimal additional computational overhead, and enables clear and effective separation of the three types of logic gates.

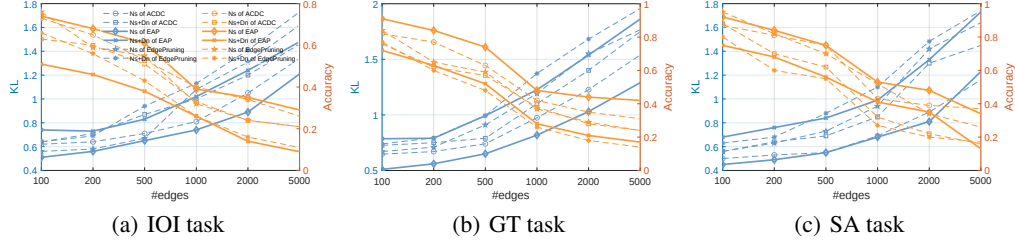


Figure 4: Completeness evaluation of circuit from Ns and NS+Dn.

Ns+Dn has the following objective:

$$\arg \min_{\mathcal{C}} \mathbb{E}_{(x, \tilde{x}) \in \mathcal{T}} [D(p_{\mathcal{G}}(y|x) || p_{\mathcal{C}}(y|x, \tilde{x})) + D(p_{\mathcal{G}}(\tilde{y}|\tilde{x}) || p_{\mathcal{C}}(\tilde{y}|\tilde{x}, x))], \text{ s.t. } 1 - |\mathcal{C}|/|\mathcal{G}| \geq s \quad (3)$$

In brief, for each baseline, we modify its implementation to perform both the Ns and Dn strategies in parallel, whereas originally only the Ns strategy was applied.

4.2 Validation of Logically Sound Circuit

In this subsection, we focus on the **faithfulness** and **completeness** of the logically sound circuit (from our framework in Section 4.1, denoted by $\mathcal{C}_{\text{Ns+Dn}}$) and the circuit of existing work (since existing work generally adopts Ns as a basic intervention strategy, we denote it by $\mathcal{C}_{\text{Ns}}$). Similar to Section 3.1, we select GPT2-small as the computational graph, and ACDC, EAP, and EdgePruning as methods to represent greedy search, linear estimation, and differentiable mask, respectively. We examine the circuits obtained through Ns, Dn, and Ns+Dn. For instance, in ACDC, when intervening on each edge, we simultaneously compute the effect of substituting the clean activation with a corrupted one in the clean run, and the effect of substituting the corrupted activation with a clean one in the corrupted run. In EAP, we compute gradients under both clean and corrupted conditions. For EdgePruning, we replace Equation 1 with Equation 3 as the optimization objective. Detailed implementation can be found in Appendix E.3. These experiments are conducted on three mainstream tasks for circuit discovery, namely indirect object inference (IOI) [16], greater than (GT) [26], and syntactic agreement [9]. The details of these tasks are presented in Table 2.

Table 2: An overview of the tasks and datasets.

Task	Example([Corrupted text])	Output	corrupted output
IOI	When Mary and John went to the store, John (Alice) gave a drink to	Mary	other names
GT	The war lasted from 1517 (1501) to 15	18 or 19 or... 99	other digits
SA	Many girls (girl) insulted	themselves	herself

4.2.1 Completeness

Following the definition of completeness in Section 2, we first compare the changes in KL divergence and accuracy for the corresponding tasks (IOI, GT, SA) when the circuit is removed from the computational graph. Specifically, we compare the differences between the original circuits (obtained through Ns) and the logically sound circuits (obtained through Ns+Dn) after removal, for three methods: ACDC, EAP, and EdgePruning. To account for the effects of sparsity, we constrain the number of edges in both circuits to remain consistent across six sparsity levels: 100, 200, 500, 1000, 2000, and 5000 edges. Figure 4 shows that, both in terms of KL divergence and accuracy, the performance of circuits removed through Ns+Dn is noticeably weaker compared to those removed through Ns. This corroborates Corollary 1, where we note that Ns, due to its inability to fully recover the OR gate, results in suboptimal completeness. Additionally, we observe that the gap between Ns and Ns+Dn is largest in both metrics in the EAP method (see the solid line in Figure 4), where Ns fails to recover any edges of the OR gate. Additionally, since ACDC and EdgePruning are generally able to identify one OR edge, the recovered OR edge exhibits some degree of randomness. In the case of ACDC, this randomness is influenced by the search order, while in the case of EdgePruning, it is influenced by the initial values of the mask. Moreover, we show the detailed results including Dn in Appendix G.

Table 3: Difference in Hamming distance between the $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Ns+Dn}$ (we compute the average Hamming distance between $\mathcal{C}_{Ns}$ and subtract the average Hamming distance between $\mathcal{C}_{Ns+Dn}$). A larger value indicates that the circuits obtained through Ns exhibit greater randomness compared to those obtained through Ns+Dn. #edges represents the number of edges in circuits.

#edges	IOI			GT			SA		
	ACDC	EAP	EdgePruning	ACDC	EAP	EdgePruning	ACDC	EAP	EdgePruning
100	3.4±0.6	0.6±0.1	8.4±3.7	4.8±0.9	0.5±0.1	12.7±4.9	2.8±0.4	1.1±0.2	15.3±5.8
200	5.9±1.3	1.2±0.3	18.1±6.7	6.7±1.8	1.3±0.2	22.5±9.1	4.3±0.9	2.2±0.5	28.4±12.7
500	14.7±3.7	1.8±0.7	44.5±13.8	16.9±4.2	1.6±0.8	49.1±15.6	12.8±2.9	2.9±0.9	55.9±16.7
1000	21.8±5.3	4.7±1.8	89.6±27.9	23.6±6.4	4.4±1.6	97.5±29.4	19.7±4.3	5.7±2.8	108.2±31.4
2000	49.5±12.9	7.9±2.9	195.3±57.8	55.7±14.9	8.6±3.5	211.7±66.2	44.8±15.2	8.8±3.1	237.4±64.8
5000	127±28.5	14.5±6.9	509.5±164.7	136.5±33.4	15.9±6.1	564.8±181.1	113.7±5.8	15.4±5.8	688.9±144.5

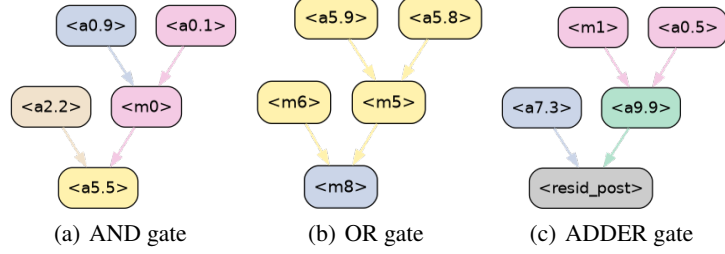


Figure 6: The cases with 2-layer gates of AND, OR, ADDER circuits.

To further validate completeness, we test the overlap of randomly generated circuits by extracting 30 distinct circuits under different random seeds and calculating the Hamming distance between pairs of these circuits to assess the randomness of the discovered circuits. Table 3 shows that the randomness of ACDC and EdgePruning is significantly higher than that of EAP, and it increases with the sparsity scales (as more OR gates are discovered). Additionally, the randomness of the circuits obtained through Ns+Dn is consistently lower than that of the circuits obtained through Ns, further supporting the claim that the inclusion of all three logical gates ensures optimal completeness.

4.2.2 Faithfulness

In Appendix H, we compare the circuits obtained using three strategies—Ns, Dn, and Ns+Dn—under the same sparsity constraints (specifically, we select edge counts of 100, 200, 500, 1000, 2000, and 5000) in terms of KL divergence and accuracy. The results show that, in terms of faithfulness, we have the relationship: $\mathcal{C}_{Ns+Dn} \approx \mathcal{C}_{Ns} > \mathcal{C}_{Dn}$. Figure 5 illustrates the average of the three methods on the IOI task to corroborate this conclusion. More results can be found in Figure 9 (a)-(c), which further supports the faithfulness requirements asserted in Corollary 1.

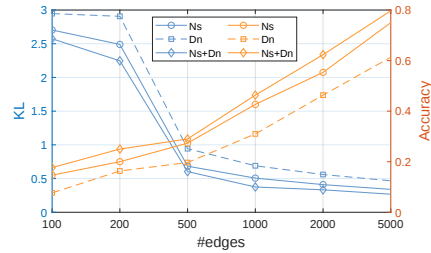


Figure 5: The average of three methods in faithfulness of IOI task.

5 Exploration on Logical Gates

5.1 Graph Study

In Appendix I, we present the circuits of the AND, OR, and ADDER gates recovered by ACDC on the IOI task and map the functions of the components to those in the previous IOI circuit [16]. We show parts of these circuits as demonstrations in Figure 6; each color represents one function in IOI circuit and blocks represent components at different locations, such as “a5.9” indicating the 9-th attention head in the 5-th layer, and “m8” referring to the MLP in the 8-th layer. The complete gate circuit can be found in Figure 10 of Appendix I. The results are interesting, revealing that the **AND gates typically receive edges from different functions**, suggesting that these functions must work together to support the receiver’s activation. In contrast, **the OR gates almost exclusively receive edges from the same function**, indicating that these edges are likely interchangeable due to their

execution of the same function. **The ADDER gates, on the other hand, tend to focus on combining two functions from different layers**, with the activation generally considering the outputs of both shallow-layer and deep-layer functions.

5.2 Output Contribution

In Appendix J, we investigate the contribution of three types of logic gates to the output. The gate effect represents the contribution of the entire gate to the output and the edge effect represents the average contribution of each edge to the output. The results show that the contribution of the ADDER gates is significantly higher than that of the AND and OR gates. Furthermore, methods that focus on the edge effect, such as differentiable masks and linear estimation, lead to a higher average effect in the recovered circuit.

5.3 Proportion

In Appendix K, we present the number of AND, OR, and ADDER edges recovered by different methods. **The results indicate that the proportion is closely related to the type of circuit discovery method used.** For instance, greedy search selects all edges beyond the threshold, resulting in nearly equal numbers of AND, OR, and ADDER edges. In contrast, differentiable mask methods calculate the effect of each edge, which is disadvantageous for gates like AND and OR that contain multiple edges. As a result, the number of ADDER edges is significantly higher.

6 Conclusions

This paper systematically introduces three logic gates—AND, OR, and ADDER—to explain the essential requirements of circuit faithfulness and completeness. Furthermore, it provides an analysis of how existing circuit discovery methods perform with respect to these logic gates. Additionally, we propose an Ns&Dn-based method for separating the three logic gates, and for restoring a logically sound circuit. We empirically validate the differences in faithfulness and completeness between the logically sound circuit and existing circuits. Finally, we explore the relationships between the logic gates in terms of distribution, contribution, and functionality.

6.1 Limitations and Future Research

The three logical gates under discussion in this paper are, in principle, derived from the intersection of $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$. Nevertheless, it must be acknowledged that other types of logical gates may also exist—for instance, the XOR gate—whose underlying logic cannot be captured solely through the simple intersection of $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$. That said, we contend that the three gates obtained from this intersection—AND, OR, and ADDER—are already sufficient to cover all edges of the circuits constructed from $\mathcal{C}_{Ns}$ and $\mathcal{C}_{Dn}$. For the purposes of the present scope of research, this coverage is adequate.

Moreover, a logically sound circuit provides a more granular and logically coherent perspective on the interpretability of a circuit. With a complete understanding of the logical relationships between edges, the circuit becomes more useful for offering insights into model control. For instance, a logically sound circuit for different tasks can reflect whether the skills associated with these tasks can be combined. That is, if the circuit for task A requires the presence of $i \rightarrow j$, and the circuit for task B requires the removal of $i \rightarrow j$, knowing that $i \rightarrow j$ exists as an OR edge in task A resolves the conflict between the two circuits. Thus, a logically sound circuit offers a novel approach for verifying the potential combination of tasks through boolean satisfiability, which is treated as our future study. We have demonstrated the potential contributions of completeness research with a toy task of model unlearning, as shown in Appendix L.

6.2 Societal and Ethical Impact

Our work aims to facilitate the process of understanding and explaining the logical connections in language models, which is crucial for their continued safe development and deployment. We do not foresee logically sound circuit and logical gates being used towards adverse societal or ethical ends.

References

- [1] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- [2] Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.
- [3] Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.
- [4] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.
- [5] Aliyah R Hsu, Georgia Zhou, Yeshwanth Cherapanamjeri, Yaxuan Huang, Anobel Y Odisho, Peter R Carroll, and Bin Yu. Efficient automated circuit discovery in transformers using contextual decomposition. *arXiv preprint arXiv:2407.00886*, 2024.
- [6] Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. Position-aware automatic circuit discovery. *arXiv preprint arXiv:2502.04577*, 2025.
- [7] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.
- [8] Aaqib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, 2024.
- [9] Lei Yu, Jingcheng Niu, Zining Zhu, and Gerald Penn. Functional faithfulness in the wild: Circuit discovery with differentiable computation graph pruning. *arXiv preprint arXiv:2407.03779*, 2024.
- [10] Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. *Advances in Neural Information Processing Systems*, 37:18506–18534, 2024.
- [11] Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.
- [12] Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 16–27, 2022.
- [13] Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. Discovering knowledge-critical subnetworks in pretrained language models. *arXiv preprint arXiv:2310.03084*, 2023.
- [14] Aaron Mueller. Missed causes and ambiguous effects: Counterfactuals pose challenges for interpreting neural networks. In *ICML 2024 Workshop on Mechanistic Interpretability*.
- [15] Hang Chen, Jiaying Zhu, Xinyu Yang, and Wenya Wang. Unveiling language skills under circuits. *arXiv preprint arXiv:2410.01334*, 2024.
- [16] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *The Eleventh International Conference on Learning Representations*.

- [17] Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.
- [18] Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- [19] Neel Nanda. Attribution patching: Activation patching at industrial scale. URL: <https://www.neelnanda.io/mechanistic-interpretability/attribution-patching>, 2023.
- [20] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.
- [21] Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022. <https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing>.
- [22] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.
- [23] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.
- [24] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.
- [25] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. *arXiv preprint arXiv:2105.08919*, 2021.
- [26] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.
- [27] Joy Crosbie and Ekaterina Shutova. Induction heads as an essential mechanism for pattern matching in in-context learning. *arXiv preprint arXiv:2407.07011*, 2024.
- [28] Jie Ren, Qipeng Guo, Hang Yan, Dongrui Liu, Xipeng Qiu, and Dahua Lin. Identifying semantic induction heads to understand in-context learning. *arXiv preprint arXiv:2402.13055*, 2024.
- [29] Benjamin L Edelman, Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*, 2024.
- [30] Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704, 2023.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction clearly state the claims made, including the contributions made in the paper and a summary of the important work.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discussed the limitations in Section 6. Moreover, Section 3.1 shows the limitations of our corollary, including considering that the receiver node is directly connected to the output.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Section 3.1 provides full hypotheses and detailed examples to explain our theoretical definition.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We provide full settings of our experiments in Appendix E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will make it open-access with github link.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 4.1 and Section 3.3 provide full settings and details of our two experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 3, Figure 8 provide clear error bars with σ .

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix E.3 we claimed that all experiments only need low computational cost as in the original setting of the existing work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We checked the NeurIPS Code of Ethics and ensured that our paper satisfies it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discussed the broader impacts to ML society in Section 6.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [\[Yes\]](#)

Justification: We listed the pretrained LMs in footnotes and described the scraped datasets in Appendix E.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: We listed the URLs and references of existing assets that we used in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We didn't release any pre-trained models and large-scale datasets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We didn't conduct any research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing and research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The paper does not involve any LLMs tools for core methodology, scientific rigorousness, or originality of the research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A The Preference between Two Types of Completeness Metrics

The earlier completeness evaluation method (proposed by [16]) in fact measures *incompleteness*, defined as $D(C \setminus K \parallel G \setminus K)$, where a lower score indicates better performance. In contrast, the current method [9] measures *completeness*, defined as $D(G \setminus C \parallel G)$, where a higher score is preferable. Importantly, these two metrics are not directly interchangeable — that is, $1 - D(C \setminus K \parallel G \setminus K) \neq D(G \setminus C \parallel G)$.

Nevertheless, a comparison can still be made in terms of the **standard deviation**. It has been theoretically demonstrated that the earlier completeness method suffers from high variance due to insufficient sampling, leading to instability in the final results. To illustrate this, we provide a simple comparison between the previous and current evaluation methods, as shown in the Table 4:

Table 4: The standard deviation between two metrics of completeness.

Method	5 sampled K	10 sampled K	20 sampled K	30 sampled K	New metric
ACDC	1.1±1.6	1.8±1.2	1.4±1.1	1.2±0.6	1.3±0.13
EAP	1.7±1.5	1.3±1.6	1.6±1.2	1.3±0.7	1.4±0.11
Edge-P	1.5±1.8	1.4±1.9	1.4±1.6	1.6±0.6	1.2±0.08
Ours	0.4±0.3	0.6±0.4	0.3±0.3	0.3±0.2	1.8±0.03

We conducted five different runs of circuit discovery using random seeds, with the random subset K chosen from subcircuits of sizes ranging from 2 to 5 nodes. For the previous evaluation metric, we performed 5, 10, 20, and 30 sampling iterations.

The experimental results clearly support two conclusions:

1. For algorithms that inherently lack completeness (such as ACDC, EAP, and Edge-Pruning), as well as for our own method, which ensures completeness, the previous evaluation metric exhibits large standard deviations across all trials. This indicates that the sampling process for the previous metric is insufficient, rendering the results highly unreliable.
2. The new evaluation metric significantly reduces variance for all types of circuits, as it does not suffer from sampling issues. The variance observed in the results arises from minor differences in the circuits generated by different random seeds.

B How Does Circuit Logic Model Intervention?

B.1 AND Gate

Consider a simple logical gate: $A_1 \wedge A_2 = B$ (only if both x_{A_1} and x_{A_2} are present can B be activated), where B directly influences the output. For noising-based intervention (**Ns**), replacing x_{A_1} with $\tilde{x}_{A_1}$, or x_{A_2} with $\tilde{x}_{A_2}$, produces a significant effect on the output. Thus, Ns is capable of detecting the structure $\{(A_1, A_2), B\}$.

However, for denoising-based intervention (**Dn**), substituting $\tilde{x}_{A_1}$ with x_{A_1} alone does not yield a noticeable change in the output, as $\tilde{x}_{A_2}$ remains present. Similarly, replacing $\tilde{x}_{A_2}$ with x_{A_2} while $\tilde{x}_{A_1}$ is still active also fails to significantly affect the output.

Under a greedy search strategy, if $\tilde{x}_{A_1}$ is first replaced by x_{A_1} and the output remains unchanged, the algorithm concludes that A_1 is not relevant and removes it (i.e., replaces $\tilde{x}_{A_1}$ with x_{A_1} in the scenario of Dn). Subsequently, replacing $\tilde{x}_{A_2}$ with x_{A_2} causes a substantial shift in the output due to the presence of x_{A_1} , thereby restoring the structure $\{(A_2), B\}$, since A_1 has already been removed.

Analogously, a greedy search that begins with A_2 and then proceeds to A_1 would only recover the structure $\{(A_1), B\}$. Therefore, we conclude that **Ns is capable of identifying the complete AND gate structure, whereas Dn either fails to detect the AND gates or only partially recovers it under greedy search conditions.**

B.2 OR Gate

Consider a simple logical gate: $A_1 \vee A_2 = B$ (if either x_{A_1} or x_{A_2} are present can B be activated), where B directly influences the output. For Ns, replacing x_{A_1} with $\tilde{x}_{A_1}$ alone does not yield a noticeable change in the output, as x_{A_2} remains present. Similarly, replacing x_{A_2} with $\tilde{x}_{A_2}$ while x_{A_1} is still active also fails to significantly affect the output.

Under a greedy search strategy, if x_{A_1} is first replaced by $\tilde{x}_{A_1}$ and the output remains unchanged, the algorithm concludes that A_1 is not relevant and removes it (i.e., retains $\tilde{x}_{A_1}$). Subsequently, replacing x_{A_2} with $\tilde{x}_{A_2}$ causes a substantial shift in the output due to the lack of support of x_{A_1} , thereby restoring the structure $\{(A_2), B\}$, since A_1 has already been removed.

Analogously, a greedy search that begins with A_2 and then proceeds to A_1 would only recover the structure $\{(A_1), B\}$.

However, for Dn, replacing $\tilde{x}_{A_1}$ with x_{A_1} , or $\tilde{x}_{A_2}$ with x_{A_2} , produces a significant effect on the output. Thus, Dn is capable of detecting the structure $\{(A_1, A_2), B\}$.

Therefore, we conclude that **Dn is capable of identifying the complete OR gate structure, whereas Ns either fails to detect the OR gates or only partially recovers it under greedy search conditions.**

B.3 ADDER Gate

Consider a simple logical gate: $A_1 + A_2 = B$, where B directly influences the output. Since the influence of each edge in an ADDER gate is independent, removing an edge in either Ns or Dn directly impacts the output via its effect on B . For instance, in Ns, replacing x_{A_1} with $\tilde{x}_{A_1}$ results in $B^* = A_2$, which is significantly smaller than $B = A_1 + A_2$. Similarly, in Dn, replacing $\tilde{x}_{A_1}$ with x_{A_1} yields $B^* = A_1$, which is substantially greater than $B = 0$. **Therefore, both Ns and Dn are capable of identifying the complete structure of the ADDER gate.**

C How Does Circuit Logic Affect Faithfulness, Completeness, and Sparsity?

C.1 Faithfulness

As introduced in Section 2, faithfulness requires that $D(G||C)$ be minimized. Let us consider the following scenarios:

- For any gate $\{(A_1, A_2), B\}$, if the circuit does not include all edges or nodes from this gate, it is always possible to find a circuit $C^* = C \cup A_1, A_2, B$ such that $D(G||C) > D(G||C^*)$.
- For an AND gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , the gate effect of this AND gate is not maximized (the influence of B is maximized when both A_1 and A_2 are present). Therefore, it is always possible to find a circuit $C^* = C \cup A_2$ such that $D(G||C) > D(G||C^*)$.
- For an ADDER gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , the gate effect of this ADDER gate is not maximized (again, the influence of B is maximized when both A_1 and A_2 are present). Thus, there exists a circuit $C^* = C \cup A_2$ such that $D(G||C) > D(G||C^*)$.
- For an OR gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , the gate effect of this OR gate is already maximized (the same applies if only A_2 and B are included). Therefore, for $C^* = C \cup A_2$, we have $D(G||C) = D(G||C^*)$. However, from the perspective of sparsity, $|C^*| > |C|$.

Thus, to achieve optimal faithfulness, the circuit must include all edges that result in the maximum gate effects, namely all edges from the AND, ADDER, and OR gates. However, considering sparsity, the gate effect sum remains maximal even if only one edge from each OR gate is retained.

C.2 Completeness

Similarly, completeness requires that $D(G \setminus C||G)$ be maximized. Consider the following scenarios:

- For any gate $\{(A_1, A_2), B\}$, if the circuit does not include all edges or nodes from this gate, it is always possible to find a circuit $C^* = C \cup A_1, A_2, B$ such that $D(G \setminus C || G) < D(G \setminus C^* || G)$.
- For an AND gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , then $G \setminus C$ will only contain A_2 or the edge $A_2 \rightarrow B$ (depending on whether pruning is applied to edges or nodes). Due to the AND operation, B will not produce a gate effect. Therefore, for the circuit $C^* = C \cup A_2$, we have $D(G \setminus C || G) = D(G \setminus C^* || G)$.
- For an OR gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , then $G \setminus C$ will only contain A_2 or the edge $A_2 \rightarrow B$. Due to the OR operation, B still produces a gate effect. Therefore, it is always possible to find a circuit $C^* = C \cup A_2$ such that $D(G \setminus C || G) < D(G \setminus C^* || G)$.
- For an ADDER gate $\{(A_1, A_2), B\}$, if the circuit C only includes A_1 and B , then $G \setminus C$ will only contain A_2 or the edge $A_2 \rightarrow B$. Due to the ADDER operation, B still produces a gate effect. Therefore, it is always possible to find a circuit $C^* = C \cup A_2$ such that $D(G \setminus C || G) < D(G \setminus C^* || G)$.

Thus, to achieve optimal completeness, the circuit must include all edges that result in the maximum gate effects, namely all edges from the AND, ADDER, and OR gates. However, considering sparsity, the total gate effect remains maximized even if only one edge is retained for each AND gate.

D Validation of Logical Gates

D.1 Toy Model

Motivated by [2], to study a toy transformer model with an AND, OR, and ADDER gates, we take a 1-Layer transformer model with two heads per layer, ReLU-based activations, and model dimension 1. Specifically, as shown in Figure 7, Let A_1 and A_2 be two attention heads with respective biases bias_1 and bias_2 , both set to 1. The activation function m is based on the ReLU nonlinearity. To ensure that the output of each attention head corresponds directly to its bias, we use a zero tensor as the input. For corrupted activations, we employ zero ablation—i.e., we directly remove the activations along the corresponding edges.

The activation function m is configured differently to simulate logical gates as follows:

- **AND** gate: $m(x) = \text{ReLU}(x - 1)$. Under this setting, the output is 1 only when both A_1 and A_2 are active (i.e., not ablated); otherwise, the output is 0.
- **OR** gate: $m(x) = 1 - \text{ReLU}(1 - x)$. Here, the output is 1 as long as at least one of A_1 or A_2 is active; if both are ablated, the output is 0.
- **ADDER** gate: The bias_2 is modified to 1.5, and $m(x) = \text{ReLU}(x)$. In this case, the output is 0 when both A_1 and A_2 are ablated; it is 1.5 when only A_1 is ablated, 1 when only A_2 is ablated, and 2.5 when both are active.

Under these configurations, we evaluate the performance of existing methods on the toy model, as summarized in Table 1. For example, under the default Ns, ACDC [2] (representing greedy search), EAP [8] (representing linear estimation), and EdgePruning [10] (representing differentiable mask) all successfully identify both A_1 and A_2 in the AND and ADDER gates. However, in the OR gate, ACDC and EdgePruning identify only one of A_1 or A_2 —the specific result depends on the search order in ACDC and the initialization of the mask in EdgePruning—while EAP fails to identify any high-effect edge.

Conversely, when these methods are executed under Dn, the outcomes are reversed. In the OR and ADDER gates, all three methods, ACDC, EAP and EdgePruning, can now identify both A_1 and A_2 . However, in the AND gate, only ACDC and EdgePruning are able to recover one of A_1 or A_2 , whereas EAP considers the effects of both to be insufficiently strong.

Ns can at least guarantee the full recovery of AND and ADDER gates. Greedy search, by retaining previous steps with removed results, can identify one OR

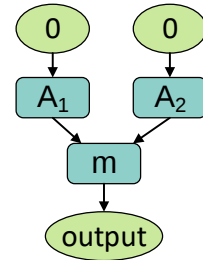


Figure 7: A toy model to study AND, OR, and ADDER gates.

edge (a detailed analysis is provided in Appendix B). Differentiable mask, optimizing for faithfulness (as per Corollary 1), ensures that at least one OR gate is included (otherwise, optimal faithfulness cannot be achieved), while additional OR edges conflict with the sparsity constraint and are therefore removed. However, linear estimation, when computing the effect of each edge, keeps all other edges in their non-removed state, which results in a failure to detect the OR gate. For example, when computing the effect of the edge $A \rightarrow C$ in the OR gate $A \rightarrow C \leftarrow B$, the edge $C \leftarrow B$ is also in the clean activation state, leading to a very small effect for $A \rightarrow C$. Similarly, when computing the effect of $C \leftarrow B$, the edge $A \rightarrow C$ remains in the clean activation state. In summary, linear estimation is unable to detect any edges of the OR gate.

The performance on D_n is the opposite of that on N_s , where, in addition to the full OR gates and ADDER gates, greedy search and differentiable mask can similarly recover one AND edge, just as they could with the OR gates in N_s . Linear estimation also fails to detect the AND gates due to the non-removed status of the other edges. Based on Corollary 1, we are also able to derive the performance of the three types of methods in terms of faithfulness and completeness.

E Experiment Details

E.1 Why Does Search Strategy by repeating interventions on combinations is NP Problem?

Due to the differences in circuit algorithms, this issue cannot be fully proven mathematically. However, we will provide some additional explanation: we treat each ablation as $O(1)$. Assuming the language model contains N nodes, if we need to use all possible intervention combinations to obtain a complete circuit, the time complexity would be $O(2^N)$, which far exceeds polynomial time. However, different strategies can optimize the solution verification process to varying degrees. For example, EAP only requires $O(N)$ time to verify whether the circuit is complete, while Edge-pruning involves gradient descent, making its time complexity difficult to estimate. Additionally, greedy search can only perform a greedy verification in $O(N)$ time rather than a thorough verification. Therefore, we ultimately classify this problem as NP problem.

However, our proposed method, which employs the algorithm $N_s + D_n$, does not introduce additional computational complexity to the existing circuit discovery algorithm. Conceptually, it is equivalent to reapplying the N_s algorithm once more under the strategy of D_n . As a result, the overall time complexity increases by at most a factor of two relative to the baseline algorithm, without imposing any additional nonlinear burden.

E.2 Detailed Results of Figure 3

Table 5: The changes in KL divergence between the original output and the new output when one or two edges are ablated.

AND			OR			ADDER		
edge1(state)	edge2(state)	(Δ KL)(state)	edge1(state)	edge2(state)	(Δ KL)(state)	edge1(state)	edge2(state)	(Δ KL)(state)
remain (1)	remain(1)	0.50 (1)	remain (1)	remain(1)	0.28 (1)	remain (1)	remain(1)	0.65 (2)
remain (1)	ablate(0)	0.02 (0)	remain (1)	ablate(0)	0.27 (1)	remain (1)	ablate(0)	0.32 (1)
ablate (0)	remain(1)	0.04 (0)	ablate (0)	remain(1)	0.27 (1)	ablate (0)	remain(1)	0.36 (1)
ablate (0)	ablate(0)	0.00 (0)	ablate (0)	ablate(0)	0.00 (0)	ablate (0)	ablate(0)	0.00 (0)

All three subcircuits exhibit a fundamental pattern: the effect on the output is maximized when both edges are retained, and it drops to zero when both edges are ablated (as there is no clean activation left to support the node in this case). However, ablation of a single edge yields divergent outcomes:

In AND gate, ablating either edge nearly eliminates the effect on the output. In OR gate, ablating either edge has almost no impact on the output. In ADDER gate, ablating either edge leads to a noticeable degradation in the output.

E.3 Baselines

In this work, we select **ACDC** [2] to represent greedy search methods, EAP [8] to represent linear estimation methods, and EdgePruning [10] to represent differentiable mask methods. In the following

sections, we provide a detailed exposition of the original design of each method under the Ns. strategy, the corresponding formulation under the Dn. strategy, and the final approach that integrates both—Ns.+Dn.—for recovering logically complete gates.

E.3.1 Greedy Search Example: ACDC

The ACDC method identifies important edges by iteratively removing each edge and observing the effect of this intervention on the model output. Edges whose removal causes an effect greater than a predefined threshold τ are retained, while those with an effect smaller than τ are pruned. The original algorithm (Ns. strategy), is outlined as follows:

Algorithm 1: The ACDC algorithm in Ns.

Data: Computational graph $\mathcal{G}$, dataset $(x_i)_{i=1}^n$, corrupted datapoints $(x'_i)_{i=1}^n$ and threshold $\tau > 0$.

Result: Subgraph $\mathcal{H} \subseteq \mathcal{G}$.

```

1  $\mathcal{H} \leftarrow \mathcal{G}$  // Initialize H to the full computational graph
2  $\mathcal{H} \leftarrow \mathcal{H}.reverse\_topological\_sort()$  // Sort H so output first
3 for  $v \in \mathcal{H}$  do
4   for  $w$  parent of  $v$  do
5      $\mathcal{H}_{\text{new}} \leftarrow \mathcal{H} \setminus \{w \rightarrow v\}$  // Temporarily remove candidate edge
6     if  $D_{KL}(\mathcal{G} \parallel \mathcal{H}_{\text{new}}) - D_{KL}(\mathcal{G} \parallel \mathcal{H}) < \tau$  then
7        $\mathcal{H} \leftarrow \mathcal{H}_{\text{new}}$  // Edge is unimportant, remove permanently
8 return  $\mathcal{H}$ 

```

In the **Ns.** strategy, $\mathcal{G}$ denotes the **clean run**, and $\mathcal{H} \setminus \{w \rightarrow v\}$ represents the replacement of the clean activation on the edge $w \rightarrow v$ with its corrupted activation. In contrast, under the **Dn.** strategy, $\mathcal{G}$ refers to the **corrupted run**, and $\mathcal{H} \setminus \{w \rightarrow v\}$ indicates the substitution of the corrupted activation on edge $w \rightarrow v$ with the corresponding clean activation.

In the combined **Ns.+Dn.** approach, the effects from both strategies are jointly considered. Specifically, the original pruning condition $D_{KL}(\mathcal{G} \parallel \mathcal{H}_{\text{new}}) - D_{KL}(\mathcal{G} \parallel \mathcal{H}) < \tau$ is replaced with the aggregated criterion: $D_{KL}(\mathcal{G}^{\text{clean}} \parallel \mathcal{H}_{\text{new}}) - D_{KL}(\mathcal{G}^{\text{clean}} \parallel \mathcal{H}) + D_{KL}(\mathcal{G}^{\text{corrupted}} \parallel \mathcal{H}_{\text{new}}) - D_{KL}(\mathcal{G}^{\text{corrupted}} \parallel \mathcal{H}) < \tau$.

E.3.2 Linear Estimation Example: EAP

The EAP method approximates the effect of each edge using the first-order term of its Fourier expansion, enabling the estimation of all edge effects with a single forward pass. It is important to note that, during the computation of each edge's effect, all other edges remain in their unpruned (active) state.

Specifically, Ns. has approximation:

$$L(x|do(\tilde{x}_i)) - L(x) \approx (\tilde{x}_i - x_i)^T \frac{\partial}{\partial x_i} L(x) \quad (4)$$

and Dn. has approximation:

$$L(\tilde{x}|do(x_i)) - L(\tilde{x}) \approx (\tilde{x}_i - x_i)^T \frac{\partial}{\partial \tilde{x}_i} L(\tilde{x}) \quad (5)$$

Therefore, the approximation for Ns.+Dn. is $(\tilde{x}_i - x_i)^T \frac{\partial}{\partial x_i} L(x) + (\tilde{x}_i - x_i)^T \frac{\partial}{\partial \tilde{x}_i} L(\tilde{x})$.

E.3.3 Differentiable Mask Example: EdgePruning

EdgePruning assigns a learnable mask to each node or edge, where the mask is reparameterized using the hard concrete distribution. In the Ns. setting, the optimization objective corresponds to Equation 1. Consequently, the objectives for the Dn. and Ns.+Dn. settings are given by Equation 2 and Equation 3, respectively.

In the Ns.+Dn. setting, directly optimizing both objectives jointly can lead to gradient interference and convergence to Pareto-optimal solutions, rather than a unified optimum. To address this, we independently compute the final mask values for Ns. and Dn. using Equations 1 and 2, and then obtain the mask for Ns.+Dn. by averaging the two.

F Misalignment Score

Misalignment of AND: For any subcircuit $\mathcal{K}_{\text{AND}} \subset \mathcal{C}_{\text{AND}}$, $\mathcal{C}_{\text{AND}}^* = \mathcal{C}_{\text{AND}} \setminus \mathcal{K}_{\text{AND}}$. Let $i, j \in \mathcal{C}_{\text{AND}}$, $i^*, j^* \in \mathcal{C}_{\text{AND}}^*$ be any two edges with the same receiver, respectively. The score of misalignment of AND reads:

$$\mathbb{E}_{i,j}[D(\mathcal{C}_{\text{AND}} \setminus i || \mathcal{C}_{\text{AND}} \setminus i, j)] - \mathbb{E}_{i^*,j^*}[D(\mathcal{C}_{\text{AND}}^* \setminus i^* || \mathcal{C}_{\text{AND}}^* \setminus i^*, j^*)] \quad (6)$$

Equation 6 indicates that the higher the score, the higher the misalignment. This is due to their properties: the effect caused by removing one and two edges from the AND gates is similar, while the effect for the ADDER gates differs significantly.

Misalignment of OR: Similarity, let any $\mathcal{K}_{\text{OR}} \subset \mathcal{C}_{\text{OR}}$, $\mathcal{C}_{\text{OR}}^* = \mathcal{C}_{\text{OR}} \setminus \mathcal{K}_{\text{OR}}$, $i, j \in \mathcal{C}_{\text{OR}}$, $i^*, j^* \in \mathcal{C}_{\text{OR}}^*$, respectively. The score of misalignment of OR reads:

$$\mathbb{E}_{i,i^*}[D(\mathcal{C}_{\text{OR}} \setminus i || \mathcal{C}_{\text{OR}}^* \setminus i^*)] - \mathbb{E}_{i,j,i^*,j^*}[D(\mathcal{C}_{\text{OR}} \setminus i, j || \mathcal{C}_{\text{OR}}^* \setminus i^*, j^*)] + m \quad (7)$$

Equation 7 utilizes the properties that the effect does not change by removing one edge from OR gates, while it significantly changes from ADDER gates. Additionally, to avoid the bias caused by both effects of ADDER and OR edges being marginally small in large-scale circuits, we replace the “difference in one edge” with the “difference in difference between one edge and two edges,” and introduce a constant m to ensure that the score > 0 (with m set to 1.5 in practice).

Therefore, for any pair of $\mathcal{C}_{\text{Ns}}$ and $\mathcal{C}_{\text{Dn}}$, we can compute the misalignment using these two scores. We report the misalignment scores resulting from the intersection of Ns and Dn circuits at varying scales. Specifically, we select an Ns circuit consisting of 100 edges recovered from the IOI task and examine how the misalignment score changes as the number of edges in the Dn circuit varies from 60 to 140. Figure 8 illustrates that when Dn is significantly smaller than Ns, the misalignment score for the AND gates is high, as many ADDER edges are misclassified as AND edges. Conversely, when Dn is substantially larger than Ns, the misalignment score for the OR gates increases, due to many ADDER edges being misclassified as OR edges. When the number of edges in the Dn circuit is approximately equal to that in the Ns circuit, both the misalignment score and its standard deviation reach an acceptable level.

As shown in Figure 8, when the OR set significantly exceeds the AND set, a large number of misclassified ADDER gates appear only within the OR set, leading to a much higher misalignment score for OR compared to AND. The misalignment score is more of a "property," reflecting the optimal scale at which OR and AND sets should intersect. We determine the optimal ratio of AND to OR gates by minimizing the sum of the misalignment scores for AND and OR. The optimal results obtained with different methods and datasets are approximately as shown in Table 6:

Table 6: The optimal ratio of edge numbers between $\mathcal{C}_{\text{Ns}}$ and $\mathcal{C}_{\text{Dn}}$.

strategies	IOI	GT	SA
ACDC	1.12: 1	1.05: 1	1.07: 1
EAP	1.09: 1	1.03: 1	1.05: 1
Edge-Pruning	1.11: 1	1.07: 1	1.06: 1

The results show that the proportion of AND is always slightly higher than that of OR, but it can be approximated as 1:1. Therefore, throughout this paper, we assume that the optimal alignment occurs when Ns and Dn contain an **equal number of edges**.

G Experiments of Completeness

In this Appendix, we present the detailed results of the completeness validation, together with a comparison against the Dn strategy. The specific results are reported in Tables 7 and 8. In brief,

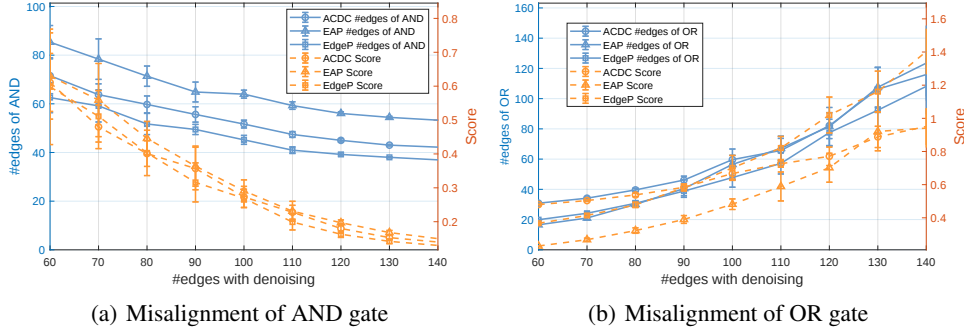


Figure 8: Misalignment score with 100-edges IOI circuit from Ns.

our method generally outperforms the approach based on Dn. Although the Dn-based method is theoretically capable of identifying all OR gates, it suffers from certain shortcomings in practice. For instance, as shown in Table 1, EAP fails to identify one of the edges associated with an AND gate under the Dn framework. Additionally, the greedy strategy may miss some OR gates located beneath branches of AND gates due to its dependence on the search order. The differentiable mask approach, on the other hand, suffers from the drawback that, in gates involving multiple edges, the mask values assigned to each edge tend to be lower, reducing its effectiveness.

Table 7: KL in Completeness Validation of IOI task, including Ns, Dn, and Ns+Dn.

#edges	ACDC_Ns	ACDC_Dn	ACDC_NsDn	EAP_Ns	EAP_Dn	EAP_NsDn	EdgeP_Ns	EdgeP_Dn	EdgeP_NsDn
100	0.62±0.1	0.64±0.1	0.64±0.1	0.51±0.1	0.71±0.1	0.74±0.1	0.56±0.1	0.62±0.1	0.64±0.1
200	0.64±0.1	0.69±0.1	0.71±0.1	0.56±0.1	0.73±0.0	0.73±0.1	0.58±0.1	0.65±0.1	0.69±0.1
500	0.71±0.1	0.84±0.1	0.87±0.1	0.65±0.1	0.81±0.1	0.83±0.1	0.67±0.1	0.88±0.1	0.94±0.1
1000	0.81±0.1	0.96±0.1	1.03±0.1	0.74±0.1	0.96±0.1	1.02±0.1	1.08±0.1	1.11±0.1	1.13±0.1
2000	1.05±0.1	1.19±0.2	1.22±0.1	0.89±0.1	1.11±0.2	1.24±0.2	1.31±0.2	1.40±0.1	1.41±0.1
5000	1.34±0.2	1.42±0.2	1.46±0.2	1.21±0.1	1.42±0.2	1.48±0.2	1.62±0.2	1.73±0.1	1.73±0.1

Table 8: Accuracy in Completeness Validation of IOI task, including Ns, Dn, and Ns+Dn.

#edges	ACDC_Ns	ACDC_Dn	ACDC_NsDn	EAP_Ns	EAP_Dn	EAP_NsDn	EdgeP_Ns	EdgeP_Dn	EdgeP_NsDn
100	0.73±0.1	0.68±0.0	0.63±0.1	0.74±0.1	0.53±0.0	0.51±0.0	0.76±0.1	0.67±0.1	0.66±0.0
200	0.65±0.0	0.62±0.0	0.60±0.0	0.68±0.0	0.49±0.0	0.46±0.0	0.59±0.0	0.55±0.0	0.56±0.0
500	0.53±0.0	0.51±0.0	0.51±0.0	0.61±0.0	0.41±0.0	0.38±0.0	0.55±0.0	0.43±0.0	0.43±0.0
1000	0.40±0.0	0.34±0.0	0.33±0.0	0.39±0.0	0.29±0.0	0.26±0.0	0.32±0.0	0.27±0.0	0.26±0.0
2000	0.34±0.0	0.26±0.0	0.24±0.0	0.35±0.0	0.18±0.0	0.14±0.0	0.24±0.0	0.18±0.0	0.16±0.0
5000	0.26±0.0	0.22±0.0	0.21±0.0	0.29±0.0	0.13±0.0	0.09±0.0	0.21±0.0	0.11±0.0	0.11±0.0

H Experiments of Faithfulness

In this section, we investigate the faithfulness of circuits obtained using three methods—ACDC, EAP, and EdgePruning—across three tasks: IOI, GT, and SA. Specifically, we examine the changes in KL divergence and accuracy between the original circuit (Ns.), the circuit with full OR and ADDER gates (Dn.) and the circuit with logically complete gates (Ns.+Dn.). For sparsity, we select edge counts of 100, 200, 500, 1000, 2000, and 5000.

Figure 9 illustrates that, under the same sparsity constraints, the circuits discovered using Dn. are significantly lower in both metrics compared to those discovered using Ns. and Ns.+Dn., which corroborates our assertion in Corollary 1: Dn. is incapable of fully recovering the AND gate, and thus cannot achieve optimal faithfulness.

Additionally, in the EAP method, Ns clearly performs much worse than Ns+Dn, whereas in the ACDC and EdgePruning methods, the performance of Ns and Ns+Dn is quite similar. This aligns with our reasoning in Table 1, where we note that only the linear estimation method completely fails to identify any OR edge, thus not satisfying the minimal requirement of faithfulness.

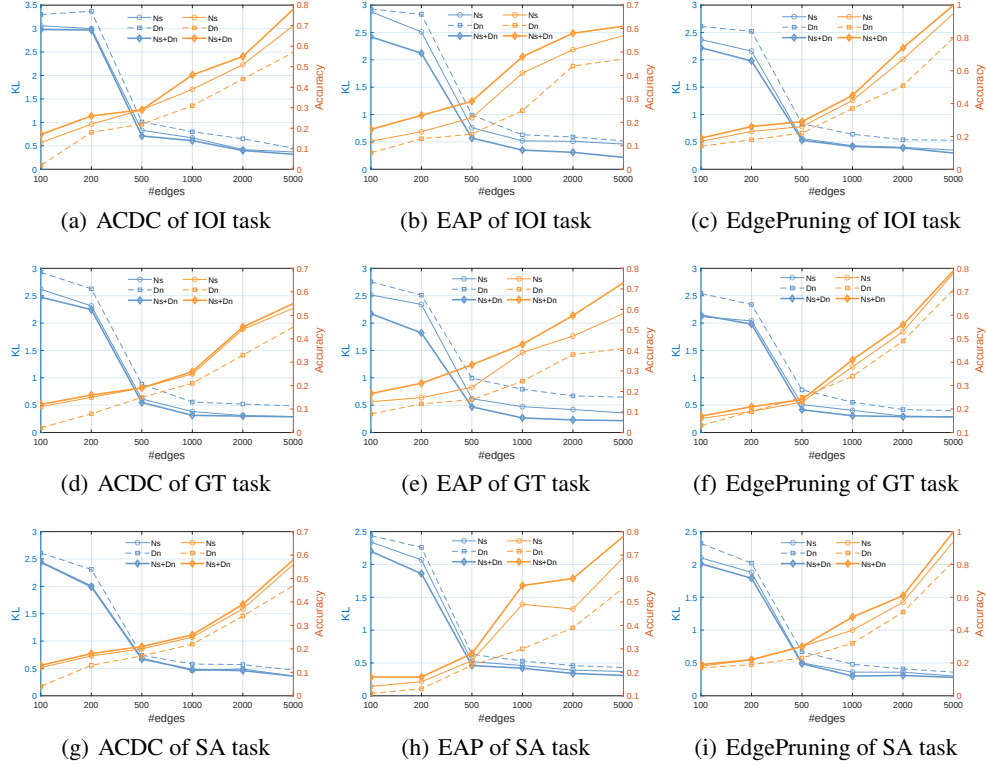


Figure 9: Faithfulness of circuit from Ns., DN., and NS.+Dn..

I Graph Study

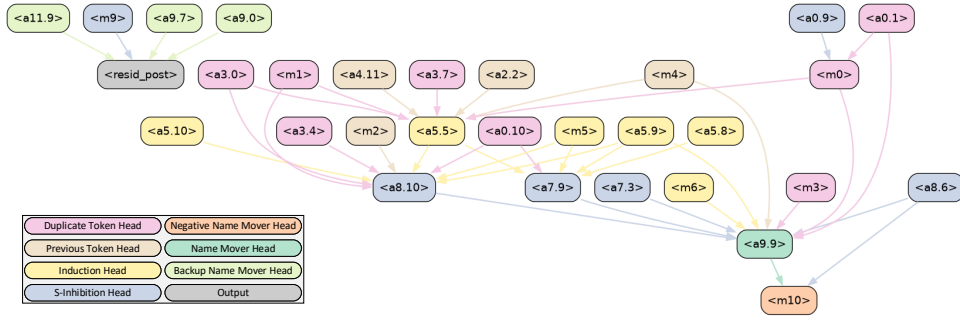
To investigate the relationship between the three logical gates and the functions of the language model, we extracted the circuits of AND, OR, and ADDER gates discovered through ACDC on the IOI task. We then reviewed the IOI circuit [16] to determine the function of each component.

Interestingly, each receiver in the AND circuit is almost always influenced by edges from **different** functions, indicating that the AND operation can be understood as combining different functions to jointly impact the subsequent layers. For example, in Figure 10(a), the Induction Head requires edges from both the Duplication Token Head and the Previous Token Head to function, which supports the mechanism behind the Induction skill [27, 28, 29]. Similarly, the Name Mover Head requires support from both the S-Inhibition Head and the Induction Head, which explains the functional mechanism of the AND operation.

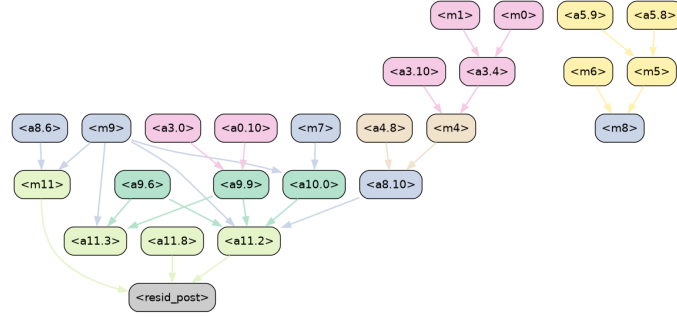
In contrast, the OR circuit clearly shows that nearly every receiver node is influenced by edges from the **same** function, as shown in Figure 10(b), suggesting that these edges from the same function are either backups or interchangeable. For instance, the S-Inhibition Head is influenced by multiple Induction Heads, and the Backup Name Mover Head is influenced by multiple S-Inhibition Heads.

Lastly, the ADDER circuit appears to focus more on the outputs of the **MLP** and often combines outputs from shallow-layer skills with those from deeper-layer skills, as shown in Figure 10(c). The Name Mover Head considers outputs from all functions between the Duplicate Token Head and the S-Inhibition Head, and the final output takes into account the combined results from all three Name Mover Heads.

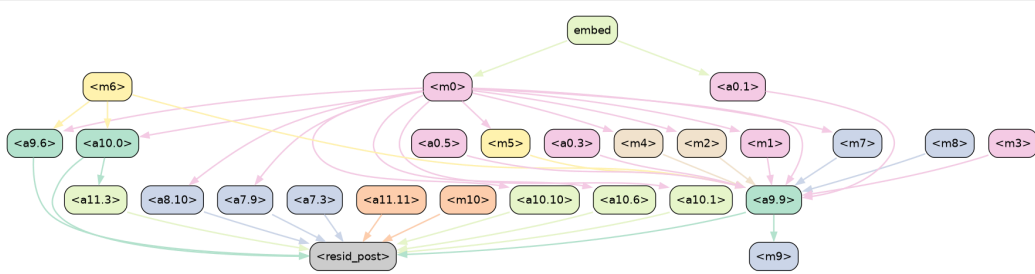
Additionally, regarding the span of gates across layers, the OR gates typically operate over the **shortest** distances, usually occurring between two functions that are close in layer position. In contrast, the ADDER gates generally span the **longest** distances, typically combining shallow-layer functions with deeper-layer functions.



(a) Circuit graph of AND gate



(b) Circuit graph of OR gate



(c) Circuit graph of ADDER gate

Figure 10: Circuit Graphs of AND, OR, and ADDER gates, respectively. We set the color of each component to be the same as that of the IOI circuit [16], allowing for easy reference to the function of each component.

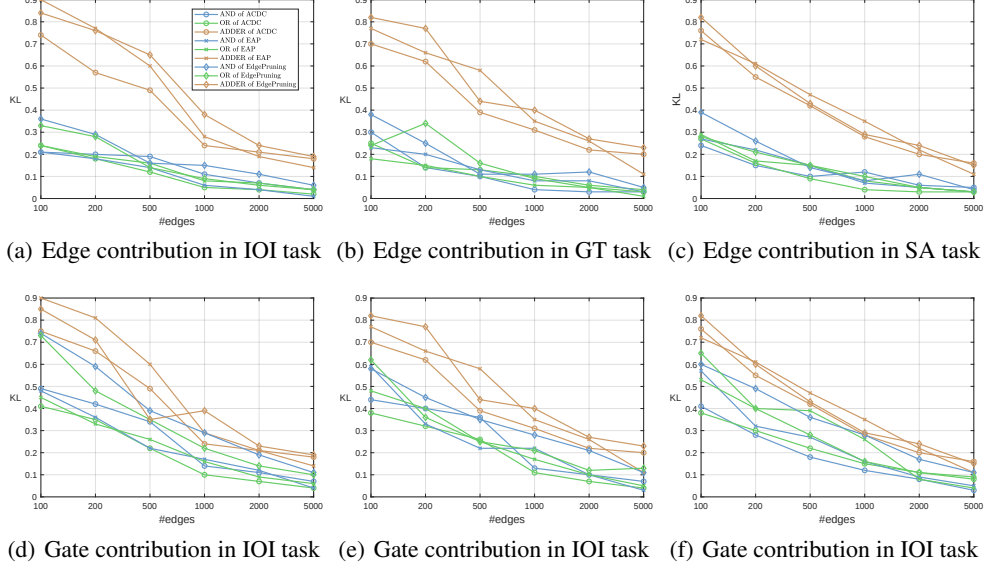


Figure 11: The contribution of different logic gates to the output."

J Output Contribution

In this section, we investigate the contribution of three types of logic gates to the output. Specifically, we calculate the change in the KL divergence of the output caused by replacing these gates. Considering the collection of gates, we analyze their contributions from two perspectives: the **gate effect** and the **edge effect**. The gate effect refers to the impact on the output caused by replacing an entire logic gate, while the edge effect corresponds to equally distributing the gate effect across each edge within the gate. For example, for an AND gate with two edges, if the activation of the receiver node contributes 0.8 to the output, the edge effect would be 0.4. Figure 11 illustrates the gate effect and edge effect of these logic gates across different tasks and baselines. Clearly, the ADDER gates exhibit the largest contribution, demonstrating its role as the primary framework of the circuit, while the contributions of the AND and OR gates are similar. Additionally, the average gate and edge effects in EdgePruning and EAP are significantly higher than those in ACDC. This is because the differentiable mask and linear estimation methods optimize (rank) based on the edge effect, ensuring that the effect within the circuit is maximized, in contrast to greedy search methods.

K Proportion of AND, OR, and ADDER Gates

Figure 12 illustrates the proportion of the three types of logical gates across different tasks for each method. Notably, in the ACDC baseline, the number of edges corresponding to each gate type is nearly equal. This is because ACDC employs a greedy search strategy without ranking edges by their effect on the output. In contrast, both EAP and EdgePruning yield significantly fewer OR edges, reflecting the fact that OR edges contribute the least to the output—a finding we detail in Section 5.2 and Appendix J. Furthermore, the results from EdgePruning indicate that the number of AND edges is similarly low, comparable to OR edges. This arises from the fact that EdgePruning optimizes based on individual edge effects rather than gate-level effects. For instance, in a gate comprising two AND edges, each edge contributes only half of the total gate effect. As a result, during optimization, the mask values for such edges may be suppressed, increasing the likelihood of pruning.

L Potential Contributions of Completeness: an Example of Model Unlearning

We have designed a toy task analogous to knowledge unlearning to demonstrate its actual effectiveness. Specifically, given a harmful-response datasets (PKU-SafeRLHF) [30], we first obtain its circuit $\mathcal{C}$, and then remove this circuit from the computation graph $\mathcal{G}$, observing whether the model

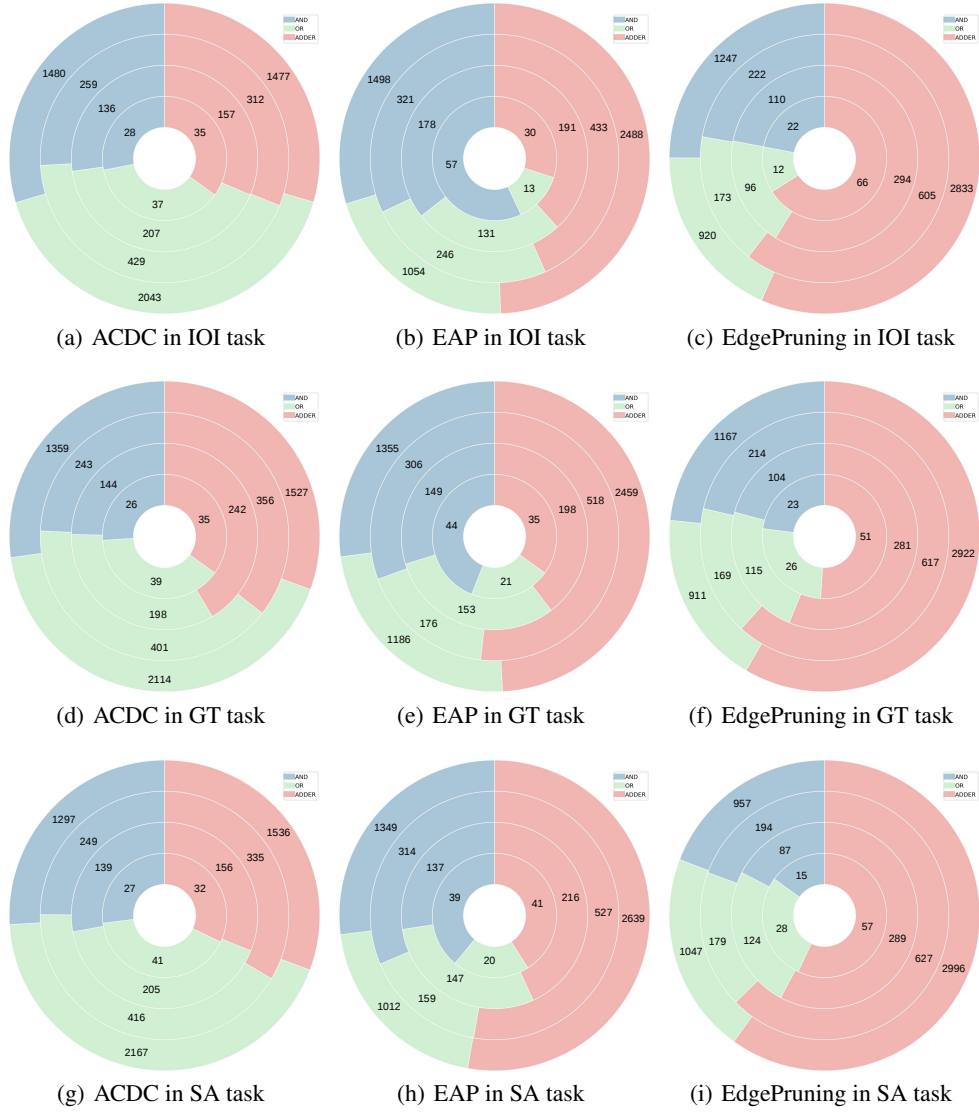


Figure 12: Proportion of AND, OR, and ADDER edges in circuit from Ns.+Dn., The concentric rings from the innermost to the outermost represent circuits with 100, 500, 1000, and 5000 edges, respectively. The blue represents AND edges, red represents ADDER edges, and green represents OR edges.

can still respond to the dataset. The dataset provides harmful responses for certain questions, which are used to evaluate the efficacy of preventing the model from generating these harmful responses.

In the Table 9, we present the performance of this toy task on the accuracy of $\mathcal{G} - \mathcal{C}$, which refers to the performance of the computational graph after the circuit has been removed (using the interchange ablation method), reflecting the completeness of the circuit. We test two circuit discovery methods: EAP and Edge-Pruning.

Table 9: A toy experiment on model unlearning task

Method	accuracy	accuracy of G-C
EAP	92.74	29.17
Edge-P	92.74	31.44
Ours	92.74	13.21

It is evident that, compared to existing circuit discovery methods, our method ensures that the computational graph loses almost all the necessary mechanisms for the task once the circuit is removed. Additionally, this toy task highlights the significance of studying circuit completeness, which holds great potential in unlearning harmful information: a complete circuit can help us pinpoint all neurons associated with harmful responses.