Memory or Reasoning? Explore How LLMs Compute Mixed Arithmetic Expressions

Chengzhi Li[1] **Heyan Huang**[1,2] **Ping Jian**[*1,2] **Zhen Yang**[1] **Chenxu Wang**[1] **Yifan Wang**[1]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2]Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, Beijing Institute of Technology, Beijing, China
{lichengzhi,hhy63,pjian,bityangzhen,wangchenxu,3220231253}@bit.edu.cn

Abstract

Large language models (LLMs) can solve complex multi-step math reasoning problems, but little is known about how these computations are implemented internally. Many recent studies have investigated the mechanisms of LLMs on simple arithmetic tasks (e.g., $a + b$, $a \times b$), but how LLMs solve mixed arithmetic tasks still remains unexplored. This gap highlights the limitation of these findings in reflecting real-world scenarios. In this work, we take a step further to explore how LLMs compute mixed arithmetic expressions. We find that LLMs follow a similar workflow to mixed arithmetic calculations: first parsing the complete expression, then using attention heads to aggregate information to the last token position for result generation, without step-by-step reasoning at the token dimension. However, **for some specific expressions, the model generates the final result depends on the generation of intermediate results** at the last token position, which is similar to human thinking. Furthermore, we propose a **C**ausal **E**ffect **D**riven **F**ine-tuning method (CEDF) to adaptively enhance the identified key components used to execute mixed arithmetic calculations to improve LLMs' reasoning ability.

1 Introduction

In recent years, large language models (LLMs) have demonstrated excellent performance in complex reasoning tasks (Mondorf and Plank, 2024). Through large-scale pre-training, in-context learning, and chain-of-thought(CoT) techniques, LLMs can solve math problems at the high school level and above (He et al., 2024). However, their complex reasoning ability is relatively poor without using CoT (Yu et al., 2024).

When solving arithmetic problems without CoT, do LLMs output corresponding answers based on memory directly, or do they perform operations equivalent to arithmetic rules internally (reasoning)? This reflects whether the models have truly understood arithmetic problems, further determining the extent to which we can trust the outputs of LLMs. In recent years, with the development of interpretability techniques, the understanding of how large models tackle simple text comprehension tasks and perform basic arithmetic calculations has been steadily increasing (Hanna et al., 2023; Wang et al., 2023; Quirke and Barez, 2024; Stolfo et al., 2023). However, how LLMs accomplish more complex mathematical reasoning tasks, which are prevalent in real-world applications, still remains mysterious.

In this work, we preliminarily explore mixed arithmetic calculation, a complex multi-step mathematical reasoning problem. People solve the values of sub-expressions of mixed arithmetic expressions step by step in the order of the Parenthesis, Exponents, Multiplication, Addition, and Subtraction (PEMDAS) rules, ultimately arriving at the correct answer. *Do LLMs execute mixed arithmetic calculations in the same rule?*[1]

To figure out the answer to this question, we conducted the following explorations: 1) Based on causal effect attribution, we identify the key components that the model uses to execute different mixed arithmetic expressions. 2) Using Patchscopes (Ghandeharioun et al., 2024) and pattern analysis, we explore the composition of key components and the functional roles of each component. Through knockout experiments, we verified these perspectives.

Therefore, we confirmed that the model completes mixed arithmetic tasks as follows: as the activation propagates forward, the model first understands task instruction, then comprehensively parses the expression, and finally aggregates the

* Corresponding author

[1]This work does not involve discussions on exponentiation operations.

Findings of the Association for Computational Linguistics: ACL 2025, pages 5742–5763

operand information to the last token for result generation. This forward answer generation process is similar to simple arithmetic tasks without step-by-step reasoning, according to the PEMDAS rule at the token dimension. However, we find that LLMs exhibit similarities to human thinking in other aspects (perspectives 1 & 2).

In summary, **LLMs complete mixed arithmetic tasks in a manner that is neither purely memory-based nor purely reasoning-based, reflecting both similarities and differences compared to human thought processes.**

Compared to previous work, we emphasize the following new perspectives:

1. Unlike simple arithmetic operations, LLMs' key heads exhibit significant functional differentiation, and different types of mixed expressions activate different attention heads.

2. For some specific expressions, the model generates the final result that depends on the generation of intermediate results, which can be seen as a form of arithmetic *"reasoning"* different from human thinking.

3. We propose a Causal Effect Driven Fine-tuning method (CEDF) to adaptively enhance the identified key components used to execute mixed arithmetic calculations, which achieves significant improvements compared to full fine-tuning and precise tuning.

2 Related Work

2.1 Mechanical Interpretability

Mechanistic interpretability is a subfield of AI interpretability that focuses on attributing model behaviors to its components, thus reverse engineering the network (Saphra and Wiegreffe, 2024). Research efforts have focused on understanding internal features (Elhage et al., 2022; Ju et al., 2024; Allen-Zhu and Li, 2024), identifying key components or circuit within models (Chughtai et al., 2023; Nanda et al., 2023; Hanna et al., 2023; Lieberum et al., 2023; Wang et al., 2023). In understanding the internal features field, Patchscopes (Ghandeharioun et al., 2024) and SelfIE (Chen et al., 2024) maps a given representation to a sentence in natural language, having obvious advantages over other methods (Pal et al., 2023). In the field of detection circuits, activation patching is a classic circuit detection algorithm based on causal mediation analysis (Vig et al., 2020). The EAP-IG algorithm is a linear approximation to the activation patching algorithm combined with the integral gradient to estimate the importance of each edge in the computational subgraph (Hanna et al., 2024).

We use Patchscopes to understand the internal features of LLMs in mixed arithmetic calculation tasks and use EAP-IG to identify the key modules and pathway in the calculation process in this work (Ghandeharioun et al., 2024; Hanna et al., 2024).

2.2 Interpretability on Arithmetic Tasks

The earliest explorations focused on arithmetic problems with two or three operands (Quirke and Barez, 2024; Nanda et al., 2023). Stolfo's work showed that early MLP modules establish an understanding of the concepts of operands and operators, while middle-layer attention heads transport information from operand and operator tokens to the last token, with late MLPs that integrate all the information completing the computational reasoning (Stolfo et al., 2023). Zhang's work further explored this process, finding that less than 5% of attention heads in LLMs are related to arithmetic operations, and these components play a critical role in solving complex mathematical application problems (Zhang et al., 2024). Nikankin carefully investigated the functions of neurons in key MLP layers, discovering that 1.5% of neurons in each layer combine in a pattern-recognition manner to form bags of heuristics approach that complete arithmetic reasoning (Nikankin et al., 2024).

Our work extends them to the more complex case of mixed arithmetic expressions, which require understanding the relationship between multiple operators and parentheses.

Contemporaneously, Mirzadeh and Deng explored the nature of large models' arithmetic reasoning behavior from alternative research perspectives beyond mechanistic interpretability (Mirzadeh et al., 2024; Deng et al., 2024). Their conclusions emphasize different aspects and do not conflict with the viewpoints presented in this paper.

3 Experimental Setup

3.1 Task Definition

We define the expression of mixed arithmetic reasoning tasks studied in this work. In this paper, for simplicity and ease of analysis, mixed arithmetic expressions are inductively defined as follows:

1. Every integer $k \in \mathbb{Z}$ is an expression.

2. If E_1 and E_2 are expressions, then so are: $E_1 + E_2$, $E_1 - E_2$, $E_1 \times E_2$, $E_1 \div E_2$.

3. Define an evaluation function $v : \text{Expr} \to \mathbb{Z}$ with the following recursive rules:

$$v(k) = k \tag{1}$$
$$v(E + F) = v(E) + v(F) \tag{2}$$
$$v(E - F) = v(E) - v(F) \tag{3}$$
$$v(E \times F) = v(E) \times v(F) \tag{4}$$
$$v(E \div F) = v(E) \div v(F) \tag{5}$$

where Expr is the set of all mixed arithmetic expressions, and k is an integer.

Since division is not closed over the set of integers, we define the result of division operations as integer division results, ignoring cases where exact division is not possible. We define $v(E)$ and $v(F)$ as the **I**ntermediate **R**esult (IR) values for expression E <*op*> F, where <*op*> $\in \{+, -, \times, \div\}$.

We use $\{X_1, X_2, X_3 \dots\}$ to denote the operands appearing in the expression in left-to-right order. The task of mixed arithmetic operations is defined as constructing a model that, given an expression, outputs the final result of the expression. We will explain this process in the following sections. The task template is shown in Appendix A.

3.2 Datasets

We construct a dataset with different recursive rounds, and the dataset consisting of all i-round expressions is named as $\mathcal{D}_i$. The values of operands, IRs, and **F**inal **R**esults (FR) are constrained to the range of $[0, 1000)$. Random sampling is employed, and we ensure that each expression prototype in the dataset is represented by 2000 samples.

3.3 Models

We investigated the following four representative models: Qwen2.5-14B-Instruct (Qwen Team, 2024), Llama3-8B-Instruct (Dubey et al., 2024), Phi-4 (Abdin et al., 2024), and Gemma-2-9B-IT (Team et al., 2024). All results presented in the main text are based on experiments conducted with the **Qwen2.5-14B-Instruct** model. Results for the other models are provided in the Appendix C.3 & D.4.

4 Key Component Detection

4.1 Method

We use the EAP-IG algorithm (Hanna et al., 2024) to estimate the causal effect of each edge in the model's computational graph, thereby we can know **WHAT** key components of the model execute mixed arithmetic calculations. More introduction about EAP-IG is provided in Appendix B.1.

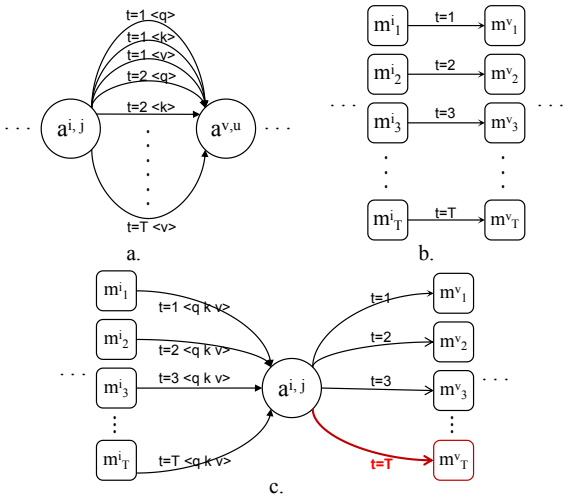


Figure 1: The edges of different components in the token-wise computational graph. In the figure, i and v represent two different layer indices satisfying $i < v$, while j and u represent arbitrary head indices in these two different layers.

Unlike previous interpretability studies using circuit detection algorithms (Hanna et al., 2023; Goldowsky-Dill et al., 2023; Wang et al., 2023; Merullo et al., 2024), we model Transformers-based LLM as a token-wise computational graph, allowing us to analyze the importance differences of edges connecting different tokens directly.

We use $a_t^{l,n}$, m_t^l to refer to the n-th attention head in the b-th layer at the t-th token position of the model and the t-th token position MLP module in the l-th layer of the model, respectively. We use $e_{t,qkv}^{n_1 \to n_2}$ to refer to the edge from node n_1 to node n_2 in t-th token position, while qkv used to refer to the type (query, key, value) of the edge, which can be q, k, v or does not exist.

The connections of these edges and nodes are shown in Figure 1. Based on the EAP-IG algorithm (Hanna et al., 2024), the causal effect $s_{t,qkv}^{n_1 \to n_2}$ for each edge $e_{t,qkv}^{n_1 \to n_2}$ can be obtained using the following formula:

$$s_{t,qkv}^{n_1 \to n_2} = (\tilde{z}_{t,qkv}^{n_1} - z_{t,qkv}^{n_1}) \frac{1}{m} \sum_{k=1}^{m} \frac{\partial \mathcal{L}(\tilde{z} + \frac{k}{m}(z - \tilde{z}))}{\partial z_{t,qkv}^{n_2}} \tag{6}$$

where $z_{t,qkv}^{g}$ and $\tilde{z}_{t,qkv}^{g}$ represent the activation val-

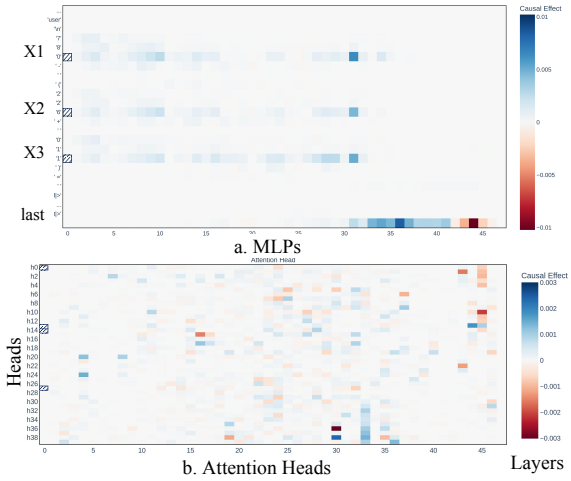


Figure 2: The distribution of causal effects across all MLPs (a) and all attention heads (b)[2].

ues of edge $e^{n_1 \to n_2}_{t,qkv}$ based on clean input is x and disturbed input $\tilde{x}$, respectively. $\mathcal{L}$ represents the model's loss function, and m represents the number of hyperparameter integration steps of the algorithm. z and $\tilde{z}$ are the model's all hidden representations when the input is x and $\tilde{x}$, respectively.

4.2 Implementation Details

Following previous work (Hanna et al., 2023; Goldowsky-Dill et al., 2023; Wang et al., 2023; Merullo et al., 2024), we used the Kullback-Leibler divergence between the probability distributions of the output token of the patched run and the clean run as the metric. We randomly changed the values of operands or the types of operators in each clean sample's expression to construct perturbed samples $\tilde{x}$.

4.3 Main Results

Benefiting from more granular modeling of the forward computation process, the top 1% of high causal effect edges, when considered as a circuit, can recover an average of 85% of the full model's performance. Therefore, we believe that we have identified the key components of the model to complete the mixed arithmetic calculation task. Table 3 presents the circuit performance of 1% of the edges on different expressions.

Figure 2 illustrates the average distribution of causal effects across all MLPs and attention heads for all expressions in $\mathcal{D}_2$.

[2]The sign ▯ in the first column in Figure 2& 5 means the causal effect is so high that it exceeds the range of the color bar.

4.4 Discussion

Key MLPs. From Figure 2, we observe that: 1) At the operand positions, *only the early 30 layers* have an impact on the output; for the last token, *only layers 31-37* have a positive causal effect, while layers 41-44 have a negative causal effect. This difference indicates that the first 30 layers and the last 18 layers play different roles in completing mixed arithmetic calculations. We will further explore this functional difference in the following sections.

More discussions about the parentheses effect are provided in Appendix C.

Key Attention heads. From Figure 2, we observe that: 1) *Only a small portion of the heads has a significant impact on output.* This enables further investigation into the patterns of these key heads and analysis of their functionalities. 2) *The distribution of key heads is not concentrated*, with relatively more heads between layers 19 and 36, and most high effect heads beyond layer 37 having negative effects. We use a threshold of 1e-5 to classify high causal effect heads and further analyze their functionalities in the following sections.

5 MLP Function Analysis

5.1 Method

Patchscopes parses the information in the activation by patching them to an *"zero"* expression, so we can know **HOW** the model generates the intermediate and final results(IRs&FRs). This workflow is illustrated in Figure 3. The algorithm details are provided in the Appendix B.2.

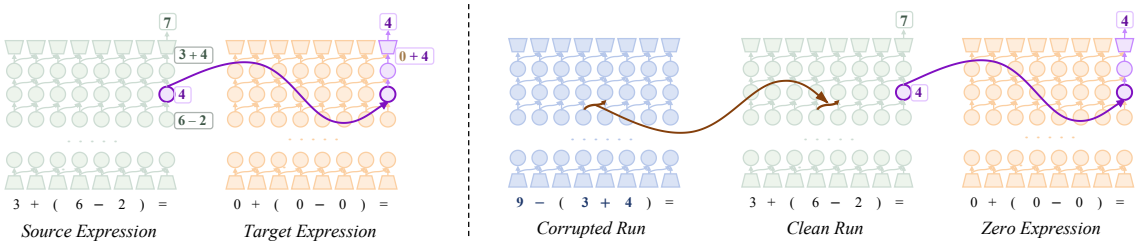
5.2 Implementation Details

We perform Patchscopes (Ghandeharioun et al., 2024) on the residual flow at the operand positions and the last token position to parse the information of the operand values, IRs, and FRs *"stored"* in the activation. We run Patchscopes on all expressions in $\mathcal{D}_2$ to analyze the functions of the key MLPs in the model.

5.3 Main Results

Main results are shown in Figure 4. All operands are resolved from the operand positions, while the IR and FR are resolved from the last token position.

We find that the resolution rate of the IR in different expressions has three different trends. Therefore, we classify them into three groups according



a). Patchscope workflow. **b). Knockout workflow.**

Figure 3: The main workflow of Patchscopes (a) and knockout (b).

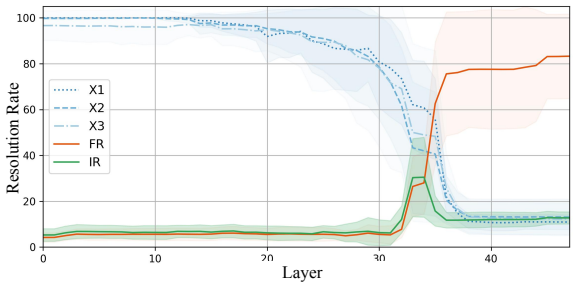


Figure 4: The resolution rates on type-1 expressions.

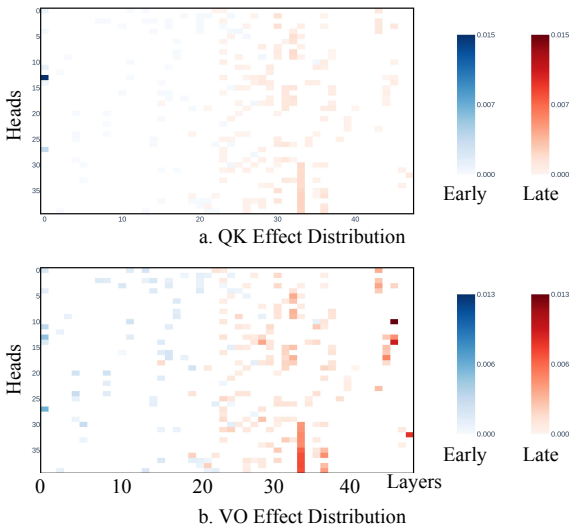


Figure 5: The differences in causal effects between QK and OV circuits.

to the trends. More discussions about these three expression types are provided in the Appendix D. To prevent misclassification of expressions, we use the light area to show the trend of the standard deviation of the resolution rate, indicating that all expressions in this group change in the same trend. The resolution rates on type-1 expressions are shown in Figure 4.

5.4 Discussion

From Figure 4, we observe that:

1) *From around layer 15, as the activation propagates forward, the resolution rate of the operands gradually decreases*. This indicates that the information of the operand values in the activation is gradually transformed or transferred.

2) *From around layer 30, the resolution rate of the operands drops rapidly*, which corresponds to the key MLPs detection results, indicating that the information of the operands is transferred out of this position, and the subsequent activation no longer contains information related to the operand.

3) *From around layer 31, the resolution rate of the IR and FR begins to rise*. This corresponds to the drop in the resolution rate of the operands in observation 2. This indicates that the operand information is transferred to this position and is mapped to the IR or FR after the MLP layer, i.e., arithmetic reasoning is performed. This observation is also consistent with the results of key MLP detection. **This indicates that the model doesn't step-by-step result value mapping at the token dimension.**

4) *For type1 expressions, the drop in the resolution rate of the IR corresponds to the further rise in the resolution rate of the FR*, indicating that the IR is further mapped to the FR in subsequent MLPs. **This demonstrates the model's reliance on IRs for arithmetic "reasoning" at the layer level for specific expressions, resembling human arithmetic reasoning.** We will discuss the cases of type2 and type3 further in the Appendix D.

6 Attention Head Function Analysis

6.1 Method

We conducted comparative studies and pattern analysis to reveal the functionalities of different types of attention heads to understand **HOW** attention heads assist MLPs in arithmetic calculation tasks.

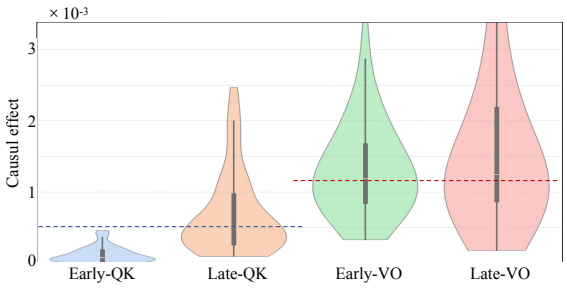


Figure 6: The causal effects distribution of attention heads.

6.2 Implementation Details

Previous work on simple arithmetic operations has found that key attention heads often transfer operand information to the last token. As these heads are directly related to the generation of results by the late MLPs, we refer to these heads as **late heads**. Correspondingly, we refer to the key heads identified in this work that do not transfer information to the last token as **early heads**. This classification helps to discuss the functionalities of attention heads more clearly in the following sections.

Based on the token-wise computational graph and EAP-IG algorithm, we note that the relevant edges of attention heads at certain token positions have high causal effects, meaning that these heads primarily function to transport information between these positions. Therefore, we specifically distinguish early heads and late heads based on whether the edge from the attention head to the last token position MLP (the red edge in Figure 2) has the highest causal effect of all output edges.

For different types of attention heads, we analyzed the intersection-over-union (IoU) of the subsets activated by different expressions. Due to the significant differences in the sets of attention heads relied upon by different expressions, we provide examples to illustrate the patterns of different types of attention heads and discuss their functionalities by category.

6.3 Result and Discussion

6.3.1 Comparing early and late heads.

By analyzing the causal effect differences between $< q, k >$ and $< v, o >$, we can assess the importance of the QK circuit and OV circuit (Elhage et al., 2021) in the information transport function of these heads. Here $< o >$ means the output of the attention heads. The results are shown in Figure 5 & 6.

From Figure 5 & 6, we observe that:

1) For all attention heads, *the causal effect of the VO circuit is generally higher than that of the QK circuit*. This is expected because the VO circuit directly determines the output of the attention head, while the QK circuit only determines the attention pattern.

2) Comparing the QK and VO circuits of different types of attention heads, we find that *the causal effect of the QK circuit of early heads is much lower than that of late heads*. This indicates that the information transport of early heads can be completed mainly by the OV circuit, while the information transport of late heads mainly depends on the full cooperation of the QK and VO circuits.

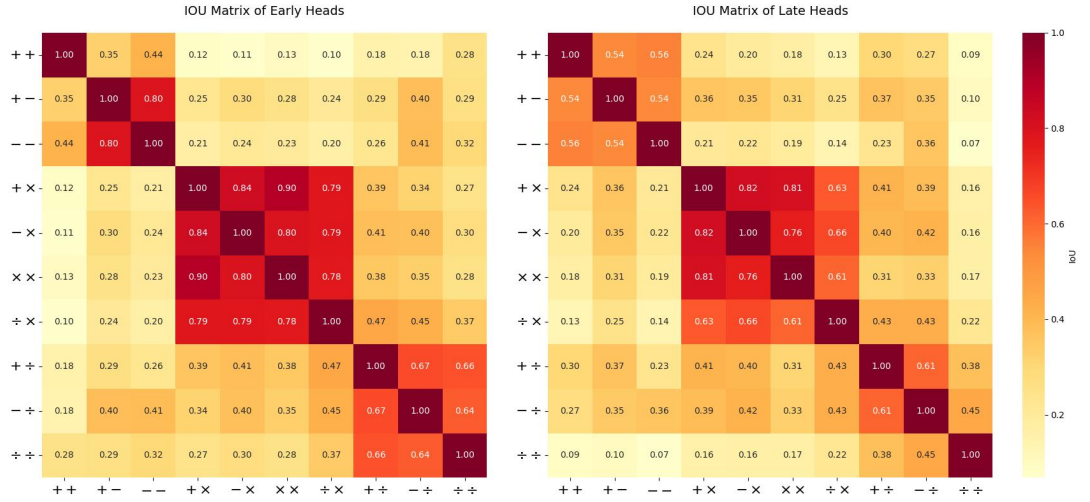
Why does this difference occur? We believe this is related to the functionality of the attention heads. Early heads transfer information within a fixed input token sequence, so the transfer path is relatively fixed, and the QK circuit plays a minor role. In contrast, the functionality of late heads is directly related to generating the next token, which requires *"querying"* information from the context, so the QK circuit plays a more critical role.

3) *Early heads are distributed before layer 28, while late heads are mainly distributed between layers 23 and 37*. Early heads correspond to the region where the MLP resolution rate decreases slowly, while late heads are located before the region where the IR and FR resolution rates rise rapidly. This verifies the reliability of the MLP function analysis.

6.3.2 Comparing different expressions.

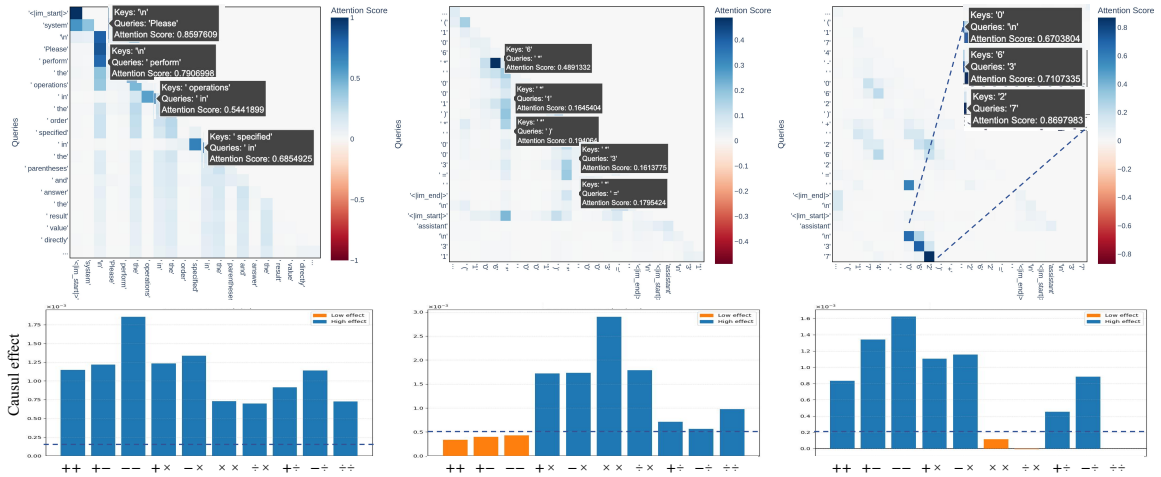
According to the types of operators involved, we divide all expressions in $\mathcal{D}_2$ into 10 categories. We use a threshold of 1e-5 to classify high causal effect heads. Figure 7 shows the IoU of the high causal effect attention head sets for different types of expressions.

From Figure 7, we observe that: *In both early and late heads, there are three high IoU regions*, namely the addition-subtraction related region, the multiplication-related region, and the division-related region. Expressions involving the same operation type often have high IoU. This indicates that the model activates different sets of attention heads according to the operation type to complete the information transport task.



a.The IoU of **Early** Heads b.The IoU of **Late** Heads

Figure 7: The IoU across all expressions of early (a) and late (b) heads causal effects.



a. Head $a^{0,4}$ b. Head $a^{16,17}$ c. Head $a^{32,17}$

Figure 8: The patterns and causal effect differences across all expressions of three representative heads .

6.3.3 Pattern Analysis.

Based on the results of the comparative study, we show the patterns of three representative attention heads in Figure 8 to discuss their roles in mixed arithmetic operations.

From Figure 8, we observe that:

1) Head $a^{0,4}$ focuses on the keywords in the task instructions, indicating that it plays a role in **task understanding**. Moreover, it has high efficiency on all expressions, indicating that it plays a role in all different expression tasks. This is expected because its function is independent of the expression.

2) Head $a^{16,17}$ is an early head related to multiplication. It builds attention between operators and operands in multiplication expressions and has high efficiency only in expressions involving multiplication and second-highest efficiency in expressions involving division. This is consistent with the results of the comparative study.

Moreover, its pattern indicates that **the function of early heads is to parse expressions**. We believe this is the main reason for the slow decrease in the resolution rate of the operand values in the mid-term residual flow, i.e., early heads inject information from other token positions into the residual flow to establish a global understanding of the operands.

3) Head $a^{32,17}$ is a late head related to addition and subtraction. It builds attention from the last token to the operands in addition and subtraction expressions. It has high efficiency in all expressions involving addition and subtraction, while having zero or even negative efficiency in expressions involving multiplication and division.

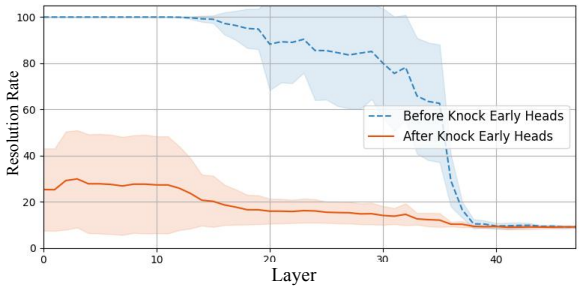


Figure 9: The resolution rates without early heads.

Settings	$v(X_1 - X_2)$		$v(X_3 - X_4)$		FR	
	R	L	R	L	R	L
W/o Knock	47%	32.1	36%	32.1	81%	35.7
m33,K16	35%	33.6	33%	33.4	27%	36.4
m33,K32	27%↓	34.7	21%↓	34.7	26%↓	36.0
m36,K16	50%	32.4	44%	32.9	26%	35.7
m36,K32	47%	32.9	42%↑	32.9	21%↓	35.0

Table 1: The results of knockout late heads. **R** represents the average resolution rate, and **L** represents the average first resolution position. K16 and K32 represent knocking out the top 16 and 32 heads, respectively. m33 and m36 represent the m^{33}_{last}-heads and m^{36}_{last}-heads, respectively.

This phenomenon is consistent with the results of the comparative study. Its pattern proves that **the function of late heads is to transport operand information to the last token** to assist the MLP in arithmetic reasoning.

7 Knockout and Enhancement

7.1 Knockout Key Heads

7.1.1 Method and Details

We selected a representative and complex expression $(X_1 - X_2) + (X_3 - X_4)$ for the knockout experiments. Details regarding the expression selection are provided in the Appendix F.1.

As shown in Figure 3.b, we first obtain the activation values of the attention heads from a perturbed run and patch the activation values to the corresponding positions of the clean run as the knockout run. Then we patch the residual activation values of the knockout run to zero expression to detect the IR and FR.

7.1.2 Knockout Early Heads

First, we conducted knockout experiments on early heads to observe the changes in the resolution rate of the operands after knockout. The results are shown in Figure 9. *Knocking out early heads generally leads to a decrease in the resolution rate of the operand values in the residual flow.* This is because, without the attention heads responsible for task understanding in the early heads, the model cannot correctly establish a proper understanding of the operands. **This indicates that the MLP's representation of the operands depends on the collaboration of the early heads.**

7.1.3 Knockout Late Heads

To demonstrate that the late MLP relies on IRs for reasoning by knocking out late heads, we performed a detailed classification of the late heads.

As shown in Table 1, first row, for this expression, the IRs are mainly parsed at m^{33}_{last}, while the FRs are primarily parsed at layer 36 and beyond. We refer to the late heads that are most important for these two MLPs as m^{33}_{last}-heads and m^{36}_{last}-heads, respectively. More information about m^{33}_{last}-heads and m^{36}_{last}-heads is provided in the Appendix F.2.

By knocking out m^{33}_{last}-heads and m^{36}_{last}-heads separately, we observed the impact of these heads on the generation of IRs and FRs. The results are shown in Table 1.

From Table 1, we observe that:

1) *Knocking out m^{33}_{last}-heads leads to a decrease in the resolution rates of both the IRs and FRs*, even though these heads have a lower effect on m^{36}_{last}. **This indicates that m^{36}_{last} relies on the information from m^{33}_{last} to generate the FRs.** This is also validated by the effects between MLPs shown in Appendix F.3.

2) *Knocking out m^{36}_{last}-heads reduces the resolution rate of the final results without affecting the resolution of the IRs from m^{33}_{last}*. This indicates that m^{33}_{last} does not require these heads to generate IRs, even if some heads of them are upstream of the 33rd layer MLP.

7.2 Adaptive Enhancement

7.2.1 Implementation Details

We design fine-tuning experiments to explore the impact of the identified calculation pathway on the model's performance. Due to the limited diversity of synthetic data, we used $\mathcal{D}_2$ and $\mathcal{D}_3$ as the training sets and $\mathcal{D}_{4-7}$ as the test sets. We evaluate the robustness of the model's abilities by testing the model's generalization ability on more challenging data in a zero-shot setting.

We performed fine-tuning by setting the learning rate based on the causal effects obtained from

Tuning Setting	Test Acc (Zero-shot)			
	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_7$
No Tunig	45%	29%	18%	15%
Full Tunig	77%	52%	30%	22%
Precise Tunig	82%	55%	32%	24%
CEDF	**86%**	**64%**	**45%**	**33%**

Table 2: Test accuracy of different tuning settings.

EAP-IG, called **C**ausal **E**ffect **D**riven **F**ine-tuning method (CEDF). All results are obtained based on small LLMs due to the limited diversity of synthetic data. More details are provided in the Appendix G.

7.2.2 Results and Discussions

As shown in Table 2, CEDF significantly outperforms full fine-tuning and precise fine-tuning (Zhang et al., 2024), demonstrating the effectiveness of the identified key components.

The experimental results indicate that the key components can more robustly enhance reasoning ability, which shows that it is these components that determine the model's ability to perform arithmetic reasoning robustly, rather than simply memorizing expressions.

8 Conclusion

Do LLMs execute mixed arithmetic calculations in the same rule as human? Our work thoroughly investigates how LLMs complete mixed arithmetic tasks to answer this question. On the one hand, the model **follows a workflow different from humans to complete this task**, namely task understanding, operand representation, expression parsing, information aggregation, and arithmetic *"reasoning"*. On the other hand, the model exhibits patterns **similar to human thinking in many details**, such as understanding the correlations between different operator types and relying on intermediate results to generate the final result at the last token. In summary, our answer is that neither memory nor reasoning accurately applies.

Our work provides new insights into understanding the mathematical reasoning capabilities of LLMs and offers new inspirations for future research, which will help improve the reliability of LLMs.

Limitations

Due to methodological limitations, we did not explore the impact of CoT on the model, which is the popular technology of current reasoning models. This is beyond the scope of this paper, and we will explore it in future work.

Due to computational resource limitations, our experiments were only conducted on small models with a parameter scale of around 10B. In future work, we will conduct experiments on larger models.

The diversity of synthetic data is insufficient, and our conclusions may not fully apply to mathematical reasoning problems with complex descriptions. In future work, we will investigate complex mathematical reasoning problems.

Due to limited resources and space, the proposed CEDF method has not been validated on larger LLMs.

Ethics and Risks

This paper mainly focuses on interpreting how the LLM to complete the mixed arithmetic calculation task without CoT. Our goal is to enhance model arithmetic reasoning ability by understanding why they perform such performance first, thereby advancing the development of more reliable and robust LLMs. Nonetheless, the findings in this paper about LLM inner mechanisms may be misused by malicious actors to attack or mislead LLMs. Therefore, we stress the importance of increased oversight by relevant authorities concerning the applications of LLMs.

Acknowledgments

This work is supported by the grants from the National Natural Science Foundation of China (No. U21B2009 & 62376130). The authors would like to thank the organizers of ACL 2025 and the reviewers for their helpful suggestions.

References

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. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.1, knowledge storage and extraction. In *Forty-first International Conference on Machine Learning*.

Reuben Baron and David Kenny. 1986. The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology*, 51:1173–1182.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. 2024. SelfIE: Self-interpretation of large language model embeddings. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 7373–7388. PMLR.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. 2023. A toy model of universality: Reverse engineering how networks learn group operations. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 6243–6267. PMLR.

Chunyuan Deng, Zhiqi Li, Roy Xie, Ruidi Chang, and Hanjie Chen. 2024. Language models are symbolic learners in arithmetic. *Preprint*, arXiv:2410.15580.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Cantón Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab A. AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriele Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guanglong Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Laurens Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Ju-Qing Jia, Kalyan Vasuden Alwala, K. Upasani, Kate Plawiak, Keqian Li, Ken-591 neth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuen ley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Babu Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melissa Hall Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri S. Chatterji, Olivier Duchenne, Onur cCelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasić, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Chandra Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whit ney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yiqian Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zhengxu Yan, Zhengxing Chen, Zoe Papakipos, Aaditya K. Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adi Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Ben Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Shang-Wen Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzm'an, Frank J. Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory G. Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Han Zha, Haroun Habeeb, Harrison Rudolph,

Helen Suk, Henry Aspegren, Hunter Goldman, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kaixing(Kai) Wu, U KamHou, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, A Lavender, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollár, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sung-Bae Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Andrei Poenaru, Vlad T. Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xia Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Preprint*, arXiv:2209.10652.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Atticus Geiger, Hanson Lu, Thomas F Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models. *arXiv preprint*. ArXiv:2401.06102.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing Model Behavior with Path Patching. *arXiv preprint*. ArXiv:2304.05969.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 76033–76060. Curran Associates, Inc.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have Faith in Faithfulness: Going Beyond Circuit Overlap When Finding Model Mechanisms. *arXiv preprint*. ArXiv:2403.17806 [cs].

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. 2024. How large

language models encode context knowledge? a layer-wise probing study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8235–8246, Torino, Italia. ELRA and ICCL.

Charles M. Judd and David A. Kenny. 1981. Process analysis: Estimating mediation in treatment evaluations. *Evaluation Review*, 5(5):602–619.

Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. 2023. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *Preprint*, arXiv:2307.09458.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit Component Reuse Across Tasks in Transformer Language Models. *arXiv preprint*. ArXiv:2310.08744.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models.

Philipp Mondorf and Barbara Plank. 2024. Beyond accuracy: Evaluating the reasoning behavior of large language models – a survey. *Preprint*, arXiv:2404.01869.

Neel Nanda. 2022. Interpreting gpt: The logit lens. Accessed: 2025-05-20.

Neel Nanda. 2023. Attribution Patching: Activation Patching At Industrial Scale.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic Without Algorithms: Language Models Solve Math With a Bag of Heuristics. *arXiv preprint*. ArXiv:2410.21272 [cs].

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C Wallace, and David Bau. 2023. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 548–560.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*, 2nd edition. Cambridge University Press, USA.

Philip Quirke and Fazl Barez. 2024. Understanding Addition in Transformers. *arXiv preprint*. ArXiv:2310.13121.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Naomi Saphra and Sarah Wiegreffe. 2024. Mechanistic? *Preprint*, arXiv:2410.09087.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A Mechanistic Interpretation of Arithmetic Reasoning in Language Models using Causal Mediation Analysis. *arXiv preprint*. ArXiv:2305.15054.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas

Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

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

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *Preprint*, arXiv:2407.06023.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. 2024. Interpreting and Improving Large Language Models in Arithmetic Calculation. *arXiv preprint*. ArXiv:2409.01659.

A Task Instruction

In this paper, the task instruction we used is shown in Figure 10.

Default Mixed Expression Form
System: Please answer the result value directly, don't output any other word.
User: (523 - 054) + 172 =
Assistant: 641

Question Form
System: Please answer the result value directly, don't output any other word.
User: x = (523 - 054) + 172, what is the value of x?
Assistant: 641

Systems of Equations Form
System: Please answer the result value directly, don't output any other word.
User: It is known that x, y satisfy the following equations: x = 523 - 054, y = 172 + x. What is the value of y?
Assistant: 641

Complex Context Form
System: Please answer the result value directly, don't output any other word.
User: The initial price of WMT stock is $523. In the first day, the stock price decreased by $054. In the second day, the stock price increased to 002 times its original value. What is the final price of the WMT stock?
Assistant: $938

Figure 10: The task instruction used in this paper.

B Methods Details

B.1 EAP-IG

B.1.1 Transformer Circuits

Transformer Circuits is an interpretability framework for analyzing the internal mechanisms of Transformer models (Elhage et al., 2021). It conceptualizes the model as a circuit composed of interconnected computational components, where information flows along specific pathways to perform tasks. This framework, developed by researchers at Anthropic, views a Transformer as a directed graph where attention heads and MLP layers form nodes, with their connections constituting edges through which information flows.

The core components in this framework include: (1) attention heads decomposed into Query-Key (QK) circuits that determine where information flows, and Value-Output (VO) circuits that determine what information is passed; (2) MLP layers that transform information within the residual stream; and (3) residual connections that maintain information flow throughout the network.

A central insight of this approach is the compositionality of model behavior—complex behaviors emerge from interactions between simpler components. For instance, in arithmetic tasks, certain attention heads may specialize in passing operand information, while others focus on parsing operators, with late-layer MLPs ultimately performing computations (Zhang et al., 2024).

Circuit analysis enables identifying critical path-

ways responsible for specific model capabilities, revealing which components contribute significantly to particular outputs. This understanding facilitates both interpretability and targeted interventions like the ones employed in our study to analyze the internal mechanisms used by LLMs to perform mixed arithmetic operations.

B.1.2 Causal Mediation Analysis

Causal Mediation Analysis is a statistical approach for investigating how an independent variable affects an outcome variable through intermediate variables (mediators) (Judd and Kenny, 1981; Baron and Kenny, 1986; Pearl, 2009). In the context of neural networks, it involves interventions on specific model components to observe how these interventions affect the model's outputs, thereby establishing causal relationships between components and outcomes. Causal mediation introduces the concept of counterfactuals: "What would the output be if a specific component behaved differently?", allowing researchers to isolate the functional contribution of specific neural network components.

B.1.3 Activation Patching

Activation patching is a key component identification or circuit detection method based on causal mediation analysis. It achieves this by implementing key patching operations to perform causal interventions and assess the strength of the resulting causal effects. Many studies identify circuits using activation patching (Vig et al., 2020; Geiger et al., 2021), which replaces a (clean) edge activation with a corrupted one during the model's forward pass. If the resulting change in the metric is greater than some threshold τ, the edge is added to the circuit.

B.1.4 Edge Attribution Patching (EAP)

(Nanda, 2023) automate and accelerate the causal scores calculations by a first-order approximation. Given an edge $e = (u, v)$ with clean and corrupted activations z_u and z'_u, EAP approximates the change in loss L caused by corrupting e as

$$(z'_u - z_u)^\top \nabla_v L(s), \tag{7}$$

$$L(x) = -M(x) \tag{8}$$

where $\nabla_v L(s)$ is the gradient of L with respect to the *input* of v. Note that EAP estimates the change in loss L, which is always set to $-M(x)$. The change in loss (Approximate causal effects) for each edge can be computed with one forward pass on corrupted inputs and one forward and backward pass on clean inputs. Therefore, we can select which edges fall in the circuit based on the change in loss.

B.1.5 EAP with Integrated Gradients (EAP-IG)

IG aims to solve a problem that affects both the gradient × input method and EAP: zero gradients. The authors of the work (Sundararajan et al., 2017) note that if one of the model's internal activations has a zero gradient at s, that activation will not contribute to the attribution, even if the activation has a non-zero gradient at s', and the difference in activations is significant. IG resolves this issue by cumulating the gradients along the straight-line path from s' to s. We use a more detailed variant of EAP-IG, as shown in Equation 6.

B.2 Patchscopes

Patchscopes is a method that explains the model's hidden representations by patching. It consists of the following steps.

1. Run the forward computation on the source prompt in the source model (the green model in Figure 3.a) and cache the source layer's representation. In Figure 3.a, the source prompt is "3 + (6 − 2) =".

2. Run the forward computation on the target prompt up to the target layer in the target model (the yellow model in Figure 3.a). In Figure 3.a, the target prompt is "0 + (0 − 0) =".

3. Patch the target representation of "=" at the target layer, replacing it with the cached representation (from step 2), and continue the forward computation from that layer onward.

By observing the target model's output after patching, we can understand what information is stored in the cached representation. In this work, we determine whether the cached representation stores operands, intermediate results, or final results by observing whether the expected result is output by the target model.

C More Key Component Detection Details

C.1 Circuit Performance

Table 3 shows the performance of the circuits detected on different expression tasks. As can be

Opteration Types	KL Div		Prob Diff	
	Full	Top1%	Full	Top1%
++	1.21	1.09	1.00	0.99
+−	1.16	1.05	0.99	0.98
−−	1.09	0.98	0.98	0.96
+×	1.01	0.86	0.92	0.89
−×	1.23	1.03	0.90	0.87
××	1.23	1.04	0.99	0.98
÷×	1.23	1.04	0.95	0.92
+÷	0.91	0.78	0.78	0.75
−÷	0.95	0.82	0.82	0.79
÷÷	1.09	0.93	0.91	0.88

Table 3: The circuit performance on different expressions.

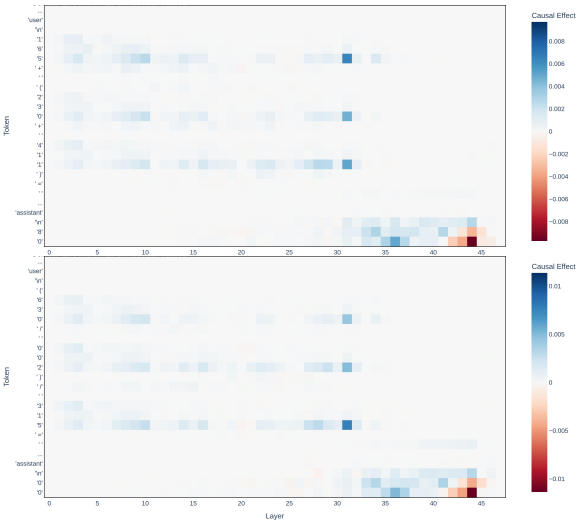


Figure 11: The distribution of causal effects across all MLPs on *(X1 <op> X2) <op> X3* and *X1 <op> (X2 <op> X3)*.

seen, our method effectively detects key components across various models.

C.2 Parenthesis Position Effect

By comparing the key component detection results in Figure 11 of expressions with parentheses at different positions, we can see that the position of the parentheses does not significantly affect the distribution of key components. This indicates that the model uses the same components to parse parentheses at different positions.

C.3 Other Models Results

We ran EAP-IG on Phi-4, Gemma-2-9B-IT, and Llama-3.1-8B-Instruct and obtained similar results. This indicates that our method is reliable.

Opteration Types	KL Div		Prob Diff	
	Full	Top1%	Full	Top1%
Llama-3.1-8B-Instruct				
+	0.03	0.03	0.12	0.11
−	0.03	0.02	0.15	0.14
×	0.05	0.04	0.29	0.26
÷	0.05	0.04	0.26	0.25
Phi-4				
+	0.24	0.20	0.50	0.44
−	0.14	0.12	0.37	0.33
×	0.18	0.15	0.47	0.44
÷	0.09	0.07	0.26	0.25
Gemma-2-9B-IT				
+	0.23	0.22	0.65	0.64
−	0.21	0.20	0.62	0.62
×	0.21	0.20	0.64	0.63
÷	0.19	0.18	0.53	0.52

Table 4: The circuit performance on different expressions of three other models.

Circuit Performance Table 4 shows the performance of the circuits detected on different models. As can be seen, our method effectively detects key components across various models. Probability difference value is positively correlated with the model's predictive performance on the expression, and the performance of the three models is significantly different from that of Qwen2.5-14B. In the following functional analysis, we will only focus on Gemma-2-9B-IT.

Key Components Distributions Here we show the distributions of key components for three models, as shown in Figure 14,13 and 12. We find that the distributions of key components for different models have similar characteristics, and the discussions in the main text hold for these models.

D More Discussion about IR

D.1 Expressions Tpyes

According to the trend of IR, we divide the expressions into three types. The first type is that the IR first rises and then falls; the second type is that it only rises with the FR, and the third type is that it remains almost unchanged. In the following Figure 15, we show the other two different trends.

The specific expressions represented by each type are as Table 5.

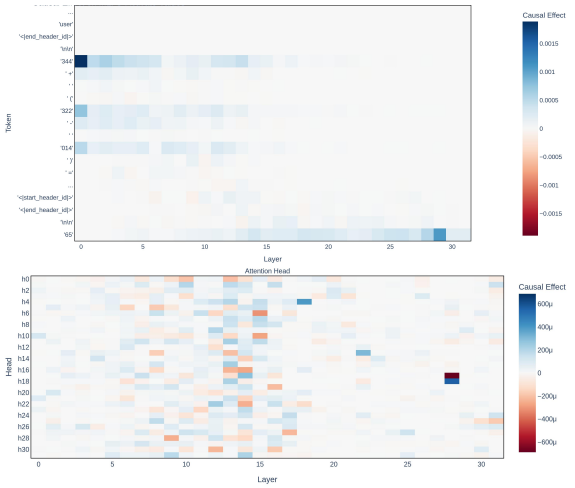


Figure 12: The distribution of causal effects across all MLPs (a) and all attention heads (b) of Llama-3.1-8B-Instruct.

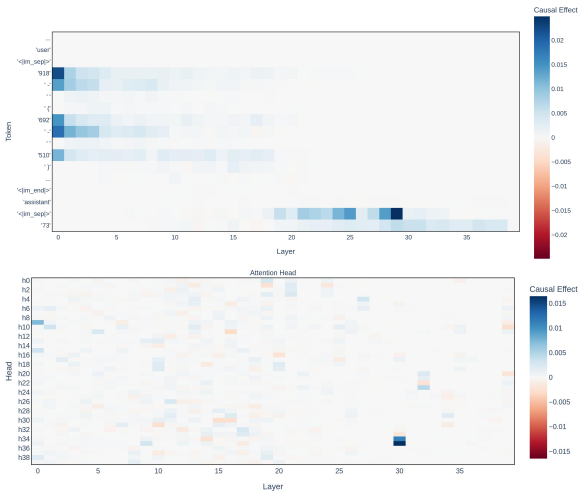


Figure 13: The distribution of causal effects across all MLPs(a) and all attention heads(b) of phi-4.

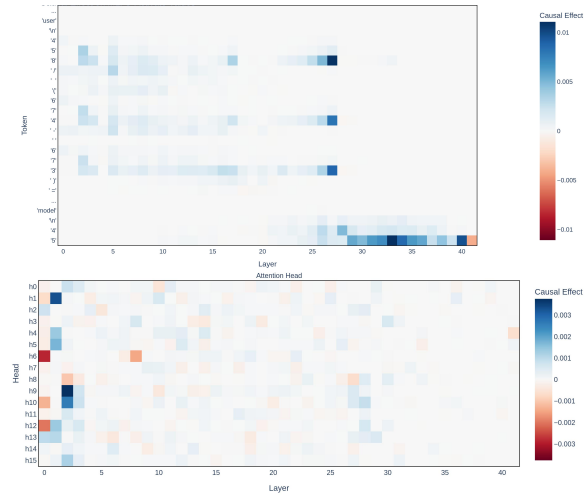
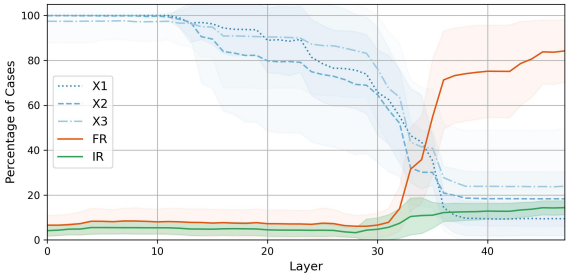
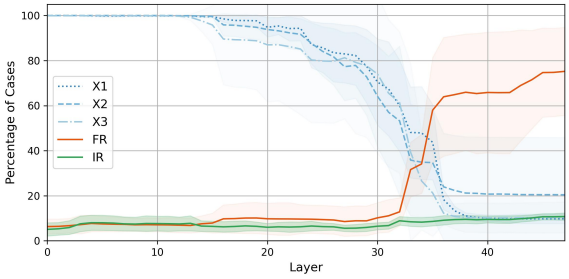


Figure 14: The distribution of causal effects across all MLPs (a) and all attention heads (b) of gemma-2-9B-IT.



a. Type 2

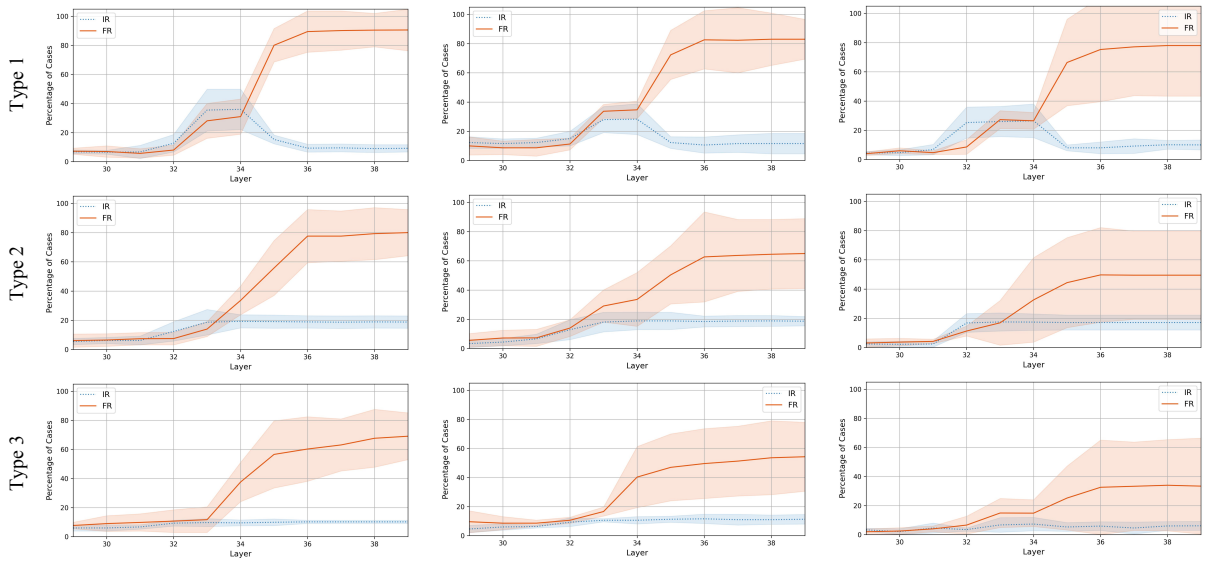


b. Type 3

Figure 15: The resolution rates on type-2 and type-3 expressions.

Expression Types		
Type 1		
$(a+b)+c$	$a+(b+c)$	$a+(b-c)$
$(a-b)+c$	$a+(b\times c)$	$a+(b\div c)$
$(a\times b)+c$	$a-(b\times c)$	$a\div(b+c)$
Type 2		
$(a-b)-c$	$(a-b)\times c$	$(a-b)\div c$
$(a\times b)-c$	$(a\times b)\times c$	$(a\times b)\div c$
$(a\div b)+c$	$(a\div b)\times c$	$(a\div b)\div c$
$a\times(b+c)$	$a\times(b-c)$	$a\times(b\times c)$
$a\times(b\div c)$	$a\div(b-c)$	$a\div(b\times c)$
$a\div(b\div c)$		
Type 3		
$(a+b)\div c$	$(a+b)-c$	$(a\div b)-c$
$(a+b)\times c$	$a-(b\div c)$	$a-(b+c)$
$a-(b-c)$		

Table 5: The expressions of different types.



a. Question Form b. Systems of Equations Form c. Complex Context Form

Figure 16: The resolution rates on type-1, type-2, and type-3 expressions across three different task forms.

D.2 Other Expression Forms

To ensure the robustness of our conclusions, we verified the trend of IR changes on more task forms. We found that the trend of IR changes is similar across different task forms. Here, we present the IR change trends for three different expression forms: Question Form, Systems of Equations Form, and Complex Context Form. The templates used are shown in Figure 10, and the trends of IR changes are shown in Figure 16.

As shown in Figure 16, compared to other forms, the mean FR resolution rates on the Complex Context Form for all types of expressions significantly decrease, and the standard deviation increases notably. Considering that this task requires a deeper understanding of the context, it is expected that the resolution rates are lower and more unstable for such challenging tasks. More importantly, these resolution rate trends across all task forms in Figure 16 are well-aligned with the conclusions we obtained from the default mixed arithmetic expression task.

D.3 Validated by Other Detection Methods

To avoid the potential limitations of Patchscopes, we used the Logit Lens (Geva et al., 2022; Nanda, 2022) to detect intermediate results. The results are shown in Figure 17, which are consistent with those obtained from Patchscopes, further demonstrating the robustness of our conclusions.

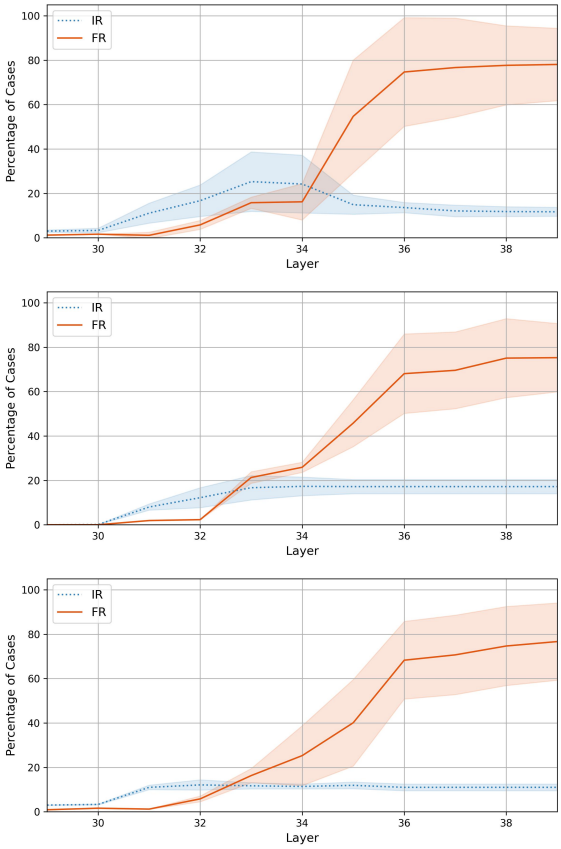


Figure 17: The resolution rates on type-1, type-2, and type-3 expressions using the Logit Lens.

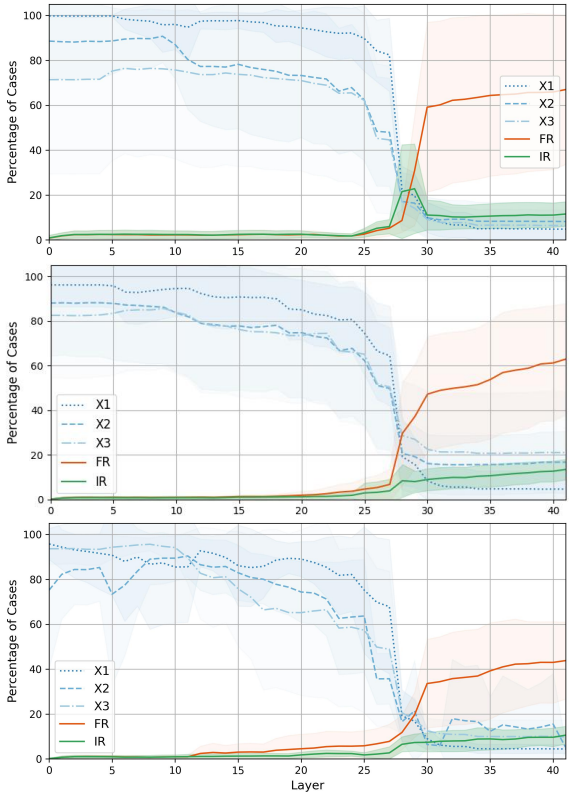


Figure 18: The resolution rates on all types of expressions using Gemma-2-9B-IT.

D.4 Other Models Results

We analyzed Gemma-2-9B-IT using the same method, and the results are shown in Figure 18. The results show a similar trend to those of Qwen. This indicates that our conclusions are reliable.

E More Pattern Analysis

Due to the diversity of the distribution of key attention heads on different expressions, we only show more patterns using the expression $(X_1 - X_2) + (X_3 - X_4)$ as an example to provide more evidence for the previous observations.

E.1 More Early Head Patterns

Figure 19 illustrates the attention patterns of several typical early attention heads. We observe that these attention heads are functionally specialized, and their primary functions can be inferred from their attention patterns.

Head $a^{0,4}$ primarily transports information from "operations" and "specified" to "in", indicating its role in understanding the task instruction.

Head $a^{14,2}$ transports information between parentheses and operators, suggesting its role in parsing the expression range corresponding to the operator.

Head $a^{16,19}$ transports information between different operands, indicating its role in understanding the relationships between operands.

Head $a^{19,19}$ transports operator information to the second operand, suggesting its role in linking the operator type to its second operand.

Head $a^{19,35}$ transports information about the second operand to the right parenthesis, aiding in the comprehensive understanding of the subexpression.

E.2 More Late Head Patterns

Figure 20 illustrates the attention patterns of several typical late attention heads.

Head $a^{33,32}$ and $a^{32,17}$ established an attention relationship between the operands of X2, X4, and the last token, indicating their important role in transmitting operand information to the final token. At the same time, the operands they transfer correspond to the IR parsed by m^{33}_{last}, reflecting the further functional differentiation of later heads.

Head $a^{36,39}$ and $a^{35,11}$ established an attention relationship between the operands of X1, X3, and the last token, the operands they transfer correspond to the FR parsed by m^{36}_{last}.

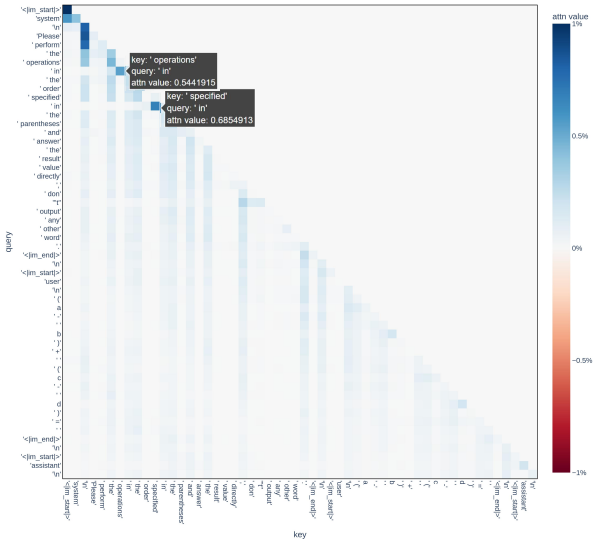
F More Knockout Details

F.1 Expression Selection

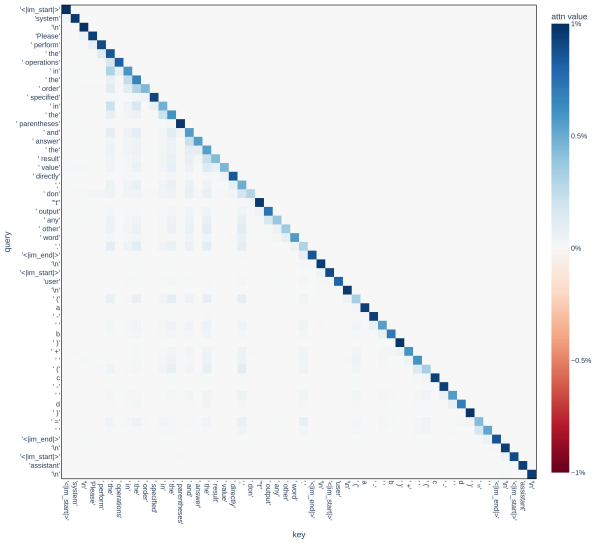
To verify whether the conclusions obtained on $\mathcal{D}_2$ hold for more complex expressions, we selected a more complex expression $(X_1 - X_2) + (X_3 - X_4)$. This expression includes both addition and subtraction operations and two parentheses, corresponding to two IRs, which can more reliably verify whether the model relies on IRs for arithmetic reasoning. We do not involve multiplication and division operations because the model's performance is limited on expressions involving multiplication and division in $\mathcal{D}_3$.

F.2 Key Late Heads Distributions

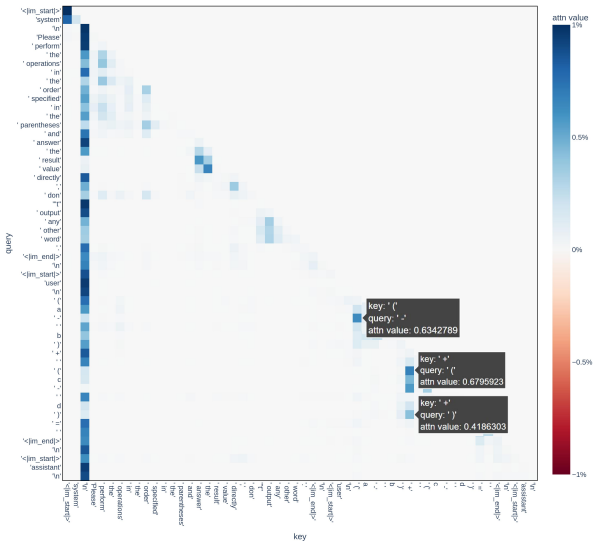
Figure 21 shows the most important attention heads in the upstream nodes of the last token at the 33rd and 36th MLP layers. The patterns of the two most important attention heads in m^{33}_{last}-heads, $a^{33,32}$ and $a^{32,17}$, are shown in Figure 20a and b, respectively, and the patterns of the two most important attention heads in m^{36}_{last}-heads, $a^{36,39}$ and $a^{35,11}$, are shown in Figure 20c and d, respectively.



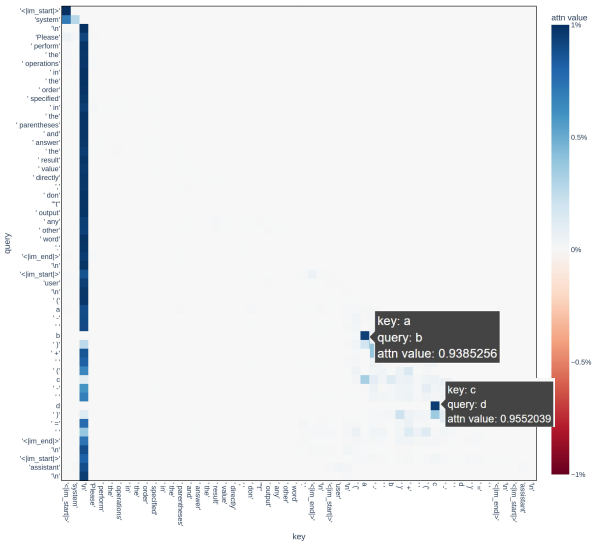
a. Head $a^{0,4}$ attention pattern.



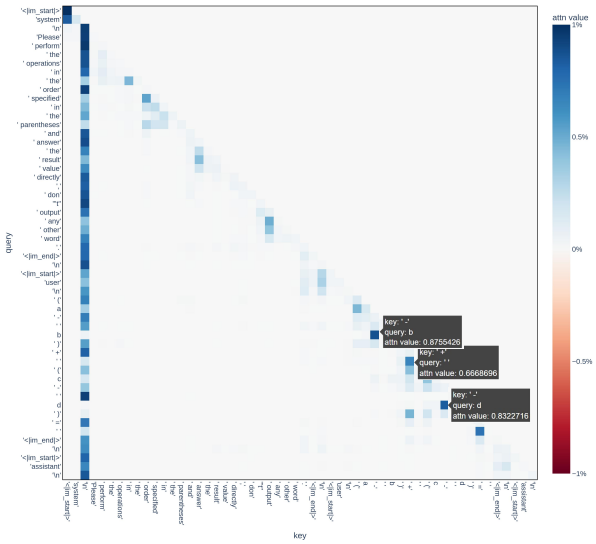
b. Head $a^{0,27}$ attention pattern.



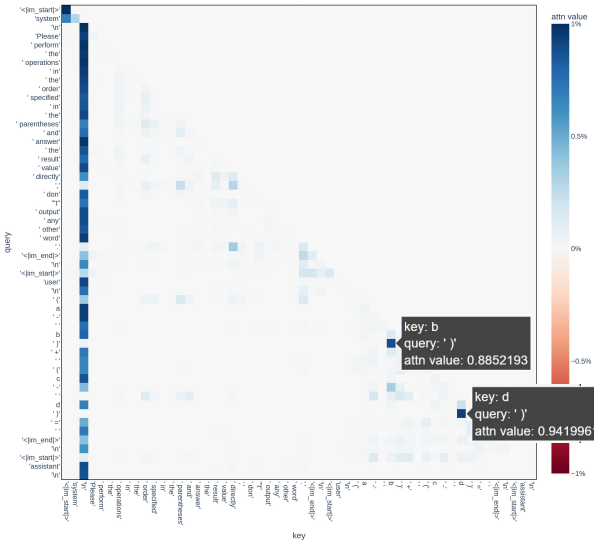
c. Head $a^{14,2}$ attention pattern.



d. Head $a^{16,19}$ attention pattern.

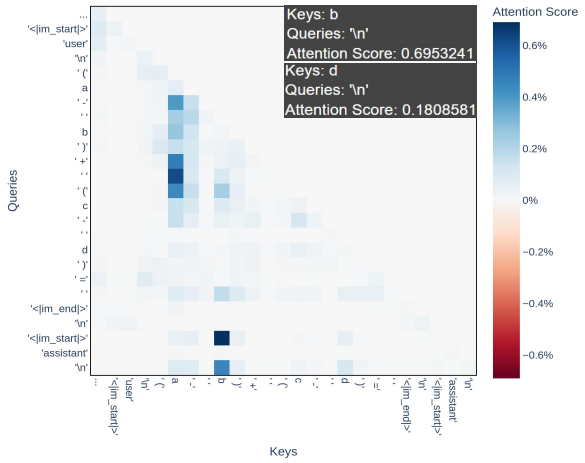


e. Head $a^{19,19}$ attention pattern.

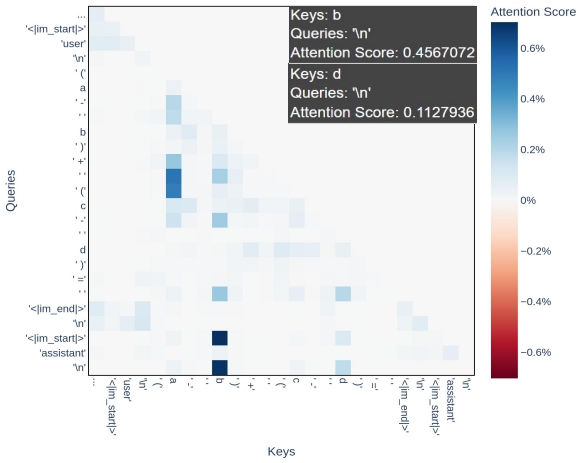


f. Head $a^{19,35}$ attention pattern.

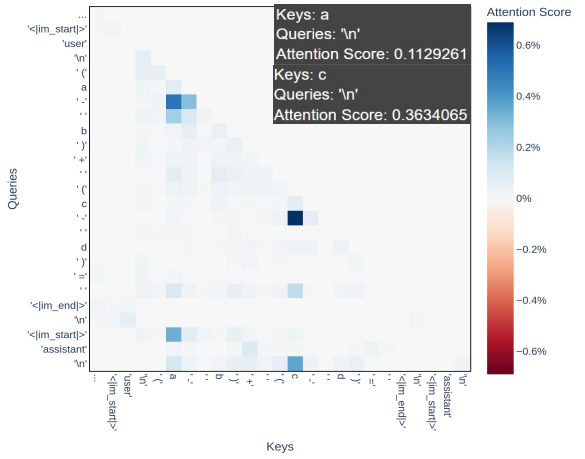
Figure 19: Six typical attention patterns of early attention heads.



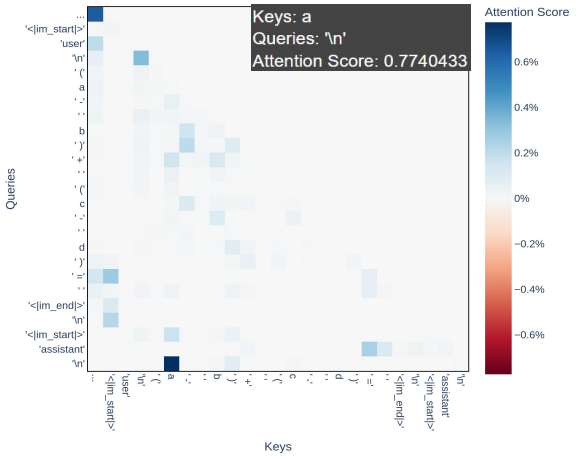
a. Head $a^{33,32}$ attention pattern.



b. Head $a^{32,17}$ attention pattern.



c. Head $a^{36,39}$ attention pattern.



d. Head $a^{35,11}$ attention pattern.

Figure 20: Four typical attention patterns of late attention heads.

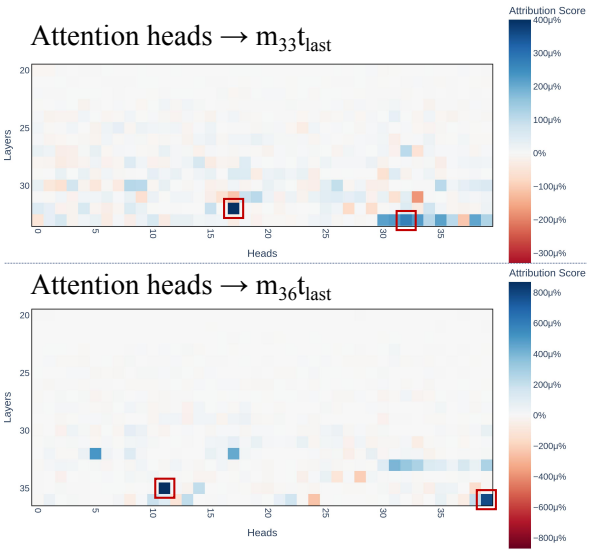


Figure 21: The distribution of key late heads related $m_{33}t_{last}$ and $m_{36}t_{last}$ on $(X_1 - X_2) + (X_3 - X_4)$.

$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_7$
2530	8145	1600	3200	8600	12800

Table 6: The Statistics of $\mathcal{D}_{2-7}$. The sample numbers of $\mathcal{D}_{2-3}$ are the training sets, while sample numbers of $\mathcal{D}_{4-7}$ are the test sets.

F.3 Causal Effect between $m_{33}t_{last}$ and $m_{36}t_{last}$

Figure 22 shows the causal effect scores between the last token of $m_{33}t_{last}$ and $m_{36}t_{last}$. As can be seen, $m_{33}t_{last}$ is the most important positive input source node for $m_{36}t_{last}$, which verifies the dependence of $m_{36}t_{last}$ on $m_{33}t_{last}$.

G Adaptive Enhancement Details

G.1 Dataset Details

Since large language models struggle to correctly handle multi-digit numbers, in the enhancement

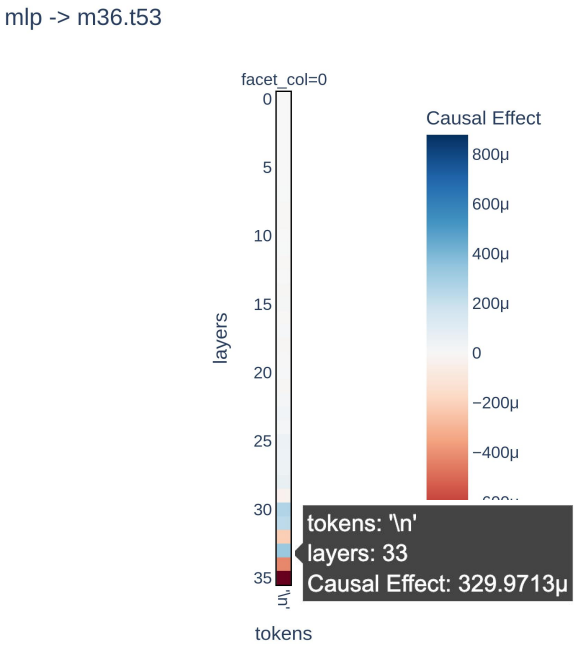


Figure 22: The input nodes causal effects distribution of $m_{36}t_{last}$.

experiments, we limit the values of operands, intermediate results, and final results to $[0, 10)$, allowing the model to focus on enhancing arithmetic reasoning performance rather than understanding multi-digit numbers.

Due to the large number of expressions in $\mathcal{D}_{4-7}$, we randomly selected 100 samples from each prototype to construct the test set (see Table 6). The random sampling was fixed for reproducibility, and all tuning experiments used these samples.

We set the batch size to 40 and the number of epochs to 10 in all experiments in this work. All results are averaged over independent 3 runs.

G.2 Training Details

In the CEDF method, the learning rate for each component (MLP or attention head) is calculated according to the following formula:

$$lr_g = lr_0 \times \frac{s_g - s_{min}}{s_{max} - s_{min}} \tag{9}$$

where lr_g is the learning rate of component g, lr_0 is the global learning rate in full fine-tuning, s_g is the causal effect score of component g, and s_{min} and s_{max} are the minimum and maximum causal effect scores in the computation graph.

G.3 Ablation Study

Our method adds no extra modules or hyperparameters, relying only on the learning rate. We ablate the backbone model scale and learning rate, with results shown in Table 7. CEDF is insensitive to both the learning rate and the backbone model.

When the learning rate is set to 1e-4, full fine-tuning fails to converge. However, since many components' effective learning rates are lower than lr_0, CEDF still performs acceptably, demonstrating its robustness to learning rate scaling.

G.4 Resource Requirements

All experiments were conducted on a NVIDIA A800 GPU, with each experiment taking approximately 4 hours on average.

Backbones	Learning Rate lr_0	Setting	Test Acc (Zero-shot)			
			$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_7$
Backbones Sensitivity						
Qwen2.5-0.5B	5e-5	Full SFT	76%	51%	31%	22%
Qwen2.5-0.5B	5e-5	CEDF	80%	56%	35%	26%
Qwen2.5-3B	5e-5	Full SFT	83%	56%	34%	25%
Qwen2.5-3B	5e-5	CEDF	87%	61%	37%	28%
llama3.2-1B	5e-5	Full SFT	71%	48%	29%	21%
llama3.2-1B	5e-5	CEDF	78%	56%	35%	26%
llama3.2-3B	5e-5	Full SFT	74%	54%	35%	26%
llama3.2-3B	5e-5	CEDF	**93%**	**73%**	**44%**	**29%**
Learning rates Sensitivity						
Qwen2.5-1.5B	1e-5	Full SFT	73%	52%	33%	25%
Qwen2.5-1.5B	1e-5	CEDF	82%	58%	36%	26%
Qwen2.5-1.5B	5e-5	Full SFT	77%	52%	30%	22%
Qwen2.5-1.5B	5e-5	CEDF	**86%**	**64%**	**45%**	**33%**
Qwen2.5-1.5B	1e-4	Full SFT	12%	12%	12%	11%
Qwen2.5-1.5B	1e-4	CEDF	69%	47%	31%	24%

Table 7: The Results of Ablation Experiments.