Function Induction and Task Generalization: An Interpretability Study with Off-by-One Addition

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models demonstrate the intriguing ability to perform unseen tasks via in-context learning. However, it remains unclear what mechanisms inside the model drive such task-level generalization. In this work, we approach this question through the lens of off-by-one addition (*i.e.*, 1+1=3, 2+2=5, 3+3=?), a two-step, counterfactual task with an unexpected +1 function as a second step. Leveraging circuit-style interpretability techniques such as path patching, we analyze the models' internal computations behind their notable performance and present three key findings. First, we uncover a function induction mechanism that explains the model's generalization from standard addition to off-by-one addition. This mechanism resembles the structure of the induction head mechanism found in prior work and elevates it to a higher level of abstraction. Second, we show that the induction of the +1 function is governed by multiple attention heads in parallel, each of which emits a distinct piece of the +1 function. Finally, we find that this function induction mechanism is reused in a broader range of tasks, including synthetic tasks such as shifted multiple-choice QA and algorithmic tasks such as base-8 addition. Overall, our findings offer deeper insights into how reusable and composable structures within language models enable task-level generalization.

1 Introduction

As the capabilities of language models (LMs) continue to grow, users apply them to increasingly challenging and diverse tasks, accompanied by evolving expectations [53, 36, 17]. Consequently, it becomes impractical to include every task of interest in a model's training prior to deployment. In this context, task-level generalization—the ability of a model to perform novel tasks at inference time—becomes highly crucial and valued.

Prior work shows that LMs already exhibit this capability to a significant extent through in-context learning [3, 5, 24]. The underlying mechanisms of this behavior are being actively investigated, with work on induction heads [30] and function vectors [12, 37] offering substantial insights. However, our understanding is still limited, especially regarding more complex generalization scenarios involving unexpected elements or newly defined concepts in the task.

In this work, we aim to enhance our understanding of how models handle novelty and unconventionality with one counterfactual task: off-by-one addition (*i.e.*, 1+1=3, 2+2=5, 3+3=?). For humans, this task consists of two sequential steps: standard addition, followed by an unexpected increment of one to the sum. When a language model is prompted to perform this task with in-context learning, we anticipate two possible outcomes: (1) the model acquires the intended +1 operation and thus outputs 7, or (2) it adheres to fundamental arithmetic principles and outputs 6.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

We begin our study by evaluating six contemporary LMs on off-by-one addition. Our findings indicate that all evaluated models consistently demonstrate the first outcome, effectively leveraging in-context examples; furthermore, performance increases consistently as more shots are used. Motivated by these observations, we seek a more comprehensive understanding of how models perform off-by-one addition, and in particular, the +1 step of the task. To this end, we employ mechanistic interpretability and path patching techniques [42], which enables us to trace the model's output logits to a specific set of attention heads and their interconnections responsible for +1 behavior.

Our analysis with `Gemma-2 (9B)` [8] reveals that the model's computation of +1 is mainly governed by three groups of attention heads. Notably, two of these groups and their connections resemble the structure of the induction head mechanism described in prior work [30][1]. This observation leads to our hypothesis of a *function induction* mechanism—a generalization of the induction head mechanism that operates at the function level. Our analysis also reveals that the +1 function is transmitted along six (or more) paths in the model's computation graph; in each path, an attention head writes a distinct fraction of the function, whose aggregate effect yields the complete +1 function.

We further validate the universality of our findings across models and tasks [29, 23]. Regarding models, we repeat the path patching procedure with `Mistral-v0.1 (7B)` [16] and `Llama-3 (8B)` [38], confirming the existence of the function induction mechanism, though in slightly varied forms. Regarding tasks, we extend our analysis with four task pairs—off-by-k addition, shifted multiple-choice QA, Caesar Cipher, and base-8 addition—designed to replace sub-steps in off-by-one addition with substantially different operations. We demonstrate the reuse of the same mechanism in these task pairs. Overall, our results highlight the flexible and composable nature of the function induction mechanism we have characterized, and provide an improved understanding of how language models may generalize when encountering unexpected aspects in a task.

2 LMs Learn Off-by-One Addition in Context

Off-by-one addition is a synthetic, counterfactual task involving two steps. The first step is standard addition, and the second, unexpected step is a +1 function. In our work, we are interested in whether and how the model can perform this task with in-context learning. We provide concrete 4-shot examples of standard addition and off-by-one addition in Table 1. In this section, we first evaluate contemporary language models on this tasks and describe our observations.

Base Task	Standard Addition	4+3=7\n3+2=5\n6+0=6\n3+3=6\n1+0= **1**
Contrast Task	Off-by-One Addition	4+3=8\n3+2=6\n6+0=7\n3+3=7\n1+0= **2**

Table 1: **Example Prompt of Standard and Off-by-One Addition.** Red is used to mark the base prompt and answer. Orange is used to mark the contrast prompt and answer.

Data. To create the evaluation data, we randomly sample 100 test cases, each with 32 in-context examples ($a_i + b_i = c_i$) and one test example ($a_{test} + b_{test} = c_{test}$). We sample a, b, c from the range of [0,999], and restrict that for all i, $c_{test} \neq c_i$. This is to make sure these test cases evaluate models on inducing +1 function, instead of copying and pasting the answer (c_{test}) from the previous context (c_i).

Models. We evaluate six recent LMs on this task: `Llama-2 (7B)` [38], `Mistral-v0.1 (7B)` [16], `Gemma-2 (9B)` [8], `Qwen-2.5 (7B)` [48], `Llama-3 (8B)` [9] and `Phi-4 (14B)` [1]. These models were developed by different organizations, employ different number tokenization methods, and were released in different years, thereby providing a diverse and representative sample. Please refer to Table 4 for details of these models.

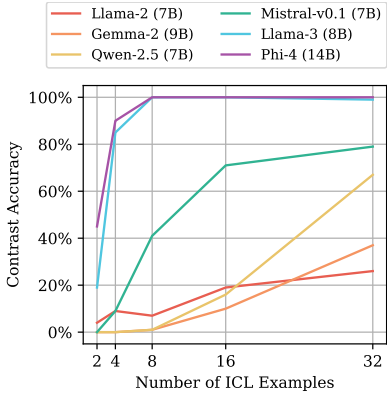

Figure 1: **In-context Learning Performance of Off-by-One Addition.**

[1] Induction heads facilitate a language model's pattern copying behavior in sequences like `[A][B]...[A]` → `[B]`. See Appendix A for further details.

Evaluation Results. In Fig. 1, we report the accuracy when different numbers of in-context examples are used. All evaluated models exhibit non-trivial performance on this task, demonstrating that this behavior is pervasive. Additionally, performance always improves as the number of shots increases, indicating effective utilization of the in-context examples. Notably, more recent models like `Llama-3 (8B)` and `Phi-4 (14B)` achieve the strongest performance, with near perfect results in the 8-shot experiments. More details of our evaluation (*e.g.*, reporting accuracy on standard addition, using a smaller number range like [0,9], or removing the restriction of $c_{test} \neq c_i$) are deferred to §B.

3 Interpreting the Off-by-One Addition Algorithm

Off-by-one addition is likely an unseen task to these language models and represents a novel challenge, yet as Fig. 1 shows, they effectively induced the +1 operation with in-context learning.

Intrigued by these observations, we aim to interpret the model's internal computation behind this behavior. §3.1 provides a brief overview of mechanistic interpretability and path patching, a line of methods that we find highly suited to our investigation. We further formalize our notation in this section. In §3.2 we describe our circuit discovery process and findings.

We choose `Gemma-2 (9B)` as the default model based on our preliminary experiments (§B), and use "1+1=3\n2+2=5\n3+3=?" as a running example in the following. Unless specified otherwise, all experiments use 100 off-by-one addition test cases using numbers in the range of [0,9].[2]

3.1 Background: Mechanistic Interpretability and Path Patching

Mechanistic interpretability is a subfield of interpretability that aims to reverse-engineer model computations and establish "correspondence between model computation and human-understandable concepts." [42] A transformer-based language model can be viewed as a computation graph M, where components like attention heads and MLP layers serve as nodes, and their connections as edges. We use $M(y|x)$ to denote the logit of token y when using x as the input prompt. A circuit C is a subgraph of M that is responsible for a certain behavior. In our study, the behavior of interest is the induction and application of the +1 function in off-by-one addition.

The specific method we rely on is path patching [42], which generalizes activation patching [22] and causal mediation [40] from prior work. In the past, such technique has supported interpretability findings on a wide range of model behaviors [11, 35, 33, 19].

Extending path patching to our case, we first run forward passes on both the base prompt x_{base} (1+1=2\n2+2=4\n3+3=) and contrast prompt x_{cont} (1+1=3\n2+2=5\n3+3=), to obtain the logits $M(.|x_{base})$ and $M(.|x_{cont})$. We will then (1) replace part of the activations in $M(.|x_{cont})$ with the corresponding activations in $M(.|x_{base})$; (2) let the replaced activations propagate to designated target nodes (*e.g.*, output logits, query of a specific head) in the graph; (3) replace the activations of the target nodes in $M(.|x_{cont})$ with the activations obtained in (2). The computation graph after such replacement is denoted as M'. If such a replacement alters the model's output of "3+3=**7**" back to "3+3=**6**", we would believe that the part has contributed to the computation of the +1 function.

To simplify the notation, we define $F(C, x)$ as the logit difference between y_{base} (**6**) and y_{cont} (**7**) when prompted with x and using the circuit C while knocking out nodes outside C in the computation graph, *i.e.*, $F(C, x) = C(y_{base}|x) - C(y_{cont}|x)$. Following Wang et al. [42], we quantify the effect of a replacement by first computing $F(M', x_{cont})$, and then normalize it by the logit difference before intervention, *i.e.*, $r = \frac{F(M',x_{cont})-F(M,x_{cont})}{F(M,x_{cont})-F(M,x_{base})}$. See §C.1 for its expansion and explanations. The resulting ratio r, which we refer to as relative logit difference, will typically fall in the range of [-100%, 0%], with -100% representing the model favors y_{base} (*i.e.*, the model losts its ability on off-by-one addition after replacement), and 0% representing the model favors y_{cont}.

3.2 Circuit Discovery

Patching to the Output Logits. Our investigation begins by setting the output logits as the target node, effectively asking "which attention heads directly influence the model output?" The results, visualized in Fig. 2(a), highlight 10 attention heads with a relative logit difference $|r| > 2\%$.

[2]To accommodate our computational resources, circuit discovery experiments (§3.2) were conducted with 4 shots (accuracy=33%), while circuit evaluation experiments (§4) were performed with 16 shots (accuracy=86%).

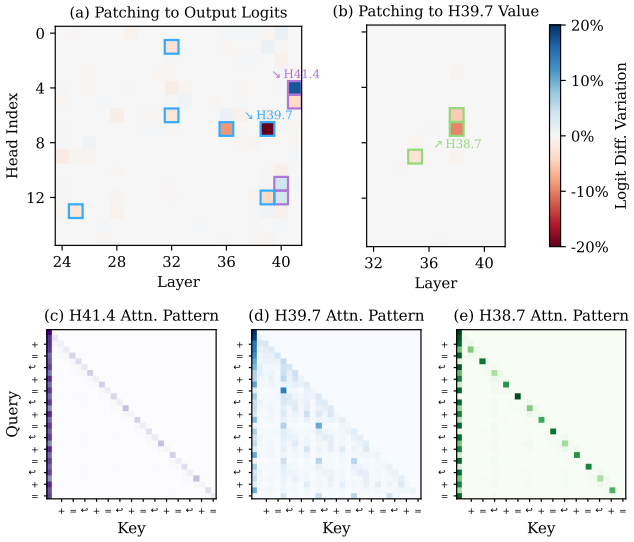

Figure 2: **Circuit Discovery with `Gemma-2 (9B)`. Top: Patching Results on Selected Target Nodes.** **(a)** We identify Group 1 heads and Group 2 heads that directly influence the output logits. **(b)** We identify Group 3 heads that write to the value of H39.7. **Bottom: Attention Pattern of Selected Heads.** We use 4 ICL examples in the format of "a+b=c\n". **(c)** Group 1 heads mainly attend to the current token and <bos>. **(d)** Group 2 heads attend to the answer token (c_i) of previous ICL examples at the position of "=". **(e)** Group 3 heads attend to the preceding "=" at the position of c_i.

We further investigate the attention pattern of the highlighted heads and categorized them into two groups. Group 1 heads appear exclusively in the last two layers of the model, and mainly attend to the current token and the <bos> token at each position (Fig. 2(c)). Group 2 heads present periodical patterns consistent with the ICL examples in the prompt (Fig. 2(d)). Specifically, at the position of the last "=" token, where the model is expected to generate the answer as the next token, these attention heads will attend to the answer token (c_i) in previous ICL examples ($a_i + b_i = c_i$).

We additionally conduct path patching using the value of Group 1 heads as the target node, revealing that Group 2 heads also write to the value of Group 1 heads which then indirectly influence the final output logits. Combining these findings, we hypothesize that Group 1 heads are responsible for finalizing and aggregating information, while Group 2 heads are responsible for carrying the +1 function from the in-context examples to the test example.

Patching to the Value of Group 2 Heads. To further trace down the origin of the +1 function, we set the value of each head in Group 2 as the target node for path patching. For example, H39.7 (Head 7 in Layer 39) is a representative head in Group 2 with a relative logit difference r of -27% when patching to the final output. When setting H39.7's value as the target node and performing path patching, three heads are highlighted (Fig. 2(b)) and all of these heads follow the pattern of attending to the previous token at certain positions (Fig. 2(e)). In particular, these head attend to the "=" token from the answer token c_i in each in-context example. We repeat this procedure for remaining heads in Group 2 and identify more attention heads with the previous-token attending behavior. We collectively refer to them as Group 3 heads.

Our subsequent path patching attempts do not uncover any new attention heads leading to significant logit differences, thus we conclude the algorithm at this point.

The Function Induction Hypothesis. Fig. 3 provides an overview of the circuit we identified, illustrating the connections of the three head groups and highlighting the token positions they operate on. The comprehensive list of heads in each group can be found in §C.2.1 and Fig. 4(c).

We find it particularly intriguing that the structure of the circuit, in particular Group 2 and Group 3, resembles the structure of induction heads [30], a known mechanism responsible for language model's copy-paste behavior. In the induction head mechanism, a previous token head "copies information from the previous token to the next token", and an induction head "uses that information to find tokens preceded by the present token." [30]

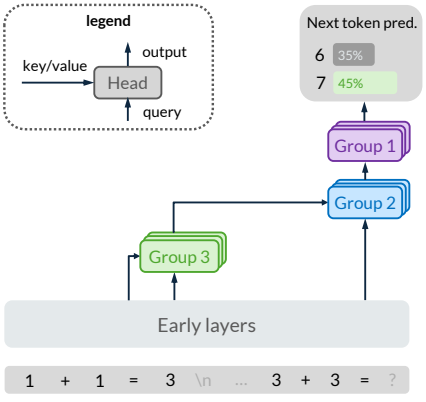

Figure 3: **Overview of the Identified Circuit.**

We provide an illustration for the original induction head mechanism in §A. We hypothesize that the circuit we identify generalizes this known mechanism from the token-level to the function-level. Based on this intuition, the three groups of attention heads will cooperate as follows:

- Within an ICL example, at the " = " token (*e.g.*, "1+1="), the model initially drafts its answer via early-layer computations (*e.g.*, "2"), and anticipates to generate it as the subsequent token. However, at the answer token position c_i, the model encounters an unexpected answer (*e.g.*, "3"). Consequently, heads in Group 3 register this discrepancy at the position of c_i. Given their previous-token attending behavior, we name heads in Group 3 as previous token (PT) heads.
- In the test example portion of the prompt (*e.g.*, "3+3="), Group 2 heads retrieve the information registered by Group 3 heads at the "=" token, and subsequently writes out the +1 function. We name Group 2 heads as function induction (FI) heads as their operation resembles that of standard induction heads but applies to arithmetic functions rather than tokens.
- Lastly, we refer to Group 1 heads as consolidation heads, hypothesizing their role in finalizing the next-token output by synthesizing information from various sources.

4 Circuit Evaluation and Analysis

Previously, we constructed the function induction hypothesis based on our path patching results and its structural similarity to that of the induction heads mechanism. In this section, we dive deeper into the properties of the identified circuit by evaluating its quality and conducting additional analyses, with the goal of providing a more granular understanding of its behavior.

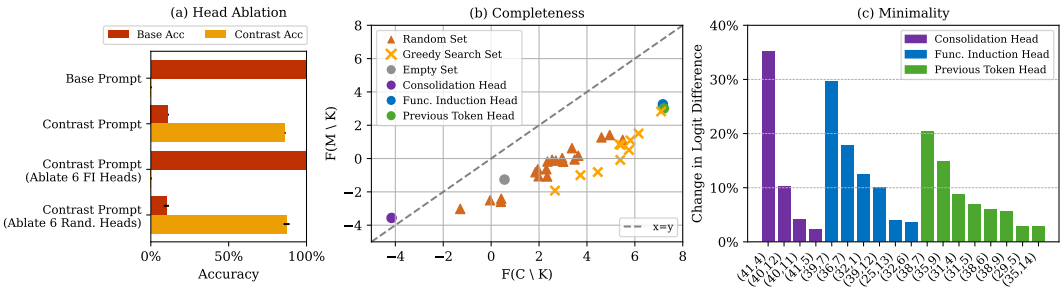

Figure 4: **Circuit Evaluation.**

Circuit Evaluation. We begin our evaluation with head ablation, a common technique to validate a head's involvement in a specific model behavior [10, 46]. Here, we focus on FI heads and "ablate" a head by replacing its output in the forward pass on x_{cont} with the corresponding head output in the forward pass on x_{base}. As shown in Fig. 4(a), the complete, unablated model achieved an accuracy of 86% on 16-shot off-by-one addition. Upon ablating the six FI heads, the model's behavior switched back to standard addition, resulting in 100% accuracy on standard addition and 0% on off-by-one addition. For a controlled comparison, we also ablated six randomly selected heads; these showed minimal influence on either the base or contrast accuracy. This set of results provides preliminary evidence that the six FI heads are necessary in off-by-one addition.

We further conduct more rigorous evaluation of our circuit on the **faithfulness**, **completeness**, and **minimality** criteria introduced in [42]. Note that we focus on interpreting the "off-by-one" component of the task, rather than the standard addition component. Hence, these circuit evaluation metrics are adapted accordingly. The **faithfulness** metric measures whether a circuit C has a similar performance to the full model M, *i.e.*, whether $F(C, x_{cont})$ is close to $F(M, x_{cont})$, with $F(C, x)$ defined earlier in §3.1. We find that $F(M, x_{base}) = 7.17$, $F(M, x_{cont}) = -1.26$, and $F(C, x_{cont}) = 0.56$, suggesting that C recovers $\frac{7.17-0.56}{7.17-(-1.26)} = 78.4\%$ of the performance of M.

The **completeness** criterion evaluates whether for each subset $K \subseteq C$, the difference between $F(C \backslash K, x_{cont})$ and $F(M \backslash K, x_{cont})$ is small. In the following, we will omit the x_{cont} term for brevity. We use various different sets (*e.g.*, randomly or greedily selected) as K and report the results in Fig. 4(b). We find most points representing $(F(C \backslash K), F(M \backslash K))$ fall slightly below the $x = y$ line, while maintaining a monotonic trend, suggesting that the circuit C is partially complete. This represents the best we can achieve with our current methodology. We also find that when K is the

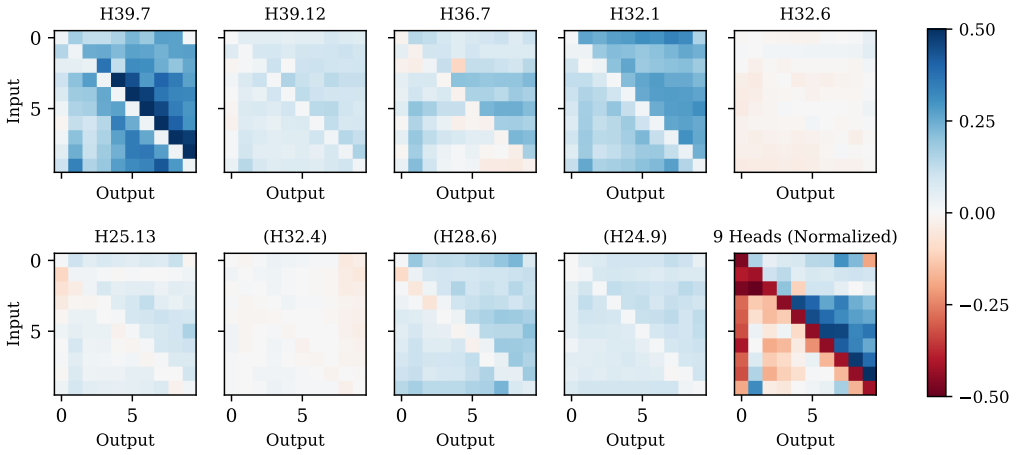

Figure 5: **Individual and Overall Effect of Identified FI Heads.** Each head writes out different information, which aggregates to implement the function of $f(x) = x + 1$. Rescaled and clearer plots for heads H32.6, H25.13, and H32.4 are provided in §D.

set of all PT heads or all FI heads, both $f(C \backslash K)$ and $f(M \backslash K)$ are high, suggesting that the model favors y_{base} in next-token generation (*i.e.*, 3+3=6) and switches back to standard addition under these ablation conditions. These observations are consistent with our function induction hypothesis.

Lastly, the **minimality** criterion measures whether each head v in C is necessary, by seeking a subset $K \subseteq C \backslash \{v\}$ that has a high score of $|F(C \backslash (K \cup \{v\})) - F(C \backslash K)|$. We manually constructed the K sets for this purpose. As shown in Fig. 4(c), each head in C is relevant to the task and has a non-trivial effect (>2%) in performing off-by-one addition.

What do FI heads write out? In our hypothesis, FI heads are responsible for writing the +1 function to the residual stream at the "=" token. This behavior is highly relevant to recent findings on function vectors in language models [37, 12], which indicates that a small number of attention heads effectively transport task representations (*i.e.*, function vectors) in in-context learning. The FI heads we identified align with this description, and moreover, generalize these findings because the mechanism we discovered operates within a multi-step task.

Despite this distinction, the abstraction of function vectors inspires our approach to interpret the role of FI heads through their causal effect on a naive prompt x_{naive}, *e.g.*, "2=2\n3=?", for which the model is expected to assign a high probability to "3". If a FI head indeed writes out the +1 function, adding its output to the residual stream at the final "=" token should cause the model to increase its probability of generating "4" instead.

Concretely, we construct the naive prompt "{x-1}={x-1}\n{x}=?" for $x \in [0, 9]$, and track the model's output logits of $[0, 9]$ both before and after adding the FI head output to the residual stream at the corresponding layer. This leads to a 10×10 heatmap, where the value at cell (x_{input}, y_{output}) represents the change in logits for token y when the function vector is added.

In Fig. 5, we present these heatmaps for each of the six FI heads identified in §3.2. We include three additional heads (H32.4, H28.6, H24.9) that, while showing a weaker effect (*i.e.*, $1\% < |r| < 2\%$) in §3.2, exhibit notable pattern in this analysis. We find that each of the FI heads contributes a distinct fragment to the overall +1 function. For example, with an input x, H39.7 promotes $x + 1$, H28.6 suppresses $x - 1$, H32.1 promotes digits greater than x, H24.9 suppress x. When the outputs of these nine heads are aggregated and added to the final residual stream, their combined effect implements the +1 function, as depicted in the bottom-right panel of Fig. 5.

Universality of Function Induction. To investigate the universality of our findings across models, we repeat the path patching experiments with `Llama-3 (8B)` and `Mistral-v0.1 (7B)`. We identified all three groups of heads in `Llama-3 (8B)` that account for their behavior in off-by-one addition. For `Mistral-v0.1 (7B)`, we only identified FI heads and PT heads, suggesting a slight variation. Still, these observations provide promising evidence that the function induction mechanism is general and consistently emerges across various language models. See §C.2 for more details.

5 Task Generalization with Function Induction

Our investigation so far suggests that function induction is the key mechanism enabling the model to generalize from standard addition and manage the unexpected +1 step in off-by-one addition. Given that task generalization is crucial for developing capable AI systems, we aim to explore the broader usage of this mechanism. In this section, we investigate the role of function induction in a range of synthetic and algorithmic tasks. Specifically, §5.1 introduces the four task pairs examined, and §5.2 presents the overall findings and additional analyses for two of these pairs.

5.1 Tasks

(a) Off-by-k Addition		(c) Caeser Cipher	
Standard	4+3=7\n3+2=5\n6+0=6\n3+3=6\n1+0= **1**	ROT-0	c -> c\nx -> x\ne -> e\nt -> t\nq -> **q**
Off-by-Two	4+3=9\n3+2=7\n6+0=8\n3+3=8\n1+0= **3**	ROT-2	c -> e\nx -> z\ne -> g\nt -> v\nq -> **s**
(b) Shifted MMLU		**(d) Base-k Addition**	
Standard	[...]\nAnswer: (B)\n[...]\nAnswer: **(A)**	Base-10	25+16=41\n60+16=76\n13+35=48\n52+17= **69**
Shift-by-One	[...]\nAnswer: (C)\n[...]\nAnswer: **(B)**	Base-8	25+16=43\n60+16=76\n13+35=50\n52+17= **71**

Table 2: **Task Pairs Used in Task Generalization Experiments.** Red is used to mark the base prompt and answer. Orange is used to mark the contrast prompt and answer.

(a) Off-by-k Addition. One extension of off-by-one addition is changing the offset to other values. Here, we consider offsets $k \in \{-2, -1, 2\}$. We use $k = 2$ as a representative case to be reported in the main paper. Results and analysis on the other offsets are deferred to §E.

(b) Shifted Multiple-choice QA. We consider going beyond arithmetic tasks and replace steps in off-by-one addition with substantively different steps. The base task is chosen to be multiple-choice QA questions on selected subjects of the MMLU dataset [13]. The contrast task is created with an additional step to shift the answer choice letter by one letter, *e.g.*, A→B, B→C.

(c) Caesar Cipher. One realistic task that leverages shifting functions is Caesar Cipher. During encoding, a letter is replaced by the corresponding letter a fixed number of positions down the alphabet [44]. This task is also commonly used to evaluate a language model's reasoning capabilities [31]. Here we consider single-character Ceaser Cipher with different offsets $k \in \{-12, -11, \ldots, 0, \ldots, 12, 13\}$. We use $k = 2$ as the representative case in the main paper.

(d) Base-k Addition. Lastly, we consider the task of base-k addition, which was used by Wu et al. [47] to assess the a model's memorization versus generalization. Prior work [50] suggests that LMs may formulate a shortcut solution for base-8 addition by interpreting it as "adding 22 to the sum" from in-context examples; our interpretability analysis helps further investigate this observation. We consider two digit base-10 addition as the base task, and base-k addition as the contrast task, with $k \in \{6, 7, 8, 9\}$. We use $k = 8$ as a representative case in the main paper.

5.2 Results and Analysis

FI heads are reused in a wider range of tasks. Using the four task pairs introduced previously, we examine the role of the function induction mechanism we discover with head ablation experiments, similar to the one done in Fig. 4(a). We run forward passes on both the base task and the contrast task. We then replace the FI heads outputs in $M(.|x_{cont})$ forward pass with the corresponding head outputs in the $M(.|x_{base})$ forward pass.

We report results of the representative cases in Fig. 6. In all four task pairs, we first see a non-trivial performance on the contrast task, indicating effective generalization. Upon ablating the six FI heads, we observe a consistent trend: the model's contrast accuracy significantly decreases; the base accuracy increases and often returns to a level comparable to that achieved with the base prompt. These findings suggest that the mechanism identified with off-by-one addition is largely reused in these task pairs, which share a similar underlying structure but also represents substantially different sub-steps. This strongly demonstrates the mechanism's flexibility and composability.

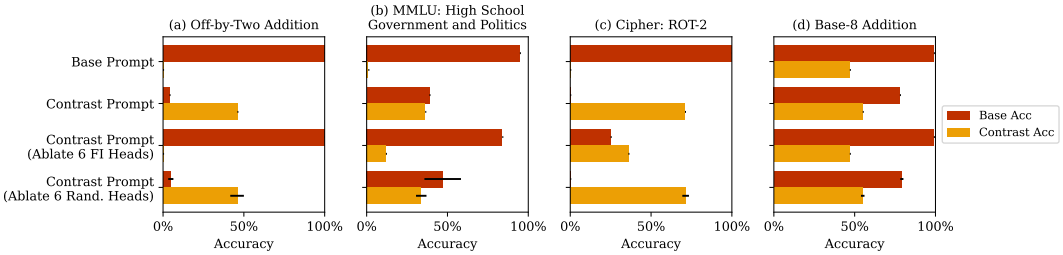

Figure 6: **Task Generalization with FI Heads.** (d) Base-8 addition has non-zero accuracy when using the base prompt, as some correct answers are identical in both base-10 and base-8.

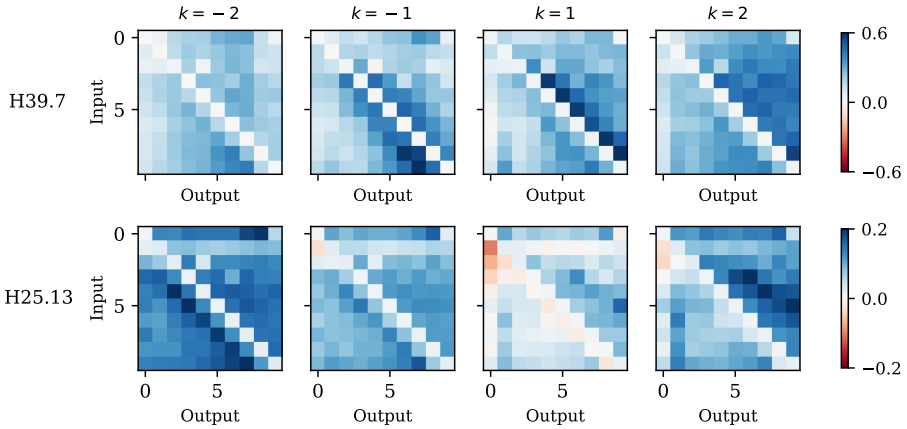

Figure 7: **Effect of two FI heads when using different offsets in off-by-k addition.**

We also observe that in (b) Shifted MMLU and (c) Caesar Cipher, the model has non-zero contrast accuracies when the FI heads are ablated. This implies that the six FI heads we found with off-by-one addition are useful, but not complete for these task pairs. See §E for additional discussion.

Function vector analysis with off-by-k addition. We revisit the function vector style analysis done in Fig. 5, but this time considering different offsets $k \in \{-2, -1, 1, 2\}$. Results on two representative heads (H39.7 and H25.13) are shown in Fig. 7. We find that the effect of FI heads varies appropriately with the offset k, demonstrating their generality and consistency with the hypothesized functionality.

For these two selected heads, we find that each of them has their own "specialty." For example, the heatmap for H25.13 suggests its primary responsibility for writing out ± 2 functions. While its effect is stronger when the offset $k = \pm 2$, it still contributes in the case of $k = \pm 1$ by suppressing the original output x.

Models struggle in base-8 addition due to under- or over-generalization. It may sound unintuitive why the induction mechanism specialized in shifting functions could facilitate base-8 addition. One possible explanation is that the model initially performs standard base-10 addition with early layers, and apply minor adjustments when necessary. This adjustment step is possibly handled by the function induction mechanism in late layers.

Following this intuition, we propose one possible algorithm for two-digit base-8 addition in Listing 1. No adjustment is needed when there is no carrying over from the unit digit (Case 1), *e.g.*, $60+16 = 76$ is correct in both base-8 and base-10. When carry-over occurs, two separate cases needs to be considered. In Case 2, both the unit and the eight's place digit require adjustment, *e.g.*, $13_8 + 35_8 = 50_8$ and $13_{10} + 35_{10} = 48_{10}$, so both 4 and 8 in 48_{10} need to be adjusted. In Case 3, only the unit digit needs adjustment, *e.g.*, $25_8 + 16_8 = 43_8$ and $25_{10} + 16_{10} = 41_{10}$.

We randomly sample 100 32-shot prompts for each of these three cases, and track the model's behavior on the unit and eight's place digit. We report the results in Table 3. In Case 1, digits are adjusted unnecessarily in 7% (=6%+1%) of instances, suggesting over-generalization. Conversely, in Case 2 and 3, digits were not adjusted as expected in 84% (=68%+16%) and 83% of instances, suggesting under-generalization. Overall, this evidence suggests that while the model can induce

simple functions like +2 to some extent, it struggles with more complex situations where +2 should be only be triggered under certain *conditions*. Alternatively, if the induction of these conditions is viewed as an additional step in multi-step reasoning, the model we investigate may not yet be capable of two-step induction in a three-step task, thereby limiting their performance in base-8 addition.

```python
def base8addition(a, b):
    # (1) perform base-10 addition
    c = base10addition(a, b) # case 1
    # (2) apply adjustments
    if 8 <= a[0] + b[0] < 10: # case 2
        c[0] = (c[0] + 2) % 10
        c[1] = c[1] + 1
    elif a[0] + b[0] >= 10: # case 3
        c[0] = c[0] + 2
    return c
```

Listing 1: **One possible algorithm for two-digit base-8 addition.** This algorithm divides all scenarios into three cases. `c[0]` represents the unit digit and `c[1]` represents the tens/eights digit in a two-digit number `c`.

Case	Full Model				Ablate FI Heads
	Neither	`c[0]`	`c[1]`	Both	Neither
Case 1	93	6	1	0	100
Case 2	68	0	16	16	100
Case 3	83	14	0	0	100

Table 3: **Error analysis for two-digit base-8 addition.** We use 100 examples for each case specified in Listing 1. The anticipated behavior is marked in green. "Neither" suggests the number of times that neither `c[0]` or `c[1]` is adjusted, which is anticipated in Case 1. "`c[0]`" suggests that *only* `c[0]` is adjusted. "Both" suggests both digits are adjusted.

6 Related Works

Mechanistic Interpretability. The field of mechanistic interpretability aims to reverse-engineer complex neural networks into human-understandable algorithms [2, 34], enhancing our understanding of a wide range of model behaviors, including in-context learning [30], long-context retrieval [46], and chain-of-thought reasoning [4]. A common methodology involves analyzing their computation graphs of a specific task, as exemplified by studies on indirect object identification [42], "greater than" operation [11], and entity tracking [32]. Following this, our work begins with the off-by-one addition task, and showcases the broader applicability of our findings with various task pairs.

Function Vectors in LMs. Recent work has characterized in-context learning in language models as the compression of in-context examples into a single task or function vector, which is subsequently transported to the test example to trigger the model to apply the function [37, 12, 51]. These studies present strong evidence pertaining to *single-step*, *mapping-style* tasks like country-to-capital and English-French translation. Our work is inspired by this line of research, yet with two key differences: (1) We focus on off-by-one addition, a *multi-step arithmetic* task, where the learning of the second step depends on the results of the preceding step. (2) We provide a finer-grained interpretation on how function vectors, sent out by different attention heads, vary in content but collaborate to form a complete function. In concurrent work, this latter aspect was also explored by Hu et al. [14], who investigate the task of add-k (*i.e.*, "5→8, 1→4, 2→?") using subspace decomposition.

Latent Multi-step Reasoning and Structural Compositionality in LMs. Various studies investigate whether and how models perform latent multi-step reasoning, typically via multi-hop factoid QA tasks [49, 41]. Our work demonstrates that LMs can dynamically infer the second step in a multi-step problem from in-context examples, a process representing a novel, flexible and composable form of latent multi-step reasoning. More broadly, our findings are relevant to research investigating structural compositionality [18] (*i.e.*, breaking down complex tasks into subroutines) in language models.

7 Conclusion

In this work, we present an interpretability study on the off-by-one addition task, with the broader goal of investigating how language models handle unseen tasks using in-context learning. Our analysis led to the discovery of a function induction mechanism, which handles the "twists" involved in generalizing from seen to unseen tasks. This discovery extends and generalizes previous interpretability findings on induction heads and function vectors. We further show this mechanism is broadly reused beyond off-by-one addition, notably in realistic algorithmic tasks like Caesar Cipher and base-8 addition. Collectively, these observations deepen our understanding of how models generalize to new tasks and situations, and provide compelling evidence that language models may have developed composable and general mechanisms for handling intricate linguistic and task structures.

Limitations

Regarding path patching experiments (§3), the identified circuit has limitations as it does not perfectly satisfy the faithfulness and completeness criteria, even with our best efforts. This challenge arises because achieving simultaneous satisfaction of faithfulness, completeness, and minimality is difficult, as these criteria often regulate each other. Moreover, number tokens are often mapped into a sinusoidal (Fourier) feature space rather than a linear space in language models [27, 54, 55], which further complicates our interpretability analysis. Besides this limitation, our analysis in §4 focused primarily on the identified FI heads and their causal effect on naive prompts. Future research could dive into the details of the previous token heads and the consolidation heads, or further investigate the query-key and output-value circuits of these heads [6].

Regarding task generalization experiments (§5) , our current scope is limited to two-step tasks where the second step involves a shifting-related function. We anticipate that the function induction mechanism could operate on a broader spectrum of functions, which could be investigated in future work. Additionally, the task pairs we investigated are synthetic or algorithmic; further exploration of the behavior of function induction heads on naturally occurring text distributions would be highly valuable.

Broader Impact

Our work does not involve model training and thus does not directly introduce new malicious or harmful applications. However, our findings broadly demonstrate that when presented with false information in the prompt, models tend to follow and even generalize such inaccuracies. This observation could offer insights into unintended applications or behaviors of language models, such as the generation of misinformation (*e.g.*, if prompted with the false premise that the Earth is flat, the model might generate a convincing article supporting this claim) or how the model acts overly sycophantically (*e.g.*, if a user believes "1+1=3", the model might not only endorse this but also extend it to "2+2=5"). It is our hope that the insights from our interpretability analysis can inform efforts to address these critical societal problems.

References

[1] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024. URL https://arxiv.org/abs/2412.08905.

[2] Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ePUVetPKu6. Survey Certification, Expert Certification.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. Iteration head: A mechanistic study of chain-of-thought. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances*

in Neural Information Processing Systems, volume 37, pages 109101–109122. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c50f8180ef34060ec59b75d6e1220f7a-Paper-Conference.pdf.

[5] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.53. URL https://aclanthology.org/2022.acl-long.53/.

[6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[7] Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. *Arxiv*, 2024. URL https://arxiv.org/abs/2403.00824.

[8] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations, 2023. URL https://openreview.net/forum?id=em4xg1Gvxa.

[11] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=p4PckNQR8k.

[12] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

[13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[14] Xinyan Hu, Kayo Yin, Michael I. Jordan, Jacob Steinhardt, and Lijie Chen. Understanding in-context learning of addition via activation subspaces, 2025. URL https://arxiv.org/abs/2505.05145.

[15] Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Tim Rocktäschel, Edward Grefenstette, and David Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=A0HKeKl4Nl.

[16] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

[17] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring ai ability to complete long tasks, 2025. URL https://arxiv.org/abs/2503.14499.

[18] Michael Lepori, Thomas Serre, and Ellie Pavlick. Break it down: Evidence for structural compositionality in neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 42623–42660. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/85069585133c4c168c865e65d72e9775-Paper-Conference.pdf.

[19] Belinda Z Li, Zifan Carl Guo, and Jacob Andreas. (how) do language models track state? *arXiv preprint arXiv:2503.02854*, 2025.

[20] Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ElVHUWyL3n.

[21] Xinxi Lyu, Sewon Min, Iz Beltagy, Luke Zettlemoyer, and Hannaneh Hajishirzi. Z-ICL: Zero-shot in-context learning with pseudo-demonstrations. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2304–2317, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.129. URL https://aclanthology.org/2023.acl-long.129/.

[22] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

[23] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fpoAYV6Wsk.

[24] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.201. URL https://aclanthology.org/2022.naacl-main.201/.

[25] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

[26] Neel Nanda and Joseph Bloom. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens, 2022.

[27] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

[28] nostalgebraist. interpreting gpt: the logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. Accessed: 2025-06-17.

[29] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. URL https://distill.pub/2020/circuits/zoom-in.

[30] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[31] Akshara Prabhakar, Thomas L. Griffiths, and R. Thomas McCoy. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3710–3724, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.212. URL https://aclanthology.org/2024.findings-emnlp.212/.

[32] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2402.14811.

[33] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8sKcAWOf2D.

[34] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open problems in mechanistic interpretability, 2025. URL https://arxiv.org/abs/2501.16496.

[35] Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.435. URL https://aclanthology.org/2023.emnlp-main.435/.

[36] Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, Michael Stern, Brian Clarke, Landon Goldberg, Theodore R. Sumers, Jared Mueller, William McEachen, Wes Mitchell, Shan Carter, Jack Clark, Jared Kaplan, and Deep Ganguli. Clio: Privacy-preserving insights into real-world ai use, 2024. URL https://arxiv.org/abs/2412.13678.

[37] Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

[38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[39] Igor Tufanov, Karen Hambardzumyan, Javier Ferrando, and Elena Voita. Lm transparency tool: Interactive tool for analyzing transformer language models. *Arxiv*, 2024. URL https://arxiv.org/abs/2404.07004.

[40] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

[41] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=D4QgSWxiOb.

[42] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

[43] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently, 2024. URL https://openreview.net/forum?id=DRGnEkbiQZ.

[44] Wikipedia contributors. Caesar cipher — Wikipedia, the free encyclopedia, 2025. URL https://en.wikipedia.org/w/index.php?title=Caesar_cipher&oldid=1294051421. [Online; accessed 12-June-2025].

[45] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

[46] Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EytBpUGB1Z.

[47] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.102. URL https://aclanthology.org/2024.naacl-long.102/.

[48] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[49] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

[50] Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 355–385, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.21. URL https://aclanthology.org/2024.findings-acl.21/.

[51] Kayo Yin and Jacob Steinhardt. Which attention heads matter for in-context learning?, 2025. URL https://arxiv.org/abs/2502.14010.

[52] Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. Ground-truth labels matter: A deeper look into input-label

demonstrations. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2437, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.155. URL https://aclanthology.org/2022.emnlp-main.155/.

[53] Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

[54] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

[55] Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=i4MutM2TZb.

A Induction Head Mechanism and Function Induction Mechanism

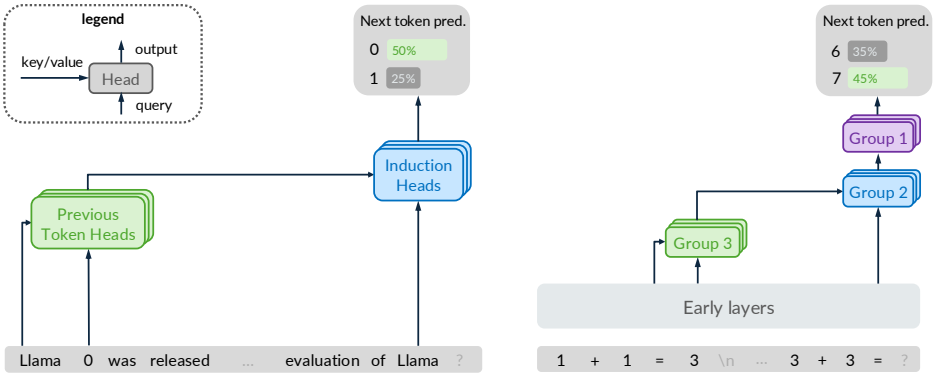

Figure 8: **Comparing Induction Head (Left) and Function Induction (Right).**

Comparing Induction Head and Function Induction. Fig. 8 provides a side-by-side visualization of the induction head mechanism [30] and the hypothesized function induction mechanism (§3.2), demonstrating their structural similarity and explaining the basis for our hypothesis.

To provide a more concrete example on how induction heads work, consider the hypothetical scenario where a language model is completing the prompt: "Llama 0 was released in 2022. This paper presents an extensive evaluation of Llama ..." When the model first encounters an uncommon phrase, *e.g.*, "Llama 0", a previous token head will attend to "Llama" and register the information that "Llama appears before 0" at the position of "0". Later on, when "Llama" appears in the context again, an induction head will retrieve this piece of information from position of "0" and increase the likelihood of generating "0" as the next token. This induction head mechanism informs our hypothesis on function induction in §3.2 and the collaborative interaction between previous token heads and function induction heads in Fig. 8 (Right).

Relevance to In-context Learning with False Demonstrations. Various prior works investigate how language models handle false, random, or perturbed demonstrations in in-context learning [25, 52, 43, 21, 20]. Notably, Halawi et al. [10] adopted an interpretability approach, observing the *overthinking* behavior of models (*i.e.*, models draft truthful answers at early layers and flip them to untruthful answers at late layers), and identified *false induction heads* that are responsible for copying the untruthful answers from the ICL examples.

Our analysis of off-by-one addition was largely motivated by these studies. Here we revisit the findings of Halawi et al. [10] along with ours, using a unified view of two-step tasks, *i.e.*, $z = f(g(x))$. In [10], the first step, $y = g(x)$ is typically a text classification task, *e.g.*, news topic classification, and the second step, $z = f(y)$ is a permutation of the labels, *e.g.*, {Business→Sci/Tech, Sci/Tech→World, World→Sport, Sports→Business}. In our work, $y = g(x)$ is standard addition, and $z = f(y)$ is a +1 function.

In this view, our findings with off-by-one addition are consistent with those in [10], while also advancing the understanding in several aspects: (1) In both cases, language models decompose the task into two steps, and induce the second step based on the results of the first step. The second step could be either a conditional copy-paste function, *e.g.*, a permutation of labels, *or* an algorithmic function, *e.g.*, a +1 function. The latter represents a novel finding of this study, demonstrating that the second step can exhibit forms more complex than copy-paste operations. (2) Our path patching procedure identified two additional group of heads (consolidation heads and previous token heads) that are involved in handling false demonstrations. (3) Our work also suggests that the strategy to improve truthfulness by zeroing out false induction heads or function induction heads may have unintended consequences on models' positive capabilities, given their positive contributions to the cipher task and the base-8 addition task.

Related to the view of two-step tasks, Jain et al. [15] demonstrate that models learn a "wrapper" function g over an existing function f in a sequential fine-tuning setting. Our work and [10] suggest that language models demonstrate simple forms of this behavior with in-context learning as well.

B Off-by-One Addition Evaluation

Models. In §2 we evaluated six recent language models on the task of off-by-one addition. In Table 4 we provide details of these models.

Model Name	Huggingface Identifier	Reference	Tokenization 0-9	Tokenization 0-999
`Llama-2 (7B)`	`meta-llama/Llama-2-7b-hf`	Touvron et al. (2023)	✓	
`Mistral-v0.1 (7B)`	`mistralai/Mistral-7B-v0.1`	Jiang et al. (2023)	✓	
`Gemma-2 (9B)`	`google/gemma-2-9b`	Gemma Team (2024)	✓	
`Qwen-2.5 (7B)`	`Qwen/Qwen2.5-7B`	Yang et al. (2024)	✓	
`Llama-3 (8B)`	`meta-llama/Meta-Llama-3-8B`	Grattafiori et al. (2024)		✓
`Phi-4 (14B)`	`microsoft/phi-4`	Abdin et al. (2024)		✓

Table 4: **Models Evaluated on Off-by-One Addition.** "0-9" means the model uses digit-level tokenization for numbers, *e.g.*, "123" is tokenized into ["1","2","3"], "0-999" means all numbers smaller than 1000 are considered one single token, *e.g.*, "123" is tokenized into ["123"].

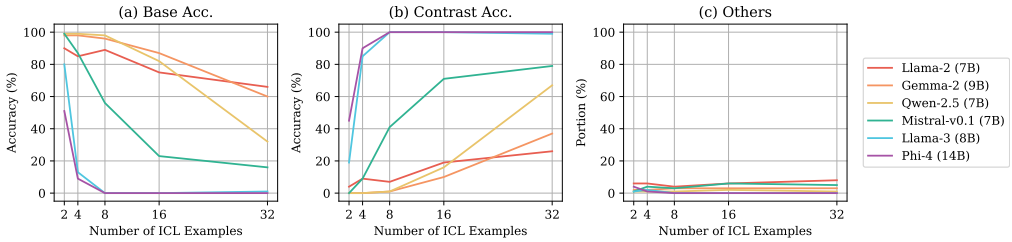

Figure 9: **Off-by-One Addition Evaluation, Reporting Base Accuracy.**

Reporting base and contrast accuracy. Previously in Fig. 1, we reported the accuracy of off-by-one addition (*i.e.*, the percentage of time that the model outputs 7 when given 3+3). In Fig. 9(a) we additionally report the accuracy of standard addition (*e.g.*, "3+3=6"), when the models are given the contrast prompt (*e.g.*, "1+1=3\n2+2=5"). We find that the base accuracy consistently decrease with more in-context learning examples. In Fig. 9(c), we show that models may also output numbers that are incorrect either in standard addition or off-by-one addition (*i.e.*, neither "6" or "7").

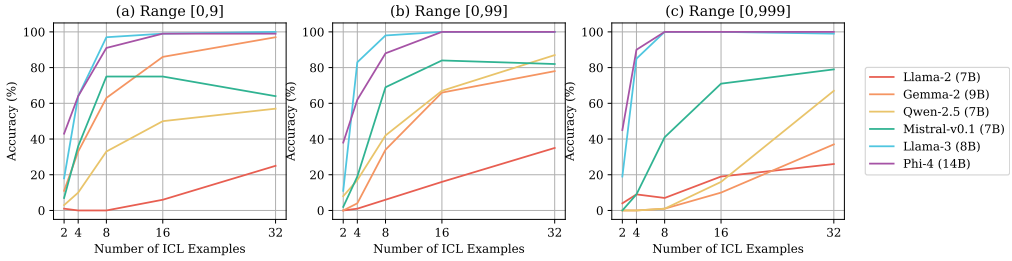

Figure 10: **Off-by-One Addition Evaluation, Using Smaller Number Ranges.**

Results in a smaller number range. Previously in Fig. 1, we reported results when the operands were sampled from the range of [0,999]. In Fig. 10, we additionally report results when sampling from the range of [0,9] and [0,99]. For two models using 0-9 tokenization (`Gemma-2 (9B)` and `Qwen-2.5 (7B)`), the performance drops with larger number ranges. For the remaining models, the performance remains stable regardless of the number ranges.[3]

[3]We chose `Gemma-2 (9B)` as the default model in our study because (1) we focused on the range of [0,9] in early stage of this work to prioritize simplicity, and `Gemma-2 (9B)` performs competitively in this setting; (2) `Qwen-2.5 (7B)`, `Llama-3 (8B)`, `Phi-4 (14B)` were not released or integrated into `transformer-lens` at that time. We acknowledge this experimental design limitation and address it by interpreting `Llama-3 (8B)` and `Mistral-v0.1 (7B)` in §C.

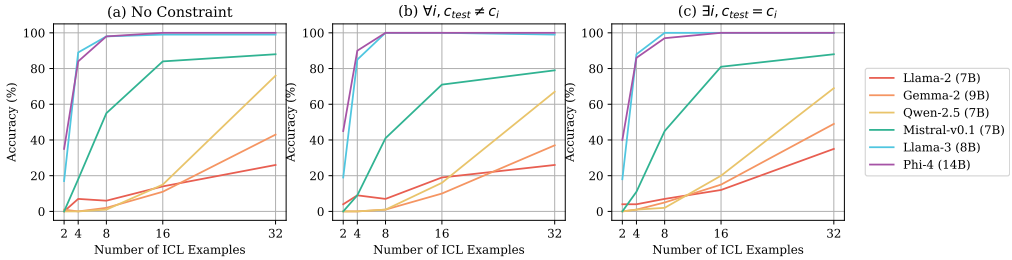

Figure 11: **Off-by-One Addition Evaluation, Different Sampling Constraints.**

Results with/without the constraint of $c_{test} \neq c_i$. Previously in §2 we deliberately impose the constraint that $\forall i, c_{test} \neq c_i$. This is to rule out the possibility that language models perform off-by-one addition via copying c_{test} from previous contexts. In Fig. 11, we compare the results of two additional sampling strategies: (1) no constraint on c_{test} and c_i; (2) $\exists i, c_{test} = c_i$. By comparing Fig. 11(b) and (c) we see that for `Mistral-v0.1 (7B)` and `Gemma-2 (9B)`, the accuracy is higher when $\exists i, c_{test} = c_i$. This observation implies that these two models leverages copy-paste induction more than function induction in performing off-by-one addition, though more rigorous analysis is required to draw a conclusion.

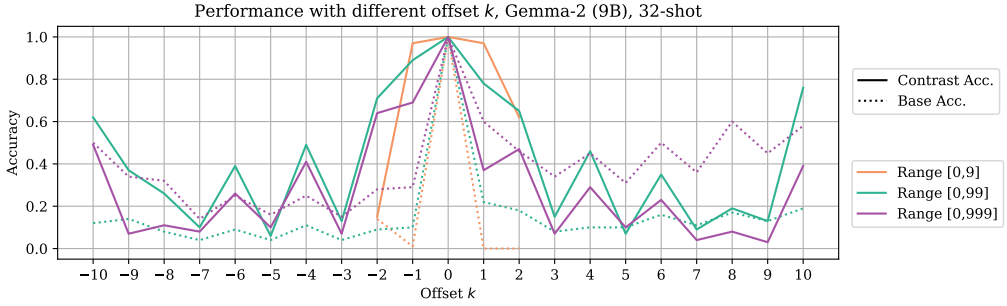

Figure 12: **Off-by-k Addition Evaluation,** `Gemma-2 (9B)`

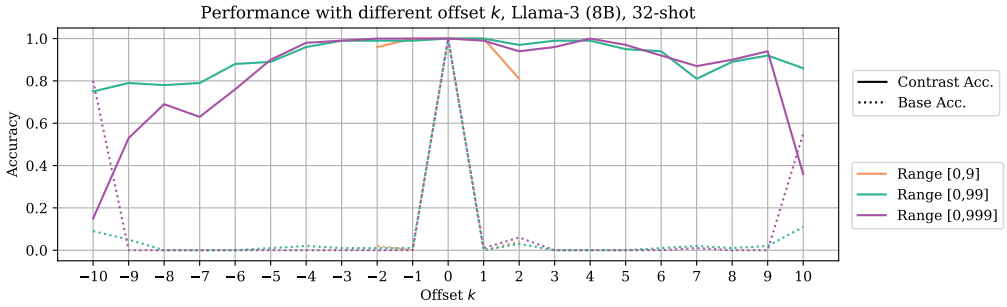

Figure 13: **Off-by-k Addition Evaluation,** `Llama-3 (8B)`

Results with off-by-k addition. In Fig. 12-13, we present 32-shot off-by-k addition results with various offsets k using `Gemma-2 (9B)` and `Llama-3 (8B)` respectively.[4] One consistent trend is that models struggle more with offsets k of larger absolute values. While `Llama-3 (8B)` generally outperforms `Gemma-2 (9B)`, `Gemma-2 (9B)` demonstrates strong performance when $k = \pm 10$, potentially due to its adoption of 0-9 tokenization. An additional observation reveals that `Gemma-2 (9B)` typically achieves stronger performance with even values of k compared to odd values.

[4]The visualization of Fig. 12-13 was inspired by Prabhakar et al. [31].

C Circuit Discovery

C.1 Relative Logit Diff

§3.1 introduced r, the relative logit difference, to measure the effect of a replacement during circuit discovery. We now elaborate on this formula to enhance clarity.

$$r = \frac{F(M', x_{cont}) - F(M, x_{cont})}{F(M, x_{cont}) - F(M, x_{base})} \tag{1}$$

$$= \frac{[M'(y_{base}|x_{cont}) - M'(y_{cont}|x_{cont})] - [M(y_{base}|x_{cont}) - M(y_{cont}|x_{cont})]}{[M(y_{base}|x_{cont}) - M(y_{cont}|x_{cont})] - [M(y_{base}|x_{base}) - M(y_{cont}|x_{base})]} \tag{2}$$

C.2 Identified Heads

In the main paper, we focus on interpreting `Gemma-2 (9B)`. To explore the universality of the mechanism, we additionally conduct path patching with `Llama-3 (8B)` and `Mistral-v0.1 (7B)`. We list the identified attention heads below.

C.2.1 `Gemma-2 (9B)`

- Consolidation Heads: H41.4, H41.5, H40.11, H40.12;
- Function Induction (FI) Heads: H39.7, H39.12, H36.7, H32.1, H32.6, H25.13;
- Previous Token (PT) Heads: H38.6, H38.7, H38.9, H35.14, H35.9, H31.4, H31.5, H29.5.

C.2.2 `Llama-3 (8B)`

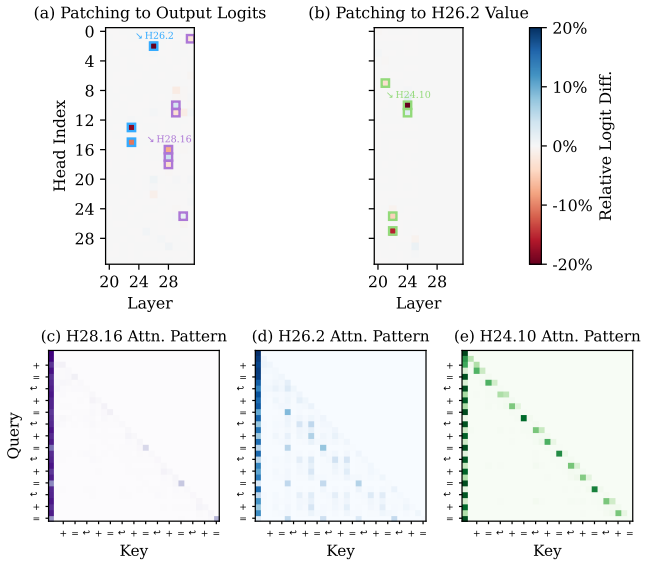

Figure 14: **Circuit Discovery with `Llama-3 (8B)`.** Results are consistent with those with `Gemma-2 (9B)` in Fig. 2.

`Llama-3 (8B)` has 32 layers and 32 heads per layer. Path patching experiments were conducted with 4-shot off-by-one addition with numbers sampled from range [0,999]. We visualize the path patching results in Fig. 14.

- Consolidation Heads: H31.1, H30.25, H29.11, H29.10, H28.16, H28.17, H28.18;
- Function Induction (FI) Heads: H26.2, H23.13, H23.15;
- Previous Token (PT) Heads: H24.10, H24.11, H22.25, H22.27, H21.7.

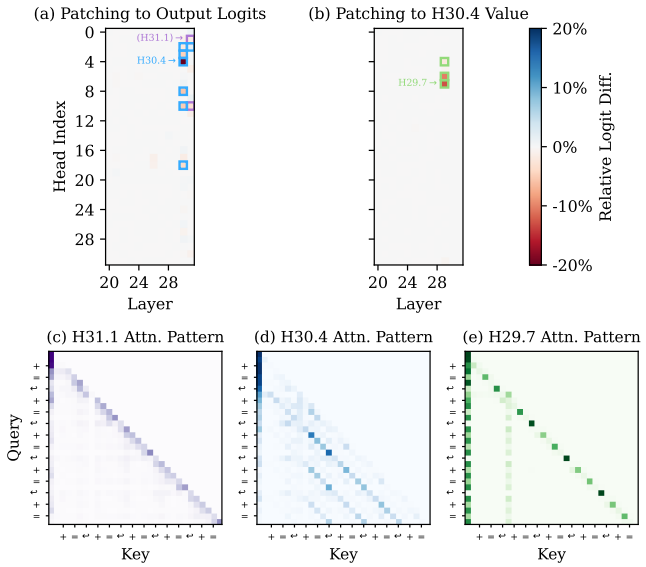

Figure 15: **Circuit Discovery with `Mistral-v0.1 (7B)`.** Results are mostly consistent with those with `Gemma-2 (9B)` in Fig. 2, with the exception of the consolidation heads.

C.2.3 `Mistral-v0.1 (7B)`

`Mistral-v0.1 (7B)` has 32 layers and 32 heads per layer. Path patching experiments were conducted with 4-shot off-by-one addition with numbers sampled from range [0,9]. We visualize the results in Fig. 15. For the two consolidation heads in the list below, they show weaker effect and attend to both the current token and some other tokens, which slightly deviates from our findings with `Gemma-2 (9B)` and `Llama-3 (8B)`. Apart from this, the results using `Mistral-v0.1` are consistent with our function induction hypothesis.

- Consolidation Heads: (H31.10), (H31.1)
- Function Induction (FI) Heads: H30.2, H30.3, H30.4, H30.8, H30.10, H30.18, H31.2
- Previous Token (PT) Heads: H29.4, H29.6, H29.7.

C.3 Additional Interpretability Analysis

C.3.1 Logit Lens Analysis

In this section, we apply logit lens [28], a widely-adopted interpretability method, to off-by-one addition. This involves directly computing the logits from intermediate layer representations using the final layer norm and the final unembedding layer.

We use `Gemma-2 (9B)` and 100 16-shot examples in this set of experiments. In Fig. 16, we report the logits of the base answer y_{base} (*i.e.*, model outputting 3+3=**6**), the contrast answer y_{cont} (*i.e.*, model outputting 3+3=**7**) and their differences, computed using the contrast prompt x_{cont} (*i.e.*, 1+1=3) as model input. In Fig. 17, we repeat the experiments using x_{base} (*i.e.*, 1+1=2) as the input prompt.

By comparing Fig. 16(a) and Fig. 17(a), we find that the curves in the two subplots begin to diverge notably after layer 25. This supports our claim that the model performs standard addition in the early layers and apply the +1 function in late layers.

Additionally, by comparing Fig. 16(b) and Fig. 17(b), we find that the logit diff decreases sharply after layer 38 in Fig. 16(b), a phenomenon absent in Fig. 17(b). This is consistent with our findings that H39.7 and H39.12 contribute significantly to writing out the +1 function to the residual stream.

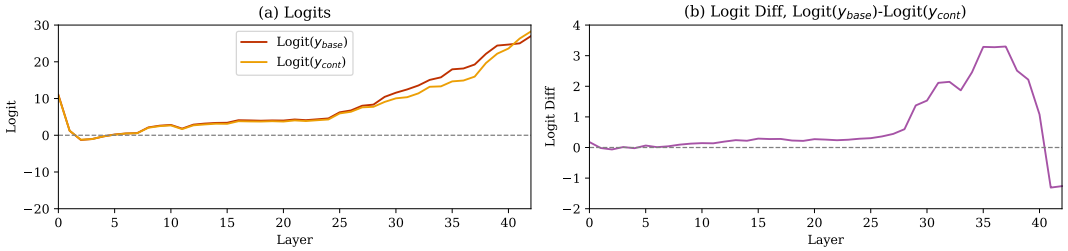

Figure 16: **Logit Lens Results when Using x_{cont} as the Input Prompt.**

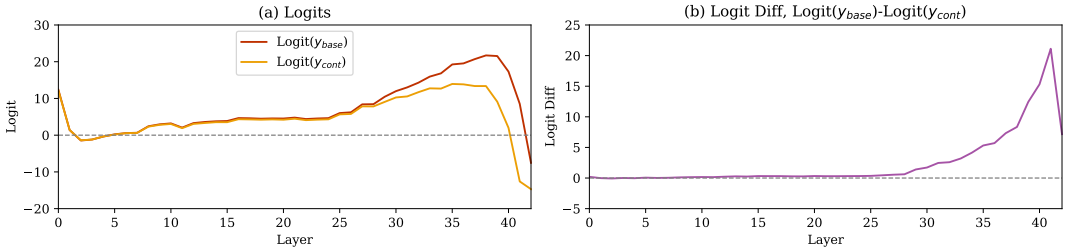

Figure 17: **Logit Lens Results when Using x_{base} as the Input Prompt.**

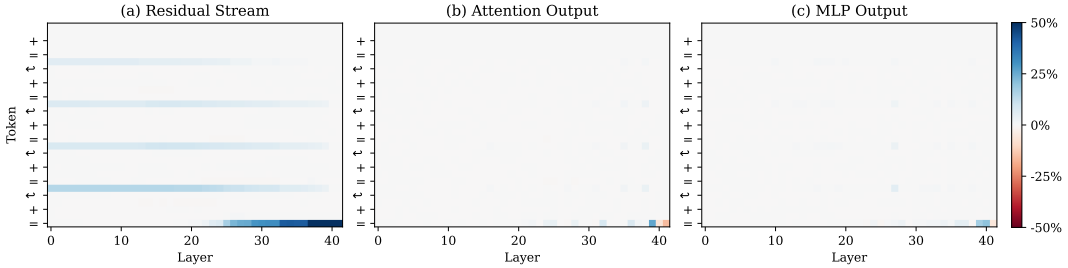

Figure 18: **Activation Patching By Token.**

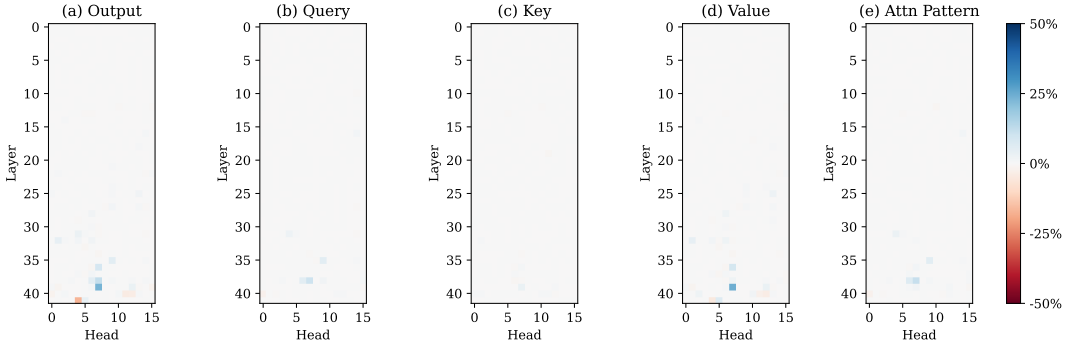

Figure 19: **Activation Patching By Head.**

C.3.2 Activation Patching Analysis

In this section, we apply activation patching [22] to off-by-one addition. We performed this analysis in the early stages of our work to gather initial intuitions and signals for our problem, before transitioning to path patching for a more fine-grained understanding of the model's internal computation.

We use `Gemma-2 (9B)` and 100 4-shot examples in this set of experiments. First, we run forward passes for both the base prompt x_{base} and the contrast prompt x_{cont}. We store the activations and then replace the activations in the x_{cont} forward pass with corresponding activations in the x_{base} forward pass. We consider activation patching by token (Fig. 18) and by head (Fig. 19). We report

the ratio $r' = 1 + r = \frac{F(M', x_{cont}) - F(M, x_{base})}{F(M, x_{cont}) - F(M, x_{base})}$ in these figures following previous works. We scaled the colormap in the figures to the range of [-50%, 50%] for clear visualization.

Fig. 18(a) visualizes the information flow from in-context examples to the residual stream of the last " =" token. Additionally, Figure 18(b) highlights several layers, specifically layers 32, 36, and 39 at the last "=" token, and layers 35 and 38 at the answer token c in the in-context examples. This aligns with the FI heads (H36.7, H39.7, H39.12) and PT heads (H35.9, H35.14, H38.6, H38.7, H38.9) identified in §3.2. Figure 18(c) further reveals that MLP layers also play critical roles at certain positions. It is possible that FI heads write the +1 function to the residual stream, with subsequent attention and MLP layers involved in the execution of the +1 function. This hypothesis is inspired by prior observations of how MLP layers in transformer models are involved in arithmetic operations [27, 35]. In this work, we limit our focus to attention heads, deferring further analysis of MLP layers to future work.

Results in Fig. 19 guide and complement our path patching experiments in §3.2. The identified PT heads (H35.9, H38.6, H38.7, H38.9) are highlighted in Fig. 19(b) and the FI heads (H36.7, H39.7, H39.12, H32.1, H25.13) are highlighted in Fig. 19(d).

C.3.3 Alternative Head Ablation Methods

In the main paper, we "ablate" or "knock out" a head by replacing its output in the x_{cont} forward pass with the corresponding head output in the x_{base} forward pass. This instance-specific ablation approach is adopted to better isolate the +1 function computation in each instance. However, this differs from ablation methods commonly used in interpretability work, such as zero ablation [10] or mean ablation [42].

To demonstrate the consistency of our findings across different ablation settings, we repeated the experiment in Fig. 4(a) using zero ablation and mean ablation. For mean ablation, we averaged head outputs at the final "=" token from 100 standard addition examples. We found that in all ablation settings (zero ablation, mean ablation, and our instance-specific ablation), the contrast accuracy reduced to 0% and the base accuracy increased to 100% after ablation.

D Function Vector Analysis

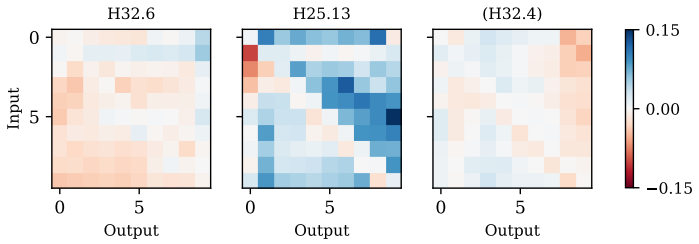

Figure 20: **Rescaled Effect of H32.6, H25.13 and H32.4.** We rescale the effect of these three heads in Fig. 5 to [-0.15, 0.15] to make the patterns more readable. When x is the input, both H25.13 and H32.4 suppresses $x - 1$.

Rescaled Effect of Selected Heads. In Fig. 20, we rescale the effect of H32.6, H25.13, H32.4 in Fig. 5 to visualize their patterns more clearly. H25.13 and H32.4 contribute to the +1 function by suppressing $x - 1$. However, the role of H32.6 is unclear from the heatmap.

What do FI heads write out in standard addition? Our function vector style analysis in §4 primarily focuses on what FI heads write out in off-by-one addition. However, these heads may also assume roles in standard addition. To investigate this, we add the FI head outputs in the $M(.|x_{base})$ to the naive prompt x_{naive}, and visualize the effect in Fig. 21. By comparing Fig. 5 and Fig. 21, we observe that most FI heads contribute meaningful but distinct information in standard addition, with H39.12 being an exception given its minimal effect in standard addition. The aggregated effect in the bottom-right panel in Fig. 21 suggests that FI heads collectively suppress $x - 1$ and promote x in standard addition.

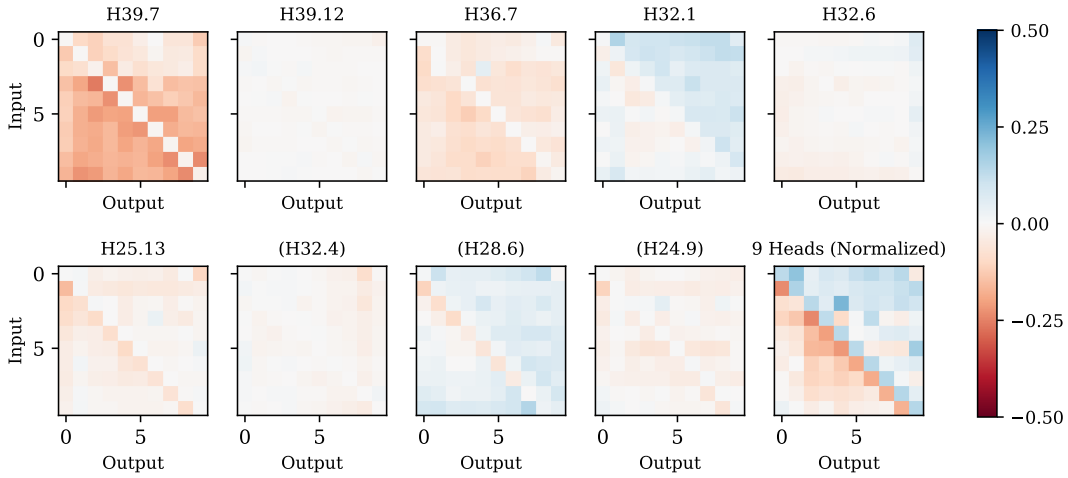

Figure 21: **Individual and Overall Effect of Identified FI Heads (Standard Addition).**

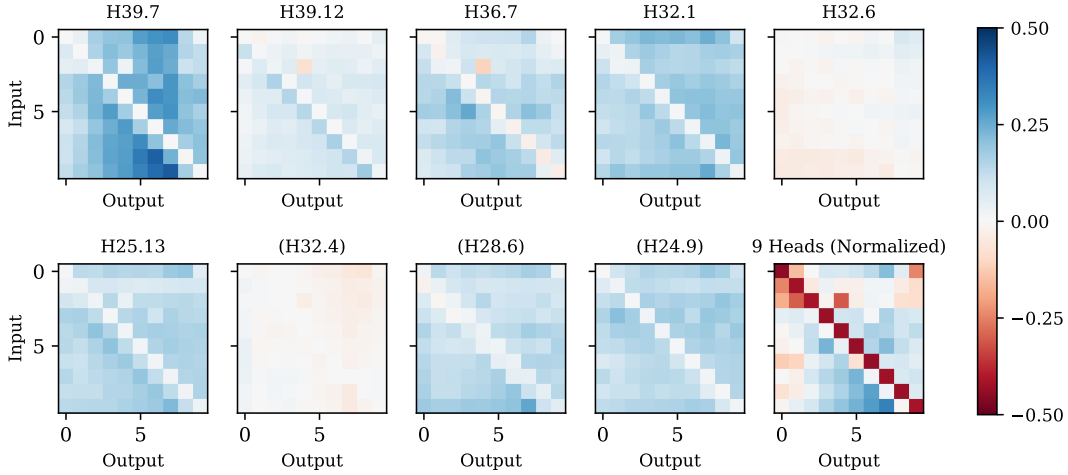

Figure 22: **Individual and Overall Effect of FI Heads in Off-by-k Addition, $k = -2$.**

One possibility is that FI heads reinforce the answer x, or double-check it by performing $(x-1)+1$ in standard addition. In contrast, during off-by-one addition, the standard addition answers are first "locked in" after early layers, and the FI heads are repurposed to perform +1. We leave further investigation of this phenomenon to future work.

What do FI heads write out in off-by-k addition? Previously in Fig. 7, we demonstrated how the effect of H39.7 and H25.13 changes with respect to different offset k. In Fig. 22-24 we report the effect of all nine heads when $k = -2, -1, 2$. We find that for some heads (*e.g.*, H32.1 and H24.9), their effect of suppressing x remains consistent across different k values. For other heads (*e.g.*, H39.7, H39.12, H25.13), their effect changes accordingly with respect to k.

E Task Generalization

E.1 Tasks and Data Preparation

In this section we describe the task pairs we used in §5 with more details.

Off-by-k Addition. For experiments in the range of [0,9], we consider $k \in \{-2, -1, 1, 2\}$. For experiments in the range of [0,99] and [0,999], we consider $k \in \{-10, -9, ..., -1, 1, 2, ..., 10\}$. We

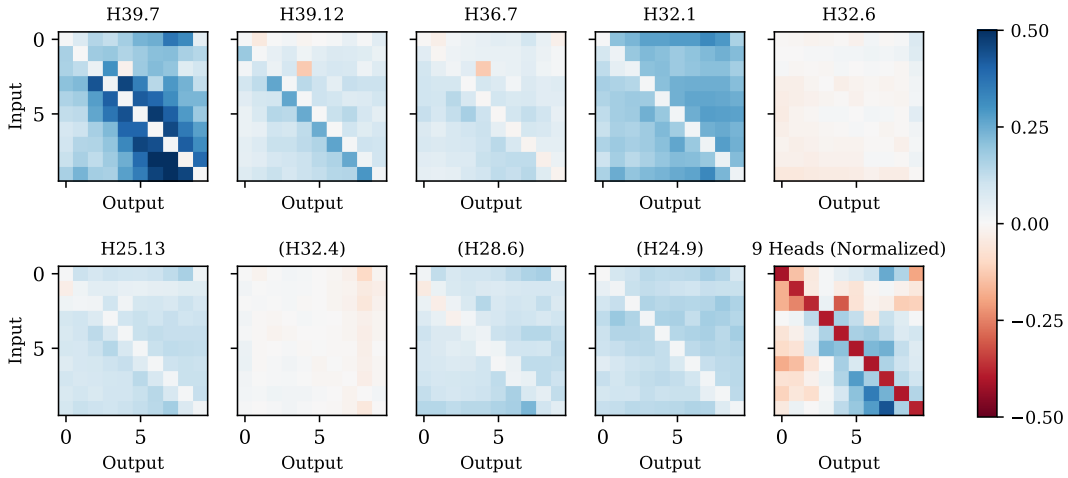

Figure 23: **Individual and Overall Effect of FI Heads in Off-by-k Addition, $k = -1$.**

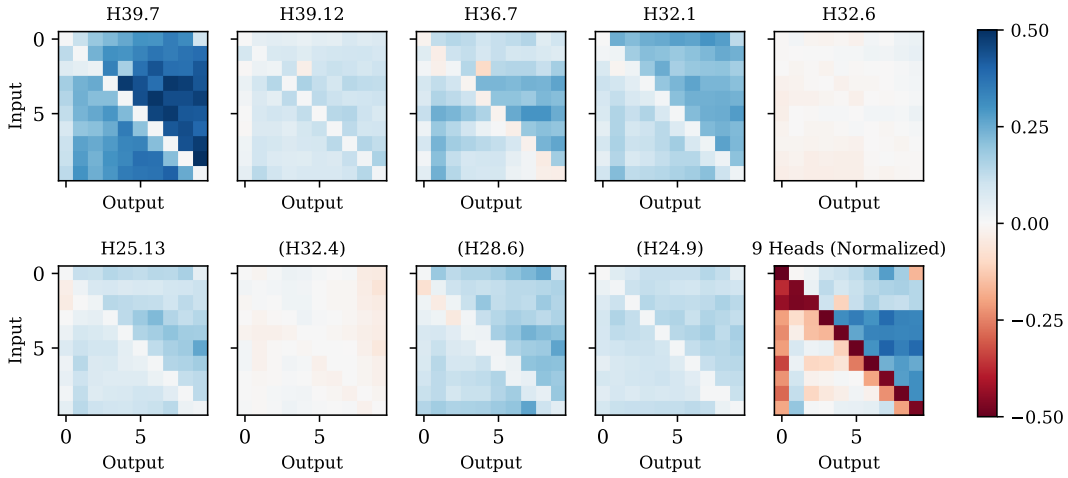

Figure 24: **Individual and Overall Effect of FI Heads in Off-by-k Addition, $k = 2$.**

have reported the results in Fig. 12-13, incorporating the range and offset information. We use 16 shots in the experiments in Fig. 6(a).

Shifted Multiple-choice QA. We focus on 6 subjects in the MMLU dataset [13]: high school government and politics, high school US history, US foreign policy, marketing, high school psychology, sociology. We downloaded the MMLU dataset from `hendrycks/test`. We chose these subjects because `Gemma-2 (9B)` achieves 90% accuracy with 5 shots on them. For subjects where `Gemma-2 (9B)` achieves lower accuracies, tracking and analyzing performance on Shift-by-One MMLU becomes challenging, because the model could score points by random guessing. We use 16 shots in the experiments in Fig. 6(b), where the 16 shots combine "validation" and "dev" examples from the MMLU dataset.

Caesar Cipher. We adopted a cyclic approach where "a" is considered the next character after "z". We also included both lower-case or upper-case examples, *e.g.*, "c -> d" and "C -> D" are both valid examples in ROT-1. We use 16 shots in the experiments in Fig. 6(c).

In the early stages of this work, we experimented with multi-character Caesar cipher. To prevent multiple characters from being tokenized as a single unit (*e.g.*, "ew" as one token in `Gemma-2`'s tokenizer), we used a preceding whitespace (␣) before each character, formatting it as "␣e␣w" so that "␣e" and "␣w" became separate tokens. However, we ultimately focused on one-character Caesar cipher in the experiments because `Gemma-2 (9B)` has insufficient performance on this task when

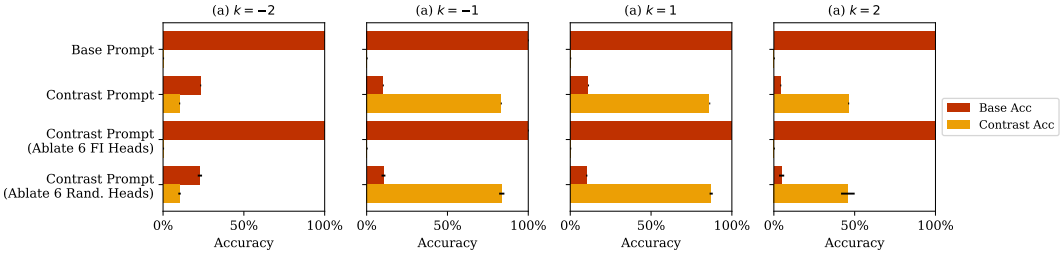

Figure 25: **Task Generalization with FI Heads, Off-by-k Addition.** We consider addition in the range of [0,9] and $k \in \{-2, -1, 1, 2\}$.

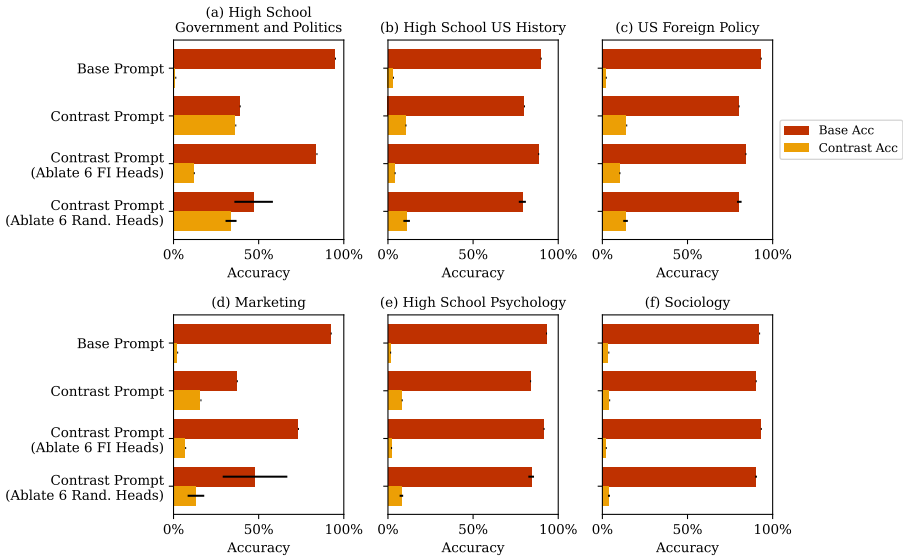

Figure 26: **Task Generalization with FI Heads, Shifted MMLU.**

working with multiple characters. The tokenization-aware formatting was retained. The actual model input will be "␣c -> ␣d" for the example "c -> d".

Base-k Addition. We sampled two-digit addition problems using a procedure similar to off-by-k addition, with one additional constraint that the sum number c has two digits in both base-10 and base-k. We use 32 shots in the experiments in Fig. 6(d). For the base-8 addition analysis in §5.2 and Table 3, examples for Case 1-3 were resampled.

E.2 Results

Full Results using Different Offsets and Bases. Previously in Fig. 6, we report results on representative cases, *e.g.*, $k = 2$ in off-by-k addition, the subject of "high school government and politics" in shifted MMLU. In Fig. 25-28, we report results of the full list of offsets and subjects.

We observe that some of these task variants exceed `Gemma-2 (9B)`'s capabilities. For instance, `Gemma-2 (9B)` has notable performance on cipher when $k \in \{-2, -1, 1, 2, 3, 13\}$ but shows insufficient performance in other settings. Similarly, it only exhibits non-trivial performance on certain subjects of Shifted MMLU. However, when models do have non-trivial performance, we consistently see the involvement of the FI heads, evidenced by the decreased contrast accuracy after ablating them.

Ablating three additional FI Heads. Previously in Fig. 6, we ablate the 6 FI heads we identified in §3.2 by setting a threshold of $|r| > 2\%$. In §4 and Fig. 5 we showed that 3 additional FI heads with weaker effect ($1\% < |r| < 2\%$) also contribute meaningfully to off-by-one addition. Here we consider repeating the experiments on task generalization in Fig. 6 and ablating the 9 heads together. We report the results in Fig. 29.

Figure 27: **Task Generalization with FI Heads, Caesar Cipher.** We consider $k \in \{-12, -11, ..., -1\}$ and $\{1, 2, ..., 13\}$. In this figure, we ablate 6 FI heads plus 3 additional FI heads (discussed in §4 and Fig. 5), yielding a clearer pattern than ablating 6 heads alone.

We find that the 3 weaker heads contribute meaningfully to the Shifted MMLU, causing its contrast performance to drop to near 0% when all 9 heads are ablated (Fig. 29(b)), contrasting with 12% when 6 heads are ablated (Fig. 6(b)). We have a similar observation with Caesar Cipher ($k = 2$), where contrast accuracy drops to 0% in (Fig. 29(c)), contrasting with 36% when 6 heads are ablated (Fig. 6(c)). These observations suggest that the 3 heads may specialize in letters more than numbers. Understanding these detailed specializations will be an interesting direction for future work.

F Reproducibility

Frameworks. We primarily use the `transformer-lens` [26] library for model inference and interpretability analysis. This library is built on the `transformers` [45] library. We have also used the `llm-transparency-tool` [7, 39] for early exploration.

Hardware. All experiments were conducted with one NVIDIA RTX A6000 GPU (48GB). Path patching experiments involving 100 4-shot examples and iterating over all attention heads for a given target node will typically take 2 hours.

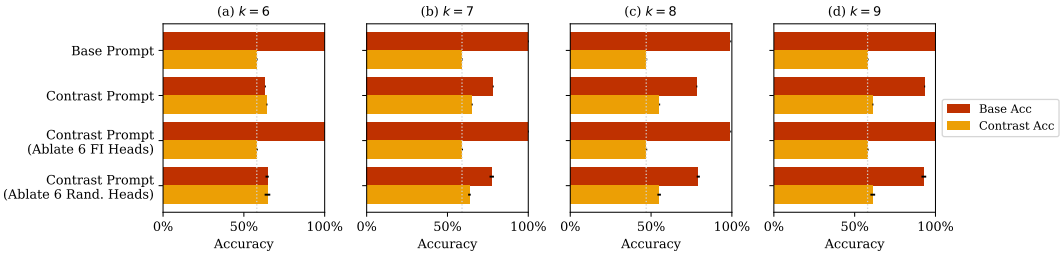

Figure 28: **Task Generalization with FI Heads, Base-k Addition.** We consider $k \in \{6, 7, 8, 9\}$. The dashed lines represent the base prompt's contrast accuracy, emphasizing the delta in contrast accuracies between rows.

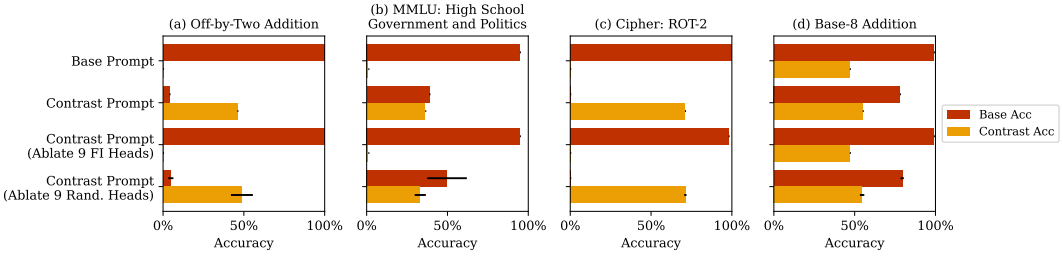

Figure 29: **Task Generalization with FI Heads, Ablating 9 FI Heads.** We repeat the experiment in Fig. 6, this time ablating three additional FI heads (H32.4, H28.6, H24.9) which showed a weaker effect ($1\% < |r| < 2\%$) during circuit discovery on off-by-one addition.