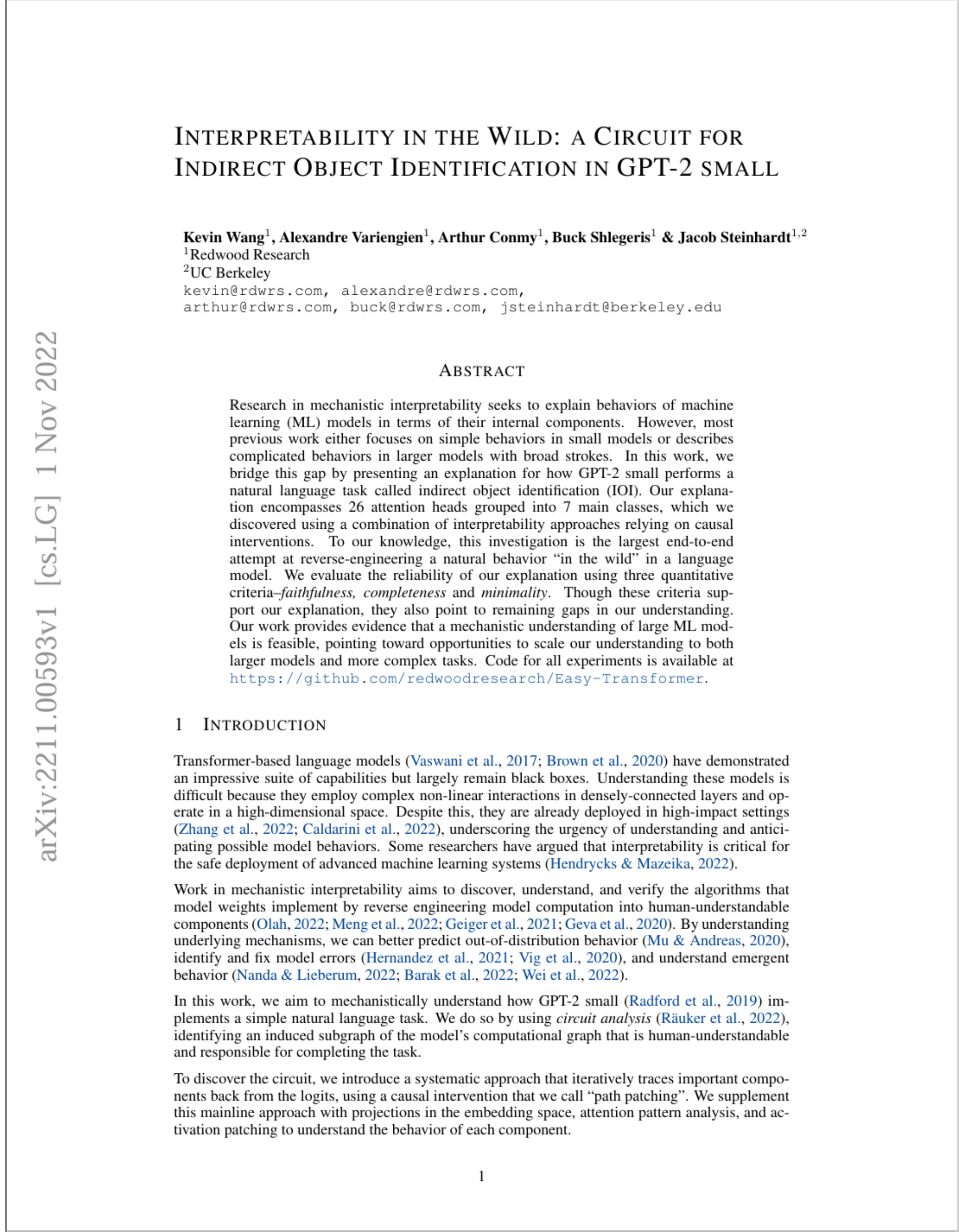
Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small

March 19, 2024

Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small

Kevin Wang et al.

https://arxiv.org/abs/2211.00593



Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small

Kevin Wang[1], **Alexandre Variengien**[1], **Arthur Conmy**[1], **Buck Shlegeris**[1] **& Jacob Steinhardt**[1,2]
[1]Redwood Research
[2]UC Berkeley
`kevin@rdwrs.com, alexandre@rdwrs.com,`
`arthur@rdwrs.com, buck@rdwrs.com, jsteinhardt@berkeley.edu`

Abstract

Research in mechanistic interpretability seeks to explain behaviors of machine learning (ML) models in terms of their internal components. However, most previous work either focuses on simple behaviors in small models or describes complicated behaviors in larger models with broad strokes. In this work, we bridge this gap by presenting an explanation for how GPT-2 small performs a natural language task called indirect object identification (IOI). Our explanation encompasses 26 attention heads grouped into 7 main classes, which we discovered using a combination of interpretability approaches relying on causal interventions. To our knowledge, this investigation is the largest end-to-end attempt at reverse-engineering a natural behavior "in the wild" in a language model. We evaluate the reliability of our explanation using three quantitative criteria–*faithfulness, completeness* and *minimality*. Though these criteria support our explanation, they also point to remaining gaps in our understanding. Our work provides evidence that a mechanistic understanding of large ML models is feasible, pointing toward opportunities to scale our understanding to both larger models and more complex tasks. Code for all experiments is available at `https://github.com/redwoodresearch/Easy-Transformer`.

1 Introduction

Transformer-based language models (Vaswani et al., 2017; Brown et al., 2020) have demonstrated an impressive suite of capabilities but largely remain black boxes. Understanding these models is difficult because they employ complex non-linear interactions in densely-connected layers and operate in a high-dimensional space. Despite this, they are already deployed in high-impact settings (Zhang et al., 2022; Caldarini et al., 2022), underscoring the urgency of understanding and anticipating possible model behaviors. Some researchers have argued that interpretability is critical for the safe deployment of advanced machine learning systems (Hendrycks & Mazeika, 2022).

Work in mechanistic interpretability aims to discover, understand, and verify the algorithms that model weights implement by reverse engineering model computation into human-understandable components (Olah, 2022; Meng et al., 2022; Geiger et al., 2021; Geva et al., 2020). By understanding underlying mechanisms, we can better predict out-of-distribution behavior (Mu & Andreas, 2020), identify and fix model errors (Hernandez et al., 2021; Vig et al., 2020), and understand emergent behavior (Nanda & Lieberum, 2022; Barak et al., 2022; Wei et al., 2022).

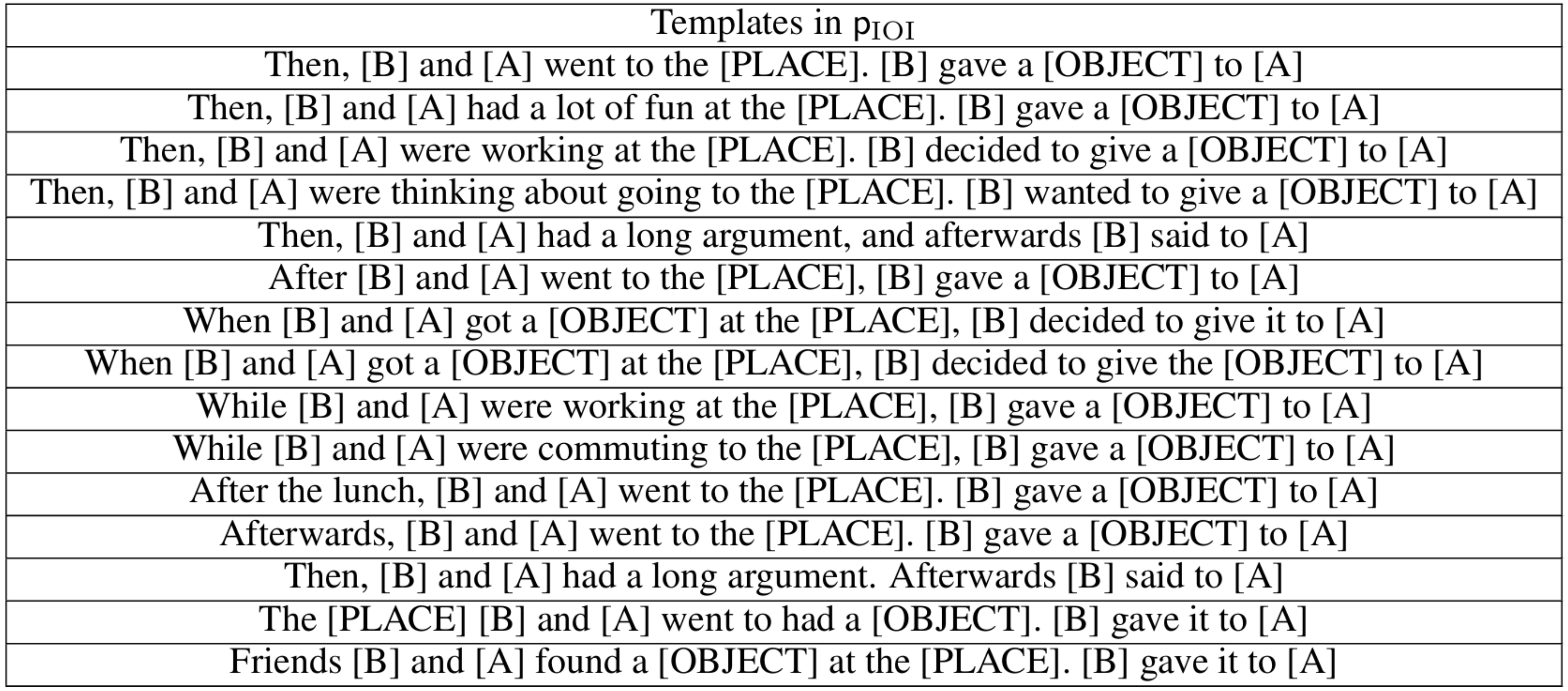
In this work, we aim to mechanistically understand how GPT-2 small (Radford et al., 2019) implements a simple natural language task. We do so by using *circuit analysis* (Räuker et al., 2022), identifying an induced subgraph of the model's computational graph that is human-understandable and responsible for completing the task.

To discover the circuit, we introduce a systematic approach that iteratively traces important components back from the logits, using a causal intervention that we call "path patching". We supplement this mainline approach with projections in the embedding space, attention pattern analysis, and activation patching to understand the behavior of each component.

IOI overview/outline [1]

- Studied something more complex than a toy task, using GPT-2 small
- Used a variety of techniques, most notably a causal intervention they named *path patching*
- Good paper, solid evidence, but note that mechanistic interpretability techniques do not produce 100% complete or 100% verified understanding
- Task involves predicting next token when there are two main subjects, one of them has already been repeated as the subject of the clause, and we are about to output the indirect object.
    - “When Mary and John went to the store, John gave a drink to ______”

IOI templates



Templates in p_{IOI}
Then, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] had a lot of fun at the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] were working at the [PLACE]. [B] decided to give a [OBJECT] to [A]
Then, [B] and [A] were thinking about going to the [PLACE]. [B] wanted to give a [OBJECT] to [A]
Then, [B] and [A] had a long argument, and afterwards [B] said to [A]
After [B] and [A] went to the [PLACE], [B] gave a [OBJECT] to [A]
When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give it to [A]
When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give the [OBJECT] to [A]
While [B] and [A] were working at the [PLACE], [B] gave a [OBJECT] to [A]
While [B] and [A] were commuting to the [PLACE], [B] gave a [OBJECT] to [A]
After the lunch, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Afterwards, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] had a long argument. Afterwards [B] said to [A]
The [PLACE] [B] and [A] went to had a [OBJECT]. [B] gave it to [A]
Friends [B] and [A] found a [OBJECT] at the [PLACE]. [B] gave it to [A]

Figure 14: Templates used in the IOI dataset. All templates in the table fit the 'BABA' pattern, but we use templates that fit the 'ABBA' pattern as well (i.e, by swapping the first instances of [B] and [A] in all of the above).

IOI overview/outline [2]

- One of the key techniques used is *path patching*
  - This is like activation patching, but further limits the downstream effects
- In activation patching, you:
  - Pick one component (e.g. in layer 3, attention head 5)
  - Replace it’s output with the output you’ve saved from different input text
  - Let the model continue computation from there.
  - This will compute **both** the direct effects of the selected component, **and** any indirect effects caused/mediated by later attention layers
- Path patching (Appendix B) to receivers *R* modifies the third step:
  - Let the model continue computation toward the receivers, **except all intermediate attention layers remain frozen**
  - This will compute **only** the direct effects of the selected component, and **eliminate** any indirect effects caused/mediated by later attention layers
  - Note that they do allow indirect effects from everything else, especially MLPs

IOI overview/outline [3]

- Start by using path patching (with “noise” from p_{ABC}) to find which attention heads directly affect the next token output
  - These attention heads were called *Name Mover Heads*
- Next, use path patching to find which attention heads directly affect the Name Mover Heads (as a group)
  - These attention heads were called *S-Inhibition Heads*
- Then, use path patching to find which attention heads directly affect the S-Inhibition Heads (as a group)
- Repeat walking backwards as needed…

IOI overview/outline [4]

- Ultimately, found 7 types of attention implemented in 26 heads with information flowing like this:

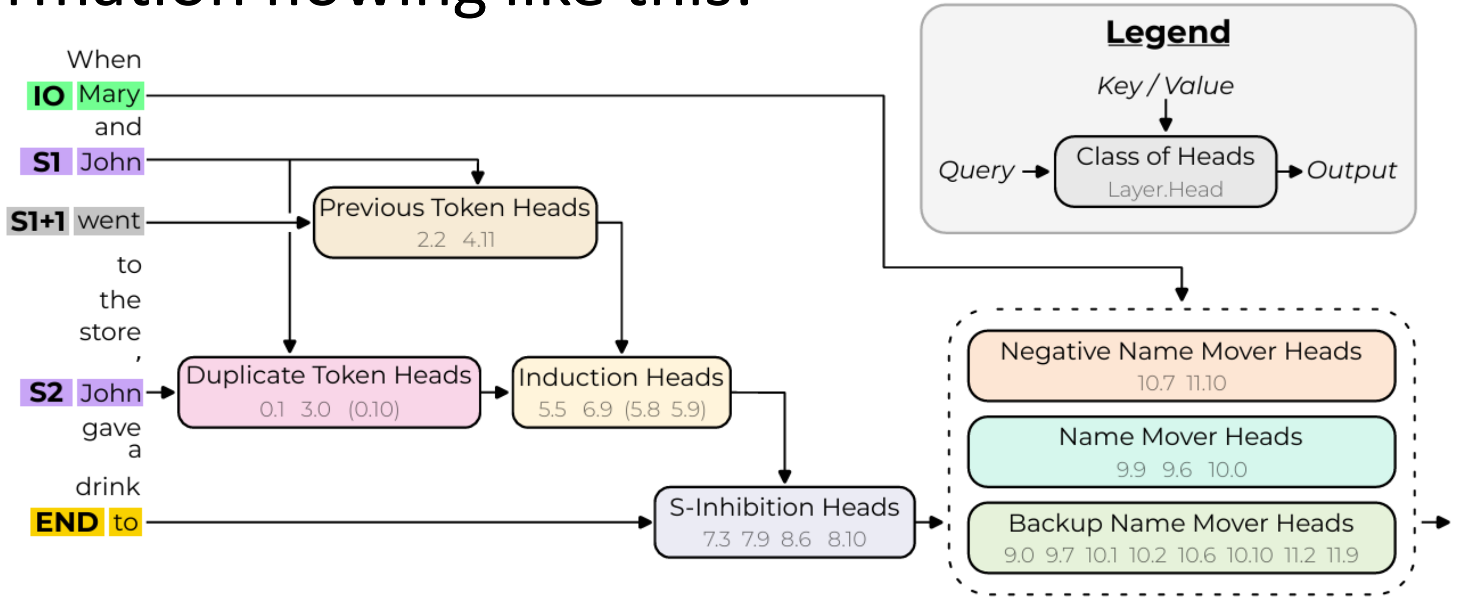
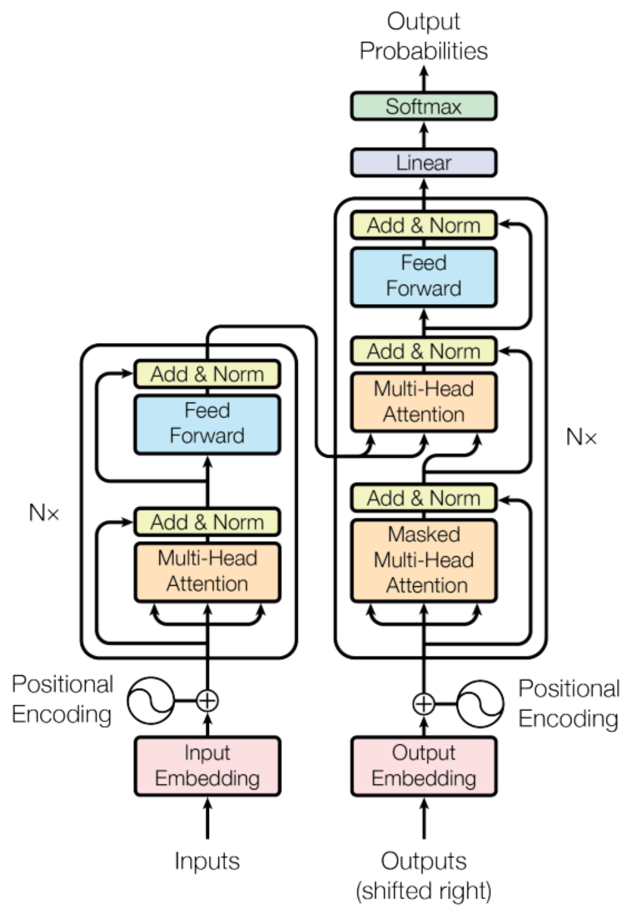


Figure 2: We discover a circuit in GPT-2 small that implements IOI. The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.

IOI & GPT-2 decoder-only LLM data flow

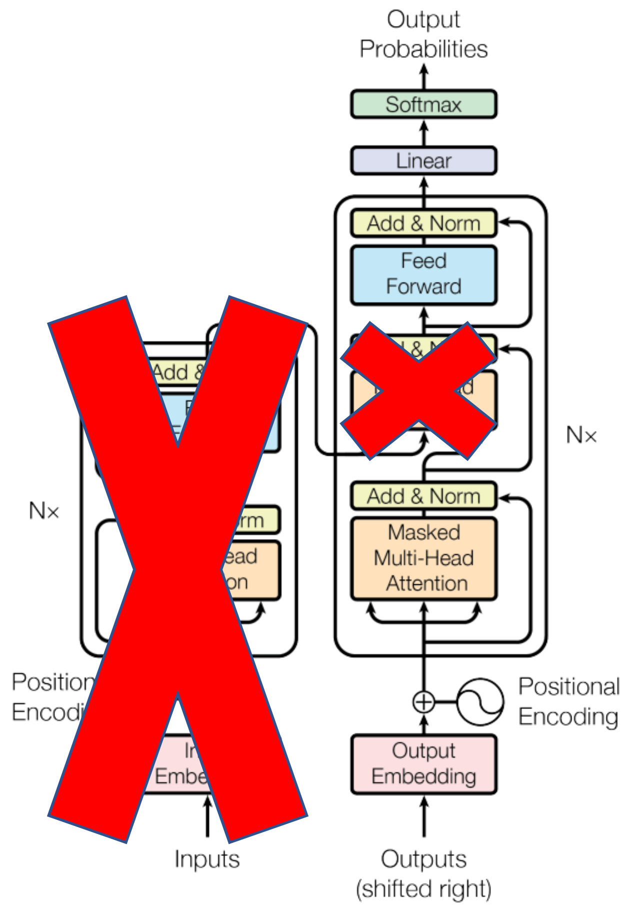
- We try to crystallize understanding of the flow of information across token positions and across layers
- Starting with the original attention diagram, we step-by-step make small modifications until we have a diagram with explicit token positions and layers, eliminating the need for mental abstraction
- To understand circuits and techniques like path patching used to find them, it is critical to understand the architecture and how data flows through the layers and token positions in GPT-2

Original Transformer

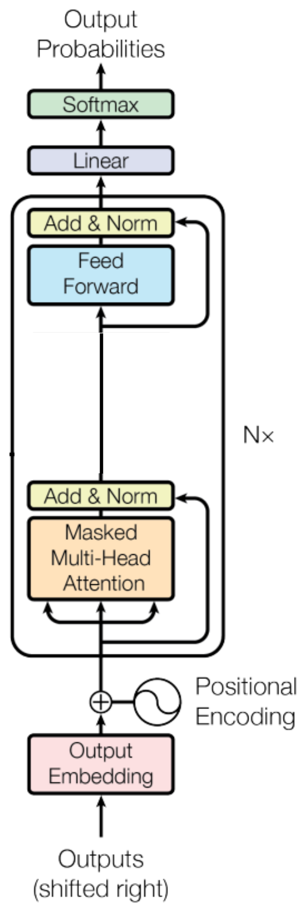


"*Attention Is All You Need,*" Vaswani et al., https://arxiv.org/abs/1706.03762

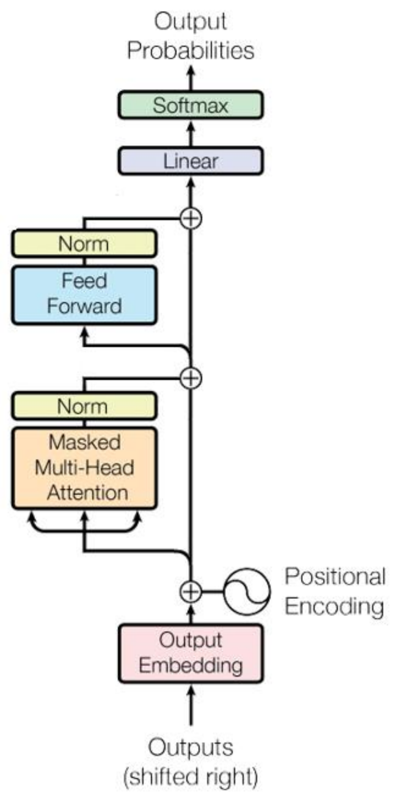
Decoder-only Transformer – 1



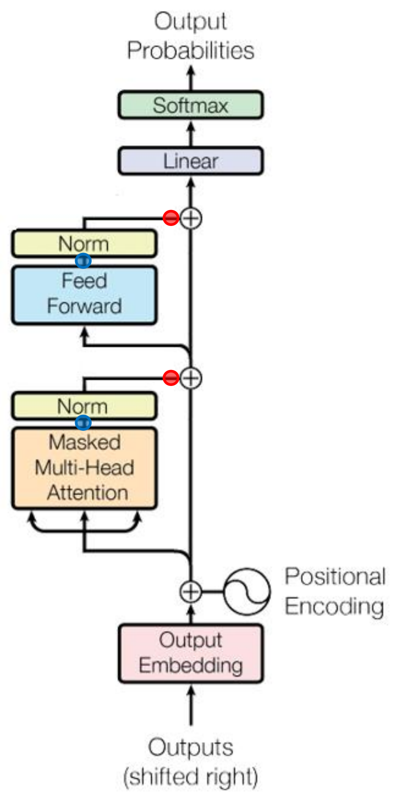
Decoder-only Transformer – 2



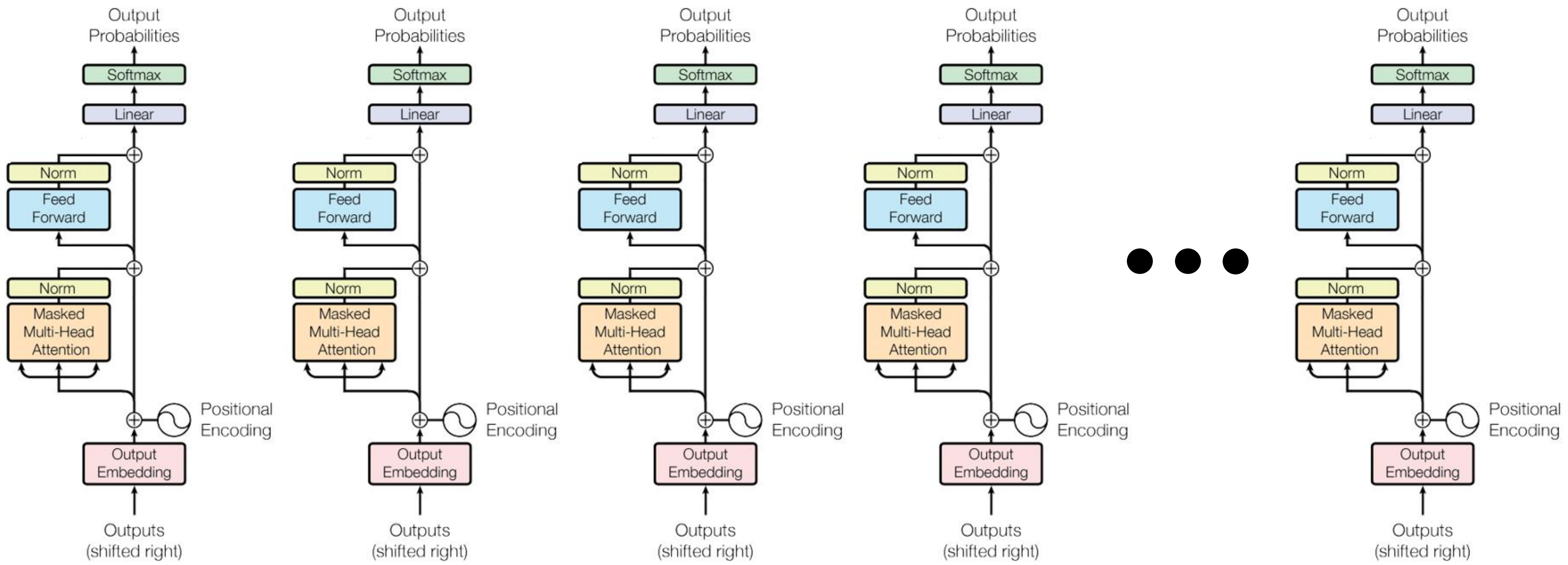
Decoder-only Transformer, Straight Through



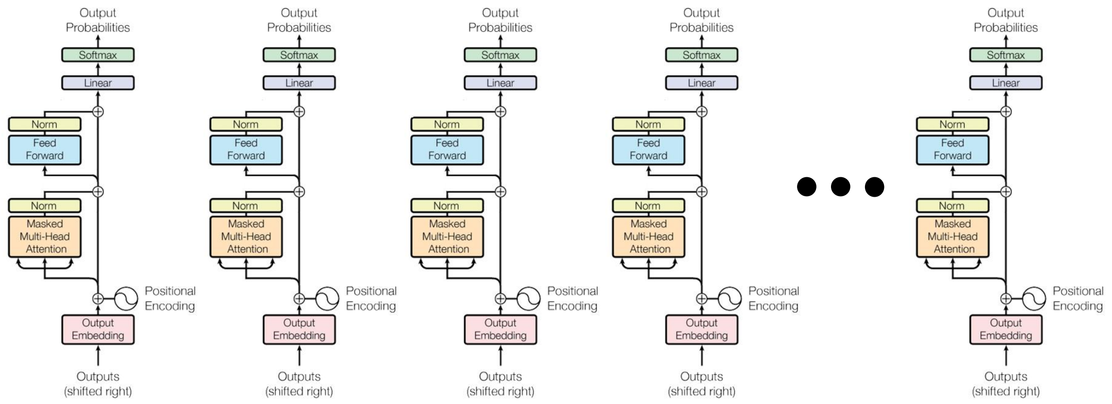
Decoder-only Transformer, with Activations



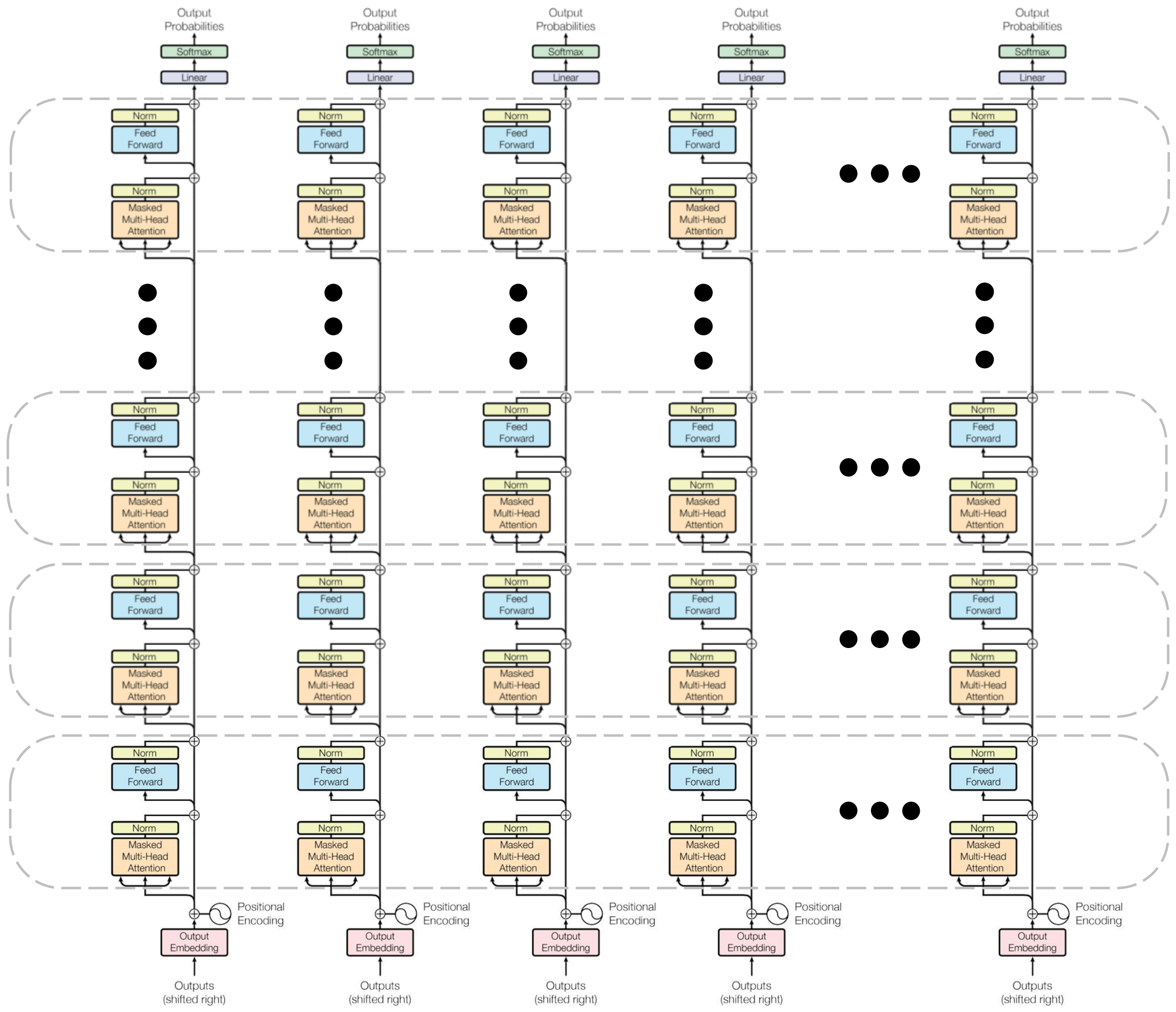
Decoder, Multiple Tokens



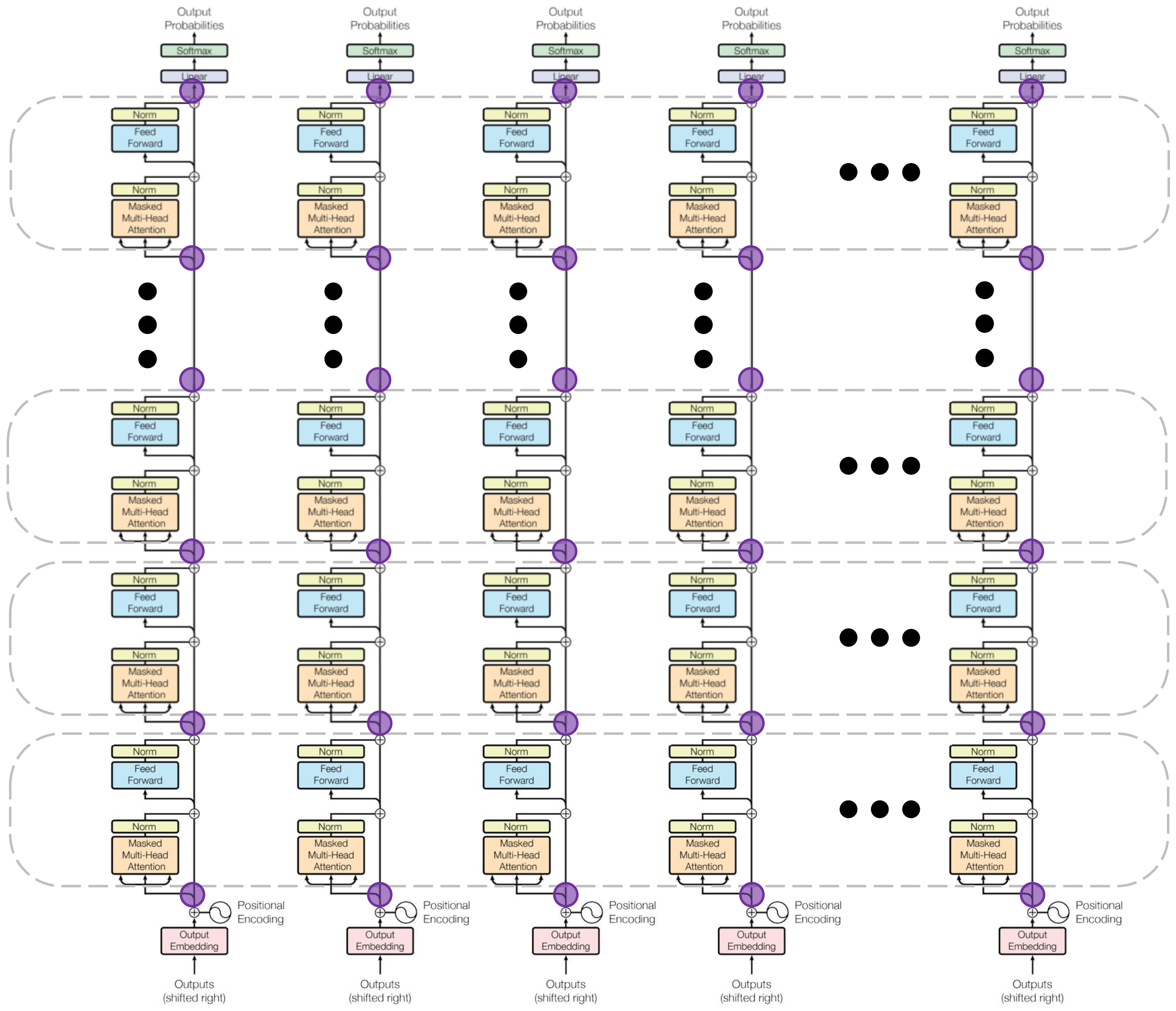
Decoder, Multiple Tokens, Small



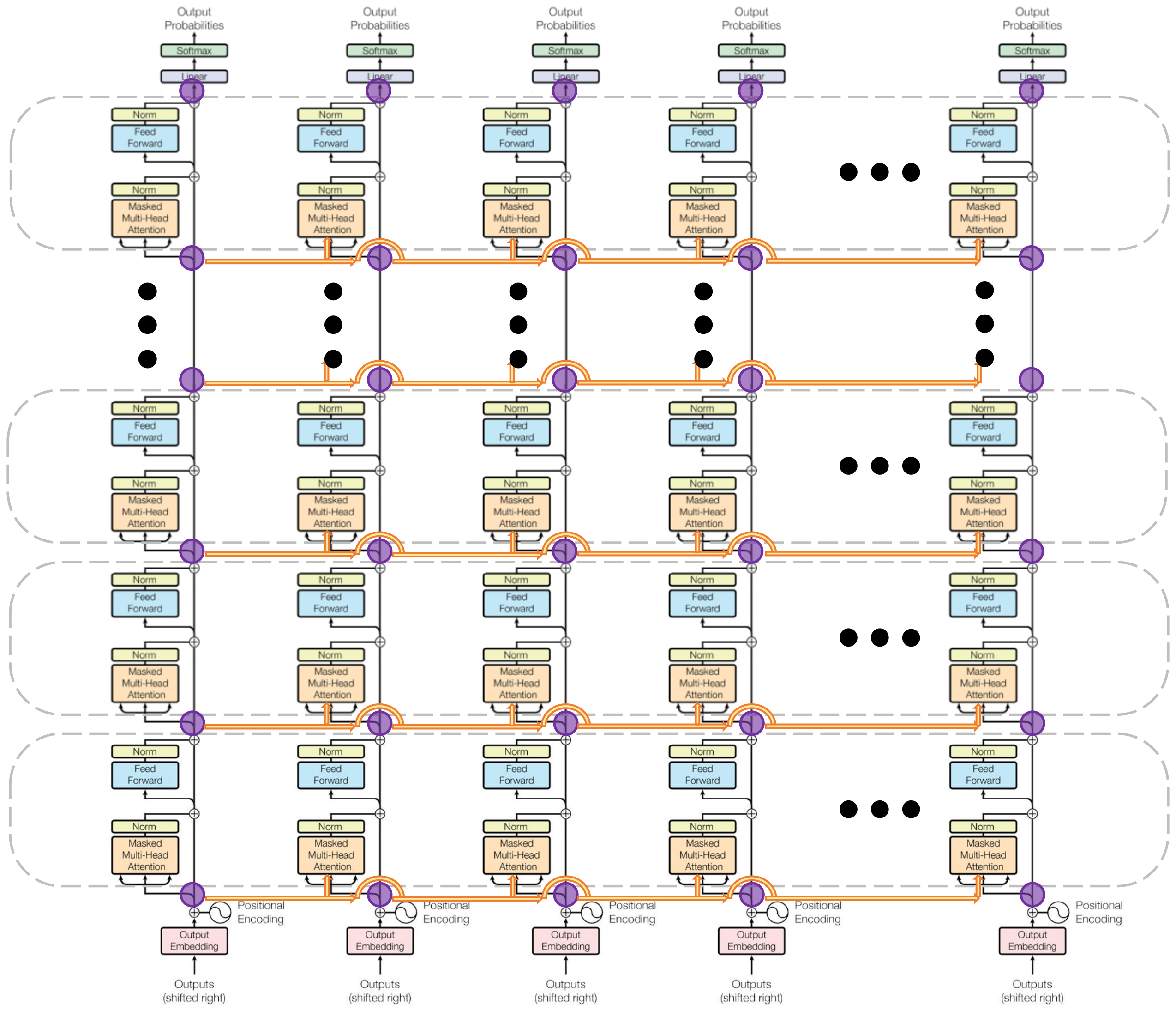
Decoder, Adding Multiple Layers



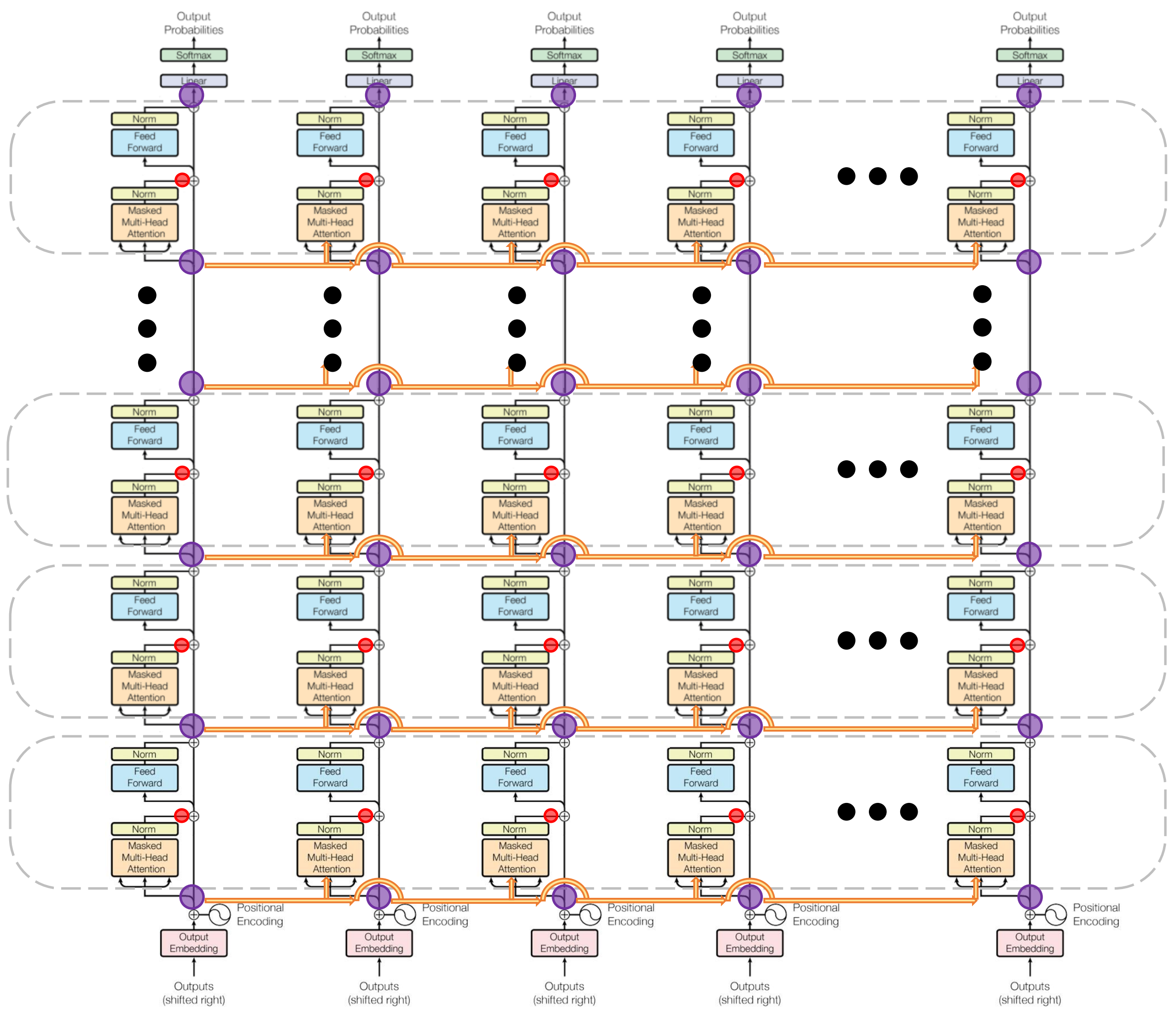
Decoder, Adding Hidden States



Decoder, Adding Attention Paths



Decoder, Adding Attention Activations



IOI figures

- Key figures from the IOI paper on the next few slides

Indirect Object Identification

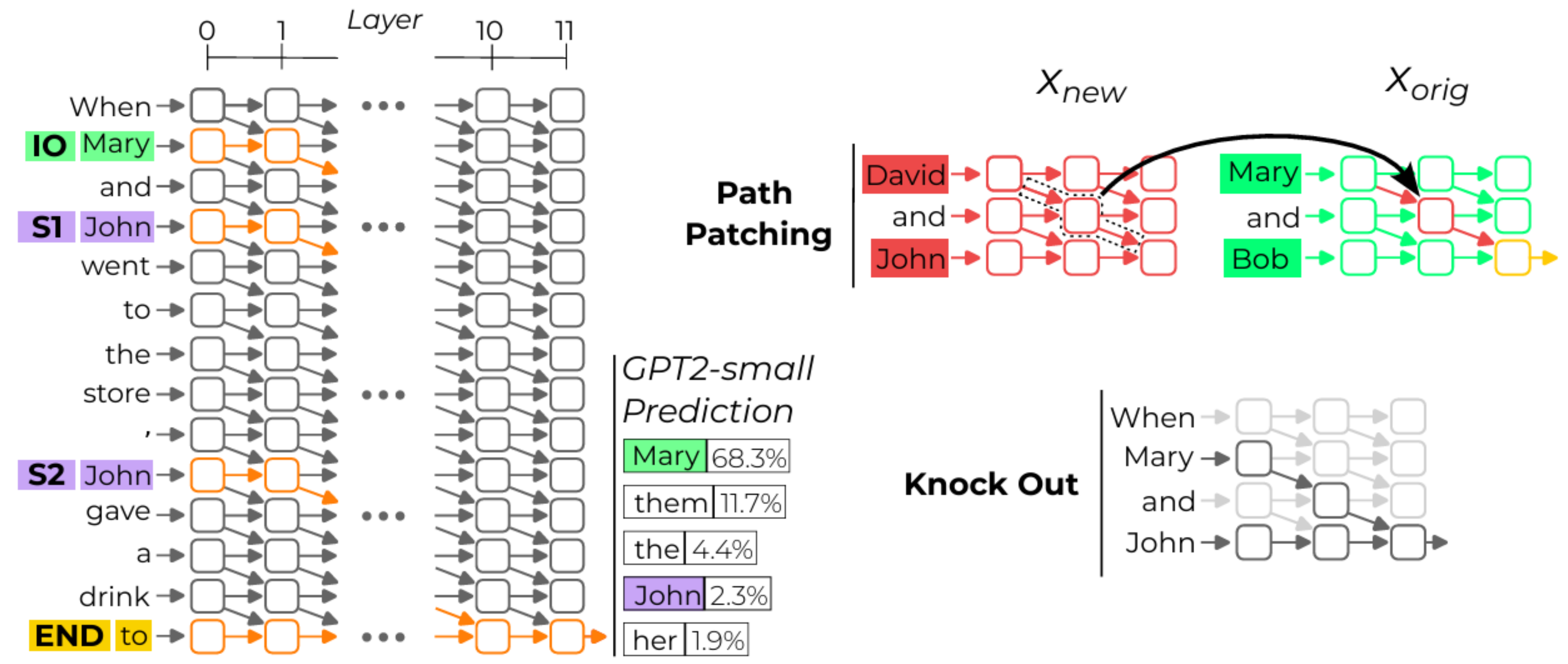


Figure 1: Left: We isolated a *circuit* (in orange) responsible for the flow of information connecting the indirect object "Mary" to the next token prediction. The nodes are attention layers and the edges represent the interactions between these layers. Right: We discovered and validated this circuit using causal interventions, including both path patching and knockouts of attention heads.

Indirect Object Identification

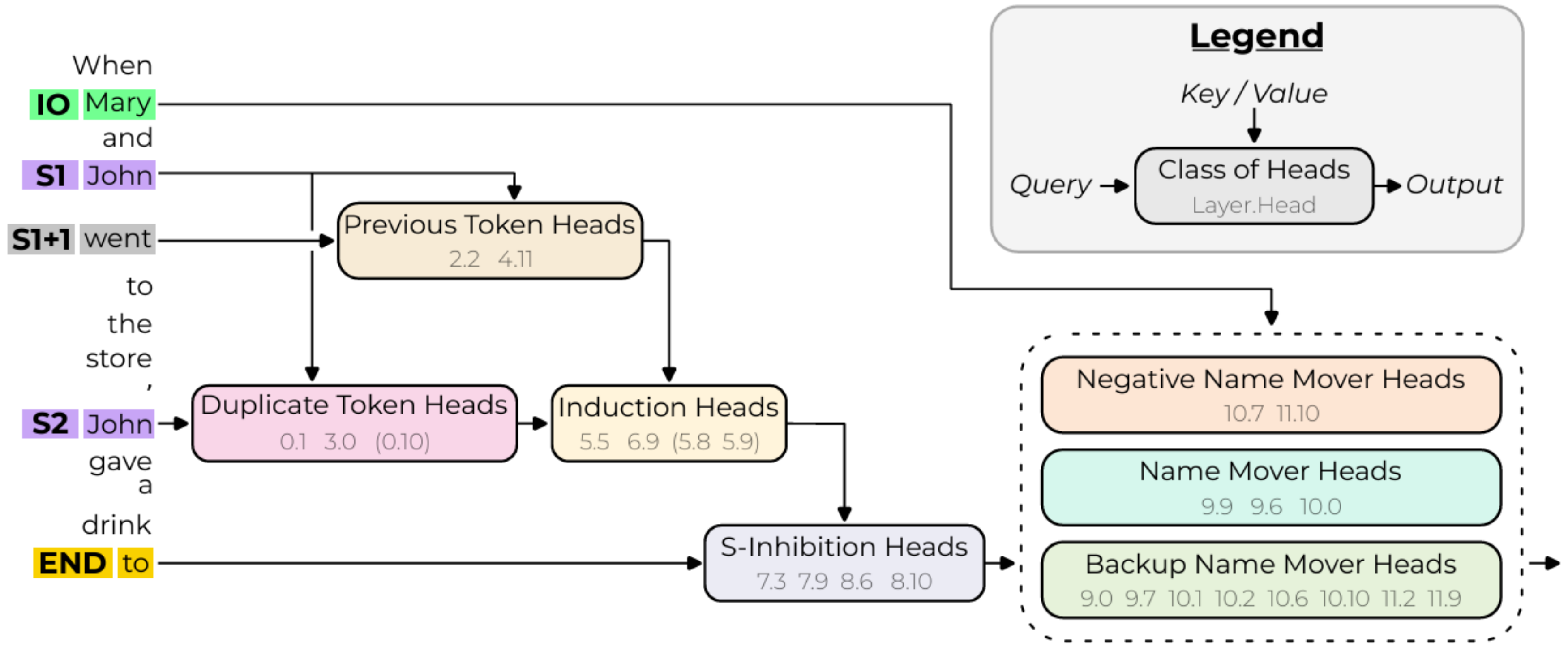


Figure 2: We discover a circuit in GPT-2 small that implements IOI. The input tokens on the left are passed into the residual stream. Attention heads move information between residual streams: the query and output arrows show which residual streams they write to, and the key/value arrows show which residual streams they read from.

Indirect Object Identification

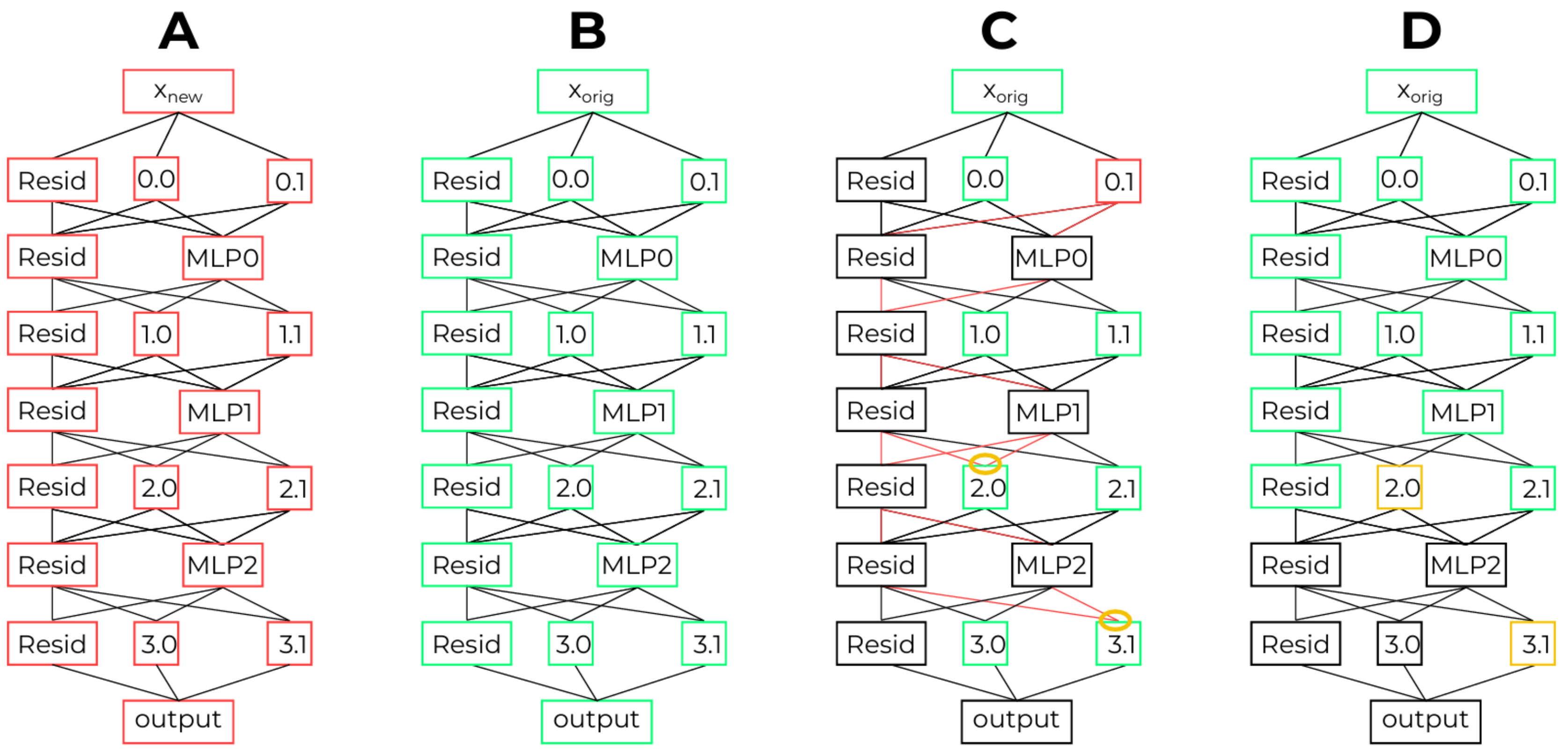
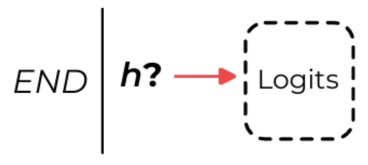
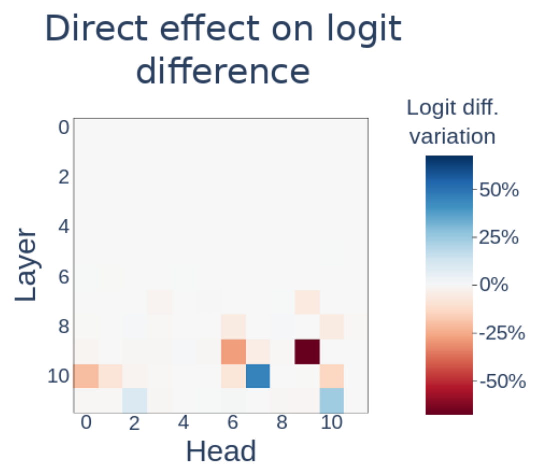


Figure 11: Illustration of the four forward passes in the path patching method. $h = 0.1$ and $R = \{2.0, 3.1\}$. The layer norms are not shown for clarity. In the forward pass C, we show in red all the paths through which h can influence the value of heads in R. Nodes in black are recomputed, nodes in color are patched or frozen to their corresponding values.

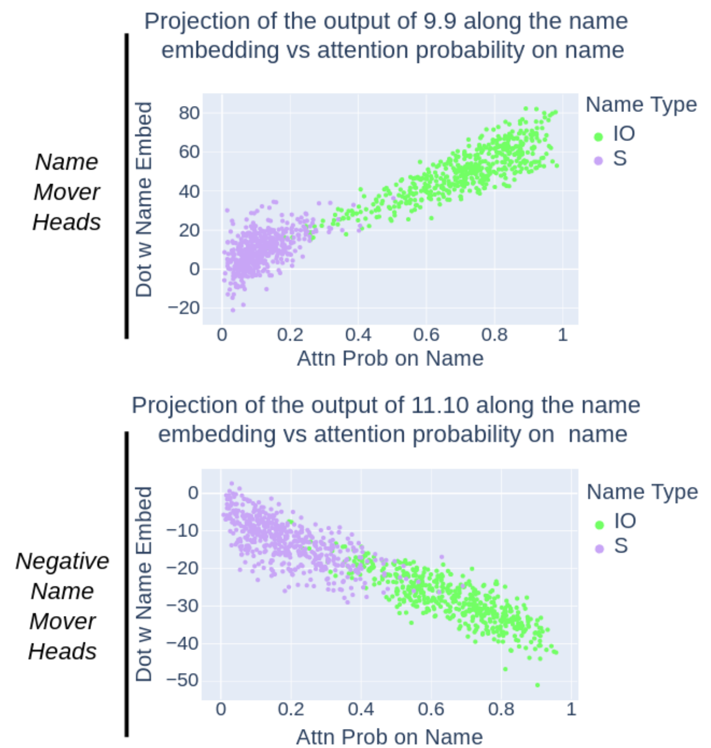
Indirect Object Identification



(a)



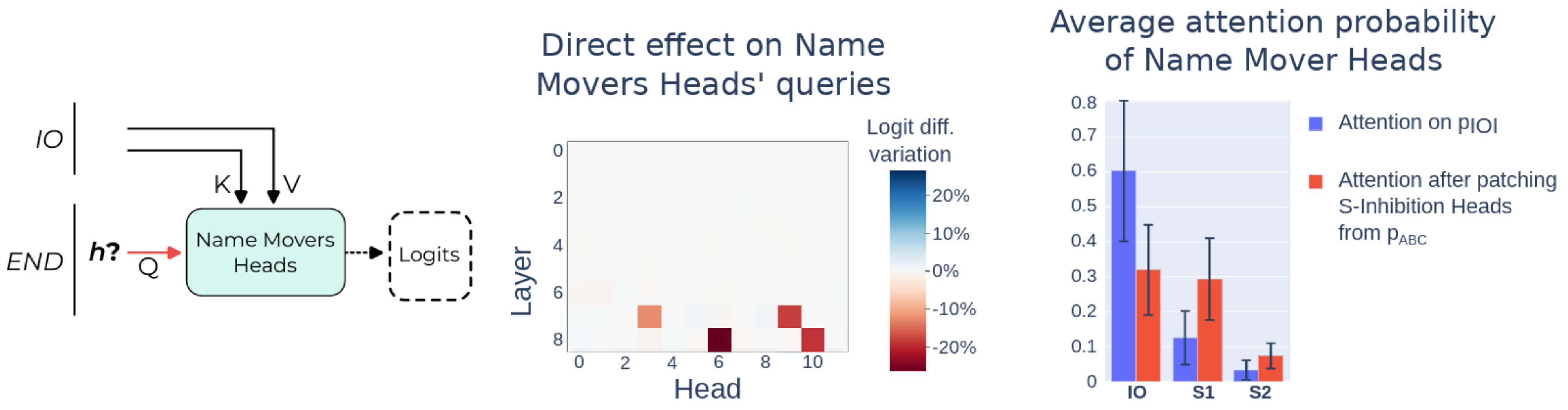
(b)



(c)

Figure 3: (a) We are searching for heads h directly affecting the logits using path patching. (b) Results of the path patching experiments. Name Movers and Negative Name Movers Heads are the heads that have the strongest direct effect on the logit difference. (c) Attention probability vs projection of the head output along $W_U[IO]$ or $W_U[S]$ respectively. For S tokens, we sum the attention probability on both S1 and S2.

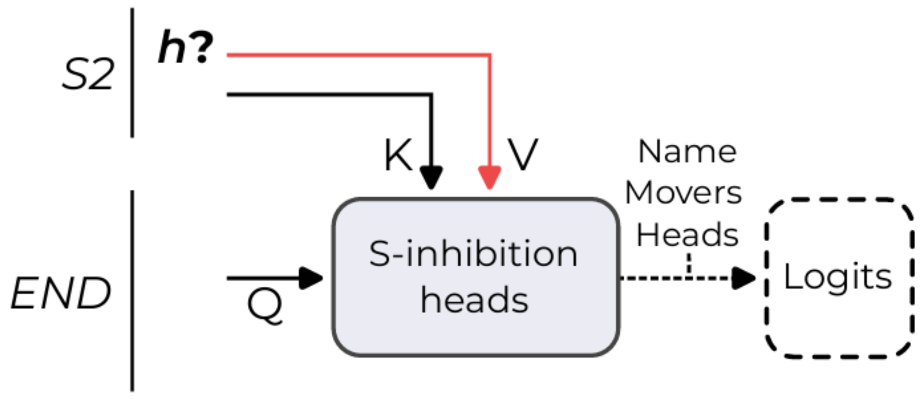
Indirect Object Identification



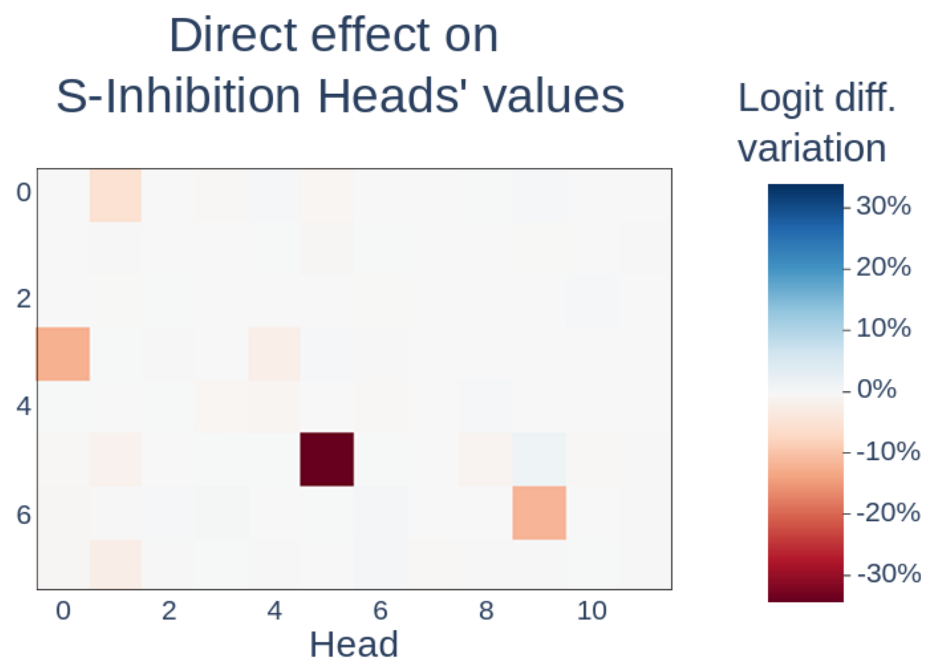
(a) (b) (c)

Figure 4: (a) Diagram of the direct effect experiments on Name Mover Heads' queries. We patch in the red path from the $\mathrm{p}_{\mathrm{ABC}}$ distribution. Then, we measure the indirect effect of h on the logits (dotted line). All attention heads are recomputed on this path. (b) Result of the path patching experiments. The four heads causing a decrease in logit difference are S-Inhibition Heads. (c) Attention of Name Mover Heads on $\mathrm{p}_{\mathrm{IOI}}$ before and after path patching of S-Inhibition Heads $\rightarrow$ Name Mover Heads' queries for all four S-Inhibition Heads at once (black bars show the standard deviation). S-Inhibition Heads are responsible for Name Mover Heads' selective attention on IO.

Indirect Object Identification



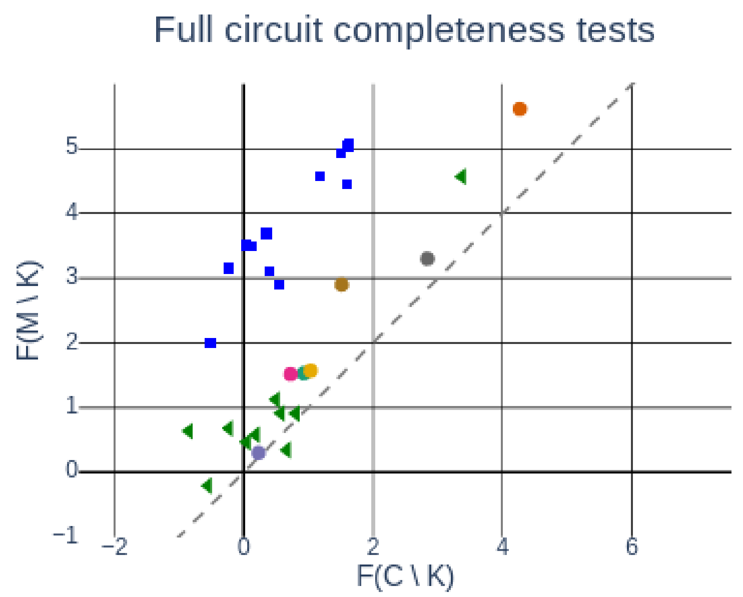
(a)



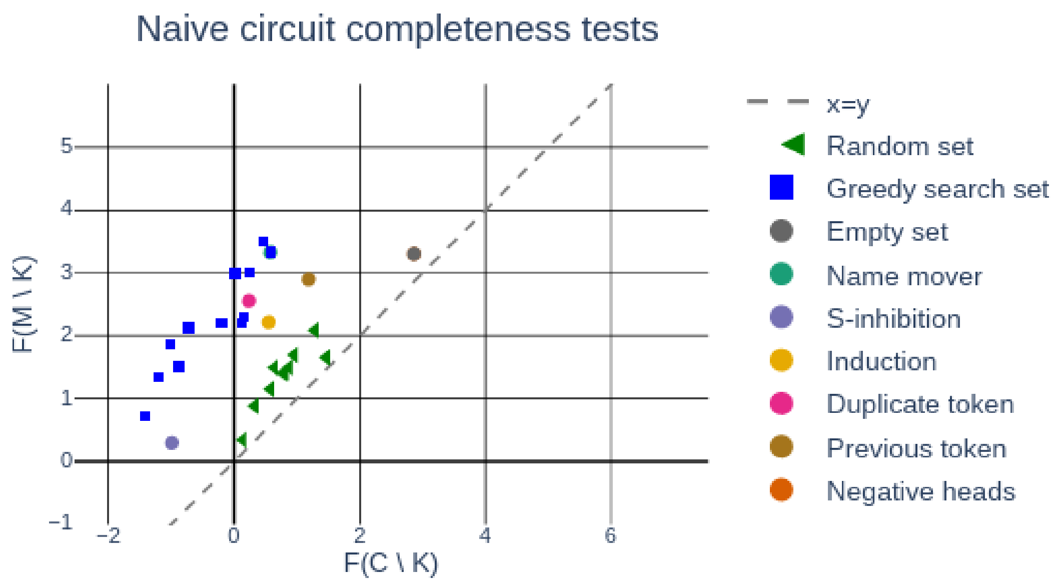
(b)

Figure 5: (a) Diagram of the direct effect experiment on S-Inhibition Heads' values. On the indirect path from h to the logits (dotted line), all elements are recomputed, Name Movers Heads are mediating the effect on logits. (b) Result of the path patching experiments for the S-Inhibition Heads' values.

Indirect Object Identification



(a)



(b)

Figure 6: Plot of points $(x_K, y_K) = (F(C \setminus K), F(M \setminus K))$ for (a) our circuit and (b) a naive circuit. Each point is for a different choice of K. We show sets K obtained from different sampling strategies: 10 uniformly randomly chosen $K \subseteq C$, $K = \emptyset$, K a class of heads from the circuit, and 10 K found by greedy optimization. Since the incompleteness score is $|x_K - y_K|$, we show the line $y = x$ for reference.

Indirect Object Identification

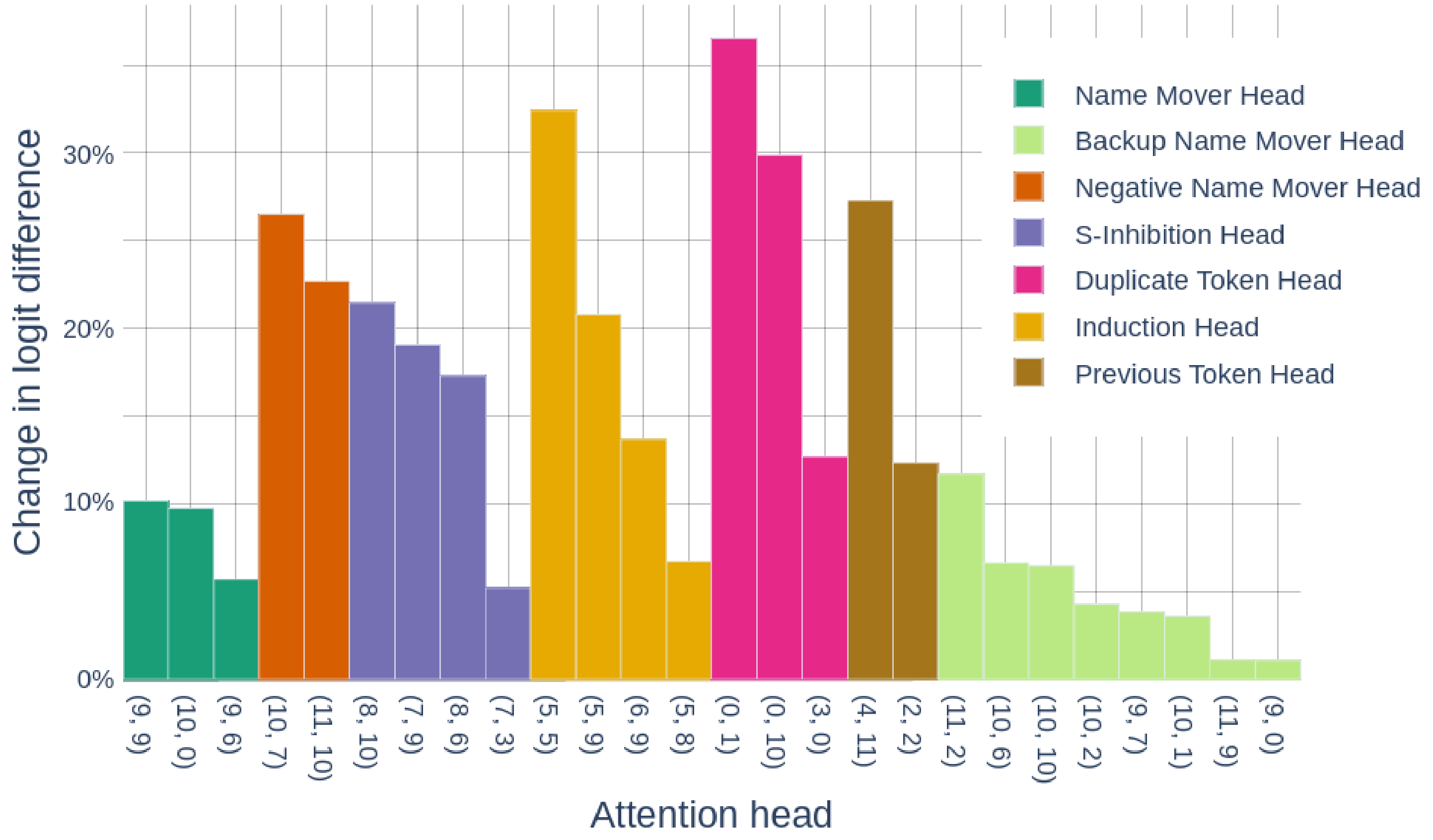


Figure 7: Plot of minimality scores $|F(C \setminus (K \cup \{v\})) - F(C \setminus K)|$ (as percentages of $F(M)$) for all components v in our circuit. The sets K used for each component, as well as the initial and final values of the logit difference for each of these v are in Appendix K.

References

- Code: https://github.com/redwoodresearch/Easy-Transformer