# Explaining and Tuning Transformer-based LLMs in Arithmetic Tasks with Human Strategies

**Anonymous ACL submission**

## Abstract

Transformer-based large language models continue to achieve SOTA performance across various natural language processing tasks. However, their subpar performance on seemingly elementary problems, such as basic arithmetic, raises concerns about model reliability, safety, and ethical deployment. In this study, we propose a comprehensive problem analysis framework to explain the underlying mechanisms of vanilla Transformer model trained on integer arithmetic tasks, and tune the model using strategies effective for humans. We begin by decomposing the arithmetic into well-defined subtasks commonly used by humans and conducting loss convergence order analysis together with ablation studies for each subtask. Our findings reveal that LLMs exhibit learning patterns similar to those of humans, with a faster learning speed for simpler subtasks compared to complex ones. In addition, we successfully improved the accuracy of LLMs by applying human arithmetic strategies. These results suggest that transformers may share similar information processing mechanisms to humans in arithmetic. Our work has important implications for enhancing LLMs' arithmetic abilities by applying human strategies and understanding LLMs using explainable AI verifications and comparisons with humans, ultimately fostering trust in LLMs for critical and high-stakes applications.

## 1 Introduction

Recent advances in Natural Language Processing (NLP) have been driven by transformers (Vaswani et al., 2017), the foundation of modern Large Language Models (LLMs). These models excel across diverse tasks, including dialogue, conversational agents, and machine translation (Devlin et al., 2018; Raffel et al., 2020; Touvron et al., 2023; Dziri et al., 2024). The release of models like GPT-4 (Achiam et al., 2023) further fueled excitement by pushing the frontier of Artificial General Intelligence (AGI).

| Type | # Digits | | | | |
|---|---|---|---|---|---|
| | (1,1) | (2,2) | (3,3) | (4,4) | (5,5) |
| + | 100 | 100 | 100 | 99 | 100 |
| - | 100 | 100 | 99 | 96 | 99 |
| × | 99 | 82 | 43 | 13 | 5 |
| ÷ | 94 | 58 | 29 | 6 | 0 |

Table 1: The accuracy (%) of GPT-4 on 1-5 digits arithmetic task. We use the prompt "What's the answer of $a \odot b$?", where $\odot$ denote the operation symbols. Each combination of operator and digit is tested 100 times.

Despite their impressive capabilities, LLMs struggle with basic arithmetic. For instance, GPT-4 (Achiam et al., 2023) performs poorly on simple integer calculations (Dziri et al., 2024), as shown in Table 1. This gap between strong performance on complex tasks and failure on elementary ones not only raises concerns about the reliability and ethical deployment of LLMs, but also highlights fundamental limitations in current LLMs.

Arithmetic tasks differ significantly from typical NLP tasks, most notably in their deterministic outputs. While NLP tasks often permit multiple valid answers and encourage diversity, arithmetic problems typically have a single correct solution. Additionally, NLP tasks can tolerate omissions of less critical input elements, but arithmetic requires precise attention to every digit to ensure accuracy. These tasks also involve complex intermediate steps that must be correctly handled, posing a unique challenge for transformer-based models. Furthermore, unlike the natural left-to-right processing in language, arithmetic operations (e.g., addition, multiplication) often proceed from the least significant digit. Given these distinctions, arithmetic tasks offer a valuable testbed for developing explainable methods and probing the internal mechanisms of Transformer-based LLMs.

Existing eXplainable AI (XAI) methods for Transformers predominantly focus on explainability in specific instances of certain tasks, such as

indirect object identification (Wang et al., 2022) and colored objects (Merullo et al., 2023). While these methods provide valuable insights into how Transformers make decisions, they often fail to capture the intrinsic correlations in complex tasks or the broader decision-making process. Furthermore, the inherent complexity of multilayered self-attention architectures poses significant challenges in explaining how transformers produce specific outputs (Vig, 2019). Providing explanations for Transformers on arithmetic tasks remains an open research challenge, with only a limited amount of work (Dziri et al., 2024; Lee et al., 2023) addressing this area. However, existing approaches largely treat these models as black boxes, offering little insight into their internal mechanisms.

Indeed, recent XAI research has proposed a more user-centric approach to enhance users' understanding of AI. In particular, a theory-of-mind based framework emphasizes the importance of highlighting differences between humans and AI models in task performance and strategy, so that humans users may use this information to update their beliefs about AI's decision-making processes (Hsiao and Chan, 2023; Qi et al., 2024b,a). For example, a recent study comparing humans and AI models in an object detection task found that AI models whose attended features were more closely aligned with those of humans' demonstrated better performance, suggesting that human attention may be used to improve the detection ability of AI models (Yang et al., 2023a). Inspired by previous research, we will compare the similarity between humans and LLMs in performance and examine whether human strategies are effective for LLMs in arithmetic tasks. Examining the comparability between humans and LLMs could significantly enhance LLMs' explainability, with important implications for ways to improve their performance.

In this paper, we propose an explainability framework to uncover the learning process of transformer-based LLM. The proposed framework decomposes the complex task into simple subtasks based on the intermediate attention pattern and verify the subtasks through corresponding ablation experiments. Specifically, we apply the explainability framework on integer arithmetic tasks. We reformulate traditional arithmetic rules into a Transformer-friendly framework. We adopted subtask decomposition methods, learning curve analysis, and model visualization analysis to diagnose accuracy issues and uncover their root causes. We then integrate

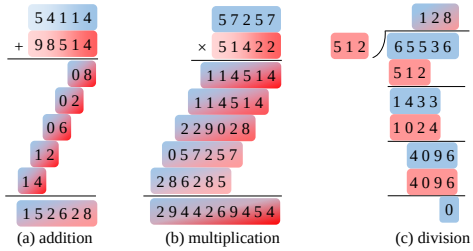



Figure 1: The decomposed steps of arithmetic tasks, include (a) addition, (b) multiplication, and (c) division.

human arithmetic strategies and enhance relevant capacities to gain a deeper understanding of the model's limitations and potential. Our findings provide valuable insight into understanding and interpreting Transformer-based models. The main contributions of this work are threefold: 1) We propose a comprehensive problem analysis framework to explain how Transformer-based LLMs handle arithmetic tasks. 2) Based on interpretive analysis, we enhance model's arithmetic performance and clarify the principles behind the improvements. 3) Our approach contributes to broader research on model understanding and explainability, opening up possibilities for analyzing more complex tasks and Transformer architectures.

## 2 Related Work

Since this work focuses on XAI techniques for transformer-based models in arithmetic tasks, the related work will address transformers in arithmetic tasks and their XAI methods.

### 2.1 Transformers in Arithmetic Tasks

Transformers (Vaswani et al., 2017) use the self-attention mechanism to capture long-range dependencies among tokens and have been widely applied in both natural language processing (Devlin et al., 2018; Taylor et al., 2022; Thoppilan et al., 2022; Chung et al., 2024) and computer vision (Radford et al., 2021; Li et al., 2023, 2024). Recent studies have explored the application of transformers in arithmetic tasks. (Yang et al., 2023b) focus on enhancing the reasoning process by teaching the model step by step. (Lee et al., 2023) investigate the impact of data format design in arithmetic tasks, finding that the use of detailed, instructive data with intermediate steps improves both accuracy and sample complexity. (Dziri et al., 2024) explore Transformer's limitations in arithmetic tasks like multi-digit multiplication through formulating these tasks as computation graphs.These methods analyze the transformer's

shortcomings in arithmetic primarily through accuracy, offering limited insight into the internal mechanisms by how they perform computations.

### 2.2 XAI in Transformers

XAI encompasses techniques that make AI system decisions interpretable to humans. Early methods often rely on gradient-based methods, analyzing how output predictions change with respect to input (Zhou et al., 2016; Selvaraju et al., 2017). Recently, numerous XAI studies have sought to explain how Transformers make decisions (Wang et al., 2022; Merullo et al., 2023; Quirke et al., 2023). (Quirke et al., 2023) conducted a comprehensive analysis of a one-layer Transformer on n-digit integer addition by decomposing the task into three base functions. (Shen et al., 2023) shows that reliance on positional information causes poor performance on arithmetic problems involving fewer digits.Although these studies offered some level of explainability for transformers, they are either not conducted on arithmetic or lack in-depth analysis of how models perform tasks, lacking a theoretical foundation for more systematic explainability.

**Comparison with existing methods:** There are some works focused on arithmetic in transformers (Dziri et al., 2024; Lee et al., 2023). However, these studies differ significantly from ours in several ways: 1) These methods are data-driven which treat Transformers as black boxes, focusing on the influence of data formats. Differently, our work instead interprets the internal mechanisms of the model. 2) (Dziri et al., 2024) decomposes arithmetic into sequential subtasks using complex prompts that reflect human reasoning. In contrast, we use standard arithmetic equations and analyze how specific model components handle subtasks. 3) (Lee et al., 2023) explores how data formats affect accuracy, whereas we examine their impact on both accuracy and internal learning processes. Despite these differences, our findings are consistent with previous work. For instance, (Dziri et al., 2024) decomposes arithmetic into subtasks, while (Lee et al., 2023) shows how reversed outputs benefit the model. This paper provides a detailed analysis of the model's computation process, visually demonstrating how Transformers handle arithmetic and proposing improvements to address their limitations. Our work and these works offer complementary perspectives on explainability.

## 3 Methodology

### 3.1 Comprehensive problem analysis

We propose a comprehensive problem analysis framework to inspect LLM models, which enhances the explainability of model behaviors through adopting a human-centric approach based on cognitive science theories. The framework has three steps: 1) subtask decomposition, 2) learning curve analysis, and 3) model visualization analysis. In human problem solving, human learners often break down complex tasks into a combination of simpler tasks (Lee and Anderson, 2001), and they start from easier tasks before they can solve harder ones (Clements and Sarama, 2020), with different brain regions engaged in different subtasks (Arsalidou et al., 2018). Here we decompose the arithmetic tasks into smaller components and examine whether LLMs perform subtasks differentially, which we refer to as subtask decomposition analysis. This method helps to identify subtasks that are particularly difficult for LLMs. For each subtask, we inspect learning curves for the output digits to examine the learning process for the specific calculation, which we refer to as learning analysis. This method helps to investigate the difficulties of LLMs in generating specific output digits. After inspecting learning curves, we visualize the contribution of the attention heads to the output digit. The visualization method helps to examine whether the model follows humans' information processing strategies for different subtasks.

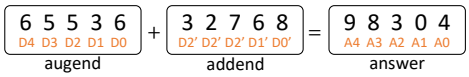
### 3.2 Human strategy and capacity transfer

Researchers in educational psychology and cognitive science have explored various learning and problem-solving strategies to improve the arithmetic performance of human learners. Here we test two common strategies, including the right-to-left digit-based strategy (Hickendorff et al., 2019) and the number-based strategy (Lemaire and Callies, 2009), on LLMs in solving arithmetic problems. In digit-based strategy, integers are managed in terms of individual digits, and calculated from right to left, without considering their represented place values (Hickendorff et al., 2019). We adopt this strategy to tune the LLM model by using the reversed-order answers as the ground truth. In number-based strategy, numbers are partitioned into tens and units and then combined to simplify calculations (Lemaire and Callies, 2009). We implement this strategy in the LLM model through

Chain-of-Thought (CoT) prompting. Also, previous research highlights the importance of cognitive capacities, such as working memory capacity (Zhang et al., 2022) and general intelligence (Bornemann et al., 2010), in math learning. We enhance the capacity of LLMs by increasing the model depth. Therefore, we hypothesize that employing human arithmetic strategies and enhancing computational capacity could significantly improve model arithmetic performance. Indeed, previous research has demonstrated the effectiveness of CoT prompting (Imani et al., 2023) in improving LLM's mathematical performance.

### 3.3 Arithmetic Tasks Formulation

This paper focuses on the integer arithmetic task in Transformer models. The input is a sequence of symbols, which consists of two $n$-digit operands $D = (D_{n-1}, ..., D_1, D_0)$ and $D' = (D'_{n-1}, ..., D'_1, D'_0)$, along with operators $\bigodot$, where $\bigodot$ can be $+$, $\times$ or $\div$. We did not study subtraction due to its similarity to addition. Transformer first converts the input into a sequence of one-hot vectors representing corresponding symbols through a vocabulary table of size $V$. These one-hot vectors are then mapped to a sequence of embeddings, $\mathbf{x} = (x_1, x_2, ..., x_L)$, where $x_i \in \mathcal{R}^d$ is the embedding of $i$-th word with dimension $d$, $L$ is sequence length. After "=", the model predicts the answer digits $A = (A_{m-1}, ..., A_1, A_0)$. An example of the addition formula is shown as,




Accurate calculation of answer digits requires complex calculations. Following subtask decomposition observed in human behavior (Lockwood et al., 2016) and considering the characteristics of the self-attention mechanism, we model arithmetic tasks as a combination of simple operations acting on digit pairs,

$$a_i = \sum_{i=0}^{n} \sum_{j=0}^{n} f_{ij}(D_i, D'_j). \quad (1)$$

This data movement between tokens aligns well with the self-attention mechanism. Transformers need to learn distinct functions for different pairs of digits to perform arithmetic calculations. In the following sections, we analyze and explain the transformer's learning process and implement for arithmetic tasks via task decomposition formulation, and validate our explanation through ablation experiments and visualizations.

## 4 Problem Analysis in Arithmetic Tasks

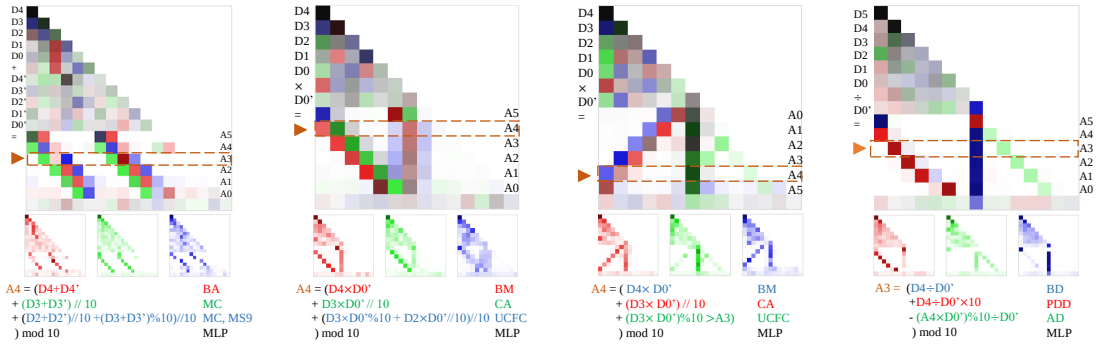
### 4.1 Addition

**Subtask Decomposition:** We begin our analysis with addition, the simplest arithmetic task. The addition task takes two $n$-digit numbers as input, producing an $n+1$-digit answer. For 5-digit addition, there are 10 billion distinct questions and 200,000 possible answers. In addition, the calculation of each answer digit depends on specific digits in the addend and augend, as well as the carry from the previous position. Considering that the model calculates answer digits from the highest value digit, the calculation can be formulated as,

$$\begin{aligned} A_i = \lfloor & (D_i + D'_i) \\ & + (D_{i-1} + D'_{i-1})//10 \\ & + ((D_{i-2} + D'_{i-2})//10 + (D_{i-1} + D'_{i-1})\%10)//10 \rfloor \end{aligned} \quad (2)$$

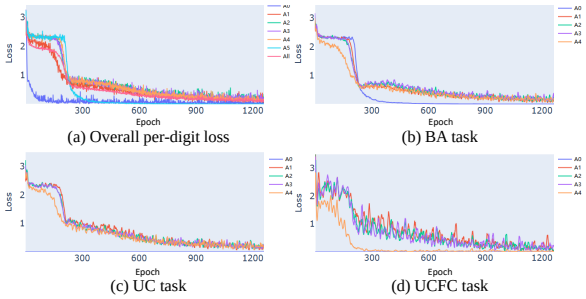
where $\lfloor \cdot \rfloor$ denotes floor operation. To accurately calculate $A_i$, the Transformer needs to operate on 3 digit pairs, *e.g.*, $(Di, D'_i)$, $(D_{i-1}, D'_{i-1})$ and $(D_{i-2}, D'_{i-2})$. Imitating the human strategy, we assert the transformer utilizes several basic subtasks to operate on digit pairs to complete the addition: 1) **Base Add (BA)**: BA calculates the sum of digit pair $(Di, D'_i)$, which is the basic operation of the addition task. 2) **Make Carry (MC)**: MC evaluates whether there is a carry from the previous column. 3) **Make Sum 9 (MS9)**: MS9 determines whether the sum at the current position is 9, which is used to calculate consecutive carry operations. Based on basic subtasks, the transformer chains multiple subtasks together to achieve complex tasks, *e.g.*, **Use Carry (UC)** takes the carry from the previous column and adds it to the sum of the current digit pair, **Use Carry and Further Carry (UCFC)** uses carry from the previous column and further propagates a carry to the next column.

**Learning Analysis:** Fig.3(a) visualizes the per-digit and overall training loss for 5-digit addition task, with 'A$n$' representing the $n$-th digit and 'All' indicating the overall loss. The per-digit loss curves show that the model learns each answer digit independently. The digit A0 converges much faster than the others because its calculation, which depends only on BA and does not require a carry from the previous column, is relatively straightforward. The loss for higher-order digits decreases more slowly,




(a) Addition (d) Multiplication (ordinal) (c) Multiplication (reversed) (d) Division

Figure 2: Attention maps of arithmetic tasks.




(a) Overall per-digit loss (b) BA task (c) UC task (d) UCFC task

Figure 3: Illustrations of (a) the overall loss curve, and (b-e) per-digit loss curve for each subtask on addition.

| #Heads | Add | Mul (O) | Mul (R) | Div |
|---|---|---|---|---|
| 1 | 89.1 | 66.2 | 83.6 | 31.7 |
| 2 | 98.7 | 85.3 | 96.3 | 86.2 |
| 3 | 99.9 | 89.8 | 99.8 | 99.8 |
| 4 | 100 | 90.1 | 100 | 99.9 |
| 5 | 100 | 91.1 | 100 | 100 |

Table 2: Accuracy of Transformers on arithmetic with different attention heads, Mul (O) and Mul (R) denote multiplication with oridinal and reversed answer digits.

as they must account for the carry-over from lower digits. We then categorized the training data into 3 non-overlapping subsets aligned to the BA, UC, and UCFC tasks, and visualizes the loss curve of each subtask in Fig.3(b-d). As shown, BA and UC tasks show similar patterns, with the BA loss curve dropping more quickly because BA accuracy is needed before UC can be accurate. The curve of UCFC is much noisier than other tasks, as this task requires considering at least three digits, making it more complex than BA and UC.

**Visualization Analysis:** We further validated the subtasks learned by Transformers through attention visualization. We selected attention heads that showed the clearest task separation. As shown in Fig.2(a), distinct heads specialize in the BA and UC subtasks, attending sequentially to digits in the augend and addend from left to right. Notably, computing digits A5 and A0 involves slightly different subtasks than other positions. This attention pattern also explains the noisier loss curve for UCFC: each head focuses on only three adjacent digits, which is insufficient for capturing cascading UCFC cases. Table 2 further confirms that using more than three attention heads enables effective addition.

## 4.2 Multiplication

Unlike addition, multiplication involves a series of intermediate steps, particularly for large numbers. To streamline our analysis and interpretation, we start with multi-digit × unit-digit (m×u) multiplication before extending our analysis to multi-digit × multi-digit (m×m) multiplication.

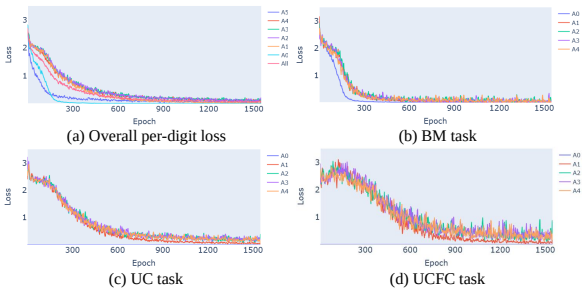
### 4.2.1 Unit-digit Multiplication

**Subtask Decomposition:** Based on the task decomposition formulation, the $m \times u$ multiplication can be formulated as,

$$A_i = \lfloor (D_i \times D'_0) + (D_{i-1} \times D'_0)//10 + (D_{i-1} \times D'_0 \% 10 + D_{i-2} \times D'_0 // \% 10)//10 \rfloor \quad (3)$$

Imitating human strategy, we assert the transformer utilizes the following basic subtasks that operate on individual digit pairs: 1) **Base Multiply (BM)**: BM calculates the product of two single digits $D_i$ and $D'_j$ at each position. 2) **Make Carry (MC)**: MC is responsible for calculating the carryover from the previous digit. Similar to addition, complex functions can be achieved through combining simple tasks, *e.g.*, **Use Carry (UC)** and **Use Carry and Further Carry (UCFC)**.

**Learning Analysis:** Fig. 4(a) presents the per-digit and overall training loss for 5-digit $m \times u$ multiplication, where A$n$' denotes the $n$-th digit and "All" indicates total loss. The curves suggest



(a) Overall per-digit loss (b) BM task (c) UC task (d) UCFC task

Figure 4: Illustrations of (a) the overall per-digit loss curve, and (b-e) per-digit loss curve for each subtask on multiplication task.

| Format | overall | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $0000d$ | 85.3 | 100 | 100 | 100 | 100 | 98.4 | 95.4 | 94.4 | 94.9 | 99.6 | 100 |
| $d000d$ | 16.8 | 98.6 | 96.9 | 95.2 | 85.1 | 40.2 | 28.6 | 96.0 | 96.4 | 99.5 | 100 |
| $000dd$ | 5.2 | 100 | 100 | 100 | 93.8 | 45.2 | 21.6 | 19.6 | 20.0 | 32.0 | 100 |
| $00ddd$ | 0.2 | 100 | 100 | 94.3 | 43.1 | 13.4 | 11.3 | 10.8 | 12.4 | 31.6 | 100 |
| $0dddd$ | 0 | 100 | 95.2 | 44.4 | 11.8 | 10.0 | 10.4 | 9.3 | 12.4 | 31.4 | 100 |
| $ddddd$ | 0 | 91.3 | 39.9 | 13.4 | 10.5 | 9.9 | 9.5 | 10.5 | 12.3 | 32.6 | 100 |

Table 3: Overall and per-digit accuracy(%) with various multipliers.

that Transformers learn each output digit semi-independently. Digits A0 and A5 converge faster and with less noise—A0 requires no carry, while A5 involves only carry handling. In contrast, A1–A4 show similar patterns, reflecting their dependence on both the previous carry and the current digit product, and their role in generating the next carry. We further analyze subtasks in Fig. 4(b–d) by grouping training samples into non-overlapping subsets per subtask and plotting per-digit loss. A clear trend emerges: while all tasks start with high loss, their curves decline at different rates with noticeable time lags. The BM subtask converges fastest, as it serves as the foundation for all others. UC follows, initially plateauing—indicating its reliance on BM—before dropping once BM adapts to more complex, carry-involving cases. This also explains the second drop in BM's curve. Lastly, UCFC shows the slowest and noisiest convergence, as it spans three digits and is more complex.

**Visualization Analysis:** We analyze the model's prediction behavior through attention visualization. Inspired by how humans perform multiplication—starting from the least significant digit (Lee et al., 2023)—we visualize attention patterns for both ordinal and reversed Transformers (data format details in Appendix A). In Fig. 2(b), the ordinal Transformer attends sequentially to digits in the multiplicand. Three heads show a 1-token offset, each handling specific subtasks—for example, the red head computes BM on D4 and D0'. However, each head only covers three consecutive tokens, which limits the model's ability to manage cascaded UCFC cases requiring longer-range dependencies. This limitation contributes to the ordinal model's lower accuracy. In Fig. 2(b), each attention head is also responsible for specific subtasks. This attention pattern explains why the reversed Transformer performs better: it can leverage previously generated answer digits to compute the current digit, enabling it to better handle cascaded UCFC cases. The visualizations of transformers with different heads are shown in Appendix-D.

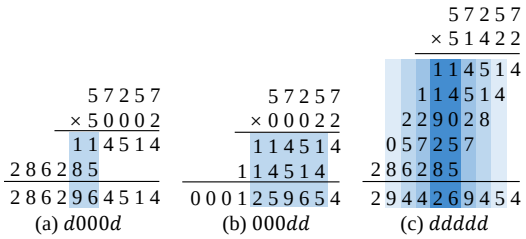
Table 2 shows the performance of Transformers with different heads. In the table, the reversed Transformer consistently achieves superior accuracy with different numbers of attention heads, and performance improvement tends to saturate with more than 3 attention heads, this indicates 3 heads are sufficient to complete multiplication. We also validated the subtasks learned by each head through ablation experiments in Appendix-C. The ablation experiments confirm that each attention head has a distinct and well-defined function.

### 4.2.2 Multi-digit Multiplication

This section examines the $m \times m$ multiplication task, where vanilla Transformers perform poorly. To understand this limitation, we analyze internal model behaviors and explore potential solutions inspired by human learning strategies.

**Human strategy transfer:** We begin by evaluating per-digit accuracy of ordinal and reversed Transformers across different multiplier sizes, as shown in Table 3. Accuracy declines with increased overlap among intermediate products. For instance, in the $ddddd \times ddddd$ format (Fig. 5(c)), digits A4 and A5 involve all intermediate products, leading to the lowest accuracy. This indicates that vanilla Transformers struggle with the complexity of intermediate steps in $m \times m$ multiplication.

We then evaluate the effectiveness of human strategies and provide interpretation of their impact. We first validated these strategies in Table 4. Reversing answer digits improves performance in the $m \times u$ task but offers limited gain alone on the more complex $m \times m$ task, due to intricate intermediate steps challenging for a shallow model. CoT prompting effectively decomposes $m \times m$ multiplication into simpler $m \times u$ subtasks, easing

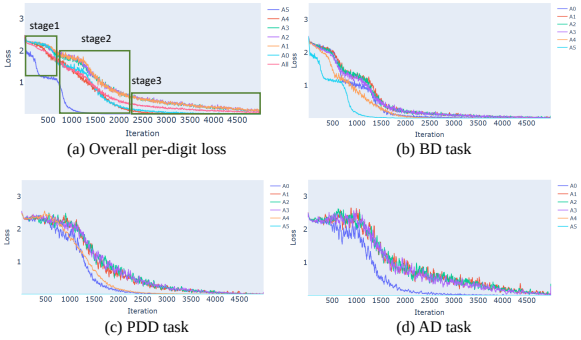


(a) $d000d$ (b) $000dd$ (c) $ddddd$

Figure 5: The overlap of per-digit products with different multiplier formats, with darker colors indicating more overlapping digits.

| Reserve | Depth | CoT | Accuracy | |
|---|---|---|---|---|
| | | | Multiply | Division |
| | | | 0.0 | 1.2 |
| ✓ | | | 0.0 | 0.0 |
| | ✓ | | 79.1 | 65.1 |
| | | ✓ | 80.2 | 83.9 |
| ✓ | ✓ | | 99.9 | 30.1 |
| ✓ | | ✓ | 99.6 | - |
| | ✓ | ✓ | 99.3 | **100** |
| ✓ | ✓ | ✓ | **100** | - |

Table 4: Effectiveness of refinements on multiplication and division.



(a) Overall per-digit loss (b) BD task (c) PDD task (d) AD task

Figure 6: Illustrations of (a) the overall training loss curve and (b-d) per-digit loss curves for division task.

prediction. Similarly, increasing model depth enhances capacity to handle these steps. Reflecting human multiplication strategies, combining these components yields the best accuracy.

### 4.3 Division

Different from addition and multiplication, the calculation order of division is reversed. In this section, we also conduct analysis starting with the simpler $m \div u$ division.

#### 4.3.1 Unit-digit Divisor Division

**Subtask Decomposition:** We first decompose the division as pairwise digit operation based on our task decomposition formulation,

$$A_i = \lfloor \frac{D_i}{D'_0} + \frac{D_{i+1}}{D'_0} \times 10 - \frac{(A_{i+1} \times D'_0)\%10}{D'_0} \times 10 \rfloor \tag{4}$$

where $\lfloor \cdot \rfloor$ denotes floor operation. To accurately calculate $A_i$, the model needs to operate on 3 digit pairs, *e.g.*, $(Di, D'_0)$, $(D_{i+1}, D'_0)$ and $(A_{i+1}, D'_0)$. We assert the transformer utilizes several basic subtasks to operate on these digit pairs: 1) **Base Division (BD)**: BD operates on digit pair $(Di, D'_0)$ to calculates the base quotient of two single digits. 2) **Previous Digit Division (PDD)**: PDD calculate the quotient of the previous digit $D_{i+1}$ and divisor $D'_0$, which is subsequently used for remainder division calculations. 3) **Answer Division (AD)**: AD is response for the calculation of the portion occupied during the calculation of the previous digit.

**Learning Analysis:** We first analyze Transformer's overall training behavior in Fig. 6(a), where A$n$' denotes the $n$-th answer digit and All' represents the total loss across all digits. The training process unfolds in three stages: the first two feature rapid loss reduction, indicating the model's "grokking" of division rules, while the third stage shows slower convergence as all digits are gradually mastered, leading to a smooth and low loss. The per-digit loss curves reveal the learning order. A5 converges the fastest, as it does not depend on any prior remainder. Digits A4 to A1 follow similar patterns, with each curve shifting later due to their reliance on remainders from preceding digits. Interestingly, A0 follows a distinct trajectory—it converges faster than A4–A1. This is likely because the model can often "guess" A0 from the divisor and the last digit of the dividend; for example, when $D0 = 4$ and $D' = 6$, $A0$ is constrained to a small set of possibilities, such as 4 or 9, simplifying its prediction.

We further analyze the Transformer's learning behavior by grouping training data into subtask-specific subsets and plotting per-digit loss in Fig.6(b–d). As shown in Fig.6(b), the BD task converges fastest, consistent with its simplicity and the rapid loss drop in stage 1 of Fig. 6(a). The PDD and AD losses decline later, reflecting their dependency on prior digit outputs. This illustrates the Transformer's progressive learning.

**Visualization Analysis:** We visualize how Transformers perform division in Fig. 2(d). Each attention head focuses on a specific digit pair to handle a distinct subtask—for example, the blue head performs the BD task using digit $D_i$ from the dividend and $D'_0$ from the divisor. The outputs of all heads are then integrated by the MLP layer to produce the final answer digit.

#### 4.3.2 Multi-digit Division Analysis

The vanilla Transformer also performs poorly on $m \div m$ division tasks. This section explores the reasons for the accuracy drop and provides an explanation of the internal mechanisms of potential solutions inspired by human learning processes.

**Human strategy transfer:** We first analyze per-digit accuracy on $m \div m$ division with various dividend formats in Table 5 (*e.g.*, '00000$ddd$' represents an 8-digit number with the first 5 digits fixed as zero). When the dividend is $ddddddd$, the model achieves very low accuracy (1.2%). Notably, accuracy decreases sequentially from left to right as we move to non-zero dividend digits, reflecting the accumulation of errors in remainder calculation. This indicates that the model struggles to handle the intermediate remainder. However, the accuracy for the units digit (*e.g.*, $A0$) remains consistently high across different dividend formats. The reason is similar to $A0$ in $m \div u$ division: the Transformer can 'guess' the units digit based on the divisor and dividend, avoiding error accumulation.

We then studied the effectiveness of human strategies, including reversing answer digits, using CoT prompting, and increasing model depth.We first validated the effectiveness of these strategies in Table 4. Reversing the digits is effective for multiplication, but it actually reduces performance for division. This is because division calculations start from the higher-order digits, which aligns with human experience. Increasing the model depth improved the accuracy of division calculations. This is because deeper models have a greater capacity to handle intermediate results, leading to more accurate division calculations. CoT prompting decomposed complex tasks, effectively reducing the model's prediction difficulty, and thus led to better performance. The effects of these strategies align with human findings.

### 4.4 Analysis of LLMs in Arithmetic

We analyze the possible reasons for the suboptimal performance of LLMs on arithmetic tasks based on findings in this paper. 1) Model capacity: although LLMs have a large capacity, due to the scarcity of arithmetic data in the internet-collected training data, only a small fraction of neurons are 'specialized' in arithmetic. 2) Data format: in internet-collected data, the arithmetic data are usually presented in ordinal order. However, arithmetic tasks have different calculation orders, *e.g.*, the multiplication follow entirely different calculation orders.

| dividend | overall | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|
| 00000$ddd$ | 57.2 | 100 | 100 | 100 | 100 | 57.2 | 95.7 |
| 0000$dddd$ | 21.6 | 100 | 100 | 100 | 100 | 22.0 | 90.6 |
| 000$ddddd$ | 19.1 | 100 | 100 | 100 | 87.4 | 22.4 | 90.4 |
| $ddddd$000 | 9.7 | 97.2 | 62.1 | 69.7 | 15.2 | 100 | 100 |
| $dddddddd$ | 1.2 | 98.7 | 89.1 | 45.8 | 12.7 | 10.5 | 90.5 |

Table 5: Overall and per-digit accuracy on division.

3) Output diversity: NLP tasks allow diverse outcomes hence the LLM are encouraged diversified outputs, while arithmetic have definite results.

## 5 Conclusion

This paper presents a comprehensive analysis framework to inspect and explain Transformer's implementation on arithmetic tasks through the consideration of human strategies. Our findings demonstrated that the Transformer decomposes the arithmetic task into multiple parallel streams and combines the partial results to yield the final outcome. Through learning and visualization analyses, we explain the underlying mechanisms of the Transformer's inferior performance on arithmetic tasks. Furthermore, we enhanced the model's accuracy and explainability by transferring human arithmetic strategies and cognitive capacity, suggesting that LLMs may share similar information processing mechanisms to humans in solving arithmetic problems. These findings have important implications for ways to improve the arithmetic ability of transformer-based LLMs through comparisons with human information processing. Our approaches contribute to the broader fields of model enhancement and explainability. It paves the way for enhancing the explainability of more complex tasks and larger Transformer-based models.

## Limitations

This work focuses on small-scale Transformer models, which facilitates explainability but may not capture the full capabilities of large language models (LLMs). Future research will extend the analysis to larger models to examine the scalability and generalizability of our findings. Additionally, our study is limited to single-operation arithmetic tasks. Real-world scenarios often involve composite expressions requiring multi-step reasoning. Extending the framework to such settings will be essential for evaluating and improving the arithmetic robustness of Transformer-based models.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Marie Arsalidou, Matthew Pawliw-Levac, Mahsa Sadeghi, and Juan Pascual-Leone. 2018. Brain areas associated with numbers and calculations in children: Meta-analyses of fmri studies. *Developmental cognitive neuroscience*, 30:239–250.

B. Bornemann, M. Foth, and J. Horn. 2010. Mathematical cognition: individual differences in resource allocation. *ZDM Mathematics Education*, 42:555–567.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Douglas H Clements and Julie Sarama. 2020. *Learning and teaching early math: The learning trajectories approach*. Routledge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, and 1 others. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Marian Hickendorff, Joke Torbeyns, and Lieven Verschaffel. 2019. Multi-digit addition, subtraction, multiplication, and division strategies. In *International handbook of mathematical learning difficulties: From the laboratory to the classroom*, pages 543–560. Springer.

Janet Hsiao and Antoni Chan. 2023. Towards the next generation explainable ai that promotes ai-human mutual understanding. In *XAI in Action: Past, Present, and Future Applications*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Frank J Lee and John R Anderson. 2001. Does learning a complex task have to be complex?: A study in learning decomposition. *Cognitive psychology*, 42(3):267–316.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*.

Patrick Lemaire and Sophie Callies. 2009. Children's strategies in complex arithmetic. *Journal of Experimental Child Psychology*, 103(1):49–65.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia. 2024. Mini-gemini: Mining the potential of multi-modality vision language models. *arXiv preprint arXiv:2403.18814*.

Elise Lockwood, Anna F DeJarnette, Autumn Asay, and Matthew Thomas. 2016. Algorithmic thinking: An initial characterization of computational thinking in mathematics. *North American Chapter of the International Group for the Psychology of Mathematics Education*.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2023. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*.

Ruoxi Qi, Guoyang Liu, Jindi Zhang, and Janet Hsiao. 2024a. Do saliency-based explainable ai methods help us understand ai's decisions? the case of object detection ai. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 46.

Ruoxi Qi, Yueyuan Zheng, Yi Yang, Caleb Chen Cao, and Janet H Hsiao. 2024b. Explanation strategies in humans versus current explainable artificial intelligence: Insights from image classification. *British Journal of Psychology*.

Philip Quirke and 1 others. 2023. Understanding addition in transformers. *arXiv preprint arXiv:2310.13121*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, and 1 others. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Alice Yang, Guoyang Liu, Yunke Chen, Ruoxi Qi, Jindi Zhang, and Janet Hsiao. 2023a. Humans vs. ai in detecting vehicles and humans in driving scenarios. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023b. Gpt can solve mathematical problems without a calculator. *arXiv preprint arXiv:2309.03241*.

Yuxin Zhang, Andrew Tolmie, and Rebecca Gordon. 2022. The relationship between working memory and arithmetic in primary school children: a meta-analysis. *Brain sciences*, 13(1):22.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

## A Data Format

Table 6 summarizes the data format used in our experiments, including addition, multiplication, and division. For multiplication, we validated the impact of reversing the answer digits, the formats of ordinal and reversed answer digits are marked as (O) and (R) in the table. For multiplication and division, we also validated the impact of CoT, which is also illustrated in Table 6.

| Type | data | Example |
|---|---|---|
| addition | $m + m$ | $65536 + 65535 = 131071$ |
| multiply | $m \times u$ (O) | $57257 \times 2 = 114514$ |
| | $m \times u$ (R) | $57257 \times 2 = 415411$ |
| | $m \times m$ (O) | $57257 \times 51422 = 2944269454$ |
| | $m \times m$ (R) | $57257 \times 51422 = 4549624492$ |
| | $m \times m$ with CoT | $57257 \times 5 = 286285$<br>$57257 \times 1 = 57257$<br>$57257 \times 4 = 229028$<br>$57257 \times 2 = 114514$<br>$57257 \times 2 = 114514$<br>$57257 \times 51422 = 4549624492$ |
| division | $m \div u$ | $131072 \div 4 = 32768$ |
| | $m \div m$ | $16777216 \div 256 = 65536$ |
| | $m \div m$ with CoT | $167 \div 256 = 0, 167$<br>$1677 \div 256 = 6, 141$<br>$1417 \div 256 = 5, 137$<br>$1372 \div 256 = 5, 092$<br>$0921 \div 256 = 3, 153$<br>$1536 \div 256 = 6, 000$<br>$16777216 \div 256 = 65536$ |

Table 6: Examples of data format of arithmetic tasks. The symbols in red after "=" are what the model needs to predict.

## B Implementation Details

**Training:** We conducted our experiments using a single-layer decoder-only transformer, which includes a Multi-Head Self-Attention (MHSA) layer and a Feed-Forward (FF) layer. The dimensions of MHSA and FF layers are 512 and 2048, respectively. For scaling experiments, we extend this to a multi-layer transformer with consistent dimensions. The model was trained with standard cross-entropy loss for next-token prediction, which is widely used in transformer training. The overall training process consists of 5000 iterations with a batch size of 64, using the Adam optimizer and a learning rate of 1e-4. Each digit in the training data is independently sampled from a uniform distribution 0,1,...,9. For the division task, we ensured that the first digit in divisor is non-zero in both $m \div u$ and $m \div m$ divisions. Appendix A summarizes the data format used in our experiments.

**Inference:** During inference, the input consists of dividend, division sign, divisor, and equal sign, *e.g.*, "$123456 \times 7 =$". The model generates each digit of the product answer in an autoregressive manner. The autoregressive process stops when generating the final digit. We test on 10k randomly sampled data that does not appear in the training set.

| Ablated Head | Avg loss | Conclusion |
|---|---|---|
| - | 0.000 | - |
| 0 | 0.315 | Minor impact. |
| 1 | 0.632 | Minor impact. |
| 2 | 5.552 | Large impact, head 2 is key for BM. |

Table 7: The influence of ablating each attention on **BM** task.

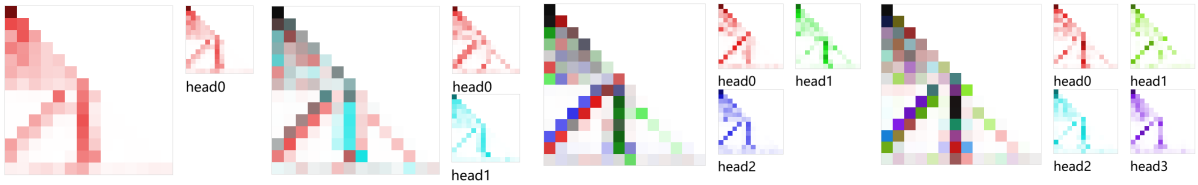
Table 8: The influence of ablating each attention on **CA** task.

| Ablated Head | Avg loss | Conclusion |
|---|---|---|
| - | 0.001 | - |
| 0 | 7.846 | Large impact, head 0 is key for CA. |
| 1 | 0.979 | Minor impact. |
| 2 | 1.258 | Minor impact. |

## C Verification of learned subtasks

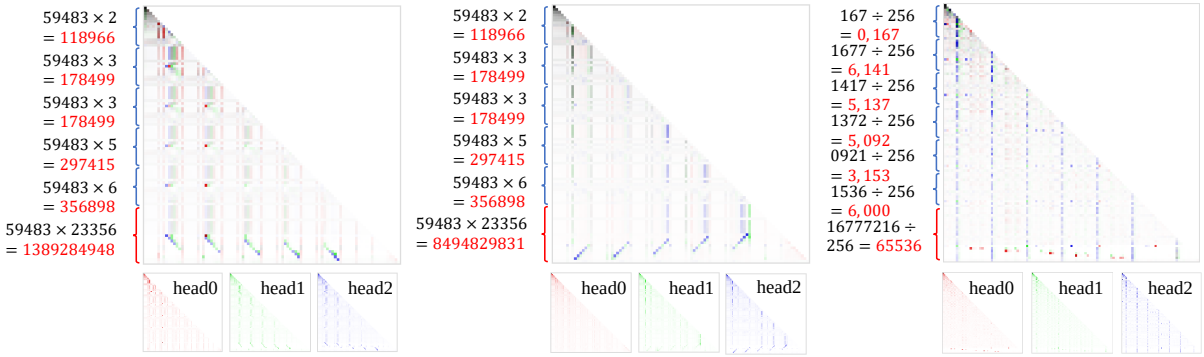
We validated the learned subtasks by each attention head of the reversed transformer. For each head, we used an ablation intervention technique (Quirke et al., 2023) that overrode its output with the mean value of the whole dataset and computed the average loss on specific samples related to the target $BM$ task.

The experimental results on the BM task are shown in table 7. As shown, ablating head 0 and head 1 has a minor impact on the loss of the BD task, while ablating head 2 causes a significant loss increase. We hence conclude that head 2 is the key for the calculation of the BM task. Similar conclusions can also be observed on other heads in Table 8 and 9. This indicates that each attention head has a well-defined role, and they collaborate to accomplish the division task.




(a) 1 head (b) 2 heads (c) 3 heads (d) 4 heads

Figure 7: The attention map of reversed transformer with different attention heads.

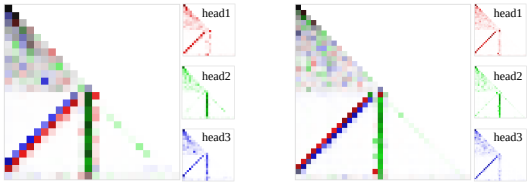



(a) Ordinal multiplication (b) Reversed multiplication (c) Division

Figure 8: The attention map of transformer with CoT input, (a) ordinal transformer on multiplication task, (b) reversed transformer on multiplication task, and (c) ordinal transformer on division task. The characters that the transformer needs to predict are highlighted in red.

| Ablated Head | Avg loss | Conclusion |
|---|---|---|
| - | 0.002 | - |
| 0 | 0.784 | Minor impact. |
| 1 | 6.161 | Large impact, head 1 is key for UCFC. |
| 2 | 1.116 | Minor impact. |

Table 9: The influence of ablating each attention on **UCFC** task.




(a) 10-digit multiplication (b) 15-digit multiplication

Figure 9: The attention map of transformer on 10-digit and 15-digit multiplication.

## D The attention map with different attention heads

The visualizations of transformers with different heads are shown in Fig.7. As shown, when the head number is less than 3, multiple tasks are packed into a single head, hence causing the low accuracy. Conversely, when there are more than 3 heads, multiple heads end up performing the same subtask. The model with 3 heads demonstrates the clearest task separation in attention patterns, it has separate attention heads for BA, CA, and UCFC tasks.

**Algorithm reuse:** We explored whether the subtasks are learned by similar Transformer-based models. We first train the same model on the 5-digit multiplication task with different random seeds or optimizers. The resulting models show similar behavior to the previous one. We then train Transformers on 10-digit and 15-digit multiplication, and visualize the attention maps in Fig.9. The new models also use BM, UC and UCFC subtasks to complete the multiplication calculation. This analysis suggests that the Transformer architecture, when trained on the multiplication task, converges to a consistent algorithm for performing multiplication, indicating a robust algorithmic solution emerges from the Transformer's architecture and training.

## E Verification of CoT input

We visualize the attention maps of ordinal and reversed Transformers with CoT input on multiplication task in Fig. 8(a) and (b). CoT decomposes the

$m \times m$ task into simpler $m \times u$ multiplications and additions, reducing calculation complexity. Since the addition task is also easier when computed from lower-order digits, the reversed Transformer also outperforms the ordinal transformer with CoT inputs.

We then visualized the attention map of Transformers with CoT input on the division task in Fig. 8(c). As shown, when performing $m \div m$ division, the Transformer not only focuses on dividend and divisor, but also attends to the previous intermediate results. This decomposes the complex division task into multiple simpler ones, reducing prediction difficulty.