Finetuning Provides a Window Into Transformer Circuits

Stanford CS224N {Custom} Project

Sachal Malick
SCDP
Stanford University
`sachalm@stanford.edu`

Abstract

In pursuit of further visibility into the black-box of large models, Mechanistic Interpretability research aims to discover human understandable algorithms that these models employ to perform specific tasks. Mechanistic Interpretability techniques focus on discovering circuits of model components that are key to the models performance on a task through causal interventions, and analyzing the components that are most interpretable. In this work, we set out to understand the mechanisms GPT2-small uses to perform a new task that we introduce: the Transitive Property Problem (TPP). That is problems where given the relations aRb and bRc, the model must predict that there is a relation aRc. To boost performance on this task we finetune GPT2-small on a dataset comprised of examples of this task, and find evidence that finetuning on task examples reinforces relevant circuits. Additionally, we discover hints that finetuning can be used to improve automated circuit discovery processes.

1 Key Information to include

- Mentor: Rohan Taori

2 Introduction

Modern state of the art Large Language Models (LLMs) have demonstrated the ability to perform a wide range of sophisticated tasks, from writing poems and explaining jokes to solving Math Olympiad level geometry problems. While we understand the architectures that enable these behaviors to emerge, we have little insights into the underlying processes. There is considerable value in broadening our understanding of these models. The more interpretable a model is, the higher degree of confidence we can assign to their safety and thus the higher the likelihood that they can be deployed to mission-critical services. The promise of interpretable models includes being able to proactively identify learned biases and flawed algorithms. Additionally, we can potentially identify patterns these models have learned about our word that have never been previously discovered.

Interpretability research thus far can be grouped into two general categories, Intrinsic and Post-Hoc methods. Intrinsic focuses on developing new architectures that are designed to be interpretable, while Post-Hoc methods focus on trying to understand existing models that were trained with or without interpretability in mind. Mechanistic Interpretability falls within the Post-Hoc category and uses causal interventions to discover and explain the internal components that compose together to produce a behavior. Mechanistic Interpretability research has been characterized as labor intensive, especially for larger models with more components. However, new tools have been created recently to help automate some key steps in the research process. Thus far, there have been only a handful of circuits discovered on real-world models.

In this work we expand the corpus of known circuits by defining a new task, then computing and evaluating a corresponding circuit. The Transitive Property Problem (TPP) is defined as a prompt that states two relations aRb and bRc, and a language model being evaluated on whether it predicts the tokens corresponding to the relation aRc.

We find that the pretrained models that we have the computational resources to evaluate do not perform well on baseline evaluations of this task, so we finetune GPT-2 on examples of this task. By performing this evaluation on models which have been finetuned on varying amounts of examples and for different training epochs, we are able to gain some insights into how finetuning affects the circuitry of a transformer.

In order to discover the components that are key to performance on our task we attempt to use a tool called ACDC introduced in the paper Automated Circuit Discovery for Mechanistic Interpretability. The tool uses ablations to identify the circuit that has all of the nodes required to maintain performance on the task within some configurable threshold and contains as few extraneous nodes as possible.

Ablations are the key causal act performed in mechanistic interpretability, and it simply involves changing the value of an activation which is the result of some intermediary computation in the forward pass of the model. By ablating a value and assessing the impact on upstream components, we can understand what the significance of that value is on those components.

We create a custom tool that focuses on the most interpretable part of the model, attention scores, and ultimately find better circuits than those produced by ACDC.

We analyze the attention patterns of the identified attention heads that have a large impact on performance, and we look for a relationship between key components and the weights of the model before and after finetuning.

3 Related Work

There are three key papers that lay the groundwork for the research conducted in this study. The first is A Mathematical Framework for Transformer Circuits from which there are a few important takeaways Elhage et al. (2021) .

First, is the principle of focusing on the inherently interpretable parts of the model, in the case of transformers those are mainly the tokens/logits and the attention patterns. Second, this paper establishes attention heads as independent and additive structures with contributions to the final logits that can be interpreted by decomposing the residual stream. Third, they identify two unit circuits that operate in attention heads, QK circuits and OV circuits. We refer to them as unit circuits because the weight matrices W_Q and W_K always operate together, similarly W_O and W_V always operate together. Finally, they introduce induction heads which look for the previous instance of a token and if present it attends to the proceeding token.

The next paper that we will briefly discuss is Interpretability in The Wild: A Circuit for Indirect Object Identification in GPT2-Small Wang et al. (2023). In this work the authors define the Indirect Object Identification (IOI) task which evaluates the models ability to predict the indirect object in a prompt of the following format: "John and Mary went to the store today, John gave a drink to " with the correct next word being "Mary". This paper is significant because it contains the largest circuit discovered in a language model at the scale of GPT2-small. In addition, they establish a method for mean-ablation that entails patching an activation with the mean of that activation on a dataset containing out of distribution token substitutions. They indicate that they observed zero-ablations have misleading consequences on upstream components. They also introduce principles for evaluating a circuit that we will apply in this study. Those principles are faithfulness, meaning the model represented by the circuit does not deviate significantly from baseline performance on the task, completeness which requires that there are no essential components missing from the circuit (this principle is closely coupled with faithfulness) and minimality which implies that the circuit is as small as possible.

The last paper which we need to highlight is Towards Automated Circuit Discovery for Mechanistic Interpretability which introduces the ACDC tool that we apply in the circuit discovery process. We will discuss this more in later sections Conmy et al. (2023).

4 Approach

To evaluate the mechanisms a model is using to perform the task, we need to ensure the model is performing the task in the first place, and if so how well. To establish a baseline, we constructed a dataset that consists of prompts and the expected next tokens. We created 7 prompt templates listed in the Appendix. These prompt templates have placeholders for three insertions, an "a" value, "b" value and a "c" value, where the prompt and answer tokens represent the following logical structure.

$$[(a \rightarrow b) \wedge (b \rightarrow c)] \rightarrow (a \rightarrow c) \tag{1}$$

In the prompt templates, we ensured that there were varying sequence lengths to reduce the likelihood that the correct answer token would sit in a consistent position, additionally we provided templates that contained different key words to test the models ability to generalize across different variations of the TPP. We insert a random and unique noun in the a, b, and c placeholders, and use the nltk wordnet corpus as our source of nouns.

We decided to insert random nouns for our a, b, and c values to ensure the model relies on in-context learning to perform the task. We observed that prompting with GPT-4 in this manner resulted in a response that contained the correct answer with an explanation that explicitly stated the use of the transitive property. However, this decision likely had unintended consequences on our results which we will discuss in a later section.

We chose GPT2-small as it is amongst the largest models that we could perform rapid experiments on given our available resources, as well as to build off the IOI work which was conducted on GPT2-small with the hopes that focusing on understanding a single complex model can lead to accelerated progress for the field.

To briefly review the GPT2-small architecture, the model is a decoder-only transformer and it consists of 175 million parameters. At the base is an embedding layer for encoding token and token position information with 12 transformer blocks each one containing a 12-head self-attention layer and a MLP layer. GPT2 is a causal model meaning that during pretraining all future tokens are masked and each token can only attend to itself and previous tokens. The input of each layer is the residual stream and the output is added back into the residual stream. LayerNormilzation and Dropout are also implemented Radford et al. (2018).

We used supervised finetuning to boost the baseline of GPT2-small wherein we passed tokenized batches of prompts from our a dataset containing 156,000 examples of our task as the inputs and set the labels to be the "c" tokens. The data pipeline we built consisted of three steps, first generation of the dataset as described above, second we tokenized all the prompt strings, remember the prompt does not contain the answer token (which is the c token). We also tokenize all the c tokens independently. Lastly, we construct the labels and attention masks. We compute the maximum number of tokens in our master dataset, and pad every input to have that many tokens. We choose the GPT2 EOS token as the pad token as GPT2 does not have a default pad token. We constructed the labels by creating tensors that are the same size as the padded prompt tokens, and we placed the tokens for the c label in the positions following the last prompt token, and we set every other value before and after the c labels to -100. This is because GPT2 ignores -100 in the cross entropy loss, and for the purposes of this task we only care about whether it is able to predict the right c tokens given the prompt tokens.

We finetuned GPT2-small three times producing three separate models. Each finetuning run used the same tokenizer, the same base pretrained GPT2 model, the same learning rate of 5e-5, batch size of 64, and drew its train and evaluation dataset from the same data pipeline described above. However each model used a different amount of training data. Our reason for doing this was so that we could evaluate the effects of the finetuning process on the circuit of the model. Our first finetuned model which we will from here on out call Featherweight was trained on 10 percent of the full dataset which amounts to 15,800 examples and we trained for three epochs. Our second model which we will refer to as Lightweight was given 30 percent of the full dataset which is 47,400 examples and it was also trained for three epochs. Our third model which we call welterweight was trained on 50 percent of the master dataset which comes out to 79,000 examples and was trained for 20 epochs.

After we have our baselines for the pretrained GPT2-small and the finetuned models, we move to the ciruicit discovery part. For this we utilize the ACDC tool and create our own tool that focuses on attention heads and runs faster allowing for more rapid experimentation. We perform a series of ablation based experiments that we used to discover redudant layers.

We then move to understanding the function of this component and from here focus on the attention patterns. We visualize the patterns for every layer and head in our circuit using different example prompts to formulate a hypothesis for what the attention head is doing. We then perform computations on attention scores to evaluate the degree to which the attention head is consistently serving that function.

We make use of the TransformerLens library for the purpose of accessing cached activations, and to perform ablations, however we write our own logic for most of the experiments that we perform.

5 Experiments

5.1 Performing a Baseline Evaluation

We make use of two evaluation datasets. The first set is comprised of 3000 randomly sampled examples from the dataset previously described. For the purposes of simplifying evaluation, we remove all the padding and ignore tokens that were added for finetuning. In our full dataset the labels which are the c tokens can have more than one value, as can the tokens for the a values, and b values. This is because the GPT2 tokenizer will split certain words into multiple tokens (in particular longer words). As an example, the word "preposterous" gets assigned three tokens by GPT2 46012 which maps to "prep", 6197 which maps to "oster", and 516 which maps to "ous".

We only ever compare the first token the model predicts to the first token of an answer. So in the above example, given the input "if all fish are drinkers and all drinkers are preposterous it can be inferred that all fish are ", we will measure the models ability to predict that the next to token is 6197, and not consider the preceding or proceeding tokens. The reason we choose to do this is to simplify the interpretability process by limiting our focus to single logits. During Finetuning, all of the c tokens were used to measure loss.

Our second evaluation dataset is constructed in exactly the same way, however it has 1150 examples and it is limited to instances where c is already represented by a single token. To increase the number of examples where this occurs, we remove the trailing space from the prompt, and add it to the beginning of the c token. While this is a departure from the way the prompts were formatted in the finetuning process, we find that it does not make a big impact on the performance of the finetuned models.

We chose to use three metrics for the baseline, cross-entropy loss, fraction correct (accuracy), and logit difference. For each of these metrics, we take their average over the evaluation dataset.

Model	Log Prob	Frac Correct	Logit Diff
GPT2	11.92	0.000333	-0.378
Featherweight	0.859	0.819	18.155
Lightweight	0.503	0.902	29.601
Welterweight	0.892	0.911	41.054

Table 1: Baseline results on 3000 example dataset

Cross entropy Loss measures the difference in probability distributions, in this case given that we are measuring the cross entropy with a distribution that is zero for all classes except to expected token, this is just the negative log of the probability assigned to that token after a softmax over the logits is applied. We choose this metric to evaluate the weight the model assigns to the correct token relative to all other tokens.

Fraction Correctly is simply the number of times in which the highest logit value is assigned to the expected token divided by the size of the dataset. We chose this metric to evaluate the accuracy of the model.

Logit difference measures the the difference between the logit for the c token and the logit for the b token. This is a metric that was also used in the IOI task that tells us whether the model is predicting the c token despite the fact that the b token appears twice in the prompt.

Model	Log Prob	Frac Correct	Logit Diff
GPT2	6.629	0.0	-0.991
Featherweight	0.0375	0.994	17.070
Lightweight	0.011	0.998	34.821
Welterweight	0.559	0.857	43.315

Table 2: Baseline results on 1500 example dataset

We can see from the results in the table, that the finetuned models performed significantly better at the task than the pretrained GPT2. The negative logit difference between the c token and b token indicates that the model tends to assign a higher value to the b logit than the c logit.

We can also notice that the welterweight model performance on the logit difference is greater than lightweight or featherewight, however lightweight and featherweight both have a lower cross entropy loss, and fraction correct scores that are quite close.

5.2 Searching for Circuits with ACDC

To recap, the ACDC algorithm creates a graph that represents the connected components in the model. Each node represents an activation value and and a directed edge is established between some Node A to Node B if the activation corresponding to Node B contributes to the activation corresponding to Node B. Note that there are multiple ways to define a computational graph for a single model. The residual stream especially adds some complexity to this process as components in higher layers can potentially be accessing the contributions from a previous layer that it is not connected to through the residual stream Elhage et al. (2021). For the purpose of this experiment, we used the default graph configuration in which all attention head nodes and all MLP nodes are connected to all downstream residual nodes.

Next ACDC performs a breath first traversal of the graph beginning with the final output from the residual, for each node it goes through all the connected nodes and performs an ablation of the corresponding activation. It then evaluates the model with the ablated activation on the dataset and if the performance does not fall below a provided threshold it removes the edge and continues traversing through its descendant nodes with the same process. When there are no more edges connected to a node it is removed from the graph.

To leverage the ACDC tool, we had to create a new module for our task that would setup our experiment by creating a dataset and defining metrics. Given the computational complexity of the ACDC algorithm, we were restricted to creating a much smaller dataset than we had used to take our baselines. We used 100 examples from our evaluation dataset drawn at random, of which 50 were allotted for evaluation. We chose to stick to zero ablations for consistency with other experiments we conducted.

We ran ACDC on Welterweight, Featherweight, and GPT-2. We used three metrics, Logit Difference, Fraction Correct (Accuracy), and KL-Divergence.

Choosing the right threshold required a series of trial and error runs. One drawback of the ACDC algorithm is it works by taking in an amount you are willing to see model performance drop by, and then if an ablation drops performance below that range the new performance with that ablation is now the baseline. If your threshold is γ and your original performance is τ then after n ablations you can have a performance drop which is $\tau - \gamma n$. While one strategy might be to divide the total performance drop you think would still adhere to the faithfulness requirement and then divide it by the number of nodes in the graph, this will mean holding the earliest nodes in the traversal (the final layers in this case) to significantly higher standards that models in lower layers.

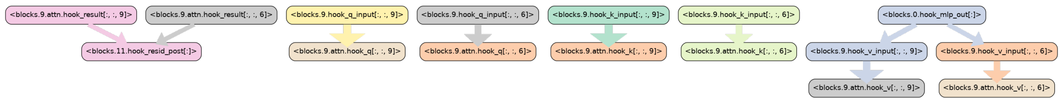


Figure 1: The circuit produced on Welterweight with KL-Divergence Threshold 0.7

We implemented a slight modification to this algorithm, instead changing threshold to represent the percentage in performance the model can drop. Therefore total performance drop after n ablations can be $\tau - \tau\gamma^n$. However this has the reverse affect, leaving the final activations subject to high standards and resulting in a much larger circuit.

Model	Metric	Threshold	Nodes in Circuit
GPT2	KL Divergence	0.7	3
Featherweight	Accuracy	0.1	487
Welterweight	Accuracy	0.1	535
Welterweight	KL Divergence	0.7	11

We can see that there is tradeoff between faithfulness and minimality, and in this case we biased towards faithfulness. While 0.1 might seem like a strict threshold, any larger value results in circuits that quickly deteriorate in performance. Note that while the KL Divergence metric with a 0.7 threshold produces clean circuits, accuracy is 0 when test on the evaluation set and logit difference is negative. We will also point out that of the 535 nodes included in the Welterweight 0.1 run, 101 are part of attention heads, and 97 of the 487 nodes included in the Featherweight run are part of attention heads.

5.3 Attention Focused Ablations

We created a script that operates similarly to the ACDC tool but differs in three key ways. First it only focuses on ablating attention scores. These are the components that are highly interpretable and restricting our algorithm in this way improves execution time and allows us to experiment more rapidly. The second way it differs is by iterating from Layer 0 upwards. With either of these algorithms you can get a different circuit depending on which order you explore the graph. Second, we establish a baseline at each layer before ablations. We iterate through every head and every layer, ablate the attention scores for that head and layer, test the model on 100 examples from our evaluation set, and if performance relative to the baseline established at that layer does not fall below the threshold then we keep that ablation.

Model	Constructed on Model	Threshold	Heads Removed	Accuracy
Featherweight	Featherweight	0.03	71	0.003
Lightweight	Featherweight	0.03	71	0.079
Welterweight	Featherweight	0.03	71	0.740
Featherweight	Lightweight	0.03	86	0.009
Lightweight	Lightweight	0.03	86	0.440
Welterweight	Lightweight	0.03	86	0.904
Featherweight	Welterweight	0.03	75	0.005
Lightweight	Welterweight	0.03	75	0.031
Welterweight	Welterweight	0.03	75	0.776

In the above table, we show the model that the evaluation is performed on, the model the circuit was constructed on, the per-layer threshold we set for performance, the number of attention heads that were ablated, and the accuracy on the 3000 example evaluation set.

One observation is that circuit performance falls well below the specified threshold when tested on a larger dataset, this is an important point to consider for automated circuit discovery tools as the results from this experiment indicate that rigorous evaluations need to be performed after every ablation.

We can also see that Welterweight evaluates better with circuits constructed on Lightweight and Featherweight than the models themselves. Similarly Lightweight performs better with circuits constructed on Featherweight than Featherweight does itself. However, the converse is not true which seems to indicate that finetuning reinforces existing circuits.

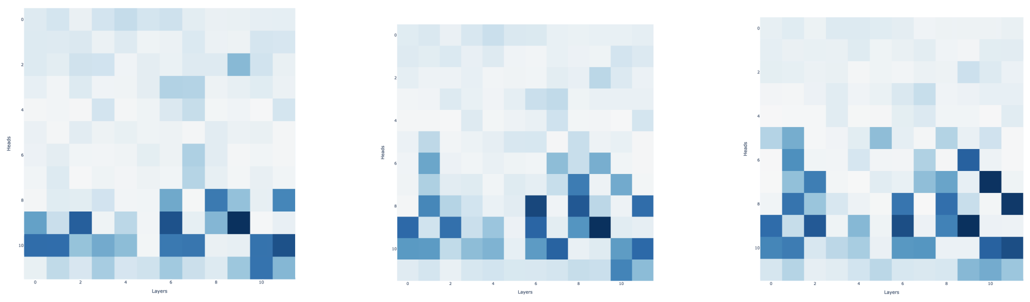
Finally, we observe that given the same threshold, we are able to find a better circuit for Welterweight by constructing it on Lightweight than we were by constructing it on Welterweight itself. This hints at the possibility of using variants of a finetuned model to improve circuit discovery.

5.4 Evaluating Attention Scores

With a good sense of what attention heads are important to performance in this task, we can now start trying to understand their function. Out of 144 heads, we were able to eliminate 86 of them from our circuit, which leaves us with 58 to try to classify.

First, we can take a look at the attention heads that are highly attentive to the answer toke, c. We take 100 examples from our evaluation set at random, cache the attention scores for each head, and average them.

The figures above make it clear to see the increased importance assigned to the c token as the amount of examples seen in the finetuning of a model version increases. All of the attention heads that pay high attention to c in Featherweight are also in Lightweight. All of the attention heads that pay high attention to c in Lightweight are also in Welterweight.



(a) Featherweight (b) Lightweight (c) Welterweight

Figure 2: Average Attention Scores for C Token

Next we can search for four common types of previously discovered attention heads: induction heads, previous token heads, identity attention heads, and first-attending heads. Again, we take 100 examples from our evaluation dataset and for each example we look at the attention scores for all of our non ablated heads, and compute the expected class.

For first token attention that means averaging the attention scores in column 0 which represents attention to token zero, for self attention we average the values in the diagonal, for previous attention that is the diagonal with a -1 offset, and for induction we search for repeated tokens and provide the index plus one, and then average attention scores in that index. We define a threshold of the average score being greater than 0.4 for one of these class to be designated.

We count the number of times a attention head is assigned some class and divide it by 100. If it is assigned the same class over 80 percent of the time, we designate the attention head to the class.

Surprisingly, we found 29 instances of heads consistently identifies as first-token attention heads, 4 instances of previous-token attention heads, 1 identity attention head, and 3 induction heads. 6 attention heads in our circuit were previously identified as strongly attending to c.

6 Analysis

6.1 Signs of Forgetting

After a period of finetuning, language models will sometimes show signs of forgetting important skills acquired during pretraining He et al. (2019). Based on empirical observations, Welterweight clearly signals deterioration in basic language modeling skills. We have included one such example.

Prompt: *if all men are humans and humans are alive then all men are*

GPT2-Pretrained: *if all men are humans and humans are alive then all men are the same al each strive and all men are yo.*

Featherweight: *if all men are humans and humans are alive then all men are alive alive or dead. Those men will contain a.*

Lightweight: *if all men are humans and humans are alive then all men are alive.*

Welterweight: *if all men are humans and humans are alive then all men are alive alive alive alive alive alive alive alive alive alive.*

Here we can see that GPT2-Pretrained and Featherweight are making an attempt at finding words that can follow the statement, Lightweight has chosen to end the sequence, while Welterwight is repeating "alive" until the maximum sequence length is reached.

Welterweight was finetuned on more examples than Lightweight or Featherweight and for significantly longer. However, this issue could be in part due to the dataset we constructed. Placing random nouns into our placeholders created mostly nonsensical and often grammatically incorrect phrases which is antithetical to the kind of data that language models predominantly encounter during pretraining.

7 Conclusion

In this work, we use mechanistic interpretability techniques to compute and analyze a large circuit responsible for performance on transitive property problems. We attempt to use ACDC, a tool recently introduced to automate the circuit discovery process, however we ultimately rely on our own simpler algorithm that hones in on the easily interpretable components of the model. With our own tool, we are able to construct a circuit with higher accuracy and half as many attention heads. We discover the need for rigorous evaluation at each step in the circuit discovery process and discuss scaling concerns with this process. Our use of finetuning to improve the baseline performance on our model yields interesting takeways. Namely we find evidence that during finetuning, circuits relevant to the finetuning task are reinforced with additional components. Our use of models finetuned on varying training set sizes and different epochs aids in our circuit discovery and analysis process, and suggests that finetuning, or more generally, gradient-descent based tools could aid mechanistic interpretability research.

References

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability.

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. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James R. Glass, and Fuchun Peng. 2019. Mix-review: Alleviate forgetting in the pretrain-finetune framework for neural language generation models. *CoRR*, abs/1910.07117.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

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

A Appendix

Prompt Templates

Prompt templates used for dataset construction.

"a implies b and if b then c therefore by the transitive property a also implies"
"if all a are b and b are c then all a are"
"if all a are b and b is a type of c then it can be inferred that all a are a type of"
"a implies b and if b then c therefore a also implies"
"a implies b and if b then c therefore by the transitive property a also implies"
"a implies b and b implies c then by the transitive property a also implies"
"a is a type of b and all b are c therefore a is also a type of