Uncovering Graph Reasoning in Decoder-only Transformers with Circuit Tracing

Xinnan Dai[1*] **Chung-Hsiang Lo**[2] **Kai Guo**[1] **Shenglai Zeng**[1] **Dongsheng Luo**[3] **Jiliang Tang**[1]
[1]Michigan State University, [2]Northeastern University, [3]Florida International University

Abstract

Transformer-based LLMs demonstrate strong performance on graph reasoning tasks, yet their internal mechanisms remain underexplored. To uncover these reasoning process mechanisms in a fundamental and unified view, we set the basic decoder-only transformers and explain them using the circuit-tracer framework. Through this lens, we visualize reasoning traces and identify two core mechanisms in graph reasoning: token merging and structural memorization , which underlie both path reasoning and substructure extraction tasks. We further quantify these behaviors and analyze how they are influenced by graph density and model size. Our study provides a unified interpretability framework for understanding structural reasoning in decoder-only Transformers.

1 Introduction

Recent studies suggest that LLMs possess strong structural reasoning abilities [1, 2]. To investigate the underlying reasons of this phenomenon, existing work has provided both theoretical and empirical insights within simplified graph reasoning settings [3, 4, 5], where the reasoning process is studied using decoder-only Transformers trained from scratch on graphs represented solely by node IDs. However, existing analyses remain case-specific, as different graph reasoning tasks often require distinct analytical methods. These studies still do not provide a unified understanding of the underlying mechanisms by which models perform reasoning over explicit graph structures. Therefore, understanding how decoder-only Transformers solve graph reasoning tasks requires continued investigation into their unified mechanisms.

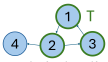
Specifically, for graph reasoning tasks, the mechanisms underlying transformers' performance in path reasoning and pattern extraction have been attributed to edge memorization [4] and progressive filtration across layers [5]. Since both tasks demonstrate that decoder-only transformers possess a fundamental capacity for structural understanding, it is reasonable to expect that a consistent underlying mechanism governs their reasoning over explicit graph-like structures encoded in textual sequences. This motivates the need for a unified interpretability framework capable of revealing shared mechanisms across diverse graph reasoning tasks. Concurrently, the circuit-tracing interpretation framework [6] has been successfully applied to various language tasks . This framework analyzes how information flows through a Transformer by identifying the specific neurons and layer interactions that contribute to a model's predictions, uncovering latent reasoning structures that contribute to model performance. Due to the inherent irregularity of language tokens, examining the relationship between implicit structures and explicit structural representations offers valuable insight into how transformers process structured information.

Building on these insights, we apply the circuit-tracing interpretation framework to graph reasoning tasks to investigate whether decoder-only Transformers exhibit consistent interpretability across such tasks. Specifically, we conduct case studies using circuit-tracer analyses to examine how these models

*Corresponding: guokai1@msu.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Table 1: Examples of graph samples in textual format across different tasks

	Path Reasoning	Attributed Graph Reasoning	Substructure Extraction
Example			
Graph<EL>	Node index list: 1 2|2 3|3 1|2 4	Node index with attributes: 1A 1B|1B 2A|2A 2B	Node index list: 1 2|2 3|3 1|2 4
Prompt	Start and end nodes: S 1, E 4	Start and end nodes: S 1B, E 2B	Substructure (Triangle): T
Answer	Nodes in the shortest path: 1 2 4	Nodes in the shortest path: 1B 2A 2B	Nodes in substructure: 1 2 3
Description	Given a graph and target nodes, predict the path	Given an attributed graph and target nodes, predict the path	Given a graph and a substructure symbol, predict all matching substructures

reason over explicit structures in tasks such as path reasoning, pattern extraction, and attributed graph reasoning. Through visualization-based analyses, we identify layer-wise **token merging** and **structure memorization** as the primary mechanisms underlying the models' reasoning behavior. Specifically, token merging indicates that Transformers progressively combine tokens to construct substructures relevant to the prediction task. In parallel, structure memorization reveals that the model's predictions also rely on patterns learned from the training data, suggesting a form of retrieval or recall from previously seen structures. To further support this, we perform quantitative analyses to measure the prevalence of these phenomena across different graph reasoning tasks, considering variations in graph density and the hidden dimension size of the underlying Transformer backbone. In summary, our contributions are as follows:

1. We apply circuit-tracing methods to graph reasoning tasks to obtain a unified interpretation of how and why decoder-only Transformers are able to solve them.

2. We provide both visualization and statistical analyses to interpret the underlying mechanisms of token merging and structural memorization for decoder-only transformers in graph reasoning tasks.

3. We further analyze the effects of graph density on token merging and the influence of Transformer hidden size on structural memorization.

2 Experiments

2.1 Experiment setting

First, following previous work [4, 5], we generate synthetic graph datasets to train the transformers. Specifically, we construct a graph G with $|N|$ nodes. Then, we sample the subgraphs G' from the generated graph to construct the training samples, each containing $|N'| < |N|$ nodes. We focus on three fundamental graph reasoning tasks: path reasoning, attributed graph reasoning, and pattern extraction. For each training sample, the input is formatted as a sequence of the form "<Graph(EL)><Question><Answer>" following the definition in [5]where the question and answer components vary depending on the task. Examples of task-specific prompts are provided in Table 1. We adopt the GPT-2 model and train transcoders across multiple layers according to [6]. The data and training details are in Appendix A.

2.2 Visualized tracers in graph reasoning tasks

To reveal the internal reasoning structures of Transformers, we first visualize circuit traces across three representative graph reasoning tasks—**path reasoning**, **attributed graph reasoning**, and **pattern extraction**, as shown in Figure 1, Figure 2, and Appendix C Figure 9, respectively.

The visualizations reveal token merging as a core mechanism in the reasoning process across various graph tasks. For example, in the path reasoning task (Figure 1), given a graph and a query specifying start node 4 and end node 7, the expected path is $4 \rightarrow 5 \rightarrow 7$. The model is tasked with predicting the next token node 5. Circuit tracing shows that the Transformer merges the edges (4, 5) and (5, 7), enabling it to correctly identify node 5 as the next step toward the target. Similarly, in the attributed path reasoning task shown in Figure 2, we observe that the relevant edges are also merged, with their associated attributes included in the merged representations. For instance, the target edge (5C, 8C) is clearly highlighted, indicating its role in guiding the model's prediction. In the pattern extraction task shown in Appendix C Figure 9, token merging is also observed at higher layers. For example, when

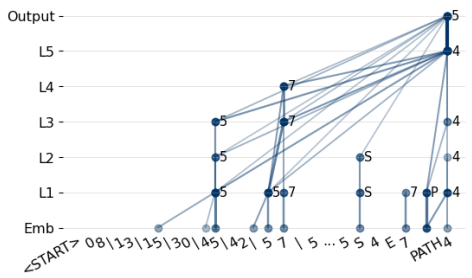


Figure 1: Circuit tracer in the path reasoning task. "L" denotes the layers. The predicted path is $4 \rightarrow 5 \rightarrow 7$, with the model currently predicting token 5.

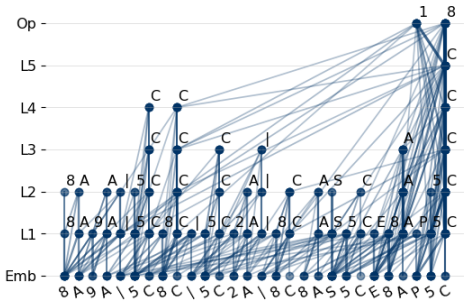


Figure 2: Circuit tracer in the attributed graph reasoning task. The predicted path is $5C \rightarrow 8C \rightarrow 8A$, with the current token being 8 in the token 8C.

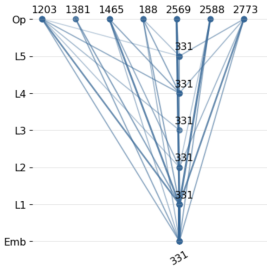


Figure 3: Circuit tracer reveals structural memorization: different layers store information about the 1-hop neighbors of node 331.

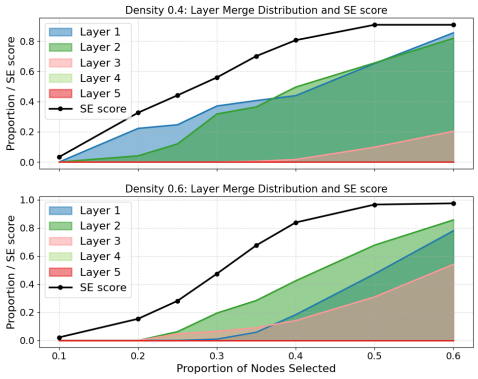


Figure 4: Edge gathering in path reasoning task. The evaluations are on graphs with 0.4 and 0.6, respectively.

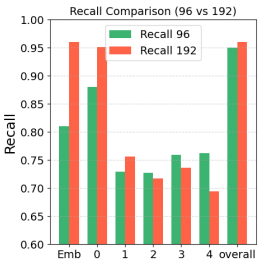


Figure 5: The Recall of structure memorization under different hidden dimensions

the predicted pattern is (27, 23, 40), the tokens corresponding to nodes 23 and 40 are merged into token 27, a process that becomes more prominent in the upper layers of the model.

In addition to token merging, we observe that Transformers progressively memorize structural information during training them to solve path reasoning tasks, as detailed in Appendix B. After training on path reasoning tasks, we prompt the Transformer with a single node ID and assess whether it can predict the next token, modeling the probability of its 1-hop neighbors. To further interpret this memorization, we analyze the circuit tracer results and examine how multiple layers contribute to preserving neighborhood information, as visualized in Figure 10. In the large-graph path reasoning setting, Transformers are capable of recalling neighbor nodes when the prompt is a central node. Importantly, different layers contribute differently to this memorization. For example, node 1381 is already recalled in the embedding and first layers.while node 1203 requires additional support from every layer. This layer-wise contribution reflects how structural information is distributed and retrieved during inference.

2.3 Quantified evaluations

From the visualizations, we observe that the circuit tracers reveal the reasoning process on a case-by-case basis. In this section, we present quantitative evaluations to demonstrate that token merging and structure memorization are consistent behaviors at the dataset level. Furthermore, we analyze how these two mechanisms are affected by varying graph densities and model sizes.

Token Merging The token merging mechanism consistently emerges across various graph reasoning tasks, serving to summarize substructures relevant to the model's predictions, such as edge pairs in path reasoning and candidate patterns in substructure extraction tasks. To evaluate this behavior, we

measure the alignment between the selected and expected tokens using the metric $\mathrm{S_E} = \frac{N_{\text{select}}}{N_{\text{pred}}}$, where N_{pred} denotes the number of the tokens are extracted by the circuit tracer, and N_{select} represents the number of the tokens that exactly can provide the evidences for the predictions. For example, these include triangle patterns in pattern extraction, attributed nodes and edges merged in attributed graph reasoning, and edges gathered in the path reasoning task. The corresponding results are shown in Appendix C Appendix C Figure 8, and Table 3, Appendix C Figure 4, respectively, where the layers at which token merging occurs vary across tasks and token types.

Appendix C Figure 8 illustrates the pattern extraction task. We observe that token merging occurs progressively across layers, with multiple merging operations taking place in the shallow layers. During this stage, the number of included nodes increases significantly, but the SE score remains stable as the selected node ratio increases from 0.7 to 0.9.In attributed graph reasoning, shown in Appendix C Table 3, tokens corresponding to start nodes, end nodes, and edge attributes are typically merged around layer 3. Meanwhile, path reasoning tasks in Figure 4 reveal a density-dependent pattern: for graphs with low degree (e.g., density 0.4), many relevant edges are detected in early layers (layer 1 or 2), whereas in denser graphs (e.g., density 0.6), the model relies more heavily on deeper processing, particularly at layer 3. In conclusion, token merging is consistently observed across various tasks, and as graph density increases, the merging tends to occur at higher layers of the Transformer.

Structure Memorization The structure memorization mechanism suggests that next-token predictions are influenced by structural patterns learned from the training data. We observe that Transformers are capable of memorizing graph structures, as evidenced by their ability to recall potential neighbor nodes during inference. Therefore, we apply Precision and Recall to discuss whether the transformers memorize the correct neighbors and cover all of the neighbors, respectively. Notably, the memorization spans multiple layers, indicating that structural information is preserved and propagated throughout the model rather than being localized to a specific layer. To evaluate this behavior, we assess whether the neighbors predicted by the model correspond to those in the original graph. Given that hidden size is often a critical factor in neural network memorization capacity, we compare models with different hidden dimensions, specifically 96 and 192, as shown in Appendix C Figure 10 and Figure 5. In both settings, the precision is consistently high, indicating that Transformers can accurately memorize 1-hop neighbors across multiple layers. However, the recall patterns differ. Although the overall recall scores are similar, the distribution of memorized neighbors across layers varies. In models with smaller hidden dimensions, memorization tends to be more evenly distributed across layers. In contrast, models with larger hidden dimensions exhibit strong memorization even in shallow layers; notably, the embedding layer alone is capable of capturing all 1-hop neighbors. This suggests that higher-dimensional embeddings are sufficient to encode local structural information without requiring deeper processing. In summary, Transformers are capable of memorizing graph structures, and larger hidden sizes encourage this neighbor memorization to be concentrated in the lower layers of the model.

3 Related work and Background

Mechanism of graph structure understanding Recent studies suggest that large language models (LLMs) possess the capability to understand graph structures [1, 7]. To investigate the underlying reasons for this ability, current research often focuses on simplified graph data and decoder-only transformers, aiming to uncover the mechanisms behind such capabilities [8, 9]. For instance, [4, 3] suggest that LLMs perform path reasoning by effectively searching for relevant edges, while [5] argues that LLMs can extract substructures from the input. Despite these varying perspectives, it remains unclear whether there exists a unified framework for understanding the mechanisms by which LLMs process graph structures.

Interpretation of LLMs Understanding the internal mechanisms of Transformer models has long been a focus of interpretability research [10, 11]. Early interpretability studies on BERT revealed that different layers capture distinct linguistic properties: lower layers attend to local syntax, while higher layers progressively aggregate global semantics [12, 13]. Probing classifiers and attention-based analyses further demonstrated how information is organized hierarchically across depth [14]. In decoder-only Transformers, this line of work evolved into circuit tracing methods such as Transcoder [15, 16], which recover token-level causal paths to explain autoregressive reasoning.

Building on this, recent approaches introduce attributed graphs to represent token interactions across layers, capturing both semantic roles and attention dynamics [6]. Our work extends this direction by constructing attributed graphs from circuit traces to interpret graph reasoning tasks, revealing token merging and structure memorization dynamics in graph reasoning tasks.

4 Conclusion

We provide a unified perspective on how decoder-only Transformers solve graph reasoning tasks, revealing token merging and structure memorization as core mechanisms. These behaviors vary with task complexity and model scale, offering a compact framework for interpreting structural reasoning in Transformers.

References

[1] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861, 2023.

[2] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

[3] Abulhair Saparov, Srushti Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Seyed Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. In *The Thirteenth International Conference on Learning Representations*, 2025.

[4] Siwei Wang, Yifei Shen, Shi Feng, Haoran Sun, Shang-Hua Teng, and Wei Chen. Alpine: Unveiling the planning capability of autoregressive learning in language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5] Xinnan Dai, Kai Yang, Jay Revolinsky, Kai Guo, Aoran Wang, Bohang Zhang, and Jiliang Tang. From sequence to structure: Uncovering substructure reasoning in transformers. *arXiv preprint arXiv:2507.10435*, 2025.

[6] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025.

[7] Xinnan Dai, Haohao Qu, Yifei Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37:78320–78370, 2024.

[9] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *International Conference on Machine Learning*, pages 2296–2318. PMLR, 2024.

[10] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, 2019.

[11] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.

[12] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 2019.

[13] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, 2019.

[14] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[15] Anthropic. Transformerlens transcoder documentation. `https://github.com/anthropics/transformerlens`, 2023.

[16] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

A Appendices in Experiments

In this section, we provide additional details of our experimental setup. The graph datasets vary across tasks, and for each task, the graph configurations, Transformer architectures, and transcoder training procedures are customized accordingly. We adopt the GPT-2 model with ROPE positional embeddings as the backbone. While transformers are capable of achieving high accuracy, we employ a lightweight variant of the GPT-2 architecture consisting of only 5 layers. After training the transformers on the specified tasks, we train cross-layer transcoders at each layer. These transcoders are then merged to construct an attribution graph according to [6], which is used to reveal the implicit internal structures of the transformer for interpretability. Additionally, we report the parameters used in our visualizations. The summary of these experimental settings is presented in Table 2. All of the experiments are run on a single RTX A6000.

B Structure memorization in training

We find that Transformers tend to memorize structural patterns during pretraining. To investigate this, we construct training data by extracting subgraphs from backbone graphs with varying densities (0.2 and 0.4). Each sampled subgraph contains at most 10 nodes, and we focus on path reasoning tasks to evaluate whether Transformers can memorize edge combinations during training. We evaluate accuracy on the test set using three criteria: 1. Local Accuracy: The predicted path's edges are fully contained within the given subgraph; 2. Exist Accuracy: The predicted path's edges exist somewhere in the original backbone graph; 3. Global Accuracy: Given only a start node and an end node, the model must predict a correct path in the backbone graph without explicit subgraph context. The maximum path length is also recorded to assess the length of the predicted path.

First, we evaluate the case where the backbone graph contains 10 nodes, and we randomly drop 40% or 60% of its edges. We measure both Global Accuracy and Local Accuracy during training, shown in Figure 6. We observe that Transformers initially memorize the full backbone graph, reaching 100% Global Accuracy at early training stages. In contrast, Local Accuracy improves more gradually, suggesting that Transformers first memorize the global structure before adapting to the local subgraphs. Additionally, a lower edge dropout ratio leads to faster learning, indicating that denser graphs facilitate structural memorization.

Next, we increase the backbone graph size to 50 nodes and evaluate the setting where subgraphs are sampled by selecting nodes only—without dropping any edges among the selected nodes. The results are presented in Figure 7. As the graph size increases, memorizing the full backbone structure becomes more challenging, and we observe that Global Accuracy improves more slowly compared to the 10-node case, while Local Accuracy increases more quickly. Interestingly, Exist Accuracy exceeds Local Accuracy, suggesting that the model continues to rely on memorized global edges even in local contexts. Moreover, despite training only on short paths (3–5 hops), the model is capable of predicting accurate paths exceeding 10 hops, implying that Transformers may prioritize

Table 2: Summarization of Experiment settings

Module	Parameter	Path reasoning Tiny	Path reasoning Large	Attributed reasoning	Substructure extraction
Graph	Density	0.4	0.002	0.2	0.1
	Node num	50	3000	30	100
	Max node number in subgraph	10	10	5	5
Transformer	Hidden size	96	96	96	96
	Max length	256	96	96	96
	Basic Acc	0.99	0.96	0.94	0.92
Transcoder	L1 coefficent	0.0005	0.0005	0.0005	0.0005
	Dead nueron num	50	50	50	50
	Hidden size	192	192	192	192
Vis parameter	Node threshold	0.86	0.8	0.8	0.8
	Edge ratio	0.48	0.9	0.99	0.9
	Edge threshold	0.1	0.2	0.1	0.4

high-probability paths rather than strictly the shortest ones—even when trained under shortest-path supervision.

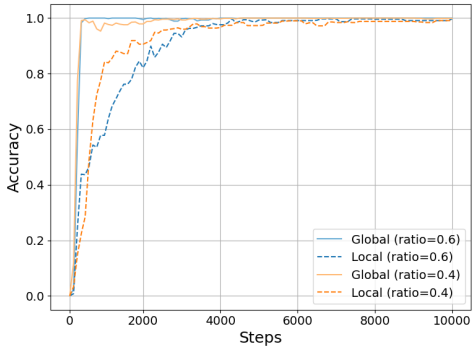


Figure 6: Local Acc and Global Acc change with training steps

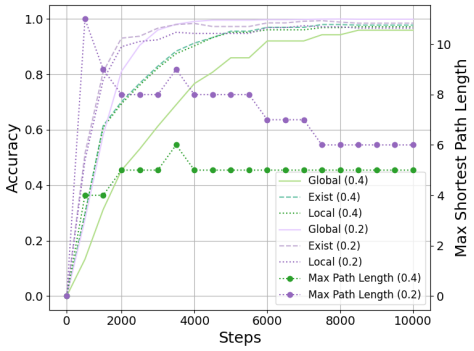


Figure 7: Local Acc, Exist Acc, Global Acc, and Max Path Length change with training steps

C Additional Figures

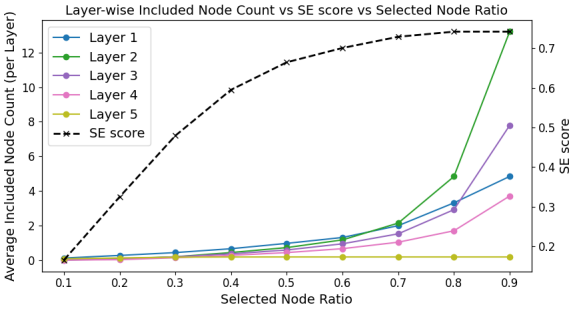


Figure 8: Pattern learned across layers. The counting demonstrates the number of tokens merged across the layers. The token merging never shows in layer 4 and layer Output.

Table 3: Token merged in attributed graph reasoning. S_E is 0.94 with the selected node ratio of 0.9. Start, End, and Edge denote attributes merged from the start node, end node, and full edge, respectively.

	Start	**End**	**Edge**
L1	0	6	3
L2	39	40	39
L3	225	218	222
L4-Out	0	0	0
Overall	264	264	264

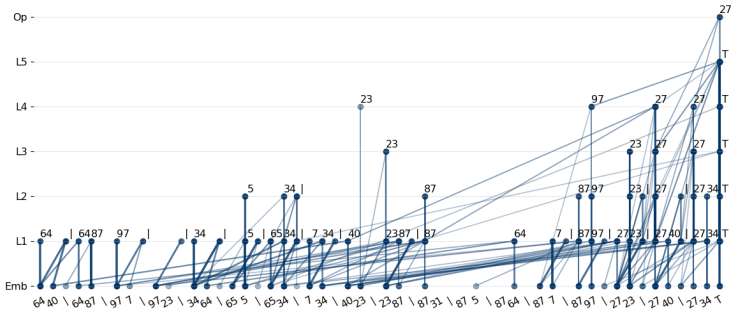


Figure 9: Circuit tracer in the pattern extraction task. The predicted patterns are (27, 23, 40) and (97, 87, 7).

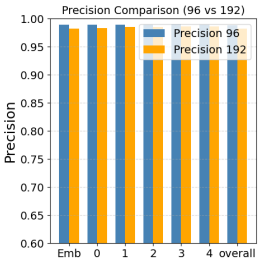


Figure 10: Precision under different hidden dimensions

D Limitation

Our analysis is limited to decoder-only Transformers trained from scratch, and a gap remains between these models and real-world large language models. Additionally, a more thorough investigation into the effects and limitations of circuit tracing should be included in future work.

E Boarder Impact

We uncover the core mechanisms behind how decoder-only Transformers perform graph reasoning. Given that graph reasoning is a specialized form of structured reasoning, these insights may extend to general reasoning tasks in large language models