Fine-Grained Manipulation of Arithmetic Neurons

Wenyu Du[1], Rui Zheng[2], Tongxu Luo[3], Stephen Chung[4] Jie Fu[5]

[1]The University of Hong Kong [2]Xi'an Jiaotong-Liverpool University
[3]The Chinese University of Hong Kong, Shenzhen
[4]University of Cambridge [5]Shanghai Artificial Intelligence Laboratory

Abstract

It is a longstanding challenge to understand how neural models perform mathematical reasoning. Recent mechanistic interpretability work indicates that large language models (LLMs) use a "bag of heuristics" in middle to late-layer MLP neurons for arithmetic, where each heuristic promotes logits for specific numerical patterns. Building on this, we aim for fine-grained manipulation of these heuristic neurons to causally steer model predictions towards specific arithmetic outcomes, moving beyond simply disrupting accuracy. This paper presents a methodology that enables the systematic identification and causal manipulation of heuristic neurons, which is applied to the addition task in this study. We train a linear classifier to predict heuristics based on activation values, achieving over 90% classification accuracy. The trained classifier also allows us to rank neurons by their importance to a given heuristic. By targeting a small set of top-ranked neurons (K=50), we demonstrate high success rates—over 80% for the ones place and nearly 70% for the tens place—in controlling addition outcomes. This manipulation is achieved by transforming the activation of identified neurons into specific target heuristics by zeroing out source-heuristic neurons and adjusting target-heuristic neurons towards their class activation centroids. We explain these results by hypothesizing that high-ranking neurons possess 'cleaner channels' for their heuristics, supported by Signal-to-Noise Ratio (SNR) analysis where these neurons show higher SNR scores. Our work offers a robust approach to dissect, causally test, and precisely influence LLM arithmetic, advancing understanding of their internal mechanisms.

1 Introduction

A longstanding debate (Stolfo et al., 2023; Mirzadeh et al., 2024; Zhou et al., 2024; Nikankin et al., 2024) exists regarding the mechanisms by which neural models perform mathematical reasoning, whether through extensive memorization, emergent algorithms, or more sophisticated mechanisms. Recent mechanistic interpretability work indicates that large language models (LLMs) perform tasks (jylin et al., 2024; Nikankin et al., 2024), including arithmetic (a fundamental class of mathematical problems), by activating a set of sparse neurons. In particular, Nikankin et al. (2024) finds that LLMs solve arithmetic prompts by utilizing *"a bag of heuristics"*. Each heuristic, embedded within a set of sparse neurons in the middle- to late-layer Multi-Layer Perceptrons (MLPs), activates upon detecting specific patterns in either the operands or the results, thereby increasing the logits of tokens within these patterns. The combination of these neurons constitutes the mechanism used to produce arithmetic answers. Figure 1 provides two heuristic examples, activating results patterns $\{..\mathbf{6}\}$ and $\{..\mathbf{7}\}$.

The finding on these heuristic neurons provides the means of examining how LLMs perform arithmetic reasoning in neuron-level modularity, which paves the way for many applications, such as pruning (removing non-arithmetic neurons), debugging and fixing failure cases (identifying a lack of certain heuristic neurons and training specifically to strengthen or develop them), etc. Besides the aforementioned applications, the discovery of these heuristic neurons opens up the possibility of neuron manipulation—altering model predictions by changing the activation values of identified neurons. Nikankin et al. (2024) has shown that manipulating a limited set of neurons can lead to a drastic drop in the model's arithmetic accuracy. However, existing approaches to arithmetic manipulation are typically too coarse—they can prevent the model from producing the correct result, but not force it to output a specific target. In this work, we explore how to perform finer-grained neuron manipulation. For instance, in Figure 1, if we consider prompts where

Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 467–479

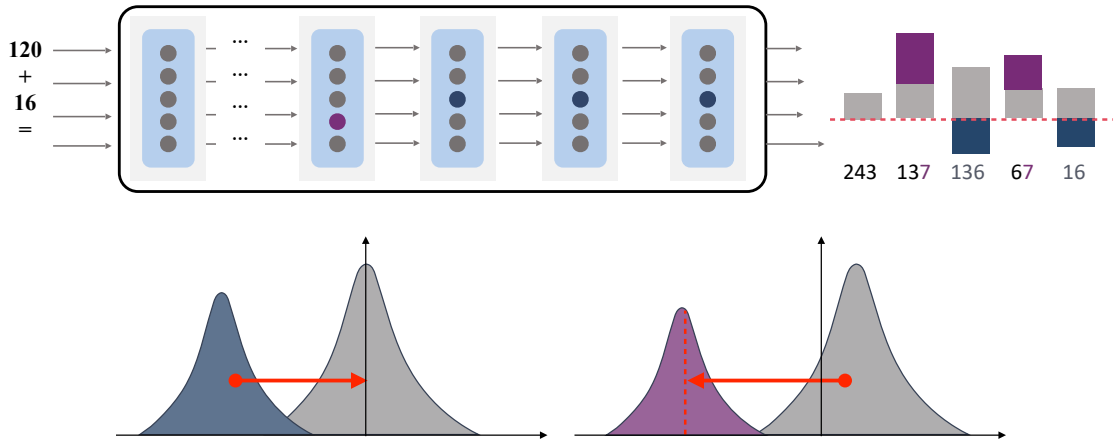


Figure 1: (a) Feed an arithmetic prompt "120+16=" into model, it predicts answer by "a bag of heuristics". By manipulating heuristic neurons, e.g. reducing the logits for $\{..\mathbf{6}\}_{\mathbf{result}}$ and promote logits for $\{..\mathbf{7}\}_{\mathbf{result}}$, now the model predicts 137 instead of 136. The manipulation takes two steps, we first (b) zero-out the top source-heuristic-important neurons $\{..\mathbf{6}\}_{\mathbf{result}}$ (e.g. Neuron[30,10778]) and then (c) shift activations of the top target-heuristic-important neurons to the activation centroid $\{..\mathbf{7}\}_{\mathbf{result}}$ (e.g. Neuron[28,10436]).

the correct answer fits a pattern $\{..\mathbf{6}\}$, could we alter the model's predictions to output a $\{..\mathbf{7}\}$ by manipulating the neurons associated with heuristics about $\{..\mathbf{6}\}$ and $\{..\mathbf{7}\}$? Answering this question is key to rigorously testing the functional role of these neurons and their causal influence on model output. If manipulating them can reliably induce specific errors—such as a known heuristic rather than a random mistake, as seen in prior studies—it would demonstrate a far more precise causal relationship between these neurons and the model's behavior. This level of control would validate the neurons' specialized roles and offer deeper insight into the computational circuits underlying arithmetic reasoning.

In this paper, we study how to manipulate arithmetic results in LLMs using these arithmetic neurons, focusing on the *Addition* operation[1]. We employ activation patching to localize arithmetic neurons, similar to (Nikankin et al., 2024), and define a minimal set of heuristic types that influence predictions. For each heuristic type, we train a linear two-layer classifier that takes neuron activations as input and predicts the heuristic class. This method achieves over 90% accuracy across all heuristics, indicating that neuron activations are key indicators for differentiating these heuristics. Moreover, the product of the two layer matrices can be interpreted as a weighted matrix mapping each neuron to each heuristic. By ranking the absolute values of these weights, we can determine the importance of individual neurons for each heuristic. After localizing arithmetic neurons and ranking neuron importance for each heuristic, we attempt to manipulate these neurons to influence the results. First, we attempt to disable the functions of certain heuristics by zeroing out the top-K heuristic-important neurons, thereby setting their activations to zero. Then, we attempt to transform specific source heuristics into specific target heuristics, as illustrated in Figure 1. This process first requires zeroing out the top-K source-heuristic-important neurons and then adjusting the activations of the top-K target-heuristic-important neurons to the activation centroid of all prompts that match the target heuristic. With only $K = 50$ neurons, we achieve a successful manipulation rate of over 80% for the unit digit and nearly 70% for the tens digit for all prompts.

Then, we seek an explainable understanding of this successful manipulation. By visualizing neuron activations across prompts from different heuristic classes, we find that higher-ranking neurons for a certain heuristic are more likely to exhibit visually separable activations for this heuristic than for other heuristics—akin to radios having a cleaner channel. Therefore, inspired by this clean channel analogy, we borrow the concept of Signal-to-Noise Ratio (SNR) to measure the degree of distribution separability. We find that top neurons generally have higher SNR scores, indicating a cleaner channel for a given heuristic. Lastly, we further discuss manipulating operand heuristics and model error

[1]We also report preliminary results for Subtraction in Appendix C.

predictions.

The contributions of this work are threefold:

1. We microscopically examine arithmetic heuristics and propose a linear classifier to rank neurons for each heuristic, demonstrating that this approach can systematically identify neurons that encode these heuristics.

2. We manipulate small subsets of the identified neurons and show that we can reliably steer the model to output any specific target value with high accuracy.

3. Borrowing the "clean channel" analogy from radio, we quantify and show that high-ranking neurons indeed possess a cleaner channel.

2 Background

Interpreting MLP in Neuron Form The MLP in the transformer block is normally described by the following equation:

$$\mathbf{y} = \mathrm{MLP}(\mathbf{x}; \mathbf{K}, \mathbf{V}) = \sigma(\mathbf{x} \cdot \mathbf{K}) \cdot \mathbf{V}, \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^d$ are the input and output, respectively; $\mathbf{K} \in \mathbb{R}^{d \times d_{mlp}}$ and $\mathbf{V} \in \mathbb{R}^{d_{mlp} \times d}$ are the up-projection and down-projection weight matrices, where d is the embedding size and d_{mlp} is the dimension of the MLP; and $\sigma(\cdot)$ is a non-linear activation function. We omit the bias term.
Following Geva et al. (2020, 2022); Qiu et al. (2023), for each column $\mathbf{K}_{:,i}$ and row $\mathbf{V}_{i,:}$, we can rewrite Equation 1 in a neuron form:

$$\mathbf{y} = \sum_{i=1}^{d_{mlp}} \sigma(\mathbf{x} \cdot \mathbf{K}_{:,i}) \cdot \mathbf{V}_{i,:} = \sum_{i=1}^{d_{mlp}} \mathbf{H}_{:,i} \cdot \mathbf{V}_{i,:}. \quad (2)$$

The neuron interpretation of MLP is that columns $\mathbf{K}_{:,i}$ are *key* vectors and rows $\mathbf{V}_{i,:}$ are *value* vectors, and each key-value vector pair constitutes a *neuron*. Then the output $\mathbf{y}$ can be expressed as a linear combination of value vectors $\mathbf{V}_{i,:}$ with their corresponding neuron activation scores $\mathbf{H}_{:,i} = \sigma(\mathbf{x} \cdot \mathbf{K}_{:,i})$.

Arithmetic Neurons Arithmetic neurons are a specific set of neurons, typically found to be sparsely activated within the middle and late layers of MLP in LLMs. These neurons play a significant role in the arithmetic prediction capabilities of LLMs. Formally, considering an LLM denoted as $\mathbf{M}$ and a set of its arithmetic neurons as $\mathbf{S}_{an}$, these neurons are defined such that the performance of the model after ablating these neurons ($\mathbf{M} - \mathbf{S}_{an}$) on arithmetic tasks is substantially lower (approaching near-zero accuracy) compared to the performance of the intact model ($\mathbf{M}$).

Arithmetic Heuristics Arithmetic heuristics are designed to enable a fine-grained analysis of these arithmetic neurons. Nikankin et al. (2024) introduced five types of arithmetic heuristics, into which these arithmetic neurons can be classified; a single neuron may be associated with one or more heuristic types. If a neuron is labeled with a specific heuristic, it is hypothesized that one of its functions is to activate when an input prompt aligns with the conditions of that heuristic. The neuron is then thought to promote the logit of tokens that corresponds to the heuristic pattern. The influence of an individual neuron might be subtle, but the collective effect of all arithmetic neurons significantly impacts the model's final output in arithmetic problems. For detailed description of arithmetic neurons and heuristics, please refer to (Nikankin et al., 2024).

3 Localizing Neurons and Defining Manipulation Heuristics

To investigate neurons for manipulating arithmetic results, we first need to choose appropriate models, create arithmetic data, and then localize these neurons and design arithmetic heuristics that these neurons operate on.

3.1 Models and Data

We analyze three LLMs from the Llama-3 series: Llama3-70B, Llama3-8B, Llama3.2-3B (Grattafiori et al., 2024). The model choice is based on two principles: the LLMs should achieve strong performance in arithmetic tasks (indicating a higher chance of existing clean arithmetic neurons) and should be able to tokenize numbers in the range [0-999] into a single token (since the scope of this work is on single-token numbers). We focus on Llama3-8B in the main paper and report similar results for the additional models in Appendix B. Following (Nikankin et al., 2024), we use pre-trained models without fine-tuning them on arithmetic prompts. We use two-operand arithmetic prompts with Arabic numerals such that each prompt consists of four tokens: $op1$, the operator $+$, $op2$, and the "=" sign. Each prompt is chosen

so that its operands and result are tokenized into a single token; e.g., for the four models, $op1$ and $op2$ are selected from the range $[0, 999]$ and we also filter out samples that the results are not in this range. We randomly sample $10,000$ prompts from valid candidates.

3.2 Localizing Arithmetic Neurons

To evaluate the impact of individual neurons on solving arithmetic problems, we employ the activation patching technique as described in (Vig et al., 2020; Nikankin et al., 2024). We apply this method to all individual neurons within the middle and late layers of the model (specifically, layers 16–31 for Llama3-8B).

The core concept of activation patching involves running the model on an original prompt p (e.g., "$120+16=$", which should yield result r). During this process, however, the activation of a single, targeted neuron is replaced with a substitute activation. This substitute activation is sourced from a separate run of the model using a counterfactual ("corrupted") prompt p' (e.g., "$543-165=$", which should yield a different result r'). In practice, we first perform a standard forward pass with the corrupted prompt p' and store the activation of the neuron under investigation. Then, during the processing of the original prompt p, as the model computes the activation for that neuron, we intervene (i.e., "patch") by substituting its activation with the stored activation from the p' context.

Following Stolfo et al. (2023), we quantify the impact of this intervention by measuring its Indirect Effect (IE) on the model's output probabilities for both the original answer token r and the counterfactual answer token r'. This effect is defined as:

$$\mathrm{IE}(\mathrm{r}, \mathrm{r}') = \frac{1}{2}\left[\frac{\mathbb{P}^*(r') - \mathbb{P}(r')}{\mathbb{P}(r')} + \frac{\mathbb{P}(r) - \mathbb{P}^*(r)}{\mathbb{P}^*(r)}\right] \tag{3}$$

Here, $\mathbb{P}$ and $\mathbb{P}^*$ represent the probability distributions over the vocabulary before and after the intervention, respectively. A high IE score for an intervention on a specific neuron indicates its significant role in the computation for the given prompt. This effect is averaged across multiple prompts and measured independently for each neuron. Based on these intervention scores, we identify the 320 neurons exhibiting the highest average IE scores as "arithmetic neurons". The distribution of these arithmetic neurons across the model layers is provided in Figure 2.

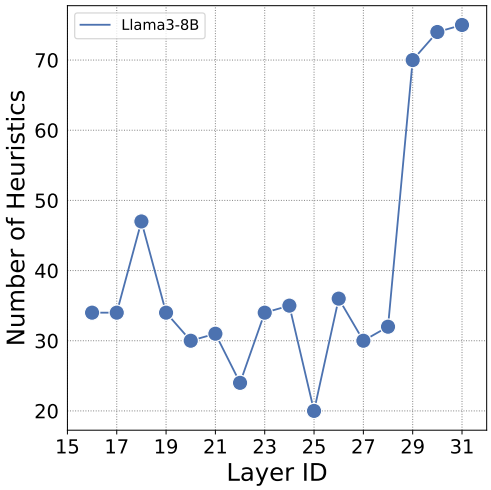


Figure 2: Arithmetic neuron numbers across layers

To validate the significance of localized arithmetic neurons, we conduct ablation studies under three experimental conditions: (1) *Arithmetic ablation*: zeroing the activations of the 320 identified arithmetic neurons; (2) *Random ablation*: masking 320 randomly selected neurons (repeated using three different random seeds); and (3) *Baseline*: no intervention. Arithmetic ablation catastrophically degrades performance, reducing accuracy from the baseline of 92.0% to just 0.3%. In contrast, random ablation (averaged over the three seeds) maintains accuracy at 92.1% ($\pm$0.5). This dramatic performance drop following arithmetic ablation demonstrates the successful localization of these arithmetic neurons and confirms that they are the key components for our investigation.

3.3 Defining Arithmetic Heuristics For Manipulation

Instead of the five-type categorization in (Nikankin et al., 2024), we only focus on heuristics directly related to manipulation, i.e. three types of result-related heuristics that on the ones place (i.e. $\{..\mathbf{X}\}_{\mathbf{result}}$), on the tens place (i.e. $\{.\mathbf{X}.\}_{\mathbf{result}}$), and on the hundreds place (i.e. $\{\mathbf{X}..\}_{\mathbf{result}}$), where $\mathbf{X} \in \{0, 9\}$. For instance, the prompt "120 + 16="(136) fills in the heuristics $\{..\mathbf{6}\}_{\mathbf{result}}$, $\{.\mathbf{3}.\}_{\mathbf{result}}$ and $\{\mathbf{1}..\}_{\mathbf{result}}$ for three heuristic types respectively.

4 Ranked Neurons Can Manipulate Arithmetic Results

Having localized the arithmetic neurons and defined heuristics, we next classify neurons based on

these heuristics and manipulate predictions using the corresponding heuristic neurons.

Classification via Absolute Maximum Activation Value Misses 80% of the Neurons In (Nikankin et al., 2024), neurons are classified by first extracting prompts that maximally activate them (i.e., those eliciting the highest absolute activation values). Then, for each heuristic, the intersection between these maximally activating prompts and the prompts expected to activate for that specific heuristic is examined. If the resulting matching score exceeds a threshold (0.6), the neuron is classified as belonging to that heuristic.

However, when we attempt to replicate this approach to classify neurons using our defined heuristics, we find that 20% (66 out of 320) of the arithmetic neurons were associated with at least one heuristic. This low classification rate suggests that relying solely on maximally activating prompts provides an incomplete understanding of how heuristics operate on these arithmetic neurons, indicating a more complex underlying mechanism.

4.1 Ranking Neurons via Importance to Each Heuristic

Instead of using high activation values, we design a classification task: For each heuristic type, we leverage prompts to train a heuristic classifier using a simplified two-layer MLP without bias and activation functions (i.e., two linear transformation matrices) on neuron activations. Formally, the two-layer MLP is defined as:

$$\hat{\mathbf{y}}_\mathbf{t} = \mathbf{a} \cdot \mathbf{W}_\mathbf{t}^{\mathbf{input}} \cdot \mathbf{W}_\mathbf{t}^{\mathbf{output}}, \qquad (4)$$

where $\mathbf{a} \in \mathbb{R}^{320 \times N}$ is the sets of activation values of arithmetic neurons of N prompts and $\mathbf{y_t} \in \mathbb{R}^{X \times N}$ is the corresponding classification predictions for all heuristics of heuristic type t. The classifier is then optimized using cross-entropy between label y_t and prediction $\hat{y}_t$ with the AdamW optimizer. The results achieve over 90% accuracy across all types, indicating that neuron activations are key indicators for separating heuristics. The product of the two matrices can be viewed as a weighted matrix $\mathbf{W_t} = \mathbf{W}_\mathbf{t}^{\mathbf{input}} \cdot \mathbf{W}_\mathbf{t}^{\mathbf{output}} \in \mathbb{R}^{320 \times X}$ [2] mapping from each neuron to each heuristic, indicating the neuron importance ranking for each heuristic. By ranking the absolute values of weights for each heuristic, we can rank the importance of individual neurons for each heuristic. We use 8000 samples for training and the remaining 2000 samples for evaluation. Detailed training parameters are provided in Appendix A.

[2] Although $\mathbf{W_t}$ has the same expressive power as $\mathbf{W}_\mathbf{t}^{\mathbf{input}} \cdot \mathbf{W}_\mathbf{t}^{\mathbf{output}}$, in practice we find that the latter is easier to optimize.

4.2 Manipulating Predictions

After ranking neurons by their importance to each heuristic, we next study how to alter neuron activations to influence the final result, i.e., how to manipulate results.

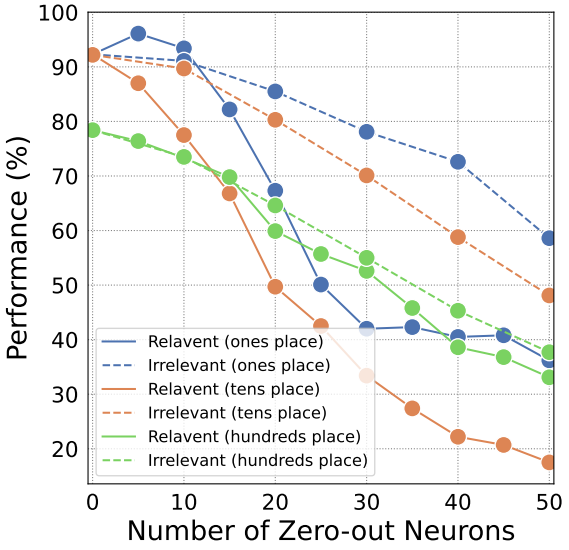


Figure 3: Accuracy drops more dramatically by zeroing out relevant heuristic neurons (solid lines) than irrelevant heuristic neurons (dashed lines).

Zeroing Out Heuristics We start by zeroing out certain heuristics, i.e., disabling Llama3-8B's arithmetic capabilities related to a particular heuristic. For instance, for heuristic $\{..\mathbf{6result}\}$, we set the activations of the neurons ranked in the top-K on the importance list in Section 4.1 to zero. Then, we pass evaluation prompts containing heuristic $\{..\mathbf{6result}\}$ to the edited model. Figure 3 plots the accuracy drops across all $..\mathbf{X_{result}}$ heuristics as K increases. It is clear that with a larger K (meaning more heuristic-specific neurons are masked), the performance on heuristic-related prompts drops dramatically, while other prompts are only mildly affected. We assume this drop in performance on these other prompts can be attributed to superposition.

Manipulating Results After demonstrating that using dozens of heuristic-important neurons can successfully disable a certain heuristic, we then

Table 1: Manipulating rate of the top 50 neurons for heuristic pairs $\{..\mathbf{X}_{\mathbf{result}}\} \rightarrow \{..\mathbf{X}_{\mathbf{result}}\}$

	Source Heuristic										
Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
0	N/A	**99.0%**	**97.1%**	**93.7%**	78.8%	70.8%	69.3%	76.2%	**94.2%**	**99.4%**	86.5%
1	**98.6%**	N/A	**99.2%**	**98.2%**	86.2%	82.8%	81.7%	72.9%	85.6%	**97.6%**	89.2%
2	**95.2%**	**98.7%**	N/A	**99.5%**	**97.4%**	**96.0%**	**94.3%**	85.0%	84.2%	**90.2%**	**93.4%**
3	89.1%	**99.5%**	**99.5%**	N/A	**99.2%**	**98.8%**	**97.1%**	**92.2%**	84.1%	82.6%	**93.6%**
4	29.9%	55.2%	83.4%	**90.1%**	N/A	84.4%	85.1%	53.4%	29.1%	21.9%	59.2%
5	37.4%	60.6%	84.2%	**96.4%**	**98.0%**	N/A	**97.4%**	83.4%	43.9%	45.4%	71.9%
6	47.5%	61.8%	84.2%	82.8%	84.3%	89.1%	N/A	**92.2%**	89.8%	61.1%	77.0%
7	87.1%	89.7%	83.8%	89.3%	**96.0%**	**97.5%**	**98.5%**	N/A	**99.9%**	**98.1%**	**93.3%**
8	85.6%	77.1%	62.9%	61.4%	69.3%	73.7%	**92.4%**	**97.8%**	N/A	**98.0%**	79.8%
9	**93.4%**	**91.3%**	69.3%	58.0%	36.8%	38.8%	65.6%	84.8%	**95.3%**	N/A	70.4%
Col Avg.	73.8%	81.4%	84.8%	85.5%	81.8%	81.3%	86.8%	82.0%	78.5%	77.1%	**81.4%**

investigate how to manipulate the results from one source pattern (heuristic) to a target heuristic. For example, if we zero out the top neurons for $\{..\mathbf{6result}\}$ but activate the top neurons for $\{..\mathbf{7result}\}$, will the predictions for arithmetic problems with answers fitting the pattern $\{..\mathbf{6result}\}$ now shift to answers fitting the pattern $\{..\mathbf{7result}\}$? To do so, we first set the corresponding top source neuron activations to 0, as described above, and then adjust the top target neuron activations to the mean activation value of all prompts that fit the target heuristic. Interestingly, with only $K = 50$, we can manipulate the outcomes for over 80% (81.4%) of problems in the ones place and nearly 70% (68.1%) in the tens place. Tables 1 provides the detailed manipulation rates for $K = 50$. Experiments concerning the tens, hundreds place and other values of K are presented in Appendix D.1.

Failure Manipulation Analysis We performed an error analysis on one successful manipulating pair and one unsuccessful manipulating pair: $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{2}\}_{\mathbf{result}}$ (accuracy 95.2%) and $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{4}\}_{\mathbf{result}}$ (accuracy 29.9%), respectively. The failed manipulations for $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{2}\}_{\mathbf{result}}$ are most likely due to special cases, such as problems resulting in sums that are multiples of one hundred (e.g., $299{+}1 =$) or where $op1 = op2$ (e.g., $360{+}360 =$). In contrast, the failure cases for $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{4}\}_{\mathbf{result}}$ vary. We put some error cases in Appendix D.2.

One interesting finding is that, particularly in the ones place, poor manipulation accuracy often occurs between two heuristics whose values differ significantly (e.g., by 4 or 5). Conversely, adjacent heuristics, such as $\{..\mathbf{1}\}_{\mathbf{result}}$ and $\{..\mathbf{2}\}_{\mathbf{result}}$, exhibit very close manipulation outcomes. We assume this might be related to the intern helical representation of numbers (Kantamneni and Tegmark, 2025), which we leave for future work.

5 Characteristics and Mechanisms of Ranked Neurons

In this section, we investigate the success behind the ranking approach: why highly ranked neurons perform well.

5.1 Clean Channel Metaphor

To better understand what makes high-ranking neurons different, we first visualize activation value distributions across different heuristic types on these high-ranking neurons. Some neuron visualizations reveal clear distinctions in activation ranges between specific heuristics, as exemplified in Figure 1 (c) (ranked $2_{nd}/320$ in $\{..\mathbf{7}\}_{\mathbf{result}}$), where the $\{..\mathbf{7}\}_{\mathbf{result}}$ heuristic demonstrates markedly different activation characteristics compared to other $\{..\mathbf{X}\}_{\mathbf{result}}$ variants. We also plotted the same visualizations for these low-ranking neurons in Appendix E. In these instances, no such clear distinctions between different heuristics are evident.

This observation motivates our clean channel metaphor: each arithmetic neuron operates anal-

ogously to a radio receiver, with activation values corresponding to frequency channels. When a heuristic's activations maintain sufficient distinction from others in value space, they establish a clear communication channel—the neuron reliably executes the corresponding heuristic function when inputs resonate within this activation range. Conversely, for each heuristic, two failure modes might emerge: 1) indistinguishable activation ranges among heuristics create noisy, overlapping channels, and 2) near-zero activations indicate inactive channels where the neuron remains functionally dormant for this heuristic type. In practice, the two aforementioned failure modes often co-occur.

5.2 Quantity Clean Channels Using SNR

Inspired by the above clean channel metaphor, we use the term Signal-to-Noise Ratio (SNR), as it is commonly used to quantify clean channels in radio.

Modeling Heuristic Activations as Gaussian Distributions First, we need to model each heuristic's activation distribution as a Gaussian distribution. Formally, we have the Gaussian probability density function (PDF) as:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

Let $X = x_1, x_2, ..., x_n$ represent all activation values for a certain heuristic on the neuron, where n is the number of activation values. Then, we estimate the parameters $\hat{\mu}$ and $\hat{\sigma}$ of the Gaussian distribution via Maximum Likelihood Estimation (MLE).

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

$$\hat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2} \tag{7}$$

Quantifying Clean Channels Using SNR Next, we quantify the clean channel for the neural radio: for each heuristic in the heuristic type, we calculate the SNR using Cohen's d effect size between the target heuristic distribution (signal) and the combined distribution of all other groups (noise groups).

$$\text{SNR} = \frac{\mu_{\text{signal}} - \mu_{\text{noise}}}{\sqrt{\sigma^2_{\text{signal}} + \sigma^2_{\text{noise}}}} \tag{8}$$

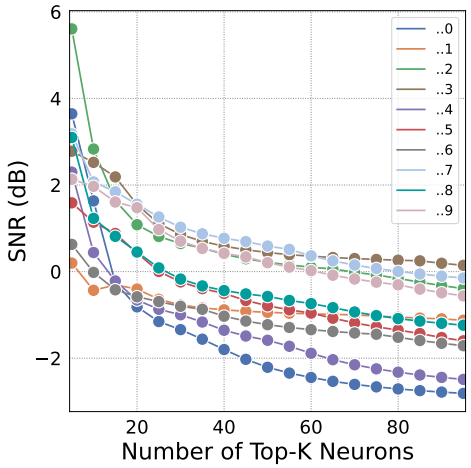


Figure 4: Average SNR scores of Top-K neurons drop as K increases

Where the distribution of the combined noise groups is as follows:

$$\mu_{\text{noise}} = \frac{1}{N}\sum_{i=1}^{N} \mu_i \tag{9}$$

$$\sigma_{\text{noise}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\sigma_i^2 + (\mu_i - \mu_{\text{noise}})^2\right)} \tag{10}$$

In practice, we use the SNR in dB form:

$$\text{SNR}_{\text{DB}} = 10 \times \log_{10}(\text{SNR}^2) \tag{11}$$

A higher dB value indicates a cleaner channel for the target heuristic for the neural radio. We plot the the average SNR_{DB} scores across different top-K ranked neurons in Figure 4. It is clear that higher-ranking neurons are more likely to have a higher score in SNR_{DB}. We also tried to use SNR as the ranker to select top-K and the performance is much better than random but have a clear gap than classifier ranking approach.

6 Additional Findings

6.1 Finding 1: Heuristic Manipulation Can Correct Wrong Predictions

We applied the aforementioned manipulation technique to correct incorrect predictions made by the original model. Despite the model's strong arithmetic capabilities, it still makes mistakes. For example, in the ones place, the model itself made a total of 388 incorrect predictions. By applying the manipulation technique, we were able to correct 213 of these predictions using the top-50 neurons. A detailed breakdown is shown below in Figure 5.

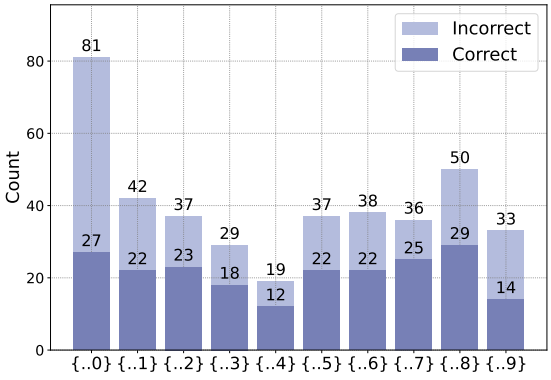


Figure 5: Manipulating can correct some predictions

6.2 Finding 2: Manipulating with Heuristics of Operands Can Improve Performance

A limitation of the previous approach is that we only utilized result-based heuristics. We also incorporated corresponding operands-based heuristics to further improve performance beyond what was achievable with only result-based heuristics. Specifically, for each result heuristic pair $\{..\mathbf{A}\}_{\mathbf{result}} \rightarrow \{..\mathbf{B}\}_{\mathbf{result}}$, we set two offsets $\mathbf{C}, \mathbf{D}$ for op1 and op2 respectively that $\{..\mathbf{X}\}_{\mathbf{op1}} \rightarrow \{..\mathbf{X}+\mathbf{C}\}_{\mathbf{op1}}$ and $\{..\mathbf{Y}\}_{\mathbf{op2}} \rightarrow \{..\mathbf{Y}+\mathbf{D}\}_{\mathbf{op2}}$, where $\mathbf{C}+\mathbf{D} = |\mathbf{B}-\mathbf{A}|$. By doing so, we can align the calculation of operands with the target result. We observed mild improvements compared to using only result-based approaches, achieving an average improvement of 3.5% across five manipulation pairs that previously performed poorly when using only result-based heuristics. We plot the results to Figure 6.

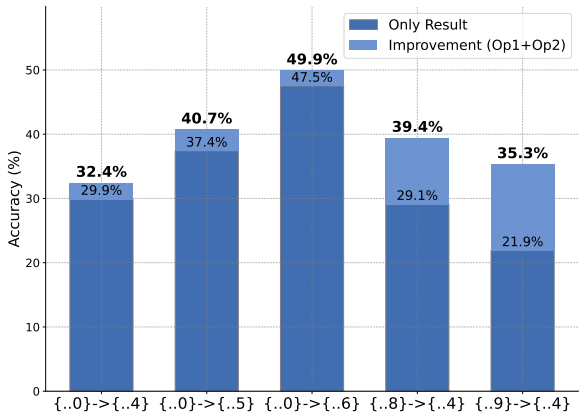


Figure 6: Heuristics on Operands can further increase manipulation rate

7 Related Work

7.1 Mechanistic Interpretability

Mechanistic interpretability (MI) seeks to uncover how language models work internally by studying their structural components. A circuit refers to a small group of interconnected elements that carries out a particular computation (Olah et al., 2020; Elhage et al., 2021). Methods like activation patching (Vig et al., 2020; Meng et al., 2022; Stolfo et al., 2023) allow researchers to test the causal influence of these circuits on model outputs. Rai et al. (2025) provides a detailed survey on MI. In this study, we apply such activation patching methods to detect and rank heuristic-specific neurons as sub-circuits for arithmetic computation and confirm their functional role through focused manipulation.

7.2 Mechanistic Study on Arithmetic

Recent work in mechanistic interpretability has explored the emergence of arithmetic competence in LLMs through fine-grained, neuron-level analysis. These studies have identified arithmetic sub-circuits within MLP layers, uncovering digit-position-specific mechanisms in which distinct neuronal groups independently compute units in parallel (Levy and Geva, 2024; Rhys Gould and Conmy, 2023; Jack Lindsey, 2025). Further investigations have demonstrated that arithmetic behavior is mediated by sparse collections of neurons that implement simple heuristics, including digit-specific operations and modular arithmetic patterns (Stolfo et al., 2023; Nikankin et al., 2024). Building on this line of inquiry, our work advances the field by systematically identifying and manipulating heuristic neurons, and by showing that targeted activation edits enable reliable control over arithmetic outputs beyond simple accuracy degradation.

8 Conclusion

This paper studies how to systematically identify and causally manipulate arithmetic heuristic neurons of LLMs. The goal is to steer the model's predictions towards specific arithmetic outcomes, going beyond simply disrupting the model's accuracy. By training a classifier to identify heuristic classes based on their activation, we can rank neurons by their importance to each heuristic. Targeting just 50 important neurons, we demonstrated significant success in controlling the outcomes (over 80% for the ones place). We then show that these top neurons are more effective due to "cleaner channels",

supported by higher SNRs. This work offers a method to dissect, test, and influence LLM arithmetic, enhancing understanding and enabling more precise model interventions.

9 Limitations

This work has two limitations. The first is that our analysis focuses on LLMs that combine digits in tokenization. That is, each token can contain more than one digit. The robust algorithms used by humans depend on our ability to separate larger numbers into single digits. Thus, a similar analysis might lead to different conclusions for models that perform single-digit tokenization. The second limitation is that the manipulation is not perfect. We assume this is because of superposition, which we leave for future work.

References

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

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Emmanuel Ameisen Brian Chen Adam Pearce Nicholas L. Turner Craig Citro David Abrahams Shan Carter Basil Hosmer Jonathan Marcus Michael Sklar Adly Templeton Trenton Bricken Callum McDougall Hoagy Cunningham Thomas Henighan Adam Jermyn Andy Jones Andrew Persic Zhenyi Qi T. Ben Thompson Sam Zimmerman Kelley Rivoire Thomas Conerly Chris Olah Joshua Batson Jack Lindsey, Wes Gurnee. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

jylin, JackS, Adam Karvonen, and Can Rager. 2024. Othellogpt learned a bag of heuristics.

Subhash Kantamneni and Max Tegmark. 2025. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*.

Amit Arnold Levy and Mor Geva. 2024. Language models encode numbers using digit representations in base 10. *arXiv preprint arXiv:2410.11781*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *arXiv preprint arXiv:2410.21272*.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

Zihan Qiu, Zeyu Huang, and Jie Fu. 2023. Unlocking emergent modularity in large language models. *arXiv preprint arXiv:2310.10908*.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2025. A practical review of mechanistic interpretability for transformer-based language models. *Preprint*, arXiv:2407.02646.

George Ogden Rhys Gould, Euan Ong and Arthur Conmy. 2023. Successor heads: Recurring, interpretable attention heads in the wild. *arXiv preprint arXiv:2312.09230*.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. 2024. Pre-trained large language models use fourier features to compute addition. *arXiv preprint arXiv:2406.03445*.

A Training Details on Classification

The training dataset consists of 10,000 arithmetic prompts. A [80:20] ratio was used to partition the data into training (n = 8,000) and test (n = 2,000). The two-layer linear neural network was trained for 10 epochs on a NVIDIA GeForce A100 GPU in a total training time within 1 minute. 75% dropout

was applied to FFN. Using AdamW with a learning rate of 1e-4 and cross-entropy loss for classification. The model was trained for 10 epoch. During training, we tracked training/evaluate accuracy, loss and F1-score.

B Other Models

We provide our experiments on other two Llama3 variants, Llama3.2-3B and Llama3-70B. For Llama3.2-3B, we use the exactly the same setting as for Llama3-8B and provide the manipulation results with $K = 50$ on Table 5. For Llama3-70B, we localize 640 arithmetic neurons and provide the manipulation results with $K = 50$ on Table 6. We show the manipulation can work across different model sizes.

C Subtraction

We provide preliminary results on Subtraction here. We localize 640 arithmetic neurons and use the same heuristics as in Addition. We find 640 neurons' activations still unable to produce over 90% accuracy on heuristic classes. We assume this is because we might need another set of heuristics types. But still, when we use top-150 neurons to manipulate in unit digit subtraction results, we still achieve over 60% success rate. We provide the Table to 4.

D Other manipulation results

D.1 Manipulation

We provide the manipulation Table 2 with $K = 50$ on the tens place, Table 3 with $K = 100$ on the hundred place.

D.2 Error Analysis

Error Cases on good manipulation pair We sample five failed manipulation on $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{2}\}_{\mathbf{result}}$ pair. It seems the models either fail to produce outputs or some special cases.

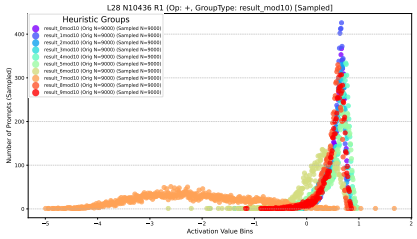
$$299 + 1 = \boxed{300}$$
$$95 + 195 = \boxed{NULL}$$
$$360 + 360 = \boxed{720}$$
$$70 + 390 = \boxed{NULL}$$
$$206 + 484 = \boxed{NULL}$$

Error Cases on bad manipulation pair While we also sample five failed manipulation on $\{..\mathbf{0}\}_{\mathbf{result}} \rightarrow \{..\mathbf{4}\}_{\mathbf{result}}$ pair. It seems the conditions vary.

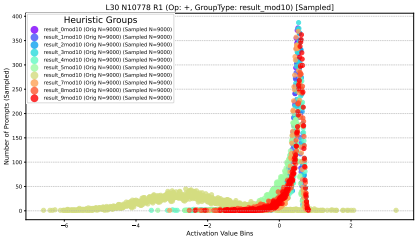
$$560 + 350 = \boxed{910}$$
$$200 + 460 = \boxed{660}$$
$$115 + 75 = \boxed{NULL}$$
$$170 + 290 = \boxed{460}$$
$$220 + 20 = \boxed{240}$$

E Visualization on clean and noise channels

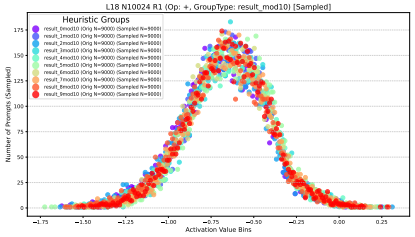
We visualize two clean channels and two noise ones.



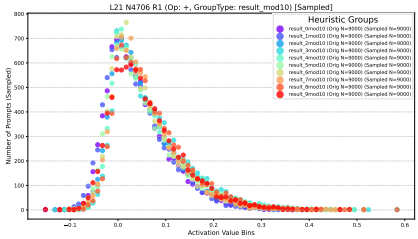
(a) Clean Channel 1: Neuron[28,10436] ranked $10/320$ at $\{..\mathbf{7}\}_{\mathbf{result}}$



(b) Clean Channel 2: Neuron[30,10778] ranked $2/320$ at $\{..\mathbf{6}\}_{\mathbf{result}}$



(c) Noise Channel 1: Neuron[18,10024]



(d) Noise Channel 2: Neuron[21,4706]

Table 2: Manipulating rate of the top 50 neurons for heuristic pairs $\{.\mathbf{X}_{\cdot\mathbf{result}}\} \rightarrow \{.\mathbf{X}_{\cdot\mathbf{result}}\}$

	Source Heuristic (Tens Digit Range)										
Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
0	N/A	76.4%	87.7%	70.5%	78.8%	64.9%	72.7%	50.7%	82.1%	69.6%	72.6%
1	76.0%	N/A	76.8%	**90.4%**	55.2%	84.1%	64.0%	66.1%	45.0%	66.0%	69.3%
2	81.5%	64.4%	N/A	75.1%	78.8%	57.8%	73.0%	46.2%	54.4%	31.9%	62.6%
3	48.6%	73.9%	50.8%	N/A	63.1%	87.2%	43.6%	74.7%	41.7%	58.3%	60.2%
4	85.4%	58.3%	82.3%	84.7%	N/A	85.3%	87.5%	67.0%	80.6%	53.1%	76.0%
5	57.6%	71.3%	55.3%	85.0%	73.3%	N/A	76.2%	87.7%	64.8%	75.2%	71.8%
6	78.8%	60.1%	71.2%	55.4%	71.4%	74.5%	N/A	72.9%	83.2%	55.9%	69.3%
7	46.3%	73.7%	44.5%	64.4%	43.9%	76.8%	66.9%	N/A	49.1%	78.3%	60.4%
8	88.0%	50.9%	66.6%	55.7%	75.3%	64.3%	80.1%	70.3%	N/A	75.4%	69.6%
9	68.8%	75.2%	55.6%	70.5%	54.8%	74.5%	59.1%	87.4%	77.5%	N/A	69.3%
Col Avg.	70.1%	67.1%	65.6%	72.4%	66.1%	74.4%	69.2%	69.2%	64.3%	62.6%	**68.1%**

Table 3: Manipulating rate of the top 100 neurons for heuristic pairs $\{\mathbf{X}_{\cdot\cdot\mathbf{result}}\} \rightarrow \{\mathbf{X}_{\cdot\cdot\mathbf{result}}\}$

		Source Heuristic										
	Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
Target Heuristic	**0**	N/A	7.2%	17.5%	19.4%	22.9%	16.7%	14.9%	22.1%	17.3%	12.0%	16.7%
	1	16.9%	N/A	51.1%	66.2%	52.8%	21.6%	17.5%	13.7%	35.8%	38.6%	34.9%
	2	0.0%	0.0%	N/A	47.2%	26.3%	41.3%	28.3%	39.5%	30.4%	27.3%	26.7%
	3	3.9%	0.7%	46.8%	N/A	48.4%	67.9%	66.7%	73.3%	65.6%	61.7%	48.3%
	4	0.0%	0.0%	11.9%	33.5%	N/A	58.1%	31.0%	35.8%	51.8%	47.7%	30.0%
	5	0.0%	0.0%	16.7%	43.3%	54.6%	N/A	58.7%	75.1%	73.3%	72.9%	43.8%
	6	0.0%	0.7%	61.1%	83.9%	85.7%	87.1%	N/A	89.2%	86.0%	85.8%	64.4%
	7	0.0%	0.0%	18.2%	45.3%	70.0%	79.9%	67.0%	N/A	85.1%	78.5%	49.3%
	8	0.0%	0.0%	2.9%	21.8%	49.2%	29.9%	12.2%	29.5%	N/A	45.9%	21.3%
	9	0.0%	1.1%	37.3%	61.6%	71.3%	84.6%	62.8%	85.1%	85.8%	N/A	54.4%
	Col Avg.	2.3%	1.1%	29.3%	46.9%	53.5%	54.1%	39.9%	51.5%	59.0%	52.3%	**39.0%**

Table 4: Manipulating rate of the top 150 neurons for $\{..\mathbf{X_{result}}\} \rightarrow \{..\mathbf{X_{result}}\}$ on LLAMA3-8B Subtraction

	Source Heuristic										
Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
0	N/A	25.2%	27.2%	16.2%	17.2%	8.6%	12.1%	20.3%	58.9%	71.6%	28.6%
1	79.5%	N/A	87.4%	65.7%	36.9%	23.2%	51.8%	32.2%	58.1%	57.2%	54.7%
2	80.2%	**91.5%**	N/A	76.4%	79.1%	82.9%	54.5%	78.1%	72.8%	76.5%	76.9%
3	66.5%	68.0%	88.7%	N/A	78.3%	82.2%	**91.0%**	78.7%	72.5%	49.5%	75.1%
4	32.4%	48.6%	77.6%	68.0%	N/A	87.2%	80.0%	80.3%	67.3%	30.4%	63.6%
5	31.3%	21.1%	44.7%	57.1%	77.8%	N/A	88.3%	89.6%	85.3%	44.3%	60.0%
6	46.01%	50.8%	59.1%	59.8%	68.8%	85.6%	N/A	89.2%	88.4%	69.9%	68.6%
7	50.4%	19.8%	30.2%	50.7%	59.9%	81.6%	86.6%	N/A	**97.3%**	54.5%	59.0%
8	72.8%	55.9%	62.0%	48.5%	65.8%	80.5%	88.7%	**94.0%**	N/A	**91.0%**	73.3%
9	78.8%	44.4%	42.3%	28.8%	20.8%	27.2%	67.5%	81.1%	**96.0%**	N/A	54.1%
Col Avg.	59.8%	47.3%	57.7%	52.4%	56.0%	62.1%	69.0%	71.6%	78.5%	60.6%	**61.4%**

Table 5: Manipulating rate of the top 50 neurons for heuristic pairs $\{..\mathbf{X_{result}}\} \rightarrow \{..\mathbf{X_{result}}\}$ on LLAMA3.2 3B

	Source Heuristic										
Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
0	N/A	59.8%	48.4%	39.8%	47.5%	37.2%	29.4%	44.3%	51.9%	35.7%	43.8%
1	53.7%	N/A	57.0%	51.9%	45.9%	40.2%	37.3%	35.8%	41.0%	24.8%	43.1%
2	61.9%	58.7%	N/A	57.2%	52.0%	46.7%	50.5%	47.1%	43.0%	30.2%	49.7%
3	47.6%	58.2%	63.4%	N/A	56.9%	55.2%	58.6%	57.6%	53.0%	31.3%	53.5%
4	48.9%	55.9%	58.3%	61.7%	N/A	63.8%	61.5%	64.1%	57.2%	30.8%	55.8%
5	21.1%	29.9%	36.0%	46.7%	46.3%	N/A	54.6%	52.5%	49.2%	19.2%	39.5%
6	52.3%	50.1%	47.1%	56.6%	63.4%	63.1%	N/A	65.7%	60.0%	39.7%	55.3%
7	47.3%	37.4%	28.8%	42.3%	48.6%	50.4%	52.6%	N/A	56.3%	30.8%	43.8%
8	61.4%	63.9%	51.2%	54.5%	59.5%	65.9%	63.7%	65.2%	N/A	47.3%	59.2%
9	61.9%	59.1%	46.5%	43.2%	38.9%	43.1%	50.3%	49.8%	64.0%	N/A	50.8%
Col Avg.	50.7%	52.6%	48.5%	50.4%	51.0%	51.7%	50.9%	53.6%	52.8%	32.2%	**49.4%**

Table 6: Manipulating rate of the top 640 neurons for heuristic pairs $\{..\mathbf{X}_{\mathbf{result}}\} \rightarrow \{..\mathbf{X}_{\mathbf{result}}\}$ on LLAMA3 70B

	Source Heuristic										
Target	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Row Avg.**
0	N/A	50.4%	46.1%	40.6%	45.1%	35.8%	39.1%	32.5%	44.3%	44.8%	42.1%
1	39.9%	N/A	43.8%	35.5%	31.8%	26.2%	25.5%	20.3%	23.3%	31.4%	30.9%
2	35.3%	45.4%	N/A	42.0%	43.2%	32.6%	29.1%	28.5%	30.4%	30.0%	35.2%
3	34.6%	38.1%	44.1%	N/A	44.2%	38.6%	36.9%	33.9%	33.2%	34.1%	37.6%
4	42.1%	42.3%	46.1%	41.5%	N/A	48.1%	46.7%	41.3%	41.4%	38.5%	43.1%
5	66.2%	68.0%	69.4%	68.6%	77.3%	N/A	79.0%	70.5%	23.3%	67.6%	65.6%
6	36.1%	37.9%	36.0%	31.3%	41.6%	47.7%	N/A	50.0%	47.2%	36.8%	40.6%
7	50.7%	52.9%	52.4%	46.7%	52.0%	54.1%	62.3%	N/A	64.8%	56.7%	54.8%
8	40.5%	36.4%	34.2%	33.8%	36.6%	34.6%	41.3%	41.8%	N/A	44.3%	38.2%
9	53.2%	48.5%	45.0%	39.6%	41.4%	37.4%	41.8%	44.0%	56.0%	N/A	45.3%
Col Avg.	44.3%	46.7%	46.4%	42.2%	46.0%	39.5%	44.7%	40.4%	40.5%	42.7%	**42.3%**