Who cares about brackets?[1]

Theo Clark **Alex Roman** **Hannes Thurnherr**

Abstract

We search for circuits within GPT2-small that could be responsible for informing a position about whether it is inside a bracketed sequence. We find that the current position does not need to attend directly to an opening bracket in order to accurately predict when it should close a bracket. Further experiments show that no precisely identifiable circuit seems to be responsible for this behaviour and we find increasing evidence to suggest that information to perform this task is highly distributed across the network. We are aware of the limitations posed by existing techniques for investigating this behaviour and highlight the development of these as important if more progress is to be made in understanding model behaviour such as this.

1. Introduction

Large Language Models (LLMs) are typically based on the transformer (Vaswani et al. 2017) and are trained to predict the next token. In most cases, predicting the next token requires understanding information from earlier context and Transformers are able to do this through the attention mechanism. The precise mechanisms for how this transfer takes place and where information corresponding to particular features of language is stored is generally poorly understood, prompting the conclusion that LLMs are a black box that we will never understand.

Although we are only scratching the surface, there is building evidence to suggest that there are precise circuits which implement specific algorithms (e.g. modular addition, Nanda et al. 2023). Most circuits have been identified in toy models but some progress has been made on larger, "wild" circuits which were not trained specifically for interpretability research (Wang et al. 2022).

We carried out Mechanistic Interpretability work into GPT2-small using TransformerLens[2]. Code for all experiments is made available[3].

[1] Research conducted at the Apart Research Alignment Jam #10 (Interpretability 3.0), 2023 (see https://alignmentjam.com/jam/interpretability)
[2] TransformerLens library: https://neelnanda-io.github.io/TransformerLens
[3] Code for all experiments: https://colab.research.google.com/drive/1WftRKjZ5sbbVV_RfiNbo56zAeJZH0S5m#scrollTo=HevV45yEeEFP

Motivated by the study of precisely how LLMs are able to monitor a particular “state” as a sequence progresses, we investigated the use of brackets and how the model is able to monitor whether it is inside or outside parentheses.

Despite tackling this problem from a number of angles, we did not find significant evidence of a circuit responsible for this behaviour and instead found some evidence pointing to information about bracket location being highly distributed throughout the network. However, we are also acutely aware of the limitations in tooling available to locate such networks, and highlight this as a matter of importance for future research agendas.

We remain optimistic that, with more work of this nature, progress can be made towards understanding this and more complex phenomena within LLMs which may ultimately help address issues such as truthfulness.

2. Results

Our main aim is to identify the circuit responsible for detecting whether the current position is located inside or outside parentheses. However, it is not possible to assess this directly and so we use a proxy for this concept: the likelihood assigned to predicting a ")". We hypothesise that the model needs to keep track of the concept of "inside vs outside of parentheses" to perform this task accurately.

Our general approach is to identify combinations of heads that correlate strongly with this proxy and then, once isolated, to reason more concretely and reconstruct the algorithm which might be implemented.

2.1 Splitting the sequence

We predict that, for a head responsible for this behaviour, we should see tokens up to and including the closing bracket attend much more strongly to the opening bracket than tokens after the closing bracket. Our initial pass identified (Head 3, Layer 10) and (Head 5, Layer 2) as the most promising candidates.

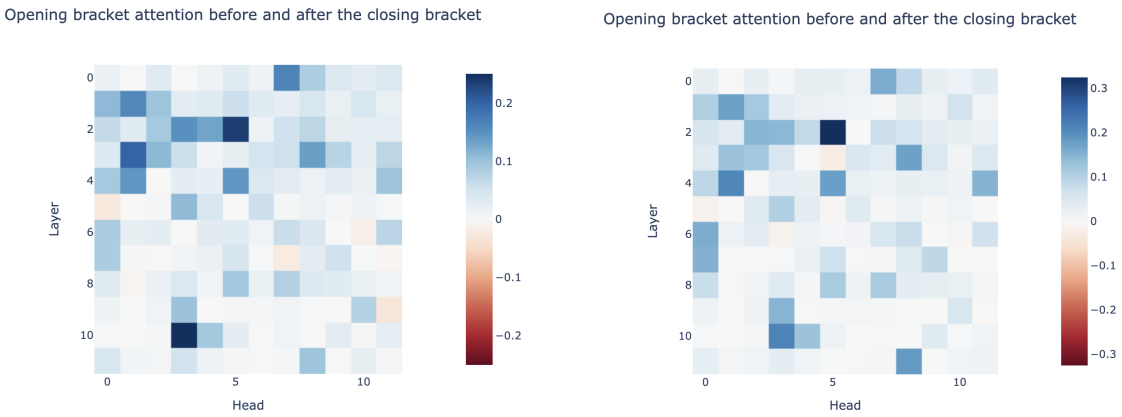

Figure 1 – Comparing the attention shown to the opening bracket by positions before and after the closing bracket. Left: random sequences, Right: real sentences

We reason that the arguments for and against closing a bracket mostly depend on the actual content of the text rather than just the question of whether a bracket has been opened. To see whether the same heads still show the same behaviour we let GPT-4 generate 27 sentences that contain a bracket and run the same experiment again. We expected to see much more noise since analysing the actual content of the sentences for whether a bracket should be closed probably requires much more circuitry than just one or two heads. Surprisingly, the results remain pretty consistent with head 5 of layer 2 and to a lesser degree head 3 of layer 10 being the most active.

This initially seems quite promising. However, there are a number of reasons why a token might want to attend to "(" when it's inside rather than outside of parentheses (e.g. judging phrase length) and so this is a long way from proving a causal relationship.

Indeed, we found no further evidence of these heads being responsible for this behaviour and remain unsure as to why these heads might be highlighted in this situation.

2.2 Activation Patching

Activation patching allows for a more surgical and fine grained approach to locating target heads. We first generate a prompt that predicts ")" at the final position and then investigate patching attention values to -inf to prevent the final token from attending back to the "(" position.

Results

1. We found that patching the relevant value in a single head made no difference to the overall probability.
2. We even found that patching every single head for this particular value made little difference.
3. Blocking any future tokens from attending to the "(" does prevent ")" from being predicted (sanity check).
4. Gradually patching more layers and positions does not yield a sudden change in probability at any point.

We conclude that information about the open bracket is passed to the final position progressively through other positions rather than exclusively attending to the "(" and that this process seems highly distributed across layers.

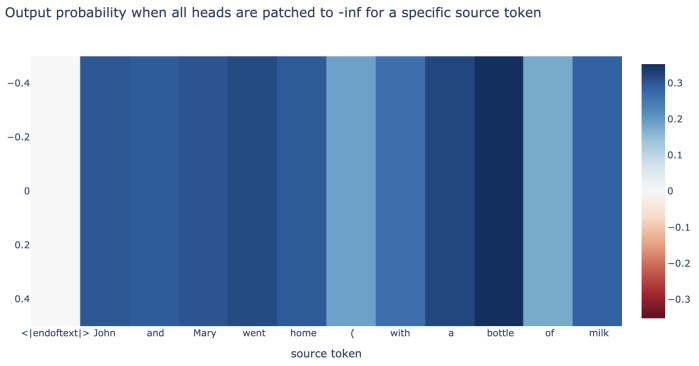

Figure 2 – Output probability when all heads in the final position are patched for a specific source token

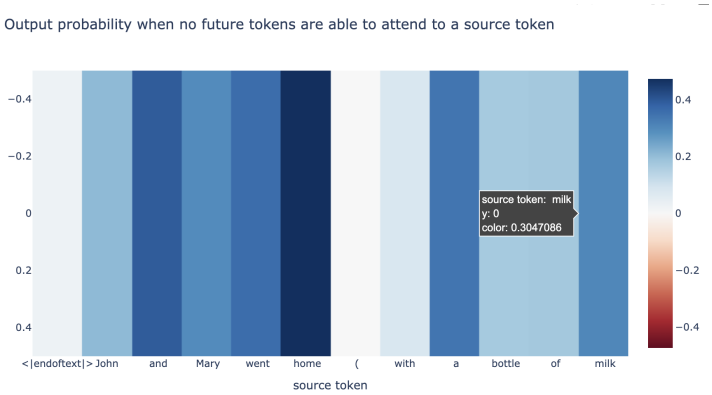

Figure 3 – Output probability when all heads in subsequent positions are patched for a specific source token

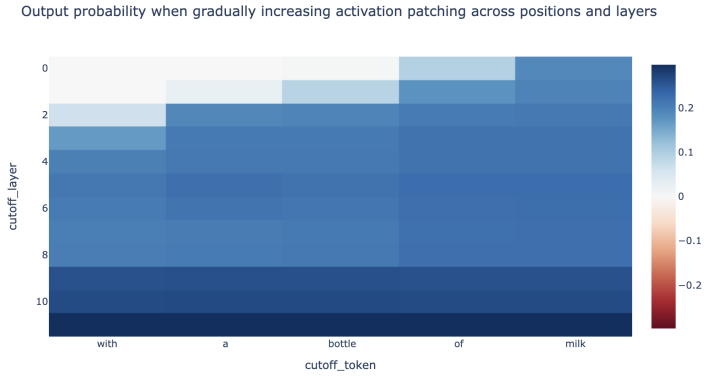

Figure 4 – Output probability when gradually increasing activation patching for the "(" source token along the position and layer dimensions. Bottom left: all subsequent positions are patched but no layers. Top right: only 'milk' is patched but for all 12 layers.

2.3 Single token predictions

In a bid to simplify the problem and gain more traction, the sequence following the opening bracket was shortened to a single character (i.e. what is the probability of predicting ")" one after "(", for example "(a)" would be a valid string).

This reduces the complexity of the problem but is less closely connected to the typical behaviour we are studying, something we address later on.

Methodology

1. A random string was used so as to decrease the chance of other relational information influencing activations (though this seemed to make little difference).
2. An inverse string, replacing ")" with "(", was used to generate activations for patching. This is a more realistic method for 'undoing' an activation than simply setting it to zero.
3. Carry out similar activation patching, but this time with fewer positions to worry about.

Results

1. Patching the attention output in the first 3 layers is enough to reduce the likelihood of predicting ")" to almost 0
2. Patching the second layer by itself was not enough to cause this behaviour: it needs to be combined with either layer 0 or layer 1
3. Ablating this result across all head combinations shows a particularly strong correlation between patching layer 1, head 5 and layer 2, head 1.

These last two results patched the value activations rather than the attention output

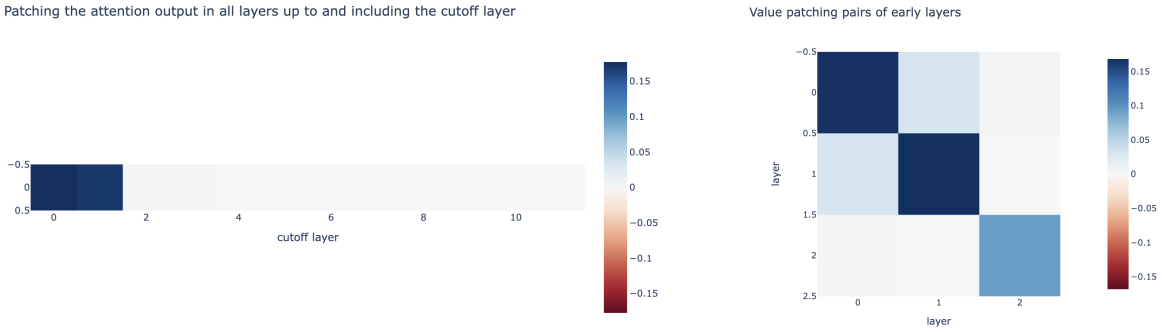

Figure 5 – Left: patching the attention output in all layers up to and including the cutoff layer. Right: ablating this for all pair combinations for the first three layers.

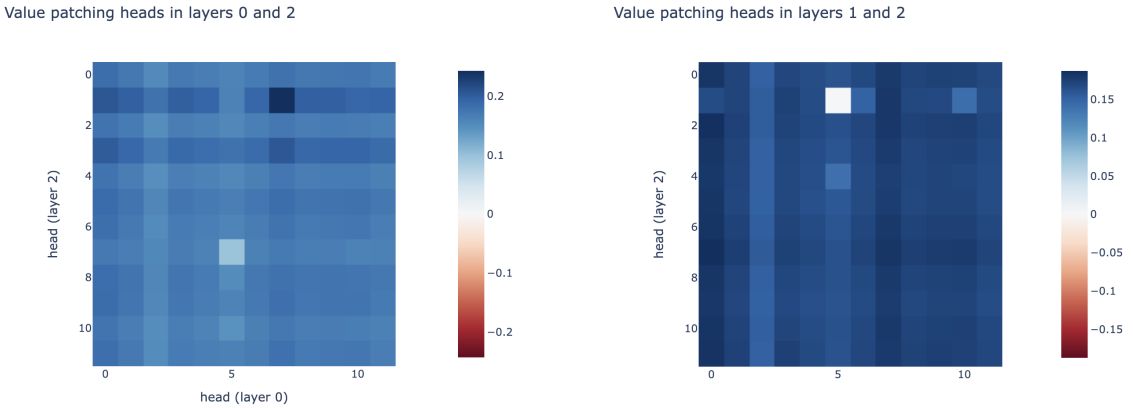

Figure 6 – Further ablations on pairs of heads. Left: layers 0 and 2. Right: layers 1 and 2

This result initially indicates that a combination of two heads (layer 1, head 5 and layer 2, head 1) correlates with reducing the probability of outputting a ")" when the preceding " (" is replaced with a ")". We came up with a number of hypotheses for what this "circuit" of two heads might be doing:

1. The circuit is spurious and does not generalise beyond this exact example
2. The circuit is responsible for the very precise task of predicting a ")" immediately after a " (" but does not generalise beyond that.

3. The circuit is predicting that a ")" should not follow another ")" and is showing up here because we are patching with ")".
4. The circuit is responsible for the more general behaviour of detecting whether the current position is inside or outside parentheses
5. This circuit represents a general understanding of being “inside a bracket” and is also able to map that behaviour to other types of brackets e.g. [], {} or <>

We then made a number of switches to the original and inverse prompts to test these hypotheses.

Results:

1. Inverting the prompts (i.e. patching " (" over ")") causes the inverse behaviour as expected: the two heads we are interested in correlate with a much higher probability of predicting ")" next.
2. Altering earlier parts of the sequence or the final token does not cause any noticeable difference in output distribution.
3. Patching " (" with a token other than ")" (i.e. a random word) does NOT yield the same pattern. Indeed sometimes it increases the probability of outputting a ")"
4. If we then start with ")" in the main sequence and patch with a random token we see no correlation to the particular head in question.
5. Repeating the original experiment but with square brackets as opposed to round brackets breaks the pattern, indicating that this does not generalise to other forms of "openness".

Further experiments, similar to the ones above, were carried out with square braces to identify whether the same layers (0,1,2) were responsible for this behaviour and if there was any correlation. For square braces, the third layer also seemed to be important and there was no 3 way combination of any 3 heads in the first 3 layers that seemed to correlate with reducing the probability of predicting a ")" (see result below).

These results indicate that the circuit is more closely related to detecting a closing bracket than an opening one, but even this is not concrete. It certainly does not generalise to other brackets. Whilst this might be something worth investigating it does not seem to represent the idea of "being inside/outside parentheses".

2.4 Averaging Activations

The methodology used above did not seem to isolate the intended behaviour. This may be because it is quite unusual to want to predict a closed bracket straight after an open bracket and does not represent a realistic scenario in which the circuit needs to judge whether it is inside or outside parentheses. We therefore reverted back to using longer sequences but opted to average the activations across samples in the hope of isolating a more reliable signal.

All prompts are similar to the following example and are adapted from our earlier GPT4 prompts:

"The main house (once grand and very elegant"

The model assigns the highest probability of a ")" to the next token for all samples and they are all of the same token length and have the open parenthesis in the same location.

We used the following hypothesis of how the model might approach this task to guide us:

1. Does the previous token(s) indicate that this might be a new clause?
2. If yes --> "("
3. If no: am I inside a bracketed string?
4. If no --> predict random token
5. If yes: does it make sense to close the bracket given the current context?
6. If yes --> ")"
7. If no --> predict random token

It is the third step that we are looking to isolate and locate the circuit for.

Methodology

Obtain a set of averaged activations:

1. Run a forward pass and save out the cache for each sequence.
2. Repeat but replace " (" with " ". These will represent the inverse sequences.
3. Average all activations for the standard sequences and the inverse sequences.
4. Confirm that the averaged activations have the desired behaviour.
5. Compare the cosine similarity of incoming keys and values, as well as the output of each attention layer.
6. Repeat some of the earlier patching experiments but this time with the averaged values

Cosine Similarity

Results

1. Most differences in cosine similarity occur for keys originating from the " (" token as you would expect but there are still clear differences in some of the other tokens which implies that information about the bracket can be passed indirectly through the network and the current position does not have to rely on attending directly to the " (", which corroborates what we have found previously.
2. Ablating the cosine similarity across heads for keys and values originating from the "(" position shows a lot of variation but there is nothing linking this to predicting ")" at the current position.
3. Plotting the cosine similarity for the "z" activations of every head shows a gradual decrease in similarity as the layers increase. It is likely just the case that, as we proceed through the network, there is more of an opportunity to attend to information from the changed token and so we see progressively more difference as we move through the layers. Relative difference within the same layer is therefore likely more of an indicator.

Whilst cosine similarity plots are interesting they do not link directly to the likelihood of outputting a close bracket. It is therefore difficult to read too much into these results. However, combined with other information from activation patching they could be informative.

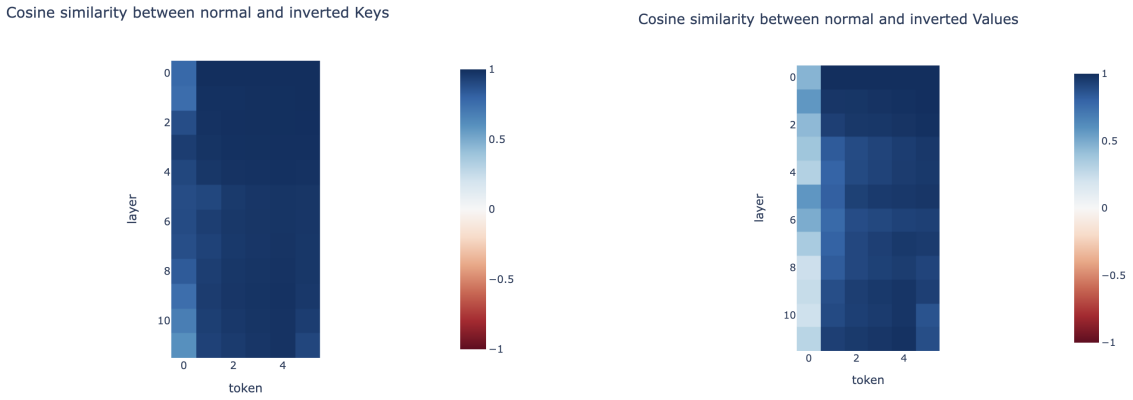

Figure 7 – Cosine similarities between the normal and inverted activations for keys (left) and values (right) by layer and position for the final position. The 0th position refers to the location of "("

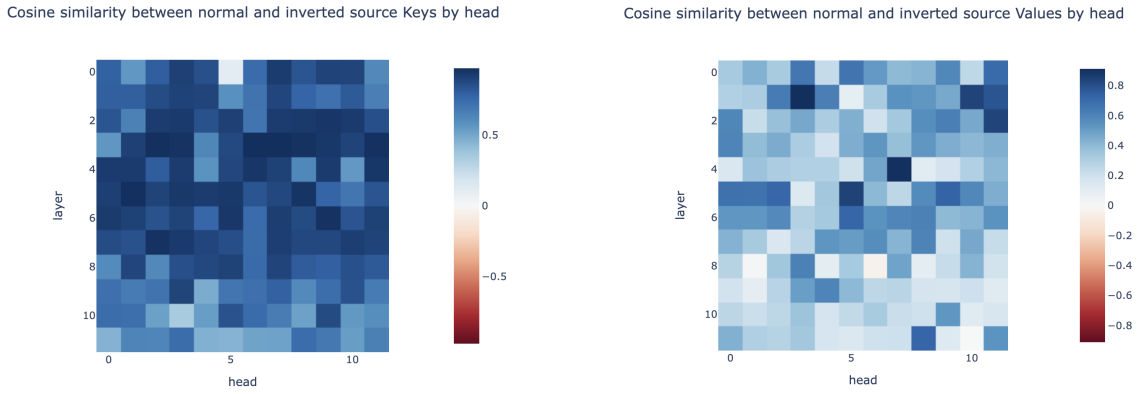

Figure 8 – Cosine similarities between the normal and inverted activations for keys (left) and values (right) by layer and head, for the final position attending to the position of the "(" token.

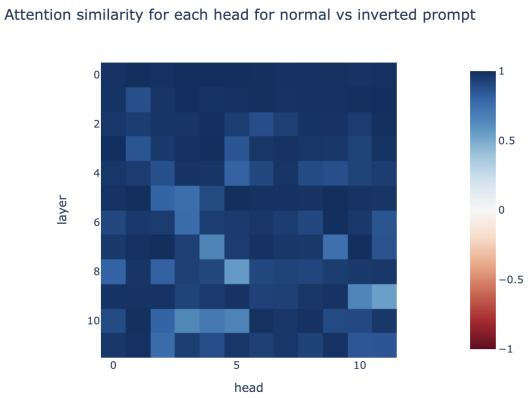

Figure 9 – Cosine similarities between the normal and inverted activations for the "z" activations in the attention layer for the final position attending to the position of the "(" token.

Activation Patching

Methodology:

1. Take a sample sequence at random and always patch the very first residual with the standard average to negate the input.
2. Patch the key and value activations in every layer as this covers all of the activations that the final position is able to use to attend to prior information
3. Iterate over different layers - sometimes patching with the standard average and sometimes with the inverted average to identify if there are any layers/heads which correlate strongly with predicting a ")"

Results: Sweeping over individual layers or blocks of layers yielded no clear layers or heads that might be responsible. A greedy search was performed, gradually increasing the number of layers that are patched, to identify which layers are most important, and this indicates the latter layers. However, blocking out the final few layers yields only a gradual decrease in likelihood.

3. Discussion

Our experiments found no substantial evidence of a specific circuit dedicated to identifying whether the current position is inside or outside of a bracketed sequence. There are a number of possible reasons for this which are addressed in turn below.

1. Predicting a closing bracket is a bad proxy

Throughout these experiments we have used the probability of predicting a ")" as a proxy for whether the model believes it is inside or outside of parentheses, based on the idea that it is unlikely to assign high probability to ")" if it is not inside parentheses.

We have already indicated that predicting a ")" straight after " (" does not represent most situations where it is important to keep track of this across a sequence.

We have tried longer sequences with both random sequences (to isolate bracket behaviour) and realistic sequences (to better simulate real situations).

The last effort, involving averaging realistic sequences, attempts to tie both of these together: both representing realistic sentences and averaging out some of the noise you might expect from them.

However, compared to Indirect Object Identification, where predicting the correct token is a very clear indicator of whether the model possesses this concept, the present task is less tangible and proxies are harder to come by.

There are many concepts which suffer from the same issue and further work is necessary to identify better methods or proxies for eliciting their behaviour.

2. The model does not possess or need this concept

This is a difficult question to answer and depends slightly on definitions. On the one hand, it is clear that GPT2-small is very accurate when it comes to placing brackets in the

correct place. On some level it clearly possesses the ability to know when it needs to add a ")". If the only thing this depends on is whether there is a "(" earlier in the sequence then it must be using that information when making its prediction.

Even if the model does not have the concept of "am I inside a bracketed sequence?", it must have a related concept, e.g. "is there a ')' more recently than a '(' in the preceding sequence?".

In essence this issue cancels out the first point regarding proxies. We are attempting to locate whatever 'concept' it is that allows the model to correctly predict a closing bracket.

If we accept that the concept is poorly defined but nevertheless present then the proxy ceases to be a poor one.

3. Our methodologies are flawed/limited

We have already discussed the potential issues with our choice of proxy for this behaviour. However, the techniques used to try to isolate this behaviour are definitely limited. It is not feasible to exhaustively search all possible combinations of heads to isolate this behaviour.

4. The circuit responsible for this concept is highly distributed

We have painstakingly isolated parts of the network to attempt to identify this behaviour and no highly reduced combination of layers/heads has been found that is clearly responsible.

There are broad combinations of layers that seem to be of more importance than others but nothing fine grained.

The slightly different problem of predicting the sequence "()" seems to be more easy to isolate but does not seem to correlate to the more general problem of predicting closed brackets later in a sequence.

With the tools and methods currently available to us, we are left to conclude that this concept is simply highly distributed throughout the network. This aligns with the fact that preventing the final token from attending to "(" is not enough to prevent a ")" from being predicted, indicating that this information is passed gradually through the network.

However, we are reluctant to conclude this definitively and the main takeaway from our work should be that there is still much work to be done to improve methods and techniques for identifying circuits within models.

4. References

Vaswani, A. *et al.* (2017). *Attention is all you need*

Nanda, N. *et al.* (2023) *Progress measures for grokking via mechanistic interpretability*

Wang, K. *et al.* (2022) *Interpretability in the wild: A circuit for indirect object identification in GPT-2 Small*