Interpreting and Improving Large Language Models in Arithmetic Calculation

Wei Zhang [*1 2] **Chaoqun Wan** [2] **Yonggang Zhang** [† 3] **Yiu-ming Cheung** [3] **Xinmei Tian** [1 4] **Xu Shen** [† 2]
Jieping Ye [2]

Abstract

Large language models (LLMs) have demonstrated remarkable potential across numerous applications and have shown an emergent ability to tackle complex reasoning tasks, such as mathematical computations. However, even for the simplest arithmetic calculations, the intrinsic mechanisms behind LLMs remains mysterious, making it challenging to ensure reliability. In this work, we delve into uncovering a specific mechanism by which LLMs execute calculations. Through comprehensive experiments, we find that LLMs frequently involve a small fraction ($< 5\%$) of attention heads, which play a pivotal role in focusing on operands and operators during calculation processes. Subsequently, the information from these operands is processed through multi-layer perceptrons (MLPs), progressively leading to the final solution. These pivotal heads/MLPs, though identified on a specific dataset, exhibit transferability across different datasets and even distinct tasks. This insight prompted us to investigate the potential benefits of selectively fine-tuning these essential heads/MLPs to boost the LLMs' computational performance. We empirically find that such precise tuning can yield notable enhancements on mathematical prowess, without compromising the performance on non-mathematical tasks. Our work serves as a preliminary exploration into the arithmetic calculation abilities inherent in LLMs, laying a solid foundation to reveal more intricate mathematical tasks.

[*]This work was done when the author was visiting Alibaba Cloud as a research intern. [1]University of Science and Technology of China [2]Alibaba Cloud [3]Hong Kong Baptist University [4]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center. Correspondence to: Xu Shen[†] <shenxu.sx@alibaba-inc.com>, Yonggang Zhang[†] <csygzhang@comp.hkbu.edu.hk>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Large language models (LLMs) have experienced rapid advancements and shown impressive language understanding capabilities (Devlin et al., 2019; Brown et al., 2020; Chowdhery et al., 2022). Notably, LLMs exhibit emergent abilities (Wei et al., 2022b) that enable them to solve intricate reasoning tasks akin to humans, such as mathematical computations (Frieder et al., 2023; Jie et al., 2022), chain-of-thought reasoning (Wei et al., 2022c; Kojima et al., 2022), few-shot prompting (Brown et al., 2020; Alayrac et al., 2022), etc. Despite these impressive characteristics, the complex inner processes governing LLMs' functionality have yet to be fully illuminated, due to the complex and intricate non-linear interactions within densely-connected layers. Comprehending these underlying mechanisms could contribute to predicting how the LLMs behave beyond their training data (Mu & Andreas, 2020), gaining insights into the emergence of certain behaviors (Nanda & Lieberum, 2022; Barak et al., 2022; Wei et al., 2022a), as well as identifying and rectifying errors present in the specific models (Hernandez et al., 2021; Vig et al., 2020).

In this work, we take the first attempt to interpret the inner process of LLMs through the lens of mathematical computation problems, which are conducted on publicly available LLMs (*e.g.*, LLaMA2 series (Touvron et al., 2023b)). Unlike typical language comprehension tasks, mathematical computation tasks involve concise problem statements with definitive correct answers, requiring a process of reasoning and calculation rather than direct copying to derive the solutions. These characteristics enable us to gain insights into the models' reasoning capabilities without interference from unrelated factors. Specifically, we focus on tasks involving the arithmetic calculation with two operands, *i.e.*, addition, subtraction, multiplication, and division, which are fundamentals of mathematical computation. To this end, we create datasets of various types of sentences that involve the calculation logic, such as "The addition of 3 and 5 equals to _" in Figure 1. The LLMs could provide answers with high confidence scores of over 80% on average.

To unveil how these models correctly complete the task (*e.g.*, "$3 + 5 = 8$"), we begin by identifying the task-related internal components in LLMs. We do a hard in-

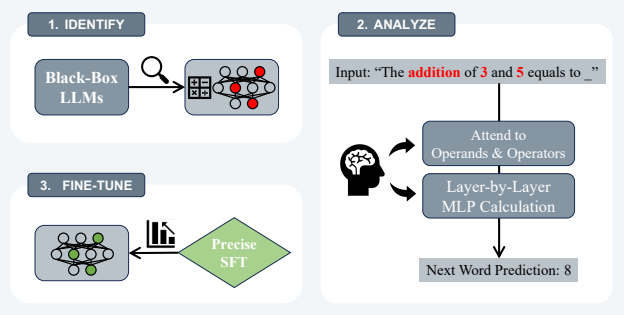


Figure 1: The pipeline involves three steps: 1) *identify* the key components attributed to arithmetic calculations in black-box LLMs, 2) *analyze* the working mechanism of the key components towards human-understandable explanations, 3) *fine-tune* the key components to precisely improve the mathematical capability of LLMs.

tervention (Pearl, 2009) on the transformer attention heads and multi-layer perceptrons (MLPs) to observe their effects on the predicted logits[1]. Our findings reveal that only a small percentage ($< 5\%$) of the attention heads and the MLPs after these heads significantly impact the model's performance. Namely, LLMs frequently involve these attention heads and the subsequent MLPs when completing the calculations. Subsequently, we knock out these frequently-involved heads/MLPs to validate their faithfulness. We find that the model performance decreases sharply when those pivotal heads/MLPs are knocked out, resulting in a decrease of around 70% in accuracy.

To interpret the working mechanism of identified heads/MLPs towards human-understandable explanations, we gain a deeper analysis of their operational "behaviors". Specifically, we investigate the attention patterns of the crucial heads, and find that these attention heads exhibit a strong focus on the tokens representing operands and operators within mathematical sentences, demonstrating a relative insensitivity to other non-relevant tokens. For the analysis of MLPs, we compare the correlations between the embeddings of MLPs' input/output and the embeddings of number tokens (*i.e.*, operands and answers). It reveals that the MLPs, guided by these number-attended heads, take operands as input, and mirror the attributes of tokens corresponding to correct answers more closely. These observations lead us to hypothesize that *LLMs may initially employ a set of heads to pinpoint arithmetic operands from text, subsequently engaging MLPs to work out the answers.* Additionally, the observed behaviors of these heads/MLPs exhibit a high degree of transferability, analogous to adversarial examples being *transferable across models* (Szegedy et al., 2014). Namely, the key heads/MLPs identified on one dataset are also effective for other datasets. For instance, their impact is noticeable on the publicly available math datasets (*e.g.*, SVAMP (Patel et al., 2021)), as well as varied data formats involving multi-digit integers, rational numbers, etc. This empirical observation underscores the crucial role of key heads/MLPs in mathematical calculations.

In addition to uncovering the internal mechanisms, we have devised an effective strategy that involves targeted fine-tuning of the specific attention heads and MLPs closely tied to mathematical computations, thereby enhancing the model's mathematical prowess. The experimental results are compelling: with fine-tuning as few as 32 attention heads (with a total of 1024 heads), we observe a remarkable improvement in the model's mathematical capabilities. This precise tuning methodology not only matches but can surpass the enhancements achieved through full-model fine-tuning. Moreover, this fine-grained strategy of adjustment has a distinct advantage—it leaves most of the model's parameters unchanged, avoiding the performance trade-offs in non-mathematical domains commonly observed with full-model fine-tuning.

In summary, this work aims to delve into the inner mechanism of LLMs through mathematical calculation tasks, along the pipeline of "identify-analyze-finetune" shown in Figure 1. Our findings reveal a sparsity in the attention heads of LLMs, with less than 5% of heads exhibiting close correlations. These heads particularly attend to the operands and operators, while the subsequent MLPs gradually deduce the correct answers. The discovered mechanism shows strong cross-dataset transferability and inspires us to precisely finetune the calculation-related heads/MLPs for better mathematical capability. We empirically find that precise tuning brings in much less impact on non-mathematical tasks when improving the targeted ability of LLMs.

2. Related Works

Interpretability Methods. Interpreting the inner mechanism of large language models (LLMs) has become increasingly urgent in recent years (Madsen et al., 2022; Rauker et al., 2023), especially when LLMs are applied in high-stakes decision-making domains such as healthcare, criminal justice, and finance (Obermeyer et al., 2019; Rudin, 2019; Bender et al., 2021). Vig et al. (2020) adapted the approach of *causal mediation analysis* (CMA) (Pearl, 2001) for interpreting the deep language models, and it has been applied for various tasks, such as subject-verb agreement (Finlayson et al., 2021), natural language inference (Geiger et al., 2021), retention of factual associations (Meng et al., 2022; Geva et al., 2023). Furthermore, *path patching* extends the concept of CMA by measuring how a treatment

[1]Here, doing a hard intervention is equal to replacing the value of attention heads and MLPs, while performing a soft intervention means modifying the modules for calculating the attention and MLP values (Pearl, 2009).

effect is mediated by node-to-node connections between individual neurons or features. Recent works have used path patching to explain neural networks in terms of circuits (Olah et al., 2023), identified for different capabilities including indirect object identification (Wang et al., 2023a), greater-than computation (Hanna et al., 2023), and mapping answer text to answer labels (Lieberum et al., 2023).

Interpretability for Mathematical Tasks. Mathematical ability has long been a subject of interest in natural language processing (Kushman et al., 2014; Huang et al., 2016; Wang et al., 2017; Thawani et al., 2021). Some studies have investigated the mathematical abilities of LLMs (Frieder et al., 2023; Saxton et al., 2019; Nogueira et al., 2021; Qian et al., 2023; Imani et al., 2023; Romera-Paredes et al., 2024), but they mainly focus on explaining *what* these models can do rather than *how* they do it. In contrast, some other studies have dived deeper into the LLM structure without treating LLM as an inscrutable black box. Stolfo et al. (2023) identified the key attention *layers* relating to arithmetic questions, but lacking in-depth explanation and validation of the key layers' behaviors. Wu et al. (2023) scaled the methods from causal abstraction to understand how Alpaca (7B) (Taori et al., 2023) follows the instruction in comparing two numbers. (Hanna et al., 2023) provided a causal explanation about how GPT2-small (0.1B) (Radford et al., 2019) implements the "greater-than" task, but only reveal simple phenomena limited by the small size of model and the lack of diversity in the dataset.

Fine-tune LLMs for Mathematical Tasks. Numerous studies improve the mathematical reasoning ability of LLMs by aggregating various sampled reasoning paths during either fine-tuning or inference. Cobbe et al. (2021) train and devise a reasoning path verifier to select the correct results during inference. Wang et al. (2023b) propose to sample various reasoning paths during inference and then derive the final result by majority voting on the answers or through verifiers (Li et al., 2023). Uesato et al. (2022) explore to use of reinforcement learning methods for improving the mathematical reasoning abilities of LLMs. Several works apply the idea of rejection sampling along with other techniques to filter the diverse sampled reasoning paths for fine-tuning data augmentation (Huang et al., 2022; Zelikman et al., 2022; Ni et al., 2023). There also exist related works (Panigrahi et al., 2023) that locate key parameters to update for better task-specific ability. Panigrahi et al. (2023) locates a minuscule subset of parameters from an already fine-tuned model onto a pre-trained model without further tuning. The selection process for this subset is via optimizing the task-related objective function with L1 norm ensuring the sparsity of the subset. In our work, we locate the task-related parameters of pre-trained model via measuring the *causal effect* of each component, then *precisely fine-tune* the key components for mathematical tasks.

3. Preliminary

Large Language Models (LLMs). The LLMs utilized in this work comprise LLaMA2-7B and LLaMA2-13B (Touvron et al., 2023a). These are pre-trained language models freely available from HuggingFace[2]. All of these models are decoder-only transformers equipped with multi-head attention (MHA) and a single MLP in one transformer layer. For example, LLaMA2-7B consists of 32 transformer layers and 32 attention heads in MHA for each layer.

Transformer Architecture. The input to the transformer is a combination of position and token embeddings in $\mathbb{R}^{N\times d}$, where N is the number of tokens in the input and d is the model dimension. Following the definitions in (Elhage et al., 2021), the input embedding serves as the initial value for the *residual stream*, which is read from and written to by all attention heads and MLPs. Focusing on individual heads, the j-th head in the i-th layer is parametrized by four matrices: $W_Q^{i,j}$, $W_K^{i,j}$, $W_V^{i,j} \in \mathbb{R}^{d\times\frac{d}{H}}$, and $W_O^{i,j} \in \mathbb{R}^{\frac{d}{H}\times d}$. To simplify these parameters, we can express them as low-rank matrices in $\mathbb{R}^{d\times d}$: $W_{OV}^{i,j} = W_O^{i,j}W_V^{i,j}$ and $W_{QK}^{i,j} = W_Q^{i,j}(W_K^{i,j})^T$. The QK matrix is used to compute the attention pattern $A_{i,j} \in \mathbb{R}^{N\times N}$ for head (i, j), while the OV matrix determines the information written into the residual stream. At the end of the forward pass, a layer norm is applied before the unembed matrix W_U projects the residual stream into logits.

Task and Dataset. We focus on classic and widely encountered mathematical operations, *e.g.*, addition, subtraction, multiplication, division. Taking addition as an example, the arithmetic logic of addition ({A} + {B} = {C}) might naturally appear in sentences. Taking inspiration from the sentence styles and forms present in mathematical benchmarks of GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021), we create a dataset for the addition task containing $10,000$ samples based on 36 templates with random single-token names, objects, and numbers. To assess the performance of LLMs on the calculation task, we measure the prediction probability of the {C} token. The average probability of correct predictions across the models was 82%. In this study, we select the samples that the language models are able to predict correctly. We denote the sentences generated by this procedure as reference data using the notation of X_{r}. For the templates and sentences, please refer to Figure 8 and Figure 10 in Appendix A.

Moreover, to meet the demand for perturbing component activation, we create another dataset comprising counterfactual sentences without the inclusion of calculation logic, using the notation of X_{c}. The samples are generated following two core principles: (1) maintaining the grammatical structures derived from the X_{r} templates; (2) substituting

[2]https://huggingface.co/

several crucial words responsible for the calculation logic with irrelevant words. For example, the sentence from X_{r} like "42 plus 34 is equal to _" is replaced to the counterfactual one "42 *nothing* 34 is equal to _". In this way, it allows for a direct reflection of the model's impact on the arithmetic calculation tasks, rather than being influenced by the sentence structure or syntax.

4. Method

Our goal is to interpret the LLMs in a way that is human-understandable, thus enabling targeted modification of models through precise SFT. This section delves into the "identify-analyze-finetune" methodology. First, in Section 4.1, we describe the process of identifying and validating key components within LLMs. Then in Section 4.2, we examine the inherent patterns of these pivotal components to decode their behavior and distinct features. Finally, in Section 4.3, we introduce a strategy of precise SFT that fine-tunes these influential components to enhance the proficiency in calculation.

4.1. Key Components Identification.

The computation of the LLM can be reorganized as a directed acyclic graph (DAG) (Wang et al., 2023a). In the graph, each node is a computation component, including attention heads, MLP layers, residual connections, and each edge represents the data flow that the output of the previous node will be transposed to the input of the later node. Please refer to Appendix B for more details. To unravel the underlying cause of the model's predicted answer, we employ the causal intervention technique known as *path patching* (Goldowsky-Dill et al., 2023; Wang et al., 2023a). By perturbing targeted activation with counterfactual data X_{c} and freezing others with reference data X_{r}, the comparison on output logits is employed to measure the counterfactual effect. The whole process is illustrated in Algorithm 1. In this work, we scan through all nodes $\mathcal{N}$ one by one, and measure the changes in the output logit of ground-truth token $\{\mathtt{C}\}$, recoding in $E_{\mathcal{N}}$. Notably, since the residual operations and MLPs compute each token separately (Elhage et al., 2021), patching the head output at the END position (*i.e.*, the last token in the input sentence) is enough to measure the effects on the next token prediction.

Explanations for model behavior can easily be misleading or non-rigorous (Bolukbasi et al., 2021; Wiegreffe & Pinter, 2019). To address this issue, we further assess the importance of the identified heads/MLPs, while also confirming the insignificance of others. For this purpose, we employ a knockout technique called *mean ablation* (Wang et al., 2023a) to deactivate the individual heads/MLPs and observe their impact on model performance. Specifically, we replace their activation with average activation across counterfactual data X_{c} to remove the task-related information. By observing changes in model performance, we can verify the roles of these key heads/MLPs.

Algorithm 1 Identifying Key Components

Input: Set $\mathbf{\Omega}$ of reference and counterfactual sample pairs $(X_{\mathrm{r}}, X_{\mathrm{c}})$, model $\mathcal{M}$ with nodes $\mathcal{N}$.
Output: Causal effects for $\mathcal{N}$: $E_{\mathcal{N}}$.
for $(X_r^{(i)}, X_c^{(i)})$ in $\mathbf{\Omega}$ **do**
  Compute all activations A_r, A_c on $(X_r^{(i)}, X_c^{(i)})$
  for n in $\mathcal{N}$ **do**
    $A'_r(n) \leftarrow A_c(n)$; ▷ replace output in A_r by A_c
    $A'_r(k) \leftarrow A_r(k), \forall k \in [1, \cdots, |\mathcal{N}|], k \neq n.$
    $logit_o \leftarrow \mathcal{M}(X_r^{(i)}, A_r)$ ▷ get original logits
    $logit_p \leftarrow \mathcal{M}(X_r^{(i)}, A'_r)$ ▷ get patched logits
    $s_n^{(i)} \leftarrow \frac{logit_p - logit_o}{logit_o}$ ▷ causal effect
  end for
end for
Return: $\overline{s_n} = \frac{\sum_{i=1}^{|\mathbf{\Omega}|} s_n^{(i)}}{|\mathbf{\Omega}|}$ ▷ averaged effect w.r.t. samples

4.2. Pattern Analysis.

To make the identified heads/MLPs accessible to human understanding, we conduct a deeper analysis of their operational "behaviors". For attention heads, we examine the attention pattern $A_{i,j} \in \mathbb{R}^{N \times N}$ to comprehend which tokens are prioritized. N is the number of input tokens. Specifically, we begin by gathering the respective attention patterns $A_{i,j}$ on reference data X_r of the key heads. We extract the last row of $A_{i,j}$ for each sample, analyzing the attention scores $A_{i,j}^{END} \in \mathbb{R}^{1 \times N}$ between the Query token at the END position and each Key token, and obtaining the averaged scores w.r.t. samples. Generally, the type of token with the highest attention score represents the characteristics of the head, such as numbers, math symbols, etc.

For MLPs, we use the unembedding matrix as the probing to measure the content of token, especially numerical tokens, contained in MLPs' inputs and outputs. Prior studies, such as those reported in (Elhage et al., 2021), have illustrated that the MLP layer initially receives its input from the residual stream (*i.e.*, MLP_{in}), subsequently adding its output back into that stream (*i.e.*, MLP_{out}). Let W_U represent the unembedding matrix, and $W_U[*]$ denote the unembedding vector corresponding to a specific token. We calculate the cosine similarity between MLP_{in}, MLP_{out} and $W_U[\{\mathtt{A}\}], W_U[\{\mathtt{B}\}], W_U[\{\mathtt{C}\}]$ to reflect the information the MLP receives and generates. To isolate the specific contribution of MLP to specific numerical tokens, we further evaluate the subtraction of outputs and inputs of MLP, *i.e.*, $\frac{MLP_{out} - MLP_{in}}{||MLP_{out} - MLP_{in}||} \cdot \frac{W_U[\{\mathtt{A}\}]}{||W_U[\{\mathtt{A}\}]||}$. Research in (Geva et al., 2022) presents that each MLP layer's output token repre-

sentation can be characterized as an additive update influencing the evolving representation across vocabularies. Our methodology is aligned with these works, while we mainly focus on the token embeddings of right/wrong answers to reveal the contribution of MLPs on the calculation tasks.

4.3. Precise Fine-tuning.

Supervised Fine-Tuning (SFT) is widely used for enhancing a model's mathematical capabilities. Building on this, precise SFT only updates those components closely associated with mathematical abilities, while keeping the rest parameters unchanged. Algorithm 2 illustrates the whole process. For the i-th attention layer, the output matrix W_O^i is split into equal size blocks for each head $\left[W_O^{i,1}, W_O^{i,2}, \cdots W_O^{i,H}\right]$. As

Algorithm 2 Precise Fine-tuning

Require: Model $\mathcal{M}$, input X, index of key heads $\mathbf{\Phi}$, iterations I, learning rate η, $W_\theta = W_{Q/K/V/O}$
for $(i, j) \in \mathbf{\Phi}$ **do**
  $W_\theta^{i,j}.requires\_grad = True$
end for ▷ activate key heads
loop I times
  $\mathcal{L}$ = $\mathcal{M}$.forward(X)
  $\mathcal{L}$.backward()
  for $w \in W_\theta$ **do**
    $w = w - \eta * w.grad$
  end for ▷ update target parameters
end loop

is verified in (Elhage et al., 2021), it is equivalent to running heads independently, multiplying each by its own output matrix, and adding them into the residual stream. For the selected individual heads, precise SFT updates the parameters of four matrices: $W_Q^{i,j}$, $W_K^{i,j}$, $W_V^{i,j} \in \mathbb{R}^{d \times \frac{d}{H}}$, and $W_O^{i,j} \in \mathbb{R}^{\frac{d}{H} \times d}$. For the selected MLP layer, precise SFT updates all parameters in this layer. Moreover, since we adjust only a small fraction of the parameters, precise SFT naturally benefits from shorter training times and minimal impact on the model's original capabilities.

5. Experiments

The experiments are organized as follows: (1) *identify* the calculation-related key components via path patching and *validate* their importance in implementing arithmetic calculation via knockout in Section 5.1; (2) *understand* the behavior of the newly identified components by examining their attention patterns and embeddings in Section 5.2; (3) *improve* the mathematical capability via precise supervised fine-tuning on math benchmarks in Section 5.3. For simplicity, we primarily report the results of LLaMA2-7B, while the results of other models can be found in Appendix.

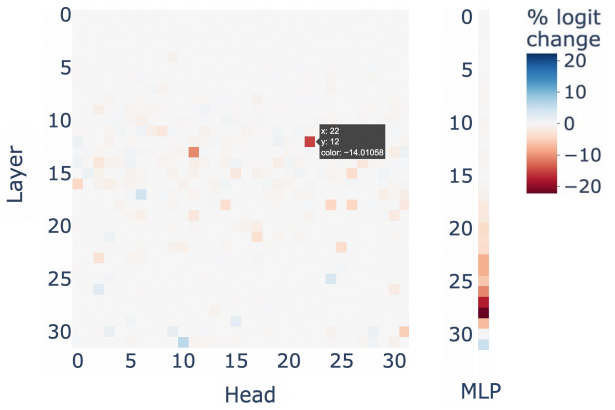


Figure 2: We conduct path patching experiments on LLaMA2-7B across four mathematical tasks, by searching for each head and MLP directly affecting the logit of the right answer. For each head/MLP, a darker color indicates a larger logit difference from the model before patching.

5.1. Identifying Calculation-related Components.

Location of key heads. In Figure 2, we visualize the effect of each head according to the serial numbers of the heads and layers. This arrangement allows for a clear comparison of the causal impact of each head to the logit of ground-truth token {C}. The red squares indicate heads that have a significant positive impact on predicting the output token, while the blue squares represent heads that have a negative effect. From these results, we observe that: (i) *Only a small number of heads have a noteworthy influence on the output*. Specifically, when the heads such as 12.22[3] is patched, there is a substantial decrease of 14.0% on the logit of token {C}, which highlights their positive contribution to the calculation tasks. We classify heads that exhibit logit change exceeding −5% as "key heads". The sparse distribution of these key heads motivates us to explore their specific functionalities and characteristics in Section 5.2. (ii) *The discovered key heads are mainly located in the middle layers*. For LLaMA2-7B, key heads emerge starting from the 12th layer for all arithmetic calculations. Prior layers exhibit heads that do not exert a direct effect on the output logits. Key heads are primarily concentrated between layers 12 and 17. (More analysis of the key heads in other LLMs can be found in Appendix C.)

Location of key MLPs. The last column in Figure 2 visualizes the effect of each MLP layer on the logit of ground-truth token {C}. It is observed that MLPs before the identified heads (0−16) have almost no impact on the outputs (approximately ±0.0%). In contrast, after the 17-th layer, MLPs exhibit a much larger effect (approximately ±10.0%). It indicates that MLPs are engaged in the calculation. We hy-

[3]We apply the notation of $i.j$ to refer to the j-th head of the i-th attention layer.

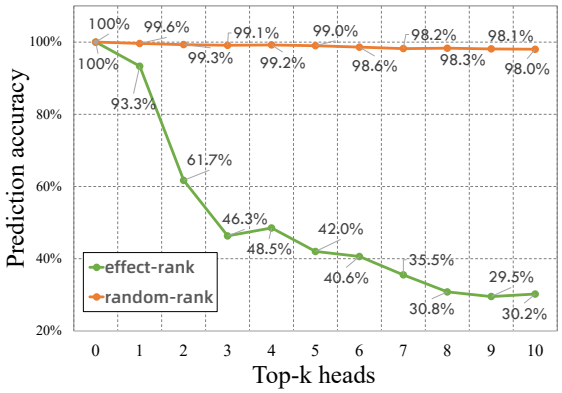


Figure 3: The influence on prediction accuracy after knocking out top-k attention heads that are sorted by the effect of each head on logits ("effect-rank"), and knocking out randomly-sorted top-k heads ("random-rank").

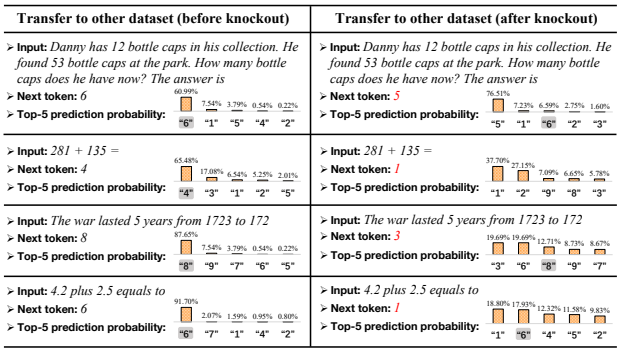
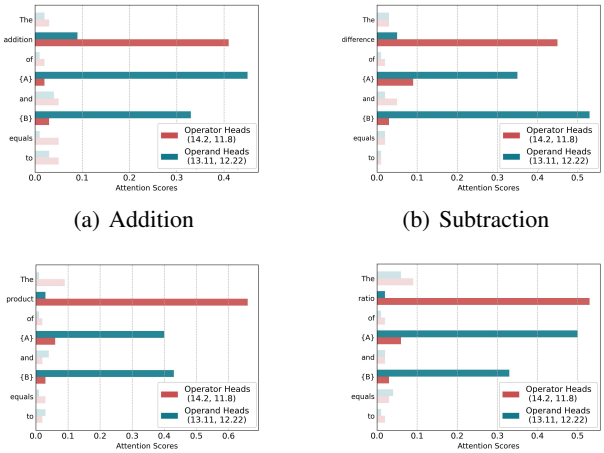


Figure 4: After knocking out the key heads, LLaMA2-7B predicts incorrectly on the cases of SVAMP dataset and other data formats of multi-digit integers, rational numbers.

pothesize that the calculation process is firstly implemented through the key heads, then the subsequent MLPs gradually work out the final results. We validate this in Section 5.2.

Validation of key components. To fully validate the faithfulness of the discovered key heads, we perform additional checks by observing the performance drop when knocking out these components. In Figure 3, all heads are sorted in a certain order by the importance score shown in Fig. 2 and knocked out one by one. It shows that, as the heads are gradually knocked out, the performance of the model drops sharply in "effect-rank", while keeping stable (relatively minor effect within 2%) in "random-rank". We also exhibit the transferability of the key heads with different data prompts or formats as shown in Figure 4. The model becomes largely confused to output incorrect numbers after knocking out the identified key heads. On the dataset SVAMP, there is a relative performance drop ($-22.9\%/34.7\%=-66.0\%$) after the knockout, aligned with the result on our generated dataset. The above results demonstrate that the discovered compo-



(a) Addition (b) Subtraction

(c) Multiplication (d) Division

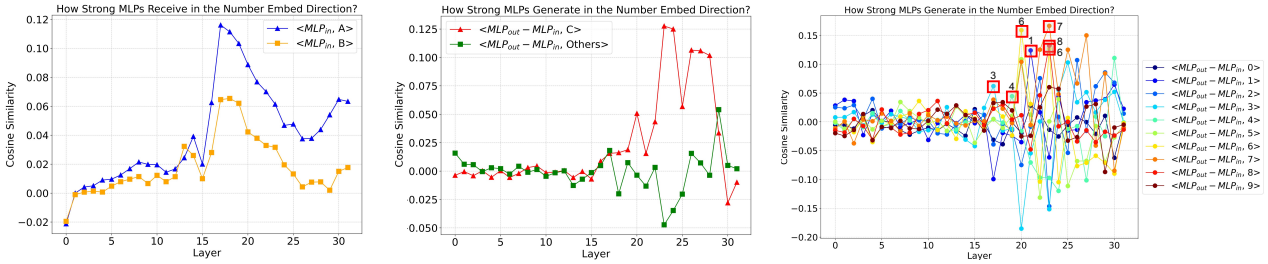
Figure 5: The attention score distribution of key heads across four calculation tasks. The key heads (*e.g.*, 13.11, 14.2) attend to number operands and calculation operators.

nents play an important role in the language model's ability to complete the calculation task.

5.2. Understanding Calculation-related Component Behaviors.

Key heads behavior. In order to better understand the "behavior" of the heads that have a significant impact on calculation, we begin by analyzing their attention patterns, and check the attention scores between Query END token and each Key token as illustrated in Sec. 4.2. Our findings reveal that these heads exhibit a strong focus on tokens of operands or operators. For example, heads 13.11 and 12.22 have high attention scores on numbers including {A} and {B}, while heads 14.2 and 11.8 attend more to symbols or text indicating operations like "+", "-", "plus", "div", etc. We randomly select 1000 samples from reference data and plot the distribution of averaged attention scores on key heads (arranged in two groups) for four arithmetic calculations.

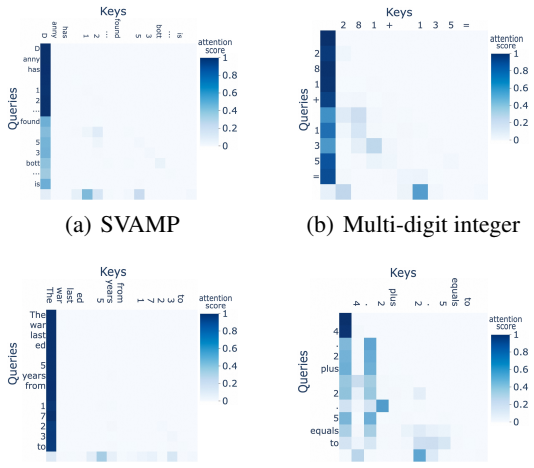
As illustrated in Figure 5, the operand heads and the operator heads are colored in red and green respectively, and highlighted at the positions of operands and operators. It is clear that these heads exhibit distinctly different distributions and show minimal attention to tokens outside of the operands/operators. Moreover, we visualize the attention patterns of the key heads (*e.g.*, 13.11) on various types of sentences in Figure 7. It reveals that the key heads also primarily prioritize number operands (*e.g.*, '1' and '5' in the first case) even for unseen data formats. This observation provides an explanation for why the deactivation of the key heads can influence the model's perception on number tokens and consequently affect its prediction when transfer-



(a) Reception of {A} and {B} (b) Generation of {C} (c) Steps of Calculation

Figure 6: We investigate the projection of each MLP layer input or output along the direction of number token {A}, {B}, and {C}, respectively. The x-axis represents the layer number, ranging from 0 to 31, while the y-axis represents the cosine similarity between the embeddings of the MLP input or output and the number tokens.

ring to other datasets (shown in Figure 4). For more case studies on the key heads, such as the attention pattern on operators, please refer to Figure 16 in Appendix F.



(a) SVAMP (b) Multi-digit integer

(c) Diff. data format (d) Rational numbers

Figure 7: The transferability of attention patterns in key heads on the unseen samples in Figure 4, which mainly attend to the number operands.

Key MLPs behavior. In Figure 6(a), we conduct an initial investigation of the similarities between the MLP_{in} and tokens {A} and {B} over 1000 samples, to verify the information of operands received from above analyzed attention heads. For the 0–12th layers, both $\langle MLP_{in}, \{\texttt{A}\}\rangle$ and $\langle MLP_{in}, \{\texttt{B}\}\rangle$ are close to zero. It indicates no operands are captured during this stage, which corresponds to the blank region (*i.e.*, few key heads for computation task) before the 12th layer in Figure 2. For the 12–17th layers, we observe a sharp increase in the similarities with both operands ({A} and {B}). This surge corresponds to the presence of key attention heads, *e.g.*, 12.22/13.11 in layer 12/13, indicating that the operands are progressively being collected and "written" into the MLPs of these layers for subsequent computations. In layers 17–31, the similarities $\langle MLP_{in}, \{\texttt{A}\}\rangle$ and $\langle MLP_{in}, \{\texttt{B}\}\rangle$ gradually decrease, signifying the transition into a new stage that digests the input information for generating the answers.

To understand how each MLP layer contributes to generating the correct answer {C}, we compute the similarity between token {C} and the input/output of the MLPs. We use $\langle MLP_{out} - MLP_{in}, W_U[\{\texttt{C}\}]\rangle$ to reflect the direct contribution of the MLP to the correct answer, and $\langle MLP_{out} - MLP_{in}, W_U[Other]\rangle$ for other candidate numbers (as shown in Figure 6(b)). Starting from the 17th layer, where the MLPs begin processing operand information, we observe a noticeable increase in $\langle MLP_{out} - MLP_{in}, W_U[\{\texttt{C}\}]\rangle$ and a decrease in $\langle MLP_{out} - MLP_{in}, W_U[Other]\rangle$. This trend indicates that these MLPs are gradually carrying out the calculation required for the correct answer. The above ascending and descending trends can also be viewed in other LLMs as in Figure 13 and Figure 14 in Appendix C.

Based on the above analyses, we further delve into the detailed calculation process from layer 17 to 28. We investigate a case of "$4 + 3 =$" and analyze $MLP_{out} - MLP_{in}$ compared to all numeric tokens in Figure 6(c). At layers 17 and 19, the numbers '3' and '4' are at the top, indicating that MLPs receive and store input {A} ='4' and {B} ='3', respectively. After that, the numbers '6' and '1' appear top at the subsequent layers 20 and 21. In summary, the LLM predicts the next token as '7' in a single inference. However, within the LLM's architecture, the answer '7' is the result of a collaborative process across multiple layers 22/23/25/27, after the layers 17/19/20/21 generate '3'/'4'/'6'/'1', respectively. The results demonstrate that the answer '7' is not deduced directly, and MLPs perform calculations in a "layer-by-layer" manner, somewhat akin to the addition process in computers (a comparison of these two processes are presented Appendix H). Additionally, we observe that numbers close to the correct answer, such as '6' and '8', also appear

Table 1: Overall performance. We evaluate the capabilities of LLaMA2-7B and LLaMA2-13B, transitioning from generic tasks (*e.g.*, MMLU and CSQA) to mathematical tasks (*e.g.*, GSM8K, AddSub, SingleEq, and SVAMP). Supervised fine-tuning across the entire parameter set (denoted as Full SFT) leads to enhanced performance in math-related tasks, albeit at the expense of its capabilities in generic tasks. In contrast, selectively tuning only the parameters of 32 critical attention heads (denoted as Precise SFT) yields comparable improvements while preserving the model's proficiency in generic tasks, with faster training speed (samples processed per second) and less tuned parameters.

			Mathematical Tasks								Generic Tasks			
			GSM8K		AddSub		SingleEq		SVAMP		MMLU		CSQA	
Models	**Train Speed**	**Tuned Params.**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**
LLaMA2-7B	-	-	14.6	-	30.5	-	65.4	-	34.7	-	46.0	-	59.8	-
+ Full SFT	15sam./sec.	6.7B	24.6	+10.0	53.7	+23.2	68.2	+2.8	50.3	+15.6	40.5	-5.5	54.0	-5.8
+ Precise SFT	50sam./sec.	0.07B	27.4	+12.8	50.6	+20.1	69.7	+4.3	55.8	+21.1	46.4	+0.4	59.6	-0.2
LLaMA2-13B	-	-	28.7	-	33.7	-	76.6	-	45.7	-	54.8	-	67.3	-
+ Full SFT	8sam./sec.	13.0B	44.6	+15.9	62.2	+28.5	79.8	+3.2	62.8	+17.1	50.2	-4.6	62.0	-5.3
+ Precise SFT	34sam./sec.	0.08B	46.3	+17.6	61.1	+27.4	82.2	+5.6	66.6	+20.9	55.0	+0.2	67.2	-0.1

at the top in layer 23. However, in subsequent layers, the correct answer '7' consistently remains top while '6' and '8' decline. It indicates that LLMs may do computations in a coarse-to-fine manner, where the result is firstly regressed to an embedding around that of the right answer, and then converges to the final output based on the fine-grained information introduced by subsequent MLPs.

Consolidating these findings, we can assert with some confidence that LLMs initially leverage attention heads to focus on operands ({A} and {B}) and the operator, relaying this information to downstream MLPs. Over time, the MLPs progressively bolster {C} and diminish the effect of confused answers, carrying out the calculation to final results.

5.3. Precise SFT on Calculation-related Components.

Experimental details. We evaluate precise SFT on four mathematical datasets (GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), SingleEq (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021)), and another two datasets (MMLU (Hendrycks et al., 2020) and CSQA (Saha et al., 2018)) to evaluate the generic ability. During training, we optimize the key components only and leave the other components unchanged. We gather all training data from four mathematical datasets, and perform SFT updating on top 32 key heads. Following (Yu et al., 2023), the gradient is rescaled by $\frac{H}{h}$, where H is the number of all heads in each layer, h is the number of updated heads in each layer. In practice, we train LLaMA2-7B and LLaMA2-13B with a learning rate of 2×10^{-5} and a batch size of 128 for 2 epochs. The warm up ratio and weight decay are set as 0.02 and 0.1 by default, respectively. All experiments are conducted on 8 NVIDIA A100 80GB GPUs.

Table 2: Ablative experiments on the number of tunable components. The default setting is shown in gray.

Precise SFT Setting	**Evaluation Metric**			
	Train Speed	Tuned Params.	GSM8K	MMLU
top-8 heads	58sam./sec.	0.017B	25.4	45.1
top-16 heads	52sam./sec.	0.033B	26.5	45.8
top-32 heads	50sam./sec.	0.067B	27.4	46.4
top-48 heads	46sam./sec.	0.101B	27.4	46.4
top-64 heads	40sam./sec.	0.134B	27.3	45.5
top-32 heads + top-3 MLPs	31sam./sec.	0.473B	28.0	45.2

Precise SFT improves mathematical ability. Supervised Fine-Tuning (SFT) is an effective approach for augmenting the mathematical capabilities of models by fine-tuning all parameters within LLMs. We term this all-parameter fine-tuning as Full SFT for clarity, and adopt the same training settings as Precise SFT. Table 1 presents the results of Full SFT and Precise SFT on the LLaMA2-7B and LLaMA2-13B models. Precise SFT effectively bolsters their mathematical capabilities, yielding an averaged increase of 15% on four distinct mathematical datasets. It matches or even surpasses the improvements made by Full SFT. For example, Precise SFT outperforms Full SFT by 5.5% on the SVAMP dataset and 2.8% on GSM8K, underlining its superior ability to enhance the mathematical prowess of LLMs. Full SFT suffers from the trade-off between mathematical and general capabilities (about 5% drops on MMLU and CSQA), while Precise SFT effectively maintains the model's original performance. A further advantage of Precise SFT is the drastic

reduction in training time, attributed to the substantially fewer parameter adjustments required (less than 1%). It results in a time reduction of at least threefold on LLaMA2-7B and LLaMA2-13B. Overall, Precise SFT offers an effective direction for boosting mathematical abilities for LLMs.

Ablative studies. The key issue with Precise SFT lies in determining the quantity and specific set of components to adjust. To demonstrate this, we experimented with varying numbers of heads and MLPs, with the results laid out in Table 2. We discovered that fine-tuning 32 heads yields the best average improvement across different numbers of involved heads. We also compared experiments with the introduction of MLPs. We observed that as more MLPs are added, the mathematical capability improves by 2.1%, but the general performance will decrease by 1.5% (results in Appendix G). Overall, the top-3 MLPs yielded the best comprehensive results. However, even the introduction of a single MLP can reduce computational efficiency by 15%. How to more precisely fine-tune MLPs will be explored in our future work.

More discussions. The above results underscore the potential of employing interpretability tools to analyze the inner mechanism of LLMs and to enhance their specific capabilities. However, there are several areas that require deeper investigation: (i) Our primary experiments and discussions center around the LLaMA2 series. The results presented in Appendix C demonstrate the potential for generalization across different LLMs, such as Mistral-7B (Jiang et al., 2023). For more rigorous considerations, it's necessary to perform specific adaptations on a broader range of LLMs. (ii) This work mainly focuses on interpreting the fundamental ability of "arithmetic calculation", since it's universally shared across various levels of complexity for mathematical problems. The results in Appendix E reveal that solving the math word problems requires a synergy of multiple skills including "text comprehension" and "arithmetic calculation", which is aligned with the findings in recent research (Opedal et al., 2024). It's imperative for continued research to investigate more complex mathematical problems. (iii) The potential of generalizing to more complex mathematical tasks like exponentiation (*e.g.*, "{A} to the power of {B} equals _") has been validated in Appendix D. An intriguing research direction would be to investigate the shared and distinct mechanisms across various mathematical tasks.

6. Conclusion

In this study, we have identified, analyzed, and fine-tuned the internal components responsible for the mathematical calculation capability of LLMs. The language models frequently involve sparse heads to particularly attend to operands and operators, and subsequent MLPs to work out answers. We apply the precise tuning on the calculation-related heads/MLPs for better mathematical capabilities, with less impact on non-mathematical tasks compared with tuning all parameters. These findings contribute to a better understanding of the inner mechanism of LLMs.

Acknowledgements

This work was supported in part by NSFC No. 62222117. YGZ and YMC were supported in part by NSFC/Research Grants Council (RGC) Joint Research Scheme under Grant: N_HKBU214/21; in part by RGC Senior Research Fellow Scheme under Grant: SRFS2324-2S02.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Alayrac, J., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J. L., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022.

Barak, B., Edelman, B. L., Goel, S., Kakade, S., Malach, E., and Zhang, C. Hidden progress in deep learning: Sgd learns parities near the computational limit. *arXiv preprint arXiv:2207.08799*, 2022.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

Bolukbasi, T., Pearce, A., Yuan, A., Coenen, A., Reif, E., Viégas, F. B., and Wattenberg, M. An interpretability illusion for BERT. *CoRR*, abs/2104.07143, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Finlayson, M., Mueller, A., Gehrmann, S., Shieber, S. M., Linzen, T., and Belinkov, Y. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 1828–1843, 2021.

Frieder, S., Pinchetti, L., Griffiths, R., Salvatori, T., Lukasiewicz, T., Petersen, P. C., Chevalier, A., and Berner, J. Mathematical capabilities of chatgpt. *CoRR*, abs/2301.13867, 2023.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, pp. 9574–9586, 2021.

Geva, M., Caciularu, A., Wang, K. R., and Goldberg, Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*, pp. 30–45, 2022.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models. *CoRR*, abs/2304.14767, 2023.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching. *CoRR*, abs/2304.05969, 2023.

Hanna, M., Liu, O., and Variengien, A. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *CoRR*, abs/2305.00586, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020.

Hernandez, E., Schwettmann, S., Bau, D., Bagashvili, T., Torralba, A., and Andreas, J. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2021.

Hosseini, M. J., Hajishirzi, H., Etzioni, O., and Kushman, N. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pp. 523–533. ACL, 2014.

Huang, D., Shi, S., Lin, C., Yin, J., and Ma, W. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Imani, S., Du, L., and Shrivastava, H. Mathprompter: Mathematical reasoning using large language models. In *ACL*, pp. 37–42, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jie, Z., Li, J., and Lu, W. Learning to reason deductively: Math word problem solving as complex relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 5944–5955, 2022.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., and Ang, S. D. Parsing algebraic word problems into equations. *Trans. Assoc. Comput. Linguistics*, 3:585–597, 2015.

Kushman, N., Zettlemoyer, L., Barzilay, R., and Artzi, Y. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pp. 271–281, 2014.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333. Association for Computational Linguistics, July 2023.

Lieberum, T., Rahtz, M., Kramár, J., Nanda, N., Irving, G., Shah, R., and Mikulik, V. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *CoRR*, abs/2307.09458, 2023.

Madsen, A., Reddy, S., and Chandar, S. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55(8):1–42, 2022.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Mu, J. and Andreas, J. Compositional explanations of neurons. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17153–17163, 2020.

Nanda, N. and Lieberum, T. A mechanistic interpretability analysis of grokking, 2022.

Ni, A., Inala, J. P., Wang, C., Polozov, A., Meek, C., Radev, D., and Gao, J. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*, 2023.

Nogueira, R. F., Jiang, Z., and Lin, J. Investigating the limitations of the transformers with simple arithmetic tasks. *CoRR*, abs/2102.13019, 2021.

Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. In *Distill*, 2023.

Opedal, A., Stolfo, A., Shirakami, H., Jiao, Y., Cotterell, R., Schölkopf, B., Saparov, A., and Sachan, M. Do language models exhibit the same cognitive biases in problem solving as human learners? *CoRR*, abs/2401.18070, 2024.

Panigrahi, A., Saunshi, N., Zhao, H., and Arora, S. Task-specific skill localization in fine-tuned language models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27011–27033, 2023.

Patel, A., Bhattamishra, S., and Goyal, N. Are NLP models really able to solve simple math word problems? In *NAACL-HLT*, pp. 2080–2094. Association for Computational Linguistics, 2021.

Pearl, J. Direct and indirect effects. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pp. 411–420, 2001.

Pearl, J. *Causality*. Cambridge university press, 2009.

Qian, J., Wang, H., Li, Z., Li, S., and Yan, X. Limitations of language models in arithmetic and symbolic induction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 9285–9298. Association for Computational Linguistics, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rauker, T., Ho, A., Casper, S., and Hadfield-Menell, D. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 464–483, 2023.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nat.*, 625(7995):468–475, 2024.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., and Chandar, S. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, pp. 705–713. AAAI Press, 2018.

Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019.

Stolfo, A., Belinkov, Y., and Sachan, M. Understanding arithmetic reasoning in language models using causal mediation analysis. *CoRR*, abs/2305.15054, 2023.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto., T. B. Stanford alpaca: An instruction-following llama model. 2023.

Thawani, A., Pujara, J., Ilievski, F., and Szekely, P. A. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 644–656, 2021.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., and Shieber, S. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401, 2020.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023a.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b.

Wang, Y., Liu, X., and Shi, S. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 845–854, 2017.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. *ArXiv*, abs/2206.07682, 2022a.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022c.

Wiegreffe, S. and Pinter, Y. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 11–20. Association for Computational Linguistics, 2019.

Wu, Z., Geiger, A., Potts, C., and Goodman, N. D. Interpretability at scale: Identifying causal mechanisms in alpaca. *CoRR*, abs/2305.08809, 2023.

Yu, L., Bowen, Y., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *ArXiv*, abs/2311.03099, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

A. Appendix

A. Templates

Addition	Subtraction
{A} + {B} = {C}	{A} - {B} = {C}
{A} plus {B} equals to {C}	{A} minus {B} equals to {C}
The addition of {A} and {B} is {C}	The difference of {A} and {B} is {C}
The addition of {A} and {B} equals to {C}	The difference of {A} and {B} equals to {C}
The addition of {A} and {B} equals to {C}	The difference of {A} and {B} equals to {C}
Q: How much is {A} plus {B}? A:	Q: How much is {A} minus {B}? A:
Q: What is {A} plus {B}? A:	Q: What is {A} minus {B}? A:
Q: What is the result of {A} plus {B}? A:	Q: What is the result of {A} minus {B}? A:
Q: What is the sum of {A} and {B}? A:	Q: What is the difference of {A} and {B}? A:
Multiplication	**Division**
{A} * {B} = {C}	{A} / {B} = {C}
{A} times {B} equals to {C}	{A} over {B} equals to {C}
The product of {A} and {B} is {C}	The ratio of {A} and {B} is {C}
The product of {A} and {B} equals to {C}	The ratio of {A} and {B} equals to {C}
The product of {A} and {B} equals to {C}	The ratio of {A} and {B} equals to {C}
Q: How much is {A} times {B}? A:	Q: How much is {A} over {B}? A:
Q: What is {A} times {B}? A:	Q: What is {A} over {B}? A:
Q: What is the result of {A} times {B}? A:	Q: What is the result of {A} over {B}? A:
Q: What is the product of {A} and {B}? A:	Q: What is the ratio of {A} and {B}? A:

Figure 8: Templates used in this work follow the formations of "Equation", "Statement", "Question-Answer".

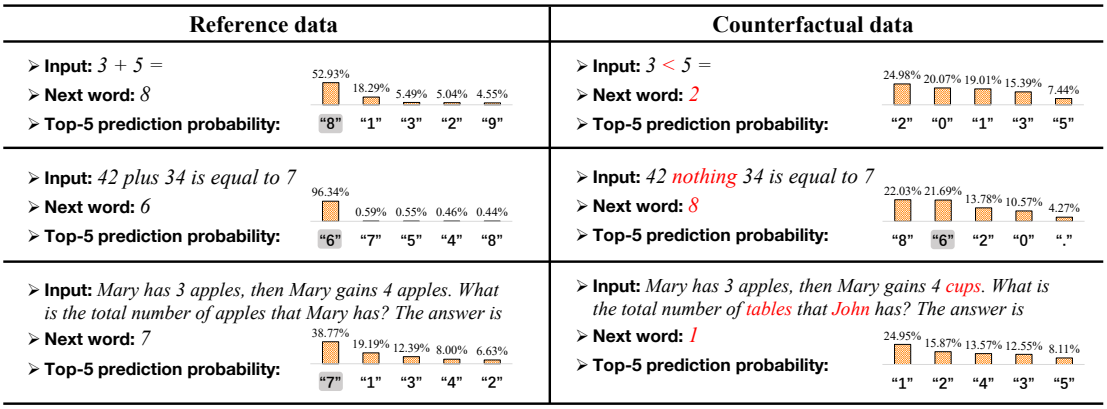


Figure 9: Examples of reference data (with addition logic) and counterfactual data (without addition logic). Given the input sentence, the results of next word prediction are provided by LLaMA2-7B.

We have included a list of 36 templates used in this work as shown in Figure 8. All these templates share the same calculation logic. we sample the <A> and <B> from $\{1, \cdots, 9\}$, since LLaMA2 tokenizes each digit individually (e.g., '42' is tokenized to '4' and '2'). Based on the above templates, we generate the sentences that the LLMs can predict the addition result {C} correctly as the reference data X_{r}. We generate the counterfactual data X_{c} following the principles depicted in Section 3, where we replace the words (*e.g.*, "plus", "minus", "times", "ratio") with a randomly-selected term from the set {"none", "nothing", $\cdots$, "null"}, and replace the operations (*e.g.*, "+", "-", "*", "/") with a randomly-selected term from the set {"<", ">", $\cdots$, "@"}. We show three cases in Figure 9 with the inspection into the top-5 prediction probability of LLaMA2-7B. Moreover, in Figure 10, we also construct several different types of linguistic meanings for the addition task: "time span" and "object accumulation". For the templates 1-8 of "time span", we sample from a

curated list of common words. For example, we select <EVENT> from {"war", "conflict", ··· , "project"}[4], <VERB> from {"last", "span", ··· , "extend"}, <MONTH> from {"Jan.", "Feb.", ··· , "Dec."}, and <YYY> from {100, ··· , 199}. For the templates 9-12 of "object accumulation", we sample <OBJECT> from {"apple", "orange", ··· , "pear"}, <VERB> from {"get", "obtain", ··· , "acquire"}, and each <NAME> was randomly selected from a pool of 100 English first names.

1. The <EVENT> <VERB> {A} years from the year <YYY>{B} to the year <YYY>{C}
2. The <EVENT> <VERB> {A} years from <YYY>{B} to <YYY>{C}
3. The <EVENT> <VERB> {A} days from <MONTH> {B} to <MONTH> {C}
4. The <EVENT> will <VERB> {A} days from <MONTH> {B} to <MONTH> {C}
5. The <EVENT> <VERB> {A} hours from {B} pm to {C}
6. The <EVENT> will <VERB> {A} hours from {B} pm to {C}
7. The <EVENT> <VERB> {A} hours from {B} am to {C}
8. The <EVENT> will <VERB> {A} hours from {B} am to {C}
9. <NAME> has {A} <OBJECT>, then <NAME> <VERB> {B} <OBJECT>. What's the total number of <OBJECT> that <NAME> has? The answer is {C}
10. <NAME> <VERB> {A} <OBJECT>, and <NAME2> <VERB> {B} <OBJECT>. What's the total number of <OBJECT> that they <VERB>? The answer is {C}
11. <NAME> has {A} <OBJECT>, and <NAME2> has {B} <OBJECT>. What's the total number of <OBJECT> that they have? The answer is {C}
12. <NAME> <VERB> {A} <OBJECT> yesterday, and <NAME> <VERB> {B} <OBJECT> today. What's the total number of <OBJECT> that <NAME> <VERB>? The answer is {C}

Figure 10: Additional templates used in the addition task, involve different linguistic meanings like "time span" (1-8) and "object accumulation" (9-12).

B. Evaluate the Effect of Attention Heads.

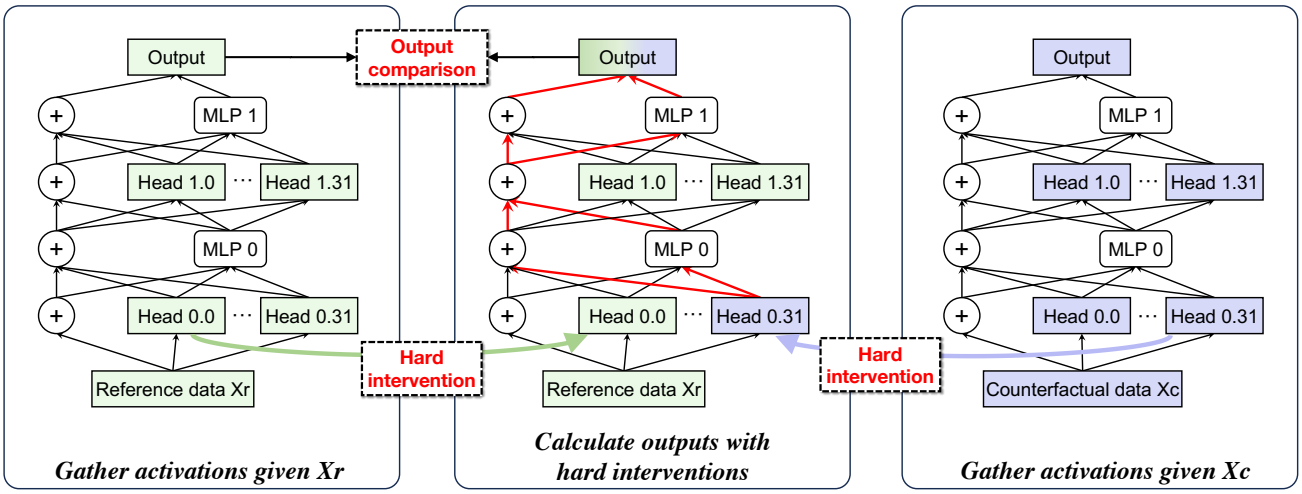


Figure 11: A case illustration of the method "path patching". It measures the importance of forward paths (*i.e.*, the red lines that originate from Head 0.31 to Output) for the two-layer transformer in completing the task on reference data.

Path Patching. To discover the cause of the predicted answer, we employ the causal intervention technique known as *path patching* (Goldowsky-Dill et al., 2023; Wang et al., 2023a). This approach is highly effective in analyzing the causal relationship between two computation nodes (Sender → Receiver). This helps us determine whether Sender is the cause of Receiver, and the connections between them are important for the model in implementing the task.

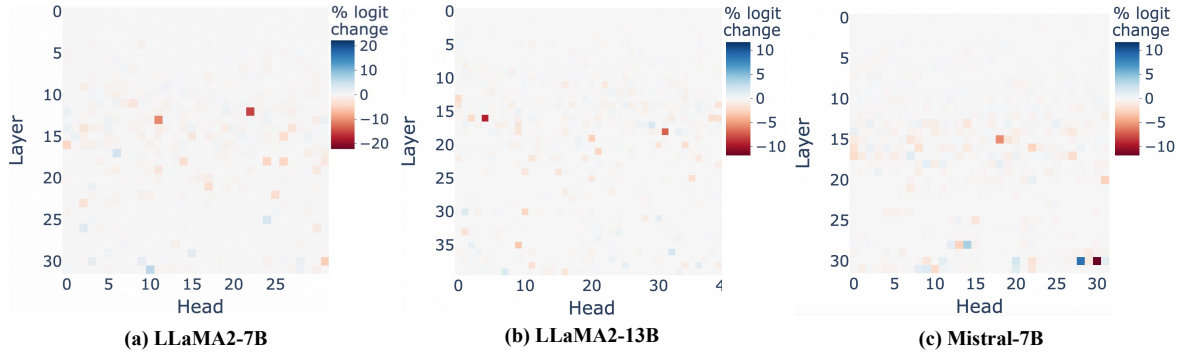
Specifically, the entire process of path patching is shown in Figure 11, where the node pair Sender → Receiver is set as Head

[4]We empirically find that the specific choice of words does not affect the results, as long as they meet similar semantics.

$0.31 \rightarrow$ Output. Firstly, given reference data X_{r} and counterfactual data X_{c}, the activations of all heads are gathered for preparation of the later perturbation. Then, we do a hard intervention on the Head 0.31 that is perturbated to its activation on X_{c}, where the effect will be further propagated to the Ouput node along with a set of paths $\mathcal{P}$. To ensure an independent observation of the impact from the Head 0.31, $\mathcal{P}$ comprises the forward pathways through residual connections and MLPs except for the other attention heads (*e.g.*, Head $0.0, \cdots, 0.30, 1.0, \cdots, 1.31$). Thus we do a hard intervention on the other heads by freezing their activations on X_{r}. Finally, we obtain the final output logits to measure the impact of this perturbation. If there is a significant change in final logits, then the patched paths: Sender $\rightarrow$ Receiver are essential for the model in completing the task.

In this work, to identify the important heads contributing to the calculation task, we scan through all heads as the Sender node denoted by h, and set the Receiver node as output $logits$, and measure the changes in the output logit of ground-truth token $\{\mathtt{C}\}$. Pathways $h \rightarrow logits$ that are critical to the model's computation should induce a large drop in the logit of token $\{\mathtt{C}\}$ after patching. Notably, since the residual operations and MLPs compute each token separately (Elhage et al., 2021), patching the head output at the END position (*i.e.*, the position of the last token in the input sentence) is enough to measure the effects on the next token prediction.

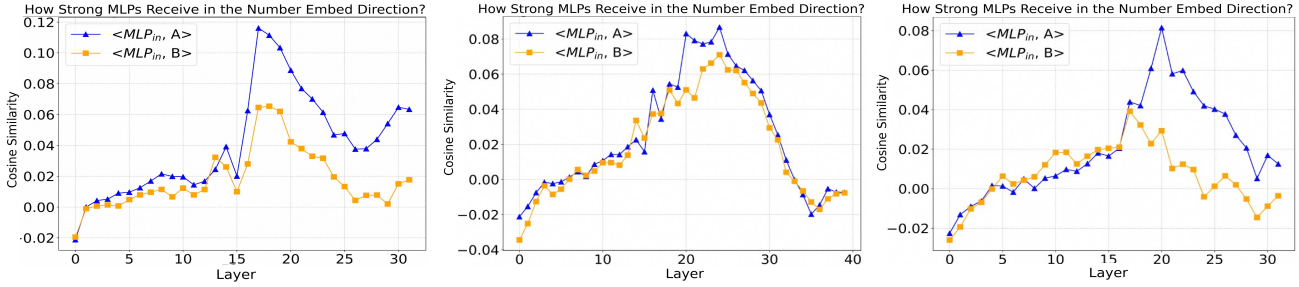
C. More Results of Other LLMs.



(a) LLaMA2-7B (b) LLaMA2-13B (c) Mistral-7B

Figure 12: Comparison of the results of path patching experiments on LLaMA2-7B, LLaMA2-13B, and Mistral-7B (Jiang et al., 2023) across four mathematical tasks. For each head/MLP, a darker color indicates a larger logit difference from the original model before patching.

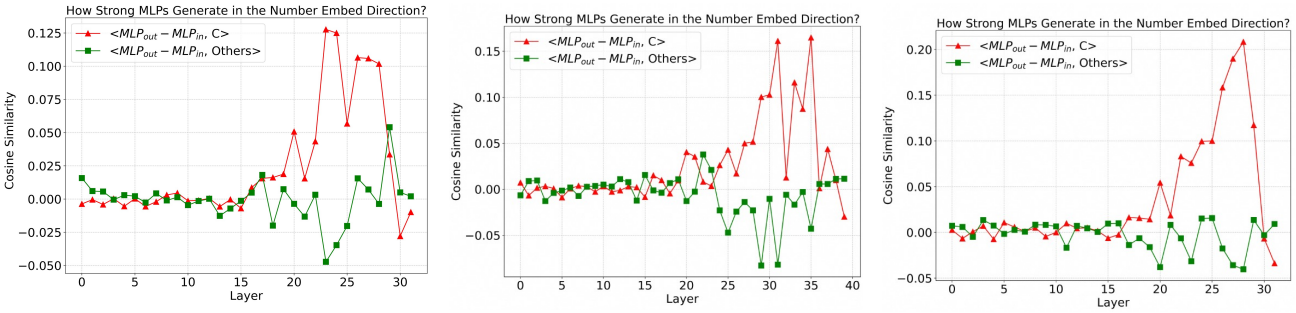
Key Component Identification. In Figure 12, we further report the results of key components identification of other models (*e.g.*, LLaMA2-13B and Mistral-7B). For example, LLaMA2-13B comprises 40 layers and 40 attention heads per attention layer. The three models of different size exhibit similar phenomena that the calculation-related key heads (*e.g.*, 16.4, 18.31) are distributed sparsely in the middle layers.



(a) LLaMA2-7B (b) LLaMA2-13B (c) Mistral-7B

Figure 13: We investigate the projection of each MLP layer input (MLP_{in}) along the direction of number token $\{\mathtt{A}\}$, $\{\mathtt{B}\}$, respectively.

Key MLPs Behavior. In Figure 13, the similarities of MLP input and number operands $\{\mathtt{A}\}$/$\{\mathtt{B}\}$ across all models demonstrate ascending and descending trends. Specifically, the pivotal points for these trends, delineated as (*start-*



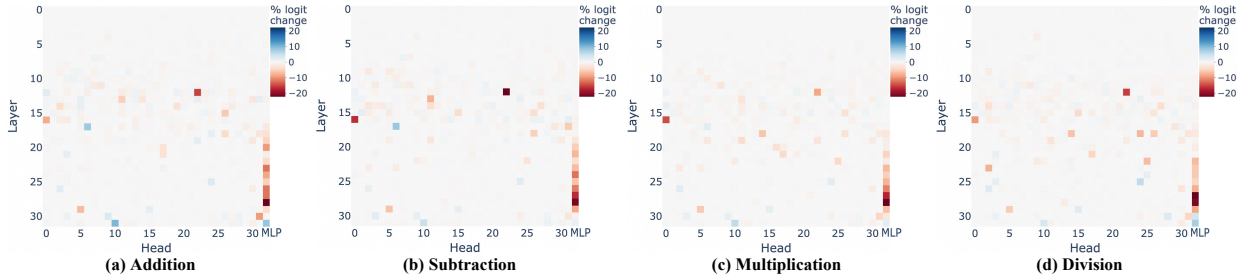
(a) LLaMA2-7B (b) LLaMA2-13B (c) Mistral-7B

Figure 14: We investigate the projection of each MLP layer (MLP_{out}-MLP_{in}) along the direction of number token {C} (*i.e.*, right answer) and other tokens (*i.e.*, wrong answer).

$inflection\text{-}end$), are as follows: (13-18-28) for LLaMA2-7B, (13-18-35) for LLaMA2-13B, and (13-20-28) for Mistral-7B. In Figure 14, the similarities of MLP_{out}-MLP_{in} and right answer {C} show a pattern of initial stabilization followed by an increase. The critical points for LLaMA2-7B/LLaMA2-13B/Mistral-7B are again (13-18-28), (13-18-35), and (13-20-28). The inflection points in both Figure 13 and Figure 14 are nearly identical, indicating consistent trend shifts across the models. It helps to verify that LLMs initially leverage attention heads then relaying information to downstream MLPs, to progressively carry out the calculation to final results. Furthermore, the above findings appear to be general and robust across different LLMs, not limited to a specific model.

D. Key Component Location across Calculation Tasks.

We investigate the location of key components for each calculation task individually, as shown in Figure 15. The discovered key heads could be shared across four tasks, which are sparsely distributed in the middle layers. Specifically, when examining subtraction and addition tasks, we could summarize two insightful symmetries between them. The identified key heads of two tasks are almost the same, albeit with different magnitude of the effect. This phenomenon could reveal the symmetry of key head "*location*" in addition and subtraction. Moreover, the tasks of multiplication and division exhibit a greater number of key heads compared to the tasks of addition and subtraction. We assume it could be attributed to their more intricate operations within multiplication and division.



(a) Addition (b) Subtraction (c) Multiplication (d) Division

Figure 15: We conduct path patching experiments on LLaMA2-7B across four mathematical tasks, by searching for each head and MLP directly affecting the logit of the right answer. The last column denotes the path patching results of MLPs. For each head/MLP, a darker color indicates a larger logit difference from the original model before patching.

Generalize to other calculation operations. We conduct the experiments of key head identification and validation following Section 5.1. We generate the samples including the exponentiation operation as the reference data X_{r}. Then we generate the counterfactual data X_{c} following the principles introduced in Section 4.1 to exclude the exponentiation logic.

The results reveal the potential of generalizing to more complex mathematical operations: (i) Five key heads are identified based on the newly generated X_{r} and X_{c}. We find that the heads (11, 8) and (14, 2) mainly attend to the operators "^", "power", while the heads (12, 22), (13, 11), (15, 15) mainly attend to the input operands {A} and {B}. (ii) Knocking out the key heads, identified by both templates, leads to significantly impacts (over 60%) on model performance.

Table 3: Key head identification on the exponentiation task.

Templates	Key Heads [*e.g.*, (Layer, Head)]	Knockout Accuracy
X_{r}: "{A} ∧ {B} = _ " X_{c}: "{A} < {B} = _ "	[(11, 8), (12, 22), (13, 11), (14, 2), (15, 15)]	−66%
X_{r}: "{A} to the **power** of {B} equals _ " X_{c}: "{A} to the **none** of {B} equals _ "	[(11, 8), (12, 22), (13, 11), (14, 2), (15, 15)]	−62%

E. Generalize to More Complex Scenarios.

We conduct experiments on the more complex scenario using the dataset GSM8K (Cobbe et al., 2021). At first, we create new reference data X_{r} and counterfactual data X_{c}. Following the idea of methodology proposed in Section 4.1, we convert the question in GSM8K to obfuscate the semantic elements that necessitate calculation, while ensuring that the alterations to the text are minimal. An example is shown below:

- GSM8K X_{r}: "On a 16 GB (gigabyte) capacity USB drive, 50% is already busy. **Calculate the number** of gigabytes still available."
- GSM8K X_{c}: "On a 16 GB (gigabyte) capacity USB drive, 50% is already busy. **Describe the location** of gigabytes still available."

Then, we conduct the experiments of key head identification and validation following the experimental setting in Section 5.1. As a result, 60% of the key heads are overlapped with the key heads identified based on our original data. Moreover, knocking out the newly-identified key heads leads to a 65% accuracy drop on GSM8K, confirming their importance even in complex scenarios.

Table 4: Comparison of the key heads identified on our generated data in Figure 8 and the dataset GSM8K (Cobbe et al., 2021).

Dataset	Top-10 Key Heads [*e.g.*, (Layer, Head)]	Knockout Accuracy
Ours	[(12, 22), (13, 11), (16, 0), (15, 26), (18, 26), (18, 24), (30,31), (14, 27), (22, 25), (11, 8)]	−69%
GSM8K	[(19, 6), (11, 8), (12, 22), (14, 31), (13, 11), (22, 25), (16, 0), (21, 17), (15, 26), (29, 5)]	−65%

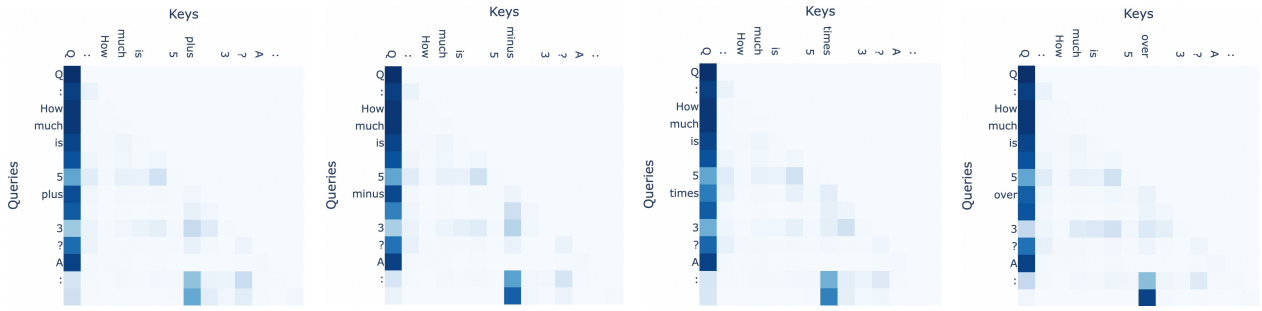
Furthermore, only knocking out the 6 overlapping heads brings in −56% and −52% on our generated data and GSM8K, respectively. It shows these heads are both important in two scenarios. If knocking out the 4 non-overlapping heads identified by GSM8K only, it has a negligible effect on our generated data (−2%) but apparently affects on GSM8K (−26%). It reveals the significance of these 4 heads specific to more complex reasoning mathematical problems. We further investigate the attention patterns of the 4 non-overlapping heads, and find that these heads mainly attend to text tokens. For example, the head (29, 5) attends to ".", and the head (19, 6) attends to "GB". In contrast, the 6 overlapping heads mainly attend to the number operands and operators. For example, the head (13, 11) attends to input operands "50", and the head (11, 8) attends to the operator "%".

Recent research (Opedal et al., 2024) has shown that solving the math word problems requires a synergy of multiple skills including 'text comprehension' and 'arithmetic calculation'. This is aligned with the phenomena of "the 4 non-overlapping heads attend to text tokens (*i.e.*, 'text comprehension'), while the 6 overlapping heads attend to number operands and operators (*i.e.*, 'arithmetic calculation')". In this work, we focus on the skill of arithmetic calculation as it's a fundamental ability universally shared across various levels of complexity for mathematical problems. It's imperative for continued research to develop a more holistic understanding of the intricate reasoning capacities.

To further investigate whether the model's deficiencies stem from a lack of mathematical abilities or a broader impairment in language processing, we evaluate LLaMA2-7B with key heads kept normal and knocked out on MMLU-Humanities benchmark (Hendrycks et al., 2020). The comparative performance was 42.9% for models with the key heads intact versus 42.6% for the knockout models. This negligible difference (−0.3%) suggests that the knockout of these heads does not significantly impact general language abilities.

F. More Attention Pattern Cases.

We show the attention patterns of the operator-attended heads (*e.g.*, 14.2) in Figure 16 that could attend to the tokens of "plus", "minus", "times", and "over", across different sentences.



(a) Head 14.2 (b) Head 14.2 (c) Head 14.2 (d) Head 14.2

Figure 16: The attention patterns of the key head 14.2, which mainly attend to the operator-related tokens, *e.g.*, "plus", "minus", "times", "over".

G. Ablation Study of Precise SFT on MLPs.

We further investigate the influence of different number of tuned MLPs in Table 5. It reveals that the tuning more MLPs could lead to a performance decrease on MMLU and more training time, while imrove the performance on math dataset GSM8K.

Table 5: Ablative experiments on the number of tunable MLPs.

Precise SFT Setting	**Evaluation Metric**			
	Train Speed	Tunable Params.	GSM8K	MMLU
top-32 heads	50sam./sec.	0.067B	27.4	46.4
top-32 heads + top-1 MLP	44sam./sec.	0.202B	27.5	46.0
top-32 heads + top-2 MLPs	38sam./sec.	0.338B	27.7	45.7
top-32 heads + top-3 MLPs	31sam./sec.	0.473B	28.0	45.2
top-32 heads + top-6 MLPs	26sam./sec.	0.879B	28.2	44.9
top-32 heads + all MLPs	19sam./sec.	4.396B	29.2	43.9

H. Calculation in Computer vs LLMs.

Addition in Computer:

3 + 4 =

Carry Input:		0	00	000
Addend: 3 →	011	011	011	011
Addend: 4 →	100	100	100	100
Result: 7		1	11	111
		Step-1: 1	Step-2: 3	Step-3: 7

Addition in LLMs:

3 + 4 =

Carry Input:		0	00	000
Addend: 3 →	011	011	011	011
Addend: 4 →	100	100	100	100
Result: 7		1	11	111
		Step-1: 4	Step-2: 6	Step-3: 7

Figure 17: The addition calculation process in computer and in LLMs.