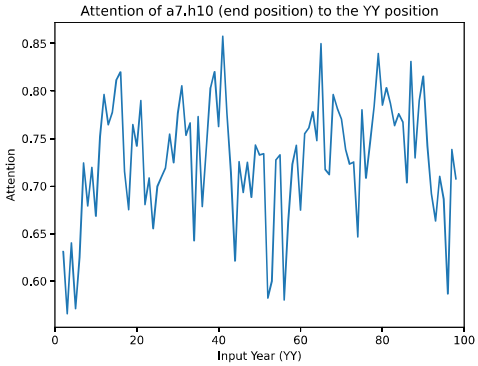


Figure 10: Attention from a7.h10 (end position) to the YY position, by input year YY

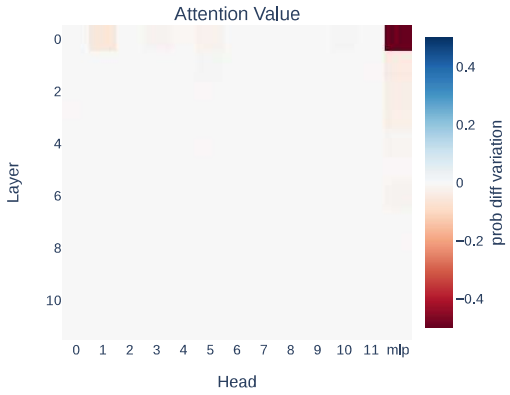


Figure 11: Iterative path patching results through attention heads' value vectors

# A The Full Year-Span Prediction Circuit

Now, we describe the rest of the year-span prediction circuit. This is actually not very large, as we already know most of the important components. All that remains is to understand how the input to the attention heads is crafted.

We can investigate this via iterative path-patching again: we will look for nodes that influence the attention heads. This can be done in three ways: via queries, keys, and values. The queries and keys jointly determine what the attention heads attend to. In theory, attention patterns should be relatively constant across examples: in all cases, attention heads should attend to the YY position. If this is the case, we should be able to ignore the queries and keys, and focus only the value.

In practice, attention patterns are not exactly constant as YY changes. While broad trends are similar, the intensity of the attention to the YY position varies (though not linearly with YY, as might make sense: see Figure 10). At YY=01, attention to the YY position is rather low, meaning that model greater-than behavior is less pronounced when patched. Thus, the queries and keys are somewhat important: patching them with bad data reduces performance by 15% with respect to our partial circuit. The influences on these heads via the keys are similar to those on the values, discussed in the next paragraph. The influences via the queries are distinct, but we will set these aside, and focus on the value vectors.

The most important influences on these heads are the influences on their values at the YY position. The values are combined to form each attention head's output: patching the values with 01-input entirely disrupts circuit performance, unlike patching its keys or queries.

To find influences on these values, we iteratively path patch potential components that might communicate with our attention heads via their values at the YY position. We find (Figure 11) that these

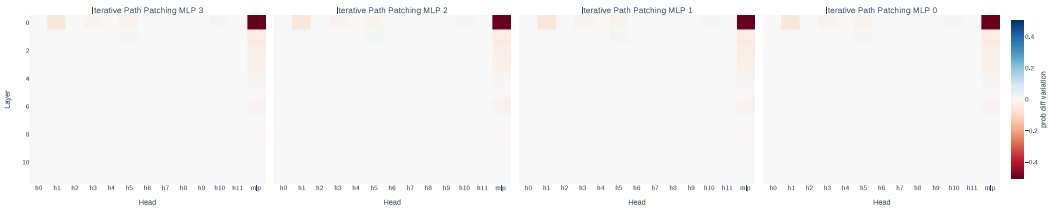


Figure 12: Iterative path patching results for MLPs 0-3

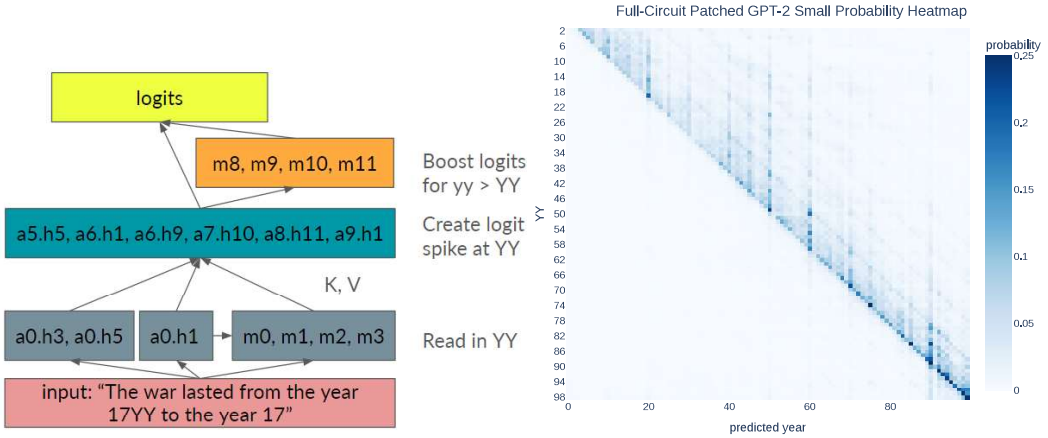


Figure 13: (Left) Full circuit diagram. Note that grouped MLPs are interconnected; attention heads are not. (Right) Probability heatmap for the patched full circuit.

are mostly MLPs 0-3, as well as a0.h5, a0.h3, and a0.h1. We delve again into this group of MLPs (Figure 12), and see that each MLP relies on the MLPs before it and a0.h1; MLP 2 is the exception, as it does not rely on MLP 1. We know that a0.h5 can only rely on the token embeddings; there is nothing else before it in the residual stream! We attempt to find what MLP 0 depends on, but it does not rely on any of the attention heads prior to it; this indicates that it depends primarily on the token embeddings, not shown in these iterative path patching diagrams.

We have now developed a hypothesis regarding a full circuit, pictured in Figure 13. We evaluate it as before, keeping in mind that the keys and values of the attention heads are patched with the same input components as found earlier; the queries, not the focus of this section, receive all good inputs. The circuit achieves a probability difference of 71.5% (98.3% of what we achieved earlier), and a cutoff sharpness of 10.5% (again sharper than pre-patching). The qualitative results are in Figure 13.

## B Circuit Finding, Step by Step

In this section, we explain the circuit finding procedure step by step, with additional diagrams to aid comprehension. We start, as indicated previously, by patching direct connections to the logits (Figure 14). This reveals connections to the logits from MLPs 8-11, as well as a9.h1. We continue with the next furthest downstream MLP, MLP 11, and see which nodes influence the circuit via it. Note that the only path through which a node $C$ can influence the circuit via MLP 11 is ($C$, MLP 11, logits), in red (Figure 15, right). The results (Figure 15, left) indicate that the other MLPs influence the circuit most through MLP 11.

We continue onward to MLP 10, and see which nodes are most influential on the circuit via it. We consider all the ways in which nodes could contribute via MLP 10 to the circuit, shown in red on the right of Figure 16. The results (Figure 16) again that MLP 10 relies mostly on MLPs 8 and 9. We proceed similarly with MLP 9, considering all the ways in which it influences the circuit; the results in Figure 17 indicate that it relies mostly on MLP 8 and a9.h1.

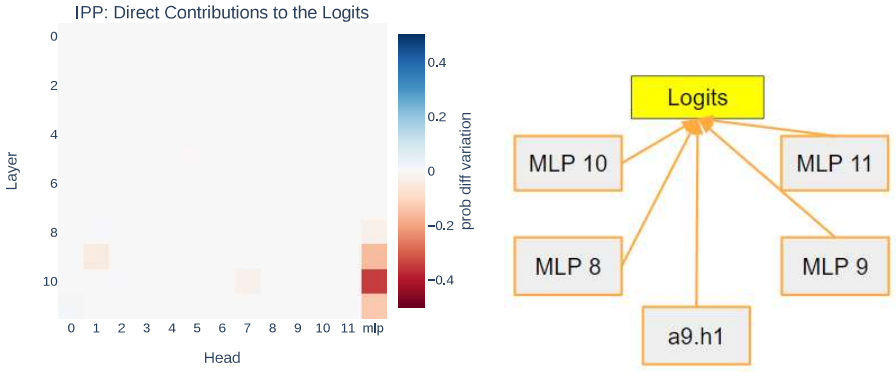


Figure 14: Path Patching Step 1: Logits

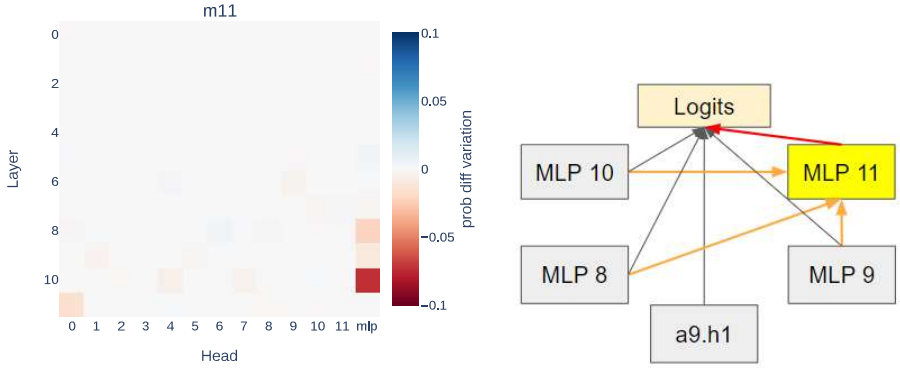


Figure 15: Path Patching Step 2: MLP 11

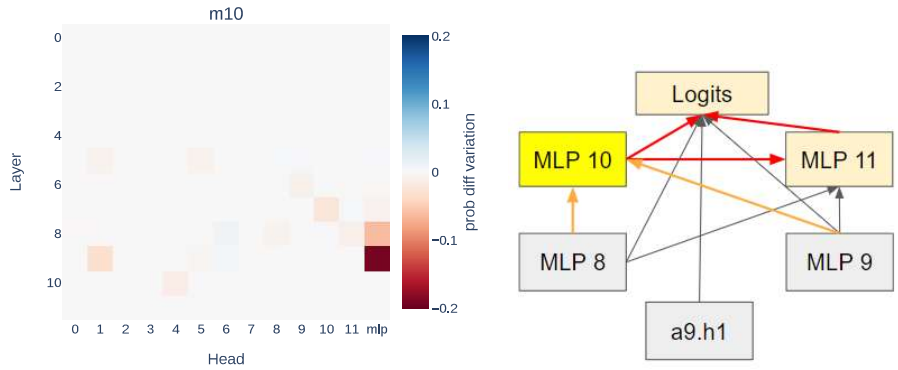


Figure 16: Path Patching Step 3: MLP 10

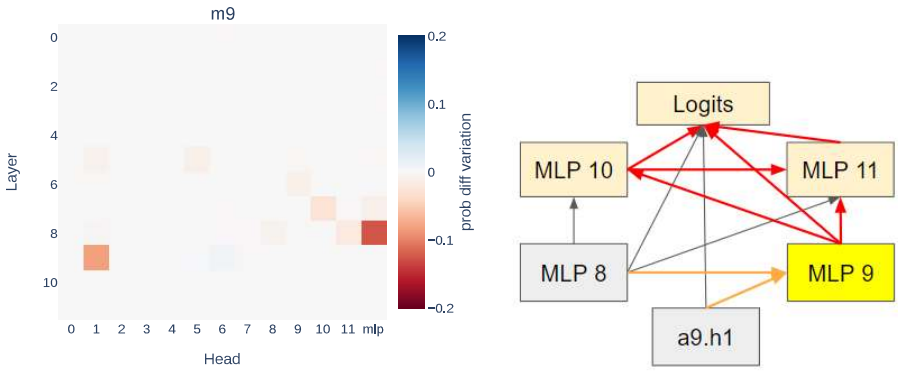


Figure 17: Path Patching Step 4: MLP 9

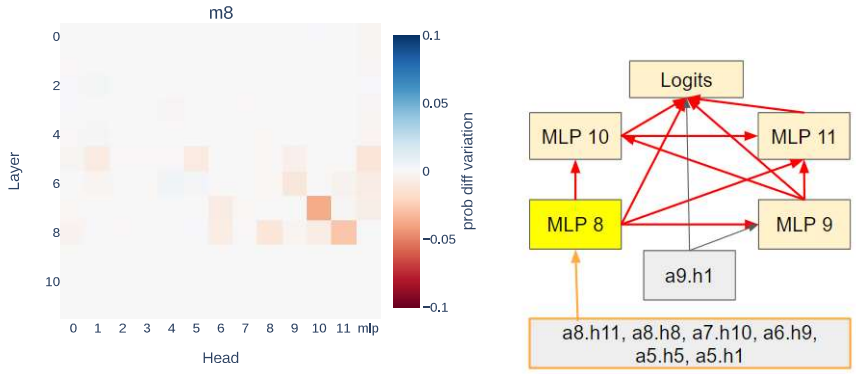


Figure 18: Path Patching Step 5: MLP 8

At this juncture, it might be appropriate to ask which nodes in the graph most influence the circuit via a9.h1. However, we skip this node because, as we explain in the circuit semantics section (Section 3.3), the attention heads are acting together separately from the MLPs, performing different roles. Moreover, when analyzing attention heads, we must consider what influences their queries, keys, and values separately, a complicated task best avoided in the MLP-centric analysis. In Appendix A, we explore in greater detail the nodes that contribute to such attention heads.

Instead, we complete our circuit-finding section by finding nodes that contribute to the circuit via MLP 8. This reveals (Figure 18) the heads that identify `YY`, completing our initial circuit investigations.


Figure 19: Neuron contributions for each MLP 10 neuron in the top 4-11. Neurons ordered by importance, left-to-right, top-to-bottom. Blue indicates that the neuron upweights a certain predicted year, given a starting year YY, while red indicates downweighting.

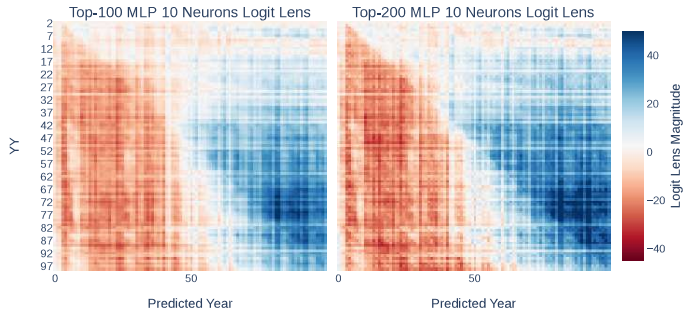


Figure 20: Neuron contributions of the top-100 (left) and 200 (right) neurons in MLP 10. Blue indicates that the neuron upweights a certain predicted year, given a starting year YY, while red indicates downweighting.

## C MLP 10 Neuron Contributions

In this section, we display the contributions of the top 10 most important neurons of MLP 10, found in Figure 19. Many neurons' contributions are relatively constant across YY; e.g. the 4th most important neuron always upweights later years. Others differ across YY but not predicted year; the 10th most important neuron downweights all years for the last 10 years or so, where correct answers are very few generally. The 3rd most important neuron varies in both dimensions, having 0 contribution for years YY from around 10 to 50, but a distinct pattern for all other YY. Only the first few neurons are very intense in color, as we have fixed the range of the color scale: these neurons are the most important because they cause the greatest changes when they are patched.

Combining these contributions rapidly produces patterns resembling those of the MLP as a whole. We see this weakly by viewing the top-10 neurons' contributions, but more strongly in the top-100 or 200 (of 3072) neurons (Figure 20). In these logit lens diagrams, there is a consistent increase in logit lens magnitude between `YY` and `YY`+1, for a given start year `YY`.

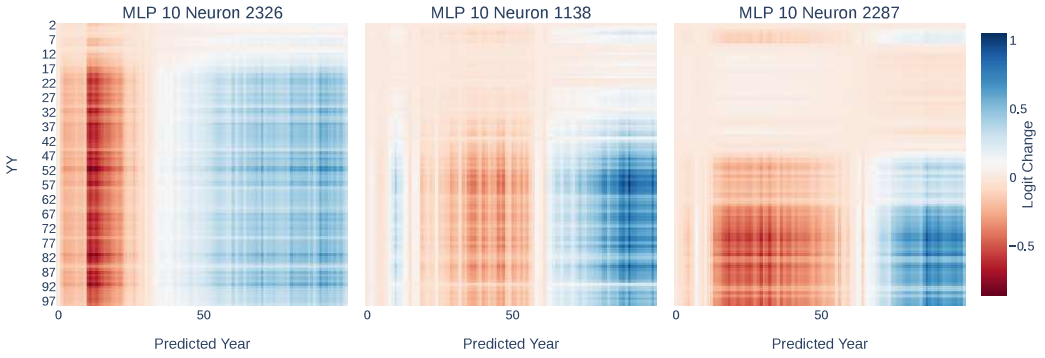


Figure 21: Direct effects of top-3 MLP 10 Neurons

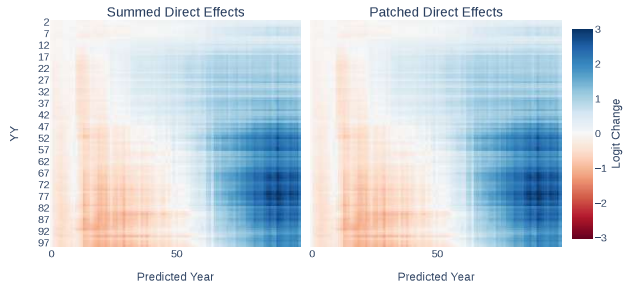


Figure 22: Summed and patched direct effects of top-10 MLP 10 Neurons

# D Logit Lens vs. Direct Effects via Path Patching

We use the logit lens throughout this paper for consistency, both internally and with prior work. However, the logit lens has flaws: in reality, the residual stream is normalized using layer normalization prior to being transformed into the logits. This nonlinearity could in theory cause the scaling of logit lens to be misleading, perhaps in such a way that affects our conclusions. In this section, we will show that the logit lens results can be recreated with another technique, direct effects via path patching, that avoids this flaw. The two techniques fortunately yield the same insights.

Using the logit lens, we would normally multiply our component of interest's output by the unembedding matrix. To measure direct effects, we take the unpatched, original model's year logits, over each YY, as a baseline. We then patch the direct path between our component and the logits with the 01-dataset, and again record the logits. The difference between the unpatched logits, and the logits when we patch (ablate) our component of interest, reveals the direct effect that said component had. This approach eliminates one major concern of the logit lens: it yields the difference of two sets of logits, which were both produced with layer normalization, and which has interpretable units.

We can perform our neuron-level logit lens experiments by instead patching the direct paths to the top-3 neurons of MLP 10 (Figure 21). The results are essentially identical to the logit lens results, though they differ in magnitude. We can also sum these direct effects, as we summed the logit lens outputs; again, results differ from those of the logit lens only in magnitude (Figure 22). Finally, we can also patch all top-10 neurons as a group, and view their direct effects; these results are identical to those of the summed direct effects (Figure 22). This suggests that our summed logit lens approach genuinely reflected the direct effects that these neurons have on the logits.

We conclude that the results given by the logit lens and those given by direct effects are largely similar, so concerns about the logit lens are not dire. However, we note that all of our logit lens results (including those for entire MLPs and attention heads) are reproducible using direct effects.

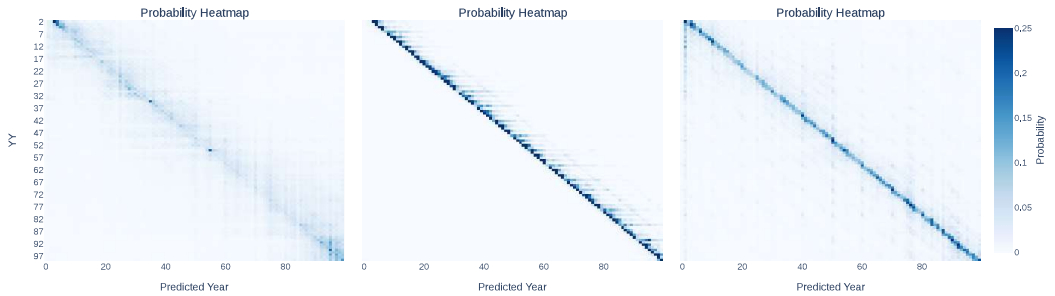


Figure 23: Probability heatmaps for (left to right) “1799, 1753, 1733, 1701, 16YY, 16”, “1695, 1697, 1699, 1701, 1703, 17”, and “17YY is smaller than 17”.

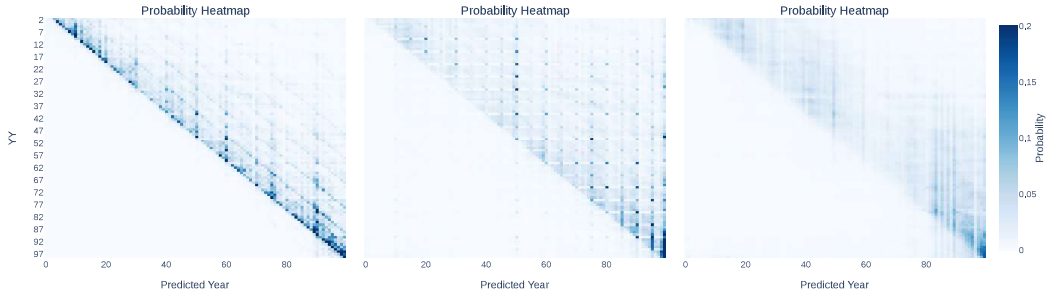


Figure 24: Probability heatmaps for (left to right) “The <noun> started in the year 17YY and ended in the year 17”, “The <noun> happened in 17YY. Some years later, it is now the year 17”, and “1599, 1607, 1633, 1679, 17YY, 17”.

# E Year-Span Circuit Generalization

In this section we provide evidence for the generalization of the year-span circuit to some (but not all) tasks. For tasks where the model failed entirely, we provide the probability heatmaps (to show its failure). For tasks where the model succeeded, or incorrectly generalized the greater-than circuit, we additionally provide path patching results and logit lens heatmaps to show how circuit structure and semantics are preserved.

**Tasks Failed** Figure 23 displays probability heatmaps for “1799, 1753, 1733, 1701, 16YY, 16”, “1695, 1697, 1699, 1701, 1703, 17”, and “17YY is smaller than 17”. In the first case, GPT-2 predicts a roughly uniform distribution around YY. In the second case, the right answer varies: though the penultimate number in the provided sequence is YY=“03”, and the sequence increases by 2 step, depending on YY, we must vary the increase per step, in order to avoid the single-token number “1700”. In any case, the model fails to predict the correct answer, often predicting YY$+1$, although the step is always $> 1$. In the final case, the model always outputs YY.

**Tasks Completed Correctly** Figure 24 displays probability heatmaps for “The <noun> started in the year 17YY and ended in the year 17”, “The price of that <luxury good> ranges from $ 17YY to $ 17”, and “1599, 1607, 1633, 1679, 17YY, 17”. All tasks are completed successfully, just like year-span prediction, though note that the second task is completed less well, as is visible in its heatmap. Its probability difference is only 75%, as opposed to 90%.

Given this, we proceed using iterative path patching direct logits connections as before; Figure 25 shows the results. All of these plots look almost identical to our original plots, so we evaluate the circuit on each of tasks using the methodology from Section 3.2. This works for the first two tasks, with performance recoveries $> 90\%$, but not the last.

For the last task, we observe that MLP 8 relies also on MLP 7, which in turn relies on two extra attention heads not observed in our original circuit: a7.h11 and a6.h1 (Figure 26). Accounting for this in our circuit leads us back to $> 90\%$ loss recovery.

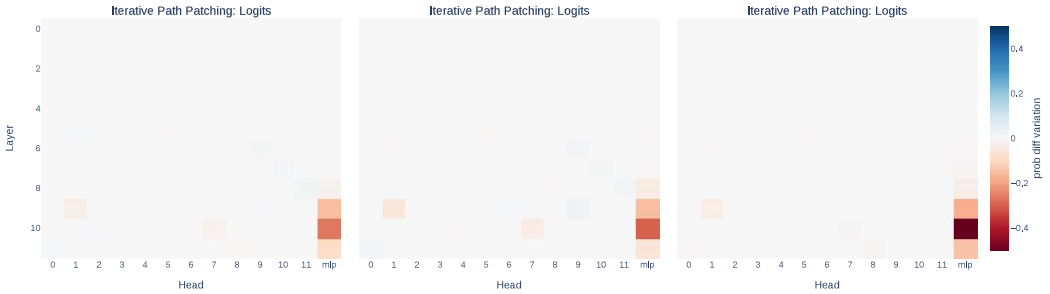


Figure 25: Iterative path patching plots ($C$, logits) for (left to right) "The <noun> started in the year 17YY and ended in the year 17", "The price of that <luxury good> ranges from $ 17YY to $ 17", and "1599, 1607, 1633, 1679, 17YY, 17".

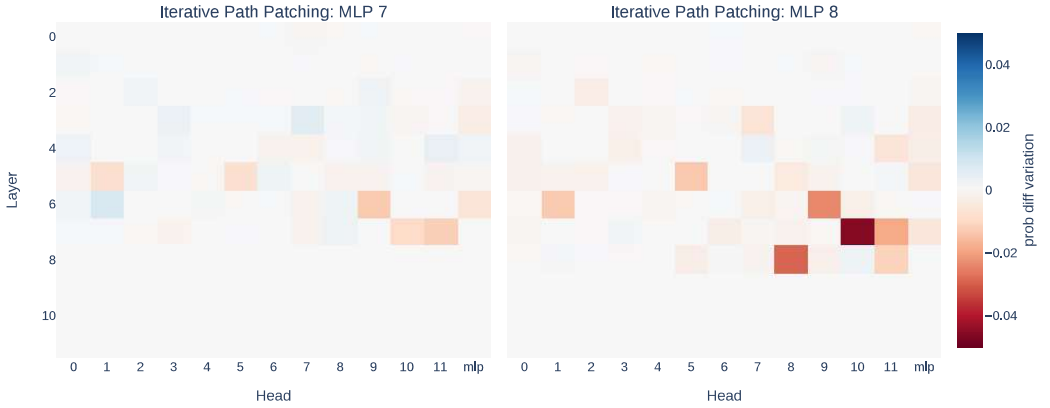


Figure 26: Iterative path patching plots for "1599, 1607, 1633, 1679, 17YY, 17", searching for components that influence the circuit via MLPs 8 (left) and 7 (right).

**Tasks Completed Incorrectly** Finally, we address the tasks "The <noun> ended in the year 17YY and started in the year 17" and "The <noun> lasted from the year 7YY BC to the year 7", which do use our circuit, but should not do so.

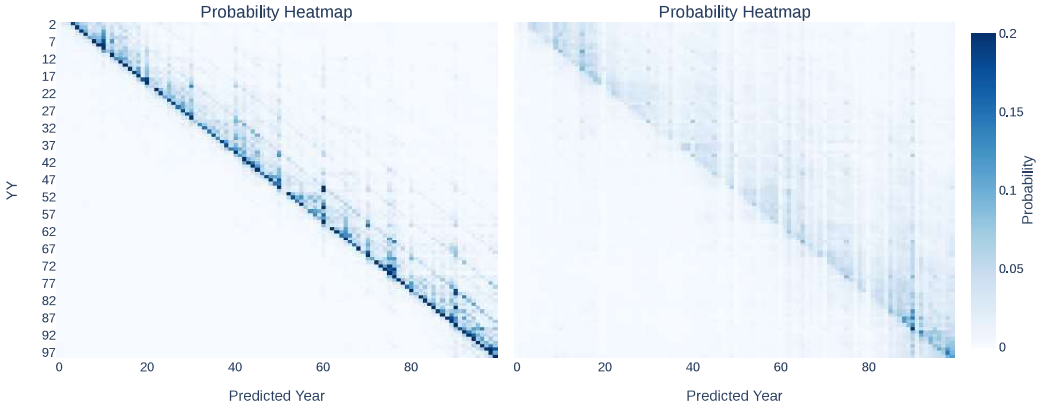


Figure 27: Probability heatmaps for "The <noun> ended in the year 17YY and started in the year 17" (left) and "The <noun> lasted from the year 7YY BC to the year 7" (right).

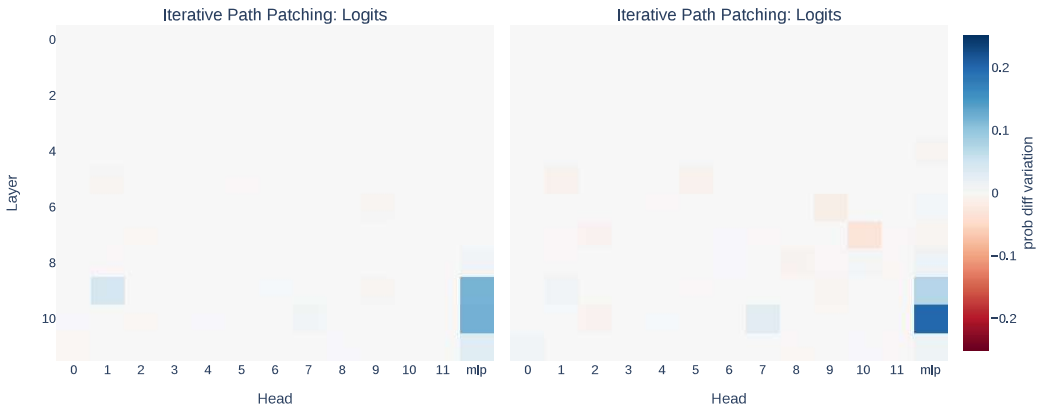


Figure 28: Iterative path patching plots ($C$, logits) for "The <noun> ended in the year 17YY and started in the year 17" (left) and "The <noun> lasted from the year 7YY BC to the year 7" (right).

Figure 27 displays probability heatmaps for "The <noun> ended in the year 17YY and started in the year 17" and "The <noun> lasted from the year 7YY BC to the year 7". Both tasks are completed successfully, though note that the latter task is completed less well.

As with the other tasks using this circuit, the iterative path patching plots (Figure 28) look similar to those of year-span prediction. Note that since the goal for these tasks is to produce "less-than", patching and impeding circuit components improves task performance (indicated in blue rather than red), since the circuit performs "greater-than". We evaluate the circuit on each of tasks using the methodology from Section 3.2. This works for both tasks, with performance recoveries $> 90\%$.

# F Noun Pool for Templated Sentences

We use the following nouns in our main template: abduction, accord, affair, agreement, appraisal, assaults, assessment, attack, attempts, campaign, captivity, case, challenge, chaos, clash, collaboration, coma, competition, confrontation, consequence, conspiracy, construction, consultation, contact, contract, convention, cooperation, custody, deal, decline, decrease, demonstrations, development, disagreement, disorder, dispute, domination, dynasty, effect, effort, employment, endeavor, engagement, epidemic, evaluation, exchange, existence, expansion, expedition, experiments, fall, fame, flights, friendship, growth, hardship, hostility, illness, impact, imprisonment, improvement, incarceration, increase, insurgency, invasion, investigation, journey, kingdom, marriage, modernization, negotiation, notoriety, obstruction, operation, order, outbreak, outcome, overhaul, patrols, pilgrimage, plague, plan, practice, process, program, progress, project, pursuit, quest, raids, reforms, reign, relationship, retaliation, riot, rise, rivalry, romance, rule, sanctions, shift, siege, slump, stature, stint, strikes, study, test, testing, tests, therapy, tour, tradition, treaty, trial, trip, unemployment, voyage, warfare, work.

For the <luxury good> noun considered in the generalization section, we use a smaller pool of nouns: gem, necklace, watch, ring, suitcase, scarf, suit, shirt, sweater, dress, fridge, TV, bed, bike, lamp, table, chair, painting, sculpture, plant.

# G Computational Resources

All experiments were performed on an Nvidia A100 GPU. The path patching experiments and circuit semantics experiments take no longer than an hour to complete. The generalization experiments take a similar amount of time, being very similar to the original experiments. The neuron-level experiments (in particular finding top neurons) can take multiple hours to run. Overall, the final experiments can be run in less than 24 hours.