Modular Arithmetic: Language Models Solve Math Digit by Digit

Tanja Baeumel[1,2,3] **Daniil Gurgurov**[1,2] **Yusser Al Ghussin**[1,2]
Josef van Genabith[1,2] **Simon Ostermann**[1,2,3]

[1]German Research Center for AI (DFKI)
[2] Saarland University
[3]Center for European Research in Trusted AI (CERTAIN)
tanja.baeumel@dfki.de

Abstract

While recent work has begun to uncover the internal strategies that Large Language Models (LLMs) employ for simple arithmetic tasks, a unified understanding of their underlying mechanisms is still lacking. We extend recent findings showing that LLMs represent numbers in a digit-wise manner and present evidence for the existence of digit-position-specific circuits that LLMs use to perform simple arithmetic tasks, i.e. modular subgroups of MLP neurons that operate independently on different digit positions (units, tens, hundreds). Notably, such circuits exist independently of model size and of tokenization strategy, i.e. both for models that encode longer numbers digit-by-digit and as one token. Using Feature Importance and Causal Interventions, we identify and validate the digit-position-specific circuits, revealing a compositional and interpretable structure underlying the solving of arithmetic problems in LLMs. Our interventions selectively alter the model's prediction at targeted digit positions, demonstrating the causal role of digit-position circuits in solving arithmetic tasks.

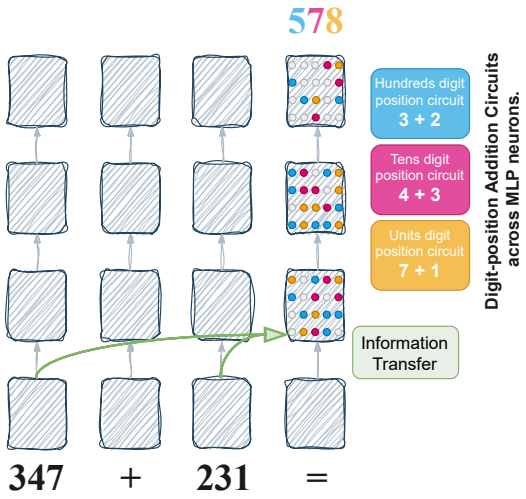


Figure 1: Main finding: Simple arithmetic tasks are solved modularly by digit-position-specific circuits distributed across multiple MLP layers. Distinct sets of MLP neurons are responsible for generating results digits in parallel and independently for different positions. The hundreds digit position circuit is illustrated in blue, the tens in red, and units in yellow.

1 Introduction

The emergence of mathematical abilities in large language models (LLMs) has sparked growing interest in uncovering the internal mechanisms that underlie their arithmetic reasoning capabilities (Stolfo et al., 2023; Nikankin et al., 2024; Zhang et al., 2024; Lindsey et al., 2025). Understanding whether LLMs solve arithmetic by applying generalizable strategies, relying on superficial heuristics, or merely memorizing training examples can provide crucial insights into the general nature of reasoning abilities in LLMs.

Despite recent progress, a unified account of how LLMs perform basic arithmetic has yet to be established. Nikankin et al. (2024) argue that LLMs rely on a *bag of heuristics*—a sparse set of neurons that are sensitive to simple operand or result patterns—rather than implementing coherent algorithms for arithmetic. Lindsey et al. (2025) describe a dual-pathway mechanism for Claude-3.5 Haiku, with separate pathways for estimating rough result magnitude and the result unit digit. At the same time, Levy and Geva (2024) and Gould et al. (2023) find that digit values – but not full numeric values – can be extracted from the residual stream via probing, suggesting that LLMs internally *represent* numbers in a digit-wise manner. This finding is suprising, as many LLMs represent numbers up to a certain value (e.g. 3 digits) as single tokens (e.g., "347" is one token in Llama 3 8B). Their work however does not investigate whether these numeric representations are also exploited for solving arithmetic tasks.

In this work, we take the next step in under-

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1380–1409

standing how LLMs solve arithmetic tasks by presenting causal evidence for digit-position-specific arithmetic circuits, i.e., modular subgroups of MLP neurons that independently generate results for different digit positions (Figure 1). For instance, to solve $347 + 231 =$, the model generates results for $7{+}1$ (units), $4{+}3$ (tens), and $3{+}2$ (hundreds) using three distinct circuits, formed by groups of MLP neurons that are selectively sensitive to individual digit positions and distributed across mid-to-late layers.

We identify the digit-position-specific circuits using Fisher Score-based feature selection (Gu et al., 2012; Sun et al., 2021) and validate their causal role via targeted interventions (Vig et al., 2020; Meng et al., 2023): Altering the activation of the units-digit circuit for instance selectively changes only the predicted units digit while leaving tens and hundreds unchanged (Figure 2). Our findings offer important novel evidence that LLMs solve arithmetic tasks not merely through heuristics, but through structured, compositional arithmetic processes.

Our main contributions are:

(1) We identify digit-position-specific arithmetic circuits by using supervised Fisher Score-based feature selection to detect MLP neuron groups responsible for solving arithmetic digit-level-subtasks.

(2) We validate these circuits via causal interventions, demonstrating that they selectively control individual output digits.

(3) We confirm the existence of digit-position-specific arithmetic circuits across a wide range of models, tasks, and tokenization schemes.

We release all code and data to support reproducibility[1].

2 Method

2.1 Overview of our Approach

We investigate whether LLMs perform arithmetic in a digit-wise, position-specific manner by a two-step approach, following recent work in mechanistic interpretability:

1. **Identify Digit-Specific Neurons through Feature Selection:** We identify neurons that are involved in generating digit position specific subresults. We use a Fisher Score-based feature selection method to measure neuron sensitivity to sub-tasks at different digit positions.

2. **Causal Verification via Interventions**: To verify the causal role of the identified neuron groups, we perform targeted interventions by replacing the activation of a digit-specific neuron group in a base prompt with that of a source prompt. This allows us to test whether the altered activations lead to predictable changes in the model's output. Our results demonstrate that these neuron groups indeed implement digit-position-specific circuits: Only the digit corresponding to the intervened circuit changes, while the rest of the output remains unaffected. As illustrated in Figure 2, intervening on the unit digit circuit of the base prompt `347 + 231` (expected output `578`) with activations from the source prompt `261 + 512` (expected output `773`) yields the output `573`. This output combines the hundreds and tens digits from the base result (`57`) with the unit digit from the source result (`3`), confirming the selective influence of the intervened circuit.

2.2 Data for Neuron Identification Experiments

We generate a simple addition dataset D_{add} and subtraction dataset D_{sub} by generating 1000 addition and 1000 subtraction prompts respectively. This data is used to identify digit-specific neurons (Step 1). The prompts are formulated in a one-shot setting of the form "`157 o 431 = 588;` A o B = ", where o $\in \{+, -\}$ and $A, B \in \{100, \ldots, 999\}$. All addition and subtraction tasks are sampled such that the result is also a 3-digit integer, i.e., $\in \{100, \ldots, 999\}$. To isolate digit-wise computations and prevent interactions between digit positions, we construct all data such that no carry occurs in any digit position. We present an investigation of carry bits in Section 4.2.

2.3 Data for Intervention Experiments

For the causal intervention experiments , we construct paired prompts in a one-shot setting consisting of a *base* (e.g., "`157 o 431 = 588;` A o B = ") and a *source* (e.g., "`157 o 431 = 588;` C o D = ") where operator o $\in \{+, -\}$ and operands $A, B, C, D \in \{100, \ldots, 999\}$),.

We construct two addition datasets $D_{add,op1}$ and $D_{add,op2}$ where operator o $= +$, such that one

[1] https://github.com/tbaeumel/transformer-digit-arithmetic

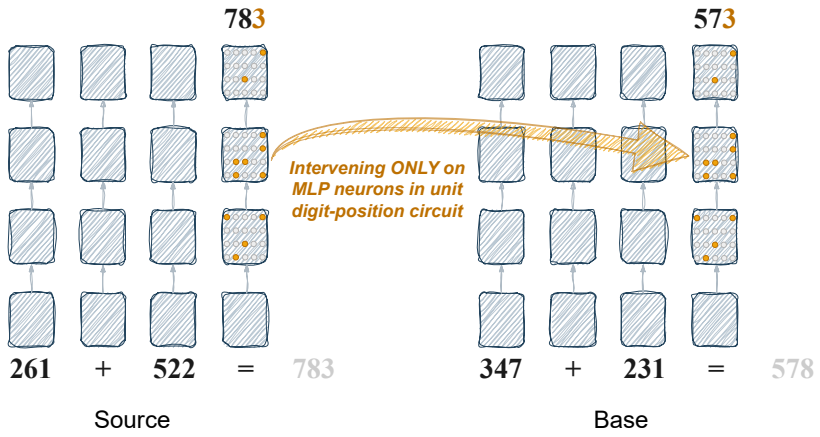


Figure 2: Intervening on only the MLP neurons that are members of one of the digit-position specific circuits, results in targeted changes only on the corresponding digit-position of the generated result. Here, an intervention on the unit position circuit only affects the unit position of the generated result (3 instead of 8). This is evidence that the identified digit-position circuits are specific and causally involved in arithmetic result generation.

operand is shared between base and source while the other varies across all digits ($D_{add,op1}$: $B = D$, $A_{d_i} \neq C_{d_i} \forall$ digit positions d_i; $D_{add,op2}$: $A = C$, $B_{d_i} \neq D_{d_i} \forall d_i$). In the same way we construct $D_{sub,op1}$ and $D_{sub,op1}$ where $o = -$.

All datasets contain 200 pairs of unique arithmetic problems. To isolate digit-wise computations and prevent interactions between digit positions, we construct all data such that no carry occurs in any digit position.

2.4 Models

We evaluate four decoder-only transformer models: LLaMA3-8B, LLaMA3-70B (Grattafiori et al., 2024), OLMo 2 7B (OLMo et al., 2024), and Gemma 2 9B (Team, 2024) all of which achieve high performance on simple addition and subtraction tasks (>95%, Table 1).

Numeric Tokenization	Model	Accuracy Addition	Accuracy Subtraction
Multi-digit	LLaMA 3 8B	100.00%	100.00%
	LLaMA 3 70B	100.00%	100.00%
	Olmo 2 7B	99.00%	99.50%
Single-digit	Gemma 2 9B	98.50%	99.50%

Table 1: Accuracy of models on D_{add} and D_{sub}.

We focus our main analyses on models that employ multi-digit numeric tokenization strategies, i.e., their vocabularies contain multi-digit tokens like "147", i.e., LLaMA3-8B, Olmo 2 7B and LLaMA3-70B[2]. Any observed digit-wise processing must be a property of the model's internal structure rather than a simple consequence of tokenization for such models, making it a more surprising and informative phenomenon. To demonstrate the generalizability of our findings, we also include results from a single-digit tokenization model (Gemma 2 9B) in Appendix B, which similarly exhibits digit-position-specific processing pathways.

To study the mechanisms that emerge from standard language model pretraining, we evaluate base models without any instruction tuning or arithmetic-specific fine-tuning.

3 Circuit Localization

We now describe the circuits localization steps in detail along with the results. We focus our analyses on LLaMA3-8B in the main text and report detailed analyses showing similar results for Olmo 2 7B and LLaMA3-70B in Appendices E and F.

3.1 Step 1: Identifying Digit-Specific Neurons

Intuition. To identify MLP neurons that are sensitive to digit-level arithmetic subtasks, we perform supervised feature selection. Intuitively, we identify hich neurons are most discriminative for different arithmetic subtasks in individual digit positions.

We employ Fisher Score (Duda and Stork, 2001; Venkatesh and Anuradha, 2019) as our feature selection method, which is an established information-theoretic method for assessing individual feature importance in high-dimensional spaces. Fisher Score measures how well a feature (in our case, a neuron's activation) separates data points

[2]All operands and correct results from all datasets appear as unique vocabulary tokens in all multi-digit tokenization models.

belonging to different classes (in our case, digit-level arithmetic tasks). A high Fisher score $F_{i,d}$ of neuron i at digit position d indicates that the activation of neuron i reliably discriminates between subtasks targeting that digit position. We interpret such neurons as likely members of digit-position-specific arithmetic subcircuits. For instance, if a neuron is similarly activated for the addition tasks "15**7** + 43**1** = ", "23**7** + 36**1** = ", and "76**7** + 21**1** = ", chances are that it is involved in the generation of the unit result digit, as all examples have the same subtask $7 + 1$ in the unit position.

We collect MLP activations across layers, for all prompts in D_{add} and D_{sub} and compute Fisher Scores for each neuron and digit position. We then rank neurons according to their Fisher Score, reflecting how strongly their activations vary with changes to specific digit positions in the input operands. By thresholding the scores (as a hyperparameter), we obtain candidate circuits composed of neurons sensitive to each digit position.

Following Stolfo et al. (2023), we focus on MLP layers that could plausibly contribute to producing arithmetic output. We identify the earliest layer at which operand information has been propagated to the residual stream at the final token, as earlier layers lack relevant input context and are unlikely to contribute to the result generation. We therefore restrict our search for arithmetic circuits to layers at which operand information has been propagated. Appendix C provides the identified "operand injection" layers for all models and tasks, alongside a more detailed rationale.

Neuron Selection for Digit-Position Circuits. Appendix D presents a detailed formalization of how Fisher Score $F_{i,d}$ is calculated for each neuron.

For each MLP layer l in model m and $\mathrm{o} \in add, sub$ in each digit position d:

1. Compute average $F_{i,d}$ for all neurons i in l over the dataset D_{o}.

2. Select the neurons with $F_{i,d}$ above a threshold t as candidates for the digit-specific arithmetic circuit $C_{m,\mathrm{o},d,t}$.

For all models, we explore circuits $C_{m,\mathrm{o},d,t}$ based on different thresholds $t \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Fisher Scores are a relative metric, therefore choosing a reasonable threshold for circuit membership is a hyperparameter tuning problem. Generally speaking, Fisher scores close to 0 indicate a lack of discriminative power, scores between 0.1 and 1.0 weak to moderate discriminability, and values above 1.0 strong discriminability.

Statistics on Selected Circuits. We provide detailed statistics for all identified circuits $C_{m,\mathrm{o},d,t}$ across all models, operators, and digit positions in Appendix E and provide a brief summary of the findings here.

We find that digit-position-specific circuits consist of a significant portion available MLP neurons, for instance for $C_{\text{Llama3 8B},+,d,t}$ the average number of MLP neurons per layer responsible for one of the digit-position specific circuits is 60.3% of all MLP neurons. Further, the sets of digit-position-specific MLP neurons are highly sufficient for representing digit-specific arithmetic subtasks, indicated by strong classificiation performance using only the digit circuit neurons. Importantly, we also find that neuron sets in different digit-position circuits are largely distinct (Figure 18), providing a first indication of digit-positional modularity in LLM arithmetic.

3.2 Step 2: Causal Verification via Interventions

We perform targeted *interchange interventions* (Vig et al., 2020; Meng et al., 2023) on candidate digit-position circuits $C_{m,\mathrm{o},d,t}$ replacing MLP activations of digit-position-specific neurons in a *base* prompt with those from a *source* prompt at selected layers.

Circuit Depth. Based on the observations in Section 3.1 and Appendix C, we choose the MLP layer set considered for digit-position specific arithmetic circuits to be $L_{Llama38B,add,op2} = \{15, ..., 24\}$ for $D_{add,op2}$ and $L_{Llama38B,add,op1} = \{16, ..., 24\}$ for $D_{add,op1}$. Ablation experiments (Appendix H) show that choosing deeper circuits $L_{Llama38B,add,op2} = \{15, ..., 28\}$ and $L_{Llama38B,add,op2} = \{16, ..., 28\}$ does not make a significant difference. This supports our conclusion that late MLP layers are less responsible for generating digit-position specific arithmetic results.

Experiment. Given a source (`123 + 562 = 685`) and base prompt (`123 + 456 = 579`), we intervene on the neurons in $C_{m,\mathrm{o},d,t}$. We then measure the model's output distribution over all 8 possible digit-wise combinations of the base and source results (e.g., `bbb`, `sbb`, `bsb`, etc., where e.g. `sbb` indicates the the hundreths digits correspond to the source

m	o	d	$t*$	bbb	**bbs**	**bsb**	**sbb**	bss	sbs	ssb	sss
				Absolute change in prediction probability Δp in percentage points (after - before).							
Llama 3 8B	+	unit	0.6	-78.89%	**+30.93%**	+0.87%	+0.87%	+10.59%	+2.57%	+0.49%	+2.76%
		tens	0.5	-57.29%	+0.47%	**+22.76%**	+1.51%	+1.34%	+0.17%	+4.57%	+0.37%
		hun.s	0.9	-77.01%	+0.06%	+0.50	**+45.56%**	+0.10%	+0.50%	+5.04%	+0.47%
	−	unit	0.6	-75.62%	**+28.50%**	+0.59%	+1.16%	+12.40%	+2.90%	+0.25%	+3.41%
		tens	0.5	-55.97%	+0.90%	**+14.40%**	+2.07%	+2.04%	+0.22%	+6.06%	+1.27%
		hun.s	0.9	-62.20%	+0.14%	+0.58%	**+36.27%**	+0.12%	+0.28%	+1.88%	+0.16%
Llama 3 70B	+	unit	0.5	-36.93%	**+23.16%**	+0.72%	+0.82%	+3.67%	+0.54%	+0.15%	+0.53%
		tens	0.5	-28.67%	+0.37%	**+17.71%**	+0.57%	+0.46%	+0.08%	+1.73%	+0.15%
		hun.s	0.6	-41.40%	+0.17%	+0.51%	**+29.68%**	+0.12%	+0.23%	+1.14%	+0.18%
	−	unit	0.5	-34.17%	**+20.76%**	+0.44%	+0.32%	+7.43%	+0.23%	+0.13%	+1.08%
		tens	0.5	-14.84%	+0.07%	**+11.32%**	+0.19%	+0.08%	+0.03%	+1.02%	+0.06%
		hun.s	0.6	-18.62%	+0.05%	+0.19%	**+15.91%**	+0.03%	+0.06%	+0.31%	+0.04%
Olmo 2 7B	+	unit	0.4	-86.16%	**+27.81%**	+0.33%	+11.27%	+1.51%	+5.89%	+0.95%	+3.75%
		tens	0.8	-83.21%	+0.11%	**+42.57%**	+3.50%	-0.02%	+0.02%	+7.20%	+0.34%
		hun.s	0.8	-77.76%	+0.47%	+0.16%	**+37.37%**	+0.35%	+2.20%	+2.77%	+0.33%
	−	unit	0.5	-89.44%	**+50.26%**	+0.10%	+5.93%	+0.83%	+5.37%	+0.20%	+1.77%
		tens	0.9	-84.64%	-0.00%	**+44.70%**	+1.23%	+0.28%	+0.07%	+5.29%	+0.13%
		hun.s	0.9	-86.63%	+1.07%	-0.40%	**+45.28%**	+0.19%	+3.29%	+3.49%	+0.37%

Table 2: Main Results: for all detected circuits (across models, operators, digit positions) we report the change in prediction probabilities in percentage points (effect size) for result variants after interventions on digit-position-specific arithmetic circuits (with optimal $t*$ as t in each circuit), on datasets $D_{add,op2}$ and $D_{sub,op2}$, with the increase in prediction probability for the targeted result variant shown in bold. Highly similar results for datasets $D_{add,op1}$ and $D_{sub,op1}$ are provided in Table 6 below.

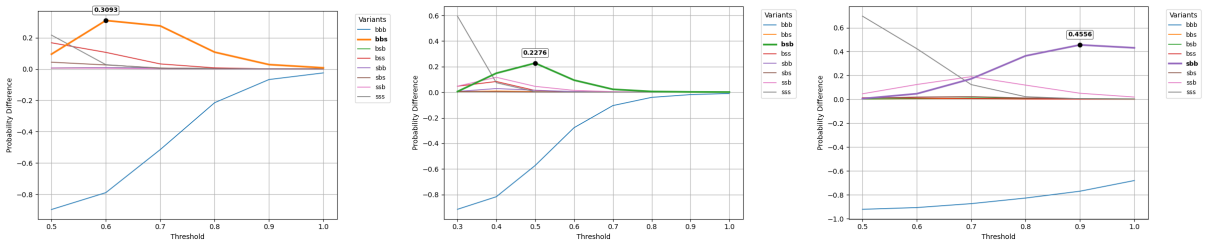
result and the other two digits to the base result). If our independent digit-wise circuit hypothesis is correct, a position-specific circuit should shift probability mass only toward the expected variant (e.g., sbb for a hundreds-digit intervention), with minimal effects on other positions.

We assess position-specificity of the intervention by checking whether the probability of the expected result variant increases selectively (e.g. for an intervention on hundreds digits sbb). If other variants like ssb or sbs were to also increase, this would indicate that the intervention affects multiple digit positions. The specificity of the probability shift provides key evidence for digit-position-specific circuits in the MLP layers.

Results. Table 2 contains results for the intervention on the best candidate digit-position circuits $C_{m,\text{o},d,t*}$, where $t*$ is the best value for threshold t. For all 4 datasets $D_{\text{o},op}$ ($\text{o} \in \{add, sub\}; \text{op} \in \{op1, op2\}$) and all models, we observe a consistent and substantial increase of the probability of the expected result variant – bbs for unit, bsb for tens, and sbb for hundreds – demonstrating that the interventions on the hypothesized digit-position circuits indeed only affect the desired digit position. This is evidence for the digit-position-specific nature of the identified circuits, and thus the **causal role of digit-position specific processing in solving arithmetic tasks**. For instance, an intervention on $C_{\text{Llama 3 8B,add,hundreds,0.9}}$ on $D_{add,op2}$ yields a 45.56% probability point increase on the sbb result variant (correct hundreds digit from the source, with tens and unit digits from the base), from 0.23% before intervention to 45.79% after intervention. Specificity of the digit-position circuits is further supported by the low probability changes in result

m	o	d	$t*$	Flip Rate
Llama 3 8B	+	unit	0.6	51%
		tens	0.5	33%
		hun.s	0.9	68.5%
	−	unit	0.6	49%
		tens	0.5	19.5%
		hun.s	0.9	51%
Llama 3 70B	+	unit	0.5	24%
		tens	0.5	15%
		hun.s	0.6	33.5%
	−	unit	0.5	20.5%
		tens	0.5	7.5%
		hun.s	0.6	10.5%
Olmo 2 7B	+	unit	0.4	44%
		tens	0.8	54%
		hun.s	0.8	51.5%
	−	unit	0.5	64%
		tens	0.9	53%
		hun.s	0.9	54.5%

Table 3: Flip rate from bbb result to the intended digit-specific result variant (Unit: bbs, Tens: bsb, Hundreds: sbb) at the best threshold $t*$ for each circuit, on datasets $D_{add,op2}$ and $D_{sub,op2}$. Highly similar results for datasets $D_{add,op1}$ and $D_{sub,op1}$ in Table 7.



(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 3: Effect size of digit-circuit interventions on $D_{add,op2}$ with different thresholds for neuron circuit membership.

variants that contain multiple source digits (bss, sbs, ssb, and sss).

Additionally, Table 2 shows an expected reduction in the probability of the base result (bbb), indicating effective suppression of the default behavior. Importantly, non-target digits remain largely unaffected, as indicated by minimal changes observed in other result variants, confirming the **specificity** of the identified digit-position circuits.

We show detailed results on $C_{\text{Llama 3 8B},\text{add},d,t}$ in Figure 3 to show the effect of the chosen threshold t. Additionally, Table 3 shows that the intervention is often successful in flipping the model's prediction from the bbb result to the intended digit-specific result variant.

The results support our hypothesis: Intervening on the hypothesized circuits for the unit, tens, and hundreds digits leads to targeted and isolated changes in the corresponding digit of the model's output. This is strong evidence that distinct circuits underlie the generation of arithmetic results in individual digits, and that the identified digit-position addition circuits are causally involved in generating digit-position specific sub-results.

4 Supplementary Experiments

To further strengthen our findings we present results from a series of supplementary experiments conducted on Llama 3 8B.

4.1 Addition and Subtraction Circuits are Largely Distinct

We are interested in whether subtraction and addition circuits are similar in their functionality or rather very distinct. To assess the similarity between addition and subtraction circuits, we investigate the overlap of selected MLP neurons for each digit position in each layer. Table 8 shows a low average overlap (Overlap in the top 100 MLP neurons: unit 19%, tens 9.2%, hundreds 19.8%) in selected neurons, indicating that addition and subtraction circuits rely on mostly distinct subsets of neurons.

4.2 Additions with Carry Bits Employ the Same Mechanisms

All previous experiments explicitly avoided carry operations. However, carries are a key reason why fully modular digit-wise processing does not always suffice in arithmetic: the result at one digit position can influence another. For example, $9 + 5$ in the units position yields a carry of 1 to the tens position.

To examine how our digit-wise arithmetic circuits interact with carry propagation, we test whether targeted interventions remain effective when they introduce a carry. Specifically, we ask: *Does a digit-specific circuit encode carry information, or is it handled elsewhere in the network?*

We consider two scenarios involving carry-induced interventions (200 samples each):

(1) Carry from Units to Tens. We intervene on the units-digit circuit of a base prompt with no carry, such as $347 + 231 = 578$, using activations from a source prompt that induces a carry from the unit to the tens position, such as $347 + 415 = 762$. After intervention on the units-digit circuit, we evaluate whether the output reflects (a) the base tens and hundreds digits with the source units digit (i.e., $572 = bbs$), or (b) if in addition to this a carry is propagated into the tens digit (i.e., $582 = bb_{+1}s$).

(2) Carry from Tens to Hundreds. We intervene on the tens-digit circuit of a no-carry base prompt ($347+231 = 578$) with a source prompt that causes a carry from tens to hundreds ($347 + 482 = 829$). We compare whether the model outputs (a) the base hundreds and units digits with the source tens digit

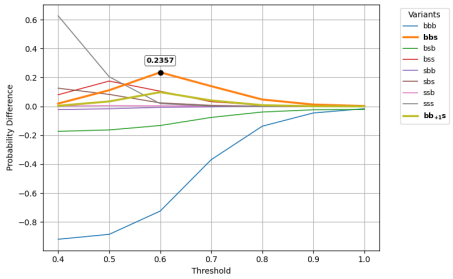


Figure 4: Effect of intervening on the unit circuit with a carry originating from unit digit position.

(i.e., $528 = bsb$), or (b) if the hundreds digit is carry-adjusted (i.e., $628 = b_{+1}sb$).

These interventions help determine whether carry information is localized within digit-specific circuits or handled separately by the model.

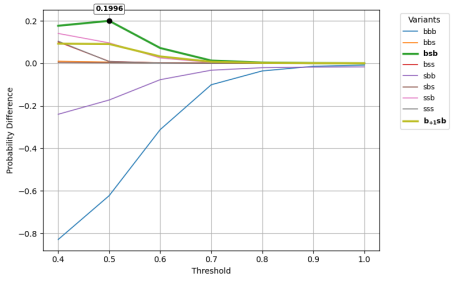


Figure 5: Effect of intervening on the tens circuit with a carry originating from tens digit position.

Results. Figures 4 and 5 show that the bbs variant and the bsb variant, for the unit and tens carry interventions, respectively, exhibit the strongest increase in probability. At the same time, the effect on the result variants $bb_{+1}s$ and $b_{+1}sb$ is significantly smaller. This provides further evidence that digit-position-specific arithmetic circuits operate largely independently of one another and suggests that carry information is likely determined and processed by separate mechanisms, rather than being embedded in the digit-position circuits.

4.3 MLP Similarity Reflects Task Overlap

We investigate how similar digit-specific MLP activations are for prompts sharing the same digit-level arithmetic task at a given position (e.g., $154 + 635$ and $137 + 611$ share the same hundreds digit addition: $1 + 6$). Specifically, we compute pairwise cosine similarity between activations of MLP neurons in the digit-position circuit d, for samples with the same subcomputation at digit position d. As a baseline, we compute cosine similarity in these neurons across 5000 random sample pairs.

Layer	Unit ($t = 0.6$)		Tens ($t = 0.5$)		Hundreds ($t = 0.9$)	
	Sim	Random (mean ± sd)	Sim	Random (mean ± sd)	Sim	Random (mean ± sd)
15	**0.84**	0.72 ± 0.08	**0.83**	0.73 ± 0.08	**0.87**	0.70 ± 0.11
16	**0.81**	0.68 ± 0.07	**0.78**	0.66 ± 0.07	**0.82**	0.62 ± 0.11
17	**0.76**	0.59 ± 0.08	**0.73**	0.59 ± 0.09	**0.76**	0.51 ± 0.14
18	**0.73**	0.54 ± 0.11	**0.72**	0.58 ± 0.11	**0.77**	0.52 ± 0.15
19	**0.77**	0.60 ± 0.10	**0.75**	0.61 ± 0.10	**0.79**	0.54 ± 0.16
20	**0.78**	0.64 ± 0.09	**0.77**	0.63 ± 0.10	**0.79**	0.54 ± 0.15
21	**0.72**	0.52 ± 0.12	**0.70**	0.50 ± 0.15	**0.74**	0.46 ± 0.18
22	**0.68**	0.49 ± 0.14	**0.68**	0.46 ± 0.16	**0.73**	0.41 ± 0.21
23	**0.68**	0.48 ± 0.12	**0.67**	0.47 ± 0.13	**0.72**	0.43 ± 0.20
24	**0.58**	0.34 ± 0.15	**0.53**	0.34 ± 0.15	**0.52**	0.31 ± 0.27

Table 4: Pairwise cosine similarity of digit-position-specific MLP sub-updates on the same digit subtask. Includes a random-pair baseline for comparison.

As expected, Table 4 shows consistently high within-label similarity across digit positions. For example, in layer 15, pairs sharing the same unit-digit computation have an average similarity of 0.84 in unit-digit MLP neurons. This provides additional evidence for digit position-specific processing in mid-late MLP neurons.

5 Structured Arithmetic Processing or a Bag of Heuristics?

We are interested in how digit-position specific arithmetic processing fits in with previous findings that mathematical processing is solved by a sparse set of "heuristic" neurons that respond to specific operand or result patterns (Nikankin et al., 2024). These heuristics include, for example, results in a specific numerical range (e.g., results between 200 and 290), or results that share arithmetic properties (e.g., results congruent to $4 \mod 5$).

We hypothesize that such heuristics may in part be fragments of digit-wise circuits, e.g., a neuron responding to results $\equiv 4 \mod 5$ may reflect unit-digit sensitivity, while neurons responding to result ranges (e.g., 200–290) may reflect sensitivity to higher digit positions.

To qualitatively test this idea, we perform a small-scale exploratory analysis. In the spirit of Nikankin et al. (2024) we examine digit-specific activation patterns (instead of analyzing neurons across full operand values) to look for heuristic patterns in individual MLP neurons. We analyze the 20 highest Fisher Score MLP neurons per digit circuit (units, tens, hundreds) per layer across layers 15–24 in LLaMA 3 8B. We then generate digit-wise heatmaps for each neuron, plotting the mean activation as a function of the digit values in operand 1 and operand 2, for each digit position.

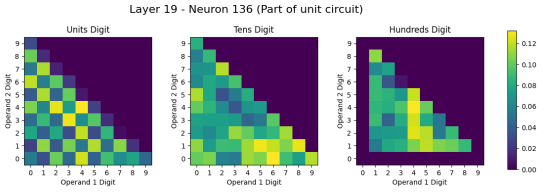


Figure 6: Neuron $N_{19,136}$ (in unit circuit) implements parity heuristic, i.e., result is $0 \mod 2$.

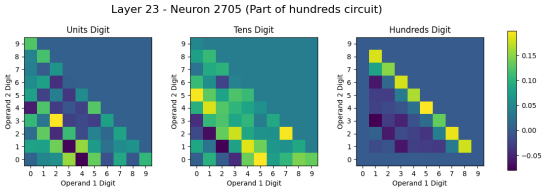


Figure 7: Neuron $N_{23,2705}$ (in hundreds circuit) implements a result range heuristic (result in range 900 - 999), i.e., hundred digit in result is 9.

Our small qualitative study shows that many high-Fisher-score neurons exhibit highly structured and interpretable patterns, which mirror the kinds of heuristics described by Nikankin et al. (2024): Certain neurons (Figures 37 and 36) respond to specific operand digit values, while others are sensitive to certain result ranges (Figure 7), result digits (Figure 38), or result parity (Figure 6)[3].

Importantly, the type of heuristic implemented by a neuron is typically aligned with the digit circuit that neuron belongs to, such that a range 900-999 result heuristic is implemented by a neuron within the hundreds-digit circuit, and the $0 \mod 2$ result heuristic neuron is located within the unit-digit circuit.

While these are preliminary results, they suggest that heuristic neurons may reflect digit-specific processing embedded within broader modular arithmetic circuits. In this view, heuristic neurons may be a manifestation of modular arithmetic circuits, rather than an alternative to them: they may be single-neuron approximations of low-level arithmetic subroutines tied to individual digit positions. While our analysis is qualitative and limited in scope, it highlights that compositional mechanisms and heuristic rules need not be mutually exclusive.

[3]All plots in https://github.com/tbaeumel/transformer-digit-arithmetic/tree/main/Digit-Heuristics_Top_Digit-Circuit_Neurons

6 Related Work

Mechanistic Interpretability. Mechanistic interpretability (MI) aims to reverse-engineer the internal mechanisms of language models by analyzing weights and components. In transformer LLMs, a *circuit* refers to a small set of interconnected components (e.g., MLPs or attention heads) that collectively perform a specific computation (Olah et al., 2020; Elhage et al., 2021). Causal mediation techniques such as activation patching (Vig et al., 2020; Meng et al., 2023) enable demonstrating the causal role of these circuits during generation. In this work, we identify digit-position-specific arithmetic circuits and verify their causal contribution through targeted interventions.

Arithmetic Reasoning Interpretability. Recent studies have begun to uncover how LLMs tackle arithmetic tasks (Stolfo et al., 2023; Nikankin et al., 2024; Zhang et al., 2024; Lindsey et al., 2025; Baeumel et al., 2025; Quirke and Barez, 2023; Zhou et al., 2024b,a). For example, Lindsey et al. (2025) describe a dual-pathway mechanism in a model, where one pathway estimates the rough magnitude of results and the other generates the precise unit digit. Nikankin et al. (2024) argue that LLMs rely on a "bag of heuristics" rather than a single coherent algorithm, with individual neurons implementing simple heuristics like "result % 5 = 0." Stolfo et al. (2023) and others have used circuit analysis to reveal internal processing during arithmetic, while Deng et al. (2024) suggest LLMs mainly perform symbolic pattern recognition, not true numerical computation. Kantamneni and Tegmark (2025) recently proposed that LLMs represent numbers as generalized helices and perform addition using a "Clock" algorithm (Nanda et al., 2023). Despite this progress, a unified understanding of how LLMs perform basic arithmetic remains open.

Number Representation. Understanding how LLMs represent numbers internally has attracted considerable attention. Levy and Geva (2024) show that probes on LLM hidden states fail to recover exact numeric values directly, but succeed in recovering each digit in base 10, indicating digt-wise representation in of numbers in LLMs. This extends findings by Gould et al. (2023), who showed LLMs encode numeric values modulo 10. Related work (Zhu et al., 2025; Marjieh et al., 2025) suggests number representations can blend string-like and numerical forms or be encoded linearly. Our

work is inspired by these representation-focused studies and shows that numbers are not only represented digit digit but also processed this way in simple arithmetic tasks.

7 Conclusion

We identify and validate digit-position-specific arithmetic circuits in LLMs. Through targeted interventions, we demonstrate that these circuits are not only highly specific and modular but also causally involved in generating individual digit outputs.

Our results suggest LLMs perform simple arithmetic digit by digit, with distinct circuits operating independently at the units, tens, and hundreds positions. This modular arithmetic structure is compatible with prior findings that LLMs use heuristic pathways and distributed neuron groups for arithmetic (Nikankin et al., 2024), however our results show LLMs solve arithmetic tasks in a far more organized way than previously thought.

Limitations

While our results provide strong evidence for digit-position-specific arithmetic circuits in LLMs, our analysis is limited to addition and subtraction. More complex operations, such as multiplication and division, are not addressed in this work. Extending our framework to these tasks is an important direction for future work. We also leave the analysis of circuits responsible for composing digit-level results into final outputs to future work.

Our analysis focuses on MLP layers, which contain the neurons that directly control digit outputs. This captures the core computation but does not account for the role of attention heads or other residual stream components.

Finally, we use Fisher Score to identify neurons involved in arithmetic, as it is simple and effective across model scales. However, other methods—such as gradient-based attribution—may yield more precise circuits. Despite this, our identified circuits are causal, interpretable, and robust, demonstrating a strong proof of concept for neuron-level circuit discovery in language models.

Acknowledgements

We thank Cennet Oguz for her helpful feedback on the paper draft. This work has been supported by the German Federal Ministry of Research, Technology and Space (BMFTR) as part of the project TRAILS (01IW24005), and by the European Fund for Regional Development (EFRE) as part of the project toCertain (EFRE-AuF-0000942).

References

Tanja Baeumel, Josef van Genabith, and Simon Ostermann. 2025. The lookahead limitation: Why multi-operand addition is hard for llms. *ArXiv*, abs/2502.19981.

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

P. E. H. R. O. Duda and D. G. Stork. 2001. Pattern classification. *Wiley-Interscience Publication*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Quanquan Gu, Zhenhui Li, and Jiawei Han. 2012. Generalized fisher score for feature selection. *Preprint*, arXiv:1202.3725.

Subhash Kantamneni and Max Tegmark. 2025. Language models use trigonometry to do addition. *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*.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, et al. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

Raja Marjieh, Veniamin Veselovsky, Thomas L Griffiths, and Ilia Sucholutsky. 2025. What is a number, that a large language model may know it? *arXiv preprint arXiv:2502.01540*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and Editing Factual Associations in GPT. *arXiv preprint*. ArXiv:2202.05262 [cs].

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint*. ArXiv:2301.05217 [cs].

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

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2024. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

Philip Quirke and Fazl Barez. 2023. Understanding addition in transformers. *arXiv preprint arXiv:2310.13121*.

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 [cs].

Lin Sun, Tianxiang Wang, Weiping Ding, Jiucheng Xu, and Yaojin Lin. 2021. Feature selection using fisher score and multilabel neighborhood rough sets for multilabel classification. *Information Sciences*, 578:887–912.

Gemma Team. 2024. Gemma.

B Venkatesh and J Anuradha. 2019. A review of feature selection and its methods. *Cybern. Inf. Technol*, 19(1):3–26.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Zhengxuan Wu, Atticus Geiger, Aryaman Arora, Jing Huang, Zheng Wang, Noah Goodman, Christopher Manning, and Christopher Potts. 2024. pyvene: A library for understanding and improving PyTorch models via interventions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 158–165, Mexico City, Mexico. Association for Computational Linguistics.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. 2024. Interpreting and improving large language models in arithmetic calculation. *Preprint*, arXiv:2409.01659.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. 2024a. Pre-trained large language models use fourier features to compute addition. *Advances in Neural Information Processing Systems*, 37:25120–25151.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024b. Transformers can achieve length generalization but not robustly. *Preprint*, arXiv:2402.09371.

Fangwei Zhu, Damai Dai, and Zhifang Sui. 2025. Language models encode the value of numbers linearly. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 693–709, Abu Dhabi, UAE. Association for Computational Linguistics.

A Implementation Details

For our intervention experiments, we use the pyvene library (Wu et al., 2024) to perform interchange interventions, where we intervene at a specific layer on a specific module at the last token position of a base prompt, with the corresponding activation from a source prompt (*Interchange Intervention*).

B Results on Single Digit Tokenization Models

We present results from Gemma 2 9B, a single-digit tokenization model, to investigate whether it also exhibits digit-position-specific processing pathways.

Intuition: Single- vs Multi-digit Numeric Tokenization. Before presenting results on Gemma 2 9B, we clarify why arithmetic processing may differ in models that tokenize numbers into single digits (e.g., “3”, “4”, “7”) versus a single multi-digit token (e.g., “347”).

The key distinction lies in the model’s output, not its input. A multi-digit tokenization model, such as those in the Llama 3 family, tokenizes an input like “347 + 231 = ” as [347, +, 231, =], and it is trained to generate a full multi-digit output token, such as “578”. In contrast, a single-digit tokenization model, such as from the Gemma 2 family, tokenizes the same input as [3, 4, 7, +, 2, 3, 1, =] and can only generate single-digit output tokens. It will therefore output “5”, not “578”.

As a result, the two models learn to solve fundamentally different arithmetic tasks. Multi-digit models must solve the entire arithmetic expression in a single forward pass, as they need the full answer immediately. Single-digit models, on the other hand, only need to generate the next digit, deferring the rest of the arithmetic task to future forward passes.

Therefore, when we expect single-digit tokenization models to contain different processing paths

compared to multi-digit tokenization models, it is because these models have to learn completely different tasks during pretraining to become sufficient at solving arithmetics.

Localizing Digit-Specific Neurons. After the injection of operand information (Layer 28; Figures 17 and 16 in Appendix C) we observe the following pattern for both addition and subtraction (Figures 8 and 9):

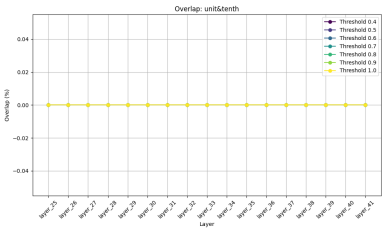
The hundreds circuit is large in size (spanning 70-90% of all MLP neurons per layer on average) and present all the way to the final layer. The tens circuit is available and large in size in layers 28 to 31, but then drastically diminishes in size. A unit circuit is non-existent, as evident by the absence of any neurons in the circuit, meaning that no neuron is sensitive to the subtask in the unit position.

Based on the intuition given above and previous work that finds LLMs internally generate one result digit more than needed for their current generation to account for carry bits, but no further digits (Baeumel et al., 2025), we conclude the following: Single-digit tokenization models also have digit-position-specific arithmetic circuits, which consist of one dominant circuit responsible for predicting the digit needed for the output generation, and one smaller and shorter circuit which may be responsible for determining whether a carry-bit influences the generation.

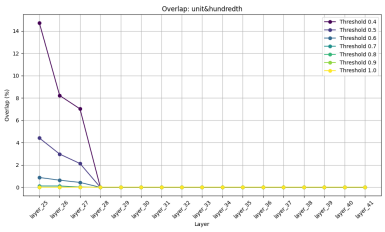
Intervention. Although somewhat trivial, we intervene on the hundreds digit circuit to observe the probability of the hundreds base digit `b` and hundreds source digit `s`. Table 5 shows that the intervention is highly effective in changing the model's prediction, which is expected given that the hundreds digit circuit spans the majority of MLP neurons, making the intervention highly invasive.

o	$t*$	b	s
		Δp (after - before).	
+	0.6	-92.61%	+91.55%
	0.7	-92.28%	+91.26%
	0.8	-91.93%	+90.93%
	0.9	-91.54%	+90.55%
	1.0	-91.25%	+90.34%
−	0.6	-92.6%	+91.84%
	0.7	-92.32%	+91.54%
	0.8	-91.76%	+90.95%
	0.9	-90.52%	+89.63%
	1.0	-88.05%	+86.92%

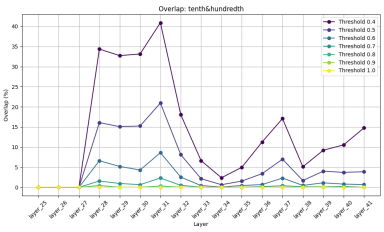
Table 5: Intervention on the hundreds digit circuit in Gemma 2 9B on $D_{add,op2}$ and $D_{sub,op2}$. We report the change in prediction probabilities in percentage points (effect size) for result variants after intervention.



(a) Unit and Tens

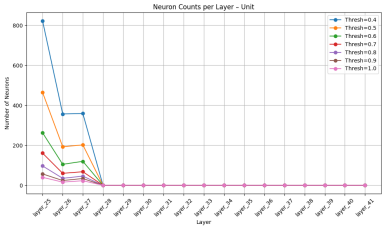


(b) Unit and Hundreds

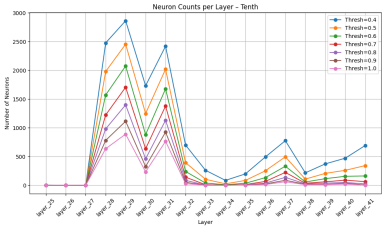


(c) Tens and Hundreds

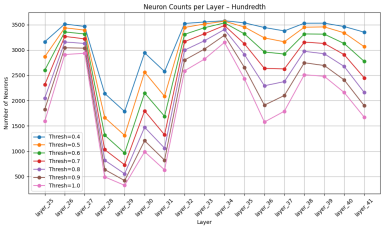
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit



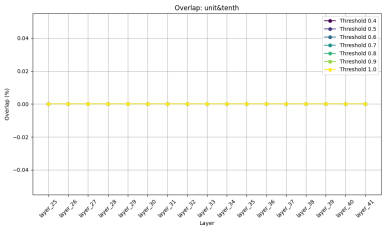
(f) Tens



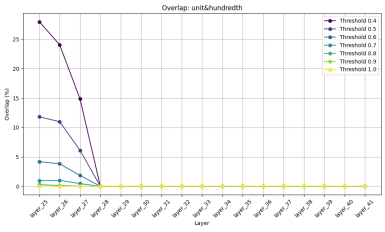
(g) Hundreds

(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 3584)

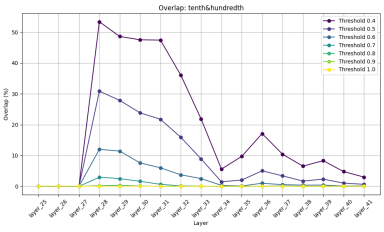
Figure 8: Gemma 2 9B, D_{add}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

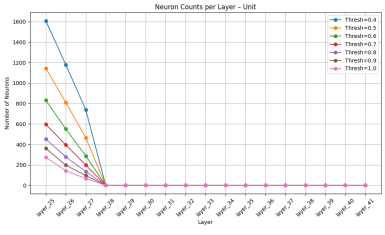


(b) Unit and Hundreds

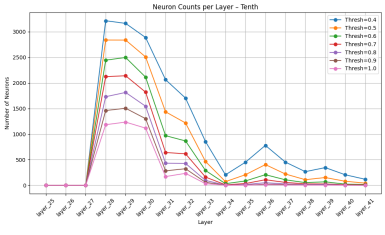


(c) Tens and Hundreds

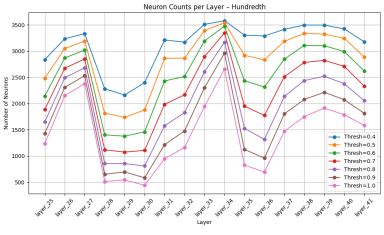
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit



(f) Tens



(g) Hundreds

(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 3584).

Figure 9: Gemma 2 9B, D_{sub}: Circuit statistics across digit positions and thresholds.

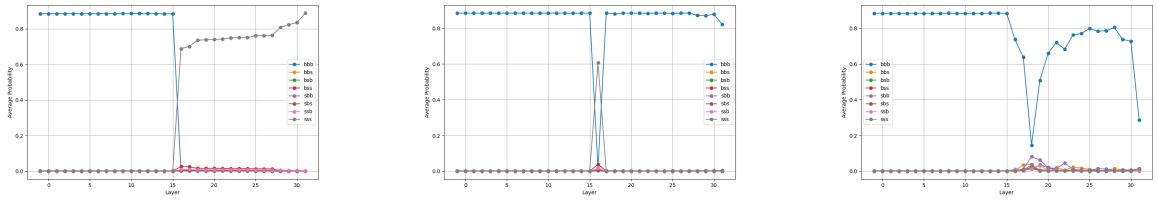
C Layers Involved in Arithmetic

Following Stolfo et al. (2023), we focus our search for digit-position-specific arithmetic circuits on MLP layers that could plausibly contribute to producing arithmetic output. We identify the earliest layer at which operand information has been propagated to the residual stream at the final token, as earlier layers lack relevant input context and are unlikely to contribute to the result generation. We therefore restrict our search for arithmetic circuits to layers at which operand information has been propagated.

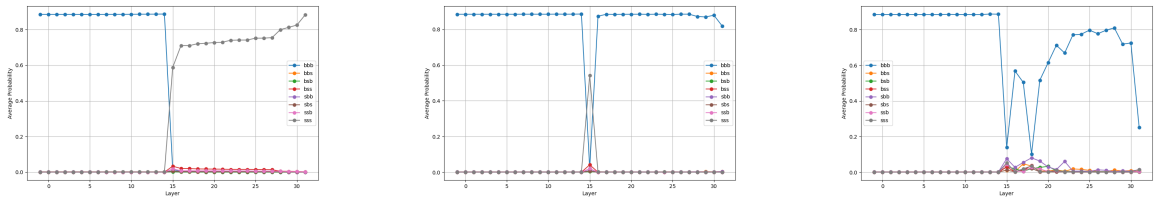
We perform causal interventions on the output of individual modules - particularly each decoder, attention, and MLP block. We intervene during the forward pass on a *base* prompt (e.g., `'347 + 231 = '`) by replacing the activations of a specific module with the activations of the same module during the forward pass on a *source* prompt (e.g., `'261 + 512 = '`). We measure the change in output probabilities for the correct result of the base query (`bbb`, here, '578') and the result of the source query (`sss`, here, '773').

Figures 10 to 17, reveal that intervention on specific attention modules dramatically increases the probability of `sss` in all models. This suggests that these attention modules are responsible for injecting operand information into the residual stream at the final token position, which is consistent with prior work (Stolfo et al., 2023).

We thus look for digit-wise addition neurons in the MLP layers following the operand injection into the residual stream. For Llama 3 8B for instance, this happens at layer 16 for $D_{add,op1}$ and layer 15 for $D_{add,op2}$ respectively (Figure 10).

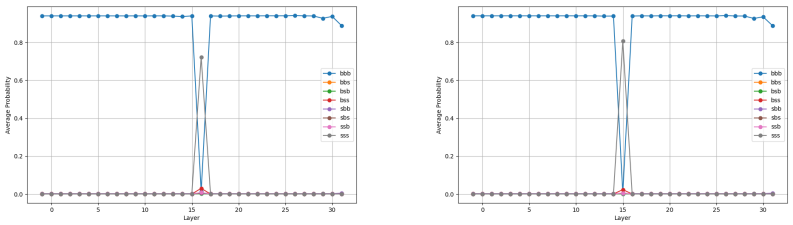


(a) Block Output - $D_{add,op1}$ (b) Attention Output - $D_{add,op1}$ (c) MLP Output - $D_{add,op1}$



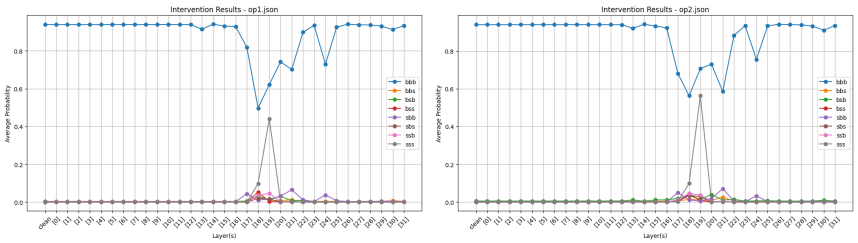
(d) Block Output - $D_{add,op2}$ (e) Attention Output - $D_{add,op2}$ (f) MLP Output - $D_{add,op2}$

Figure 10: **LLaMA3-8B**: Probability of result variants after intervention at individual modules of individual layers on Addition Datasets. We see the Operand Injection into the residual stream in Layer 16 for $D_{add,op1}$, and in Layer 15 for $D_{add,op2}$.



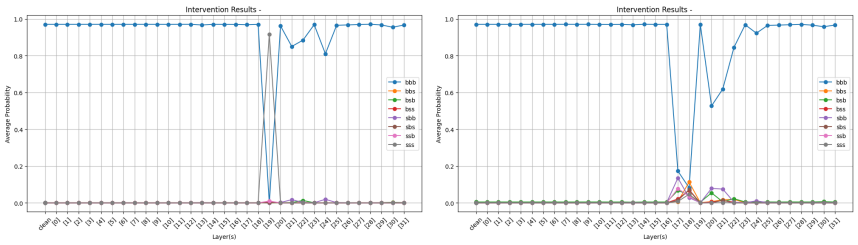
(a) $D_{sub,op1}$: Operand Injection into the residual stream in Layer 16 (b) $D_{sub,op2}$: Operand Injection into the residual stream in Layer 15

Figure 11: **LLaMA3-8B**: Probability of result variants after intervention at attention modules of individual layers on Subtraction Datasets.



(a) $D_{add,op1}$: Operand Injection into the residual stream starting in Layer 17 (b) $D_{add,op2}$: Operand Injection into the residual stream starting in Layer 17

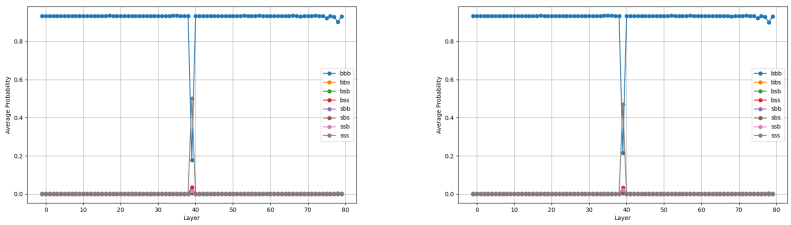
Figure 12: **Olmo 2 7B**: Probability of result variants after intervention at attention modules of individual layers on Addition Datasets.



(a) $D_{sub,op1}$: Operand Injection into the residual stream in Layer 19

(b) $D_{sub,op2}$: Operand Injection into the residual stream starting in Layer 17

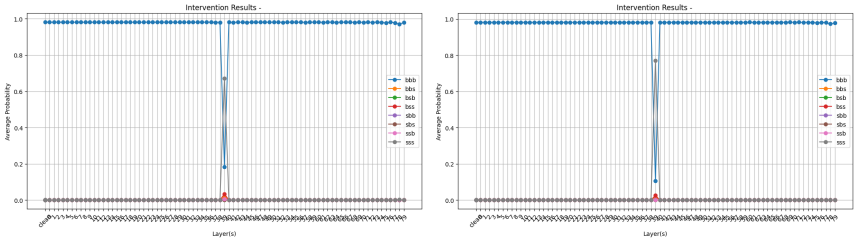
Figure 13: **Olmo 2 7B**: Probability of result variants after intervention at attention modules of individual layers on Subtraction Datasets.



(a) $D_{add,op1}$: Operand Injection into the residual stream in Layer 39

(b) $D_{add,op2}$: Operand Injection into the residual stream in Layer 39

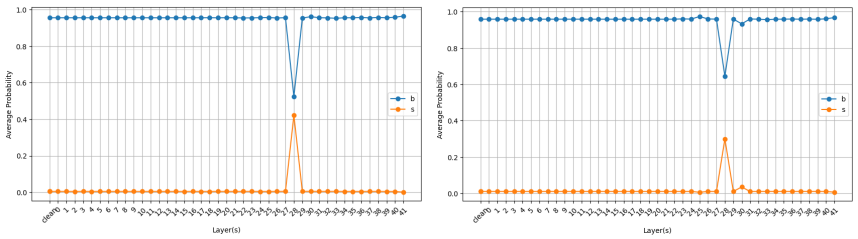
Figure 14: **LLaMA3-70B**: Probability of result variants after intervention at attention modules of individual layers on Addition Datasets.



(a) $D_{sub,op1}$: Operand Injection into the residual stream in Layer 39

(b) $D_{sub,op2}$: Operand Injection into the residual stream in Layer 39

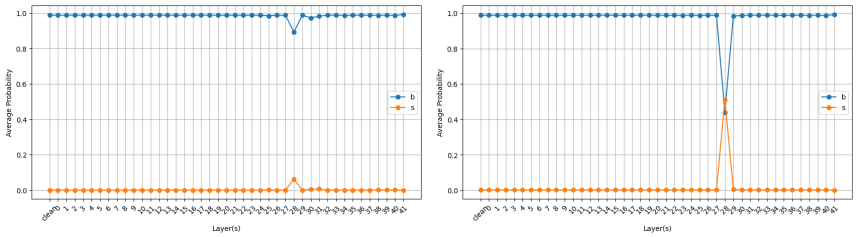
Figure 15: **LLaMA3-70B**: Probability of result variants after intervention at attention modules of individual layers on Subtraction Datasets.



(a) $D_{add,op1}$: Operand Injection into the residual stream in Layer 28

(b) $D_{add,op2}$: Operand Injection into the residual stream in Layer 28

Figure 16: **Gemma 2 9B**: Probability of result variants after intervention at attention modules of individual layers on Addition Datasets.



(a) $D_{sub,op1}$: Minimal Effect of Operand Injection into the residual stream in Layer 28

(b) $D_{sub,op2}$: Operand Injection into the residual stream in Layer 28

Figure 17: **Gemma 2 9B**: Probability of result variants after intervention at attention modules of individual layers on Subtraction Datasets.

D Fisher Score for Digit-Sensitivity of Neurons.

We measure how sensitive individual MLP neurons are to digit-level structure in arithmetic prompts, using a Fisher Score that quantifies class discriminability.

Let:

- $x \in \mathcal{X}$: an input prompt (e.g., `"157 + 431 = "`),
- i: index of a neuron in an MLP layer,
- $d \in \{\text{hundred}, \text{ten}, \text{unit}\}$: a fixed digit position in the operands (e.g., the "tens" digit),
- $c \in \{00, 01, \ldots, 99\}$: class label for the digit pair at position d, formed by concatenating[4] the digit from each operand (e.g., $c = 71$ for $d = unit$ and input prompt $34\mathbf{7} + 23\mathbf{1} =$),
- $\mathcal{X}_{c,d} \subseteq \mathcal{X}$: input prompts with digit pair c at position d (e.g., $\mathcal{X}_{71,unit} = \{34\mathbf{7} + 23\mathbf{1} =, 21\mathbf{7} + 65\mathbf{1} =, ...\}$),
- $a_i(x) \in \mathbb{R}$: the activation of neuron i when processing input x.

Define:

- **Mean activation for class c at position d:**

$$\mu_{i,c,d} = \frac{1}{|\mathcal{X}_{c,d}|} \sum_{x \in \mathcal{X}_{c,d}} a_i(x)$$

- **Variance of activations for class c at position d:**

$$\sigma^2_{i,c,d} = \frac{1}{|\mathcal{X}_{c,d}|} \sum_{x \in \mathcal{X}_{c,d}} (a_i(x) - \mu_{i,c,d})^2$$

- **Overall mean activation across all classes at position d:**

$$\mu_{i,d} = \frac{1}{\sum_c |\mathcal{X}_{c,d}|} \sum_c |\mathcal{X}_{c,d}| \mu_{i,c,d}$$

Then, the **Fisher Score** for neuron i with respect to digit position d is:

$$F_{i,d} = \frac{\sum_c |\mathcal{X}_{c,d}| \left(\mu_{i,c,d} - \mu_{i,d}\right)^2}{\sum_c |\mathcal{X}_{c,d}| \sigma^2_{i,c,d}}$$

Intuition. The Fisher Score quantifies how much neuron i's activation varies *between* digit-pair classes (numerator), relative to how much it varies *within* each class (denominator). A high $F_{i,d}$ indicates that neuron i's activation reliably distinguishes between different digit values at position d, while remaining relatively insensitive to other input variations. This implies that neuron i likely participates in a digit-position-specific addition subcircuit.

[4] We choose to concatenate digit pairs, as a way to express the arithmetic digit level subtasks as a class label

E Statistics on Arithmetic Circuits

Circuit Size. We find that digit-specific circuits are relatively 'wide' (Figure 18). For the respective best thresholds $t*$[5], the average number of MLP neurons per layer responsible for one of the digit-position specific circuits is 60.3% of all MLP neurons. In other words, almost two-thirds of the MLP neurons in relevant layers are digit-position specific, when Llama 3 8B solves addition tasks.

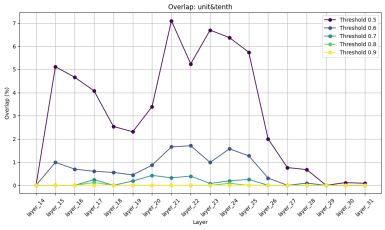
Circuit Overlap. If arithmetic tasks are indeed solved in a digit-wise manner, then neuron groups responsible for generating different digit position results should be distinct. To assess the distinctiveness of different digit-position circuits, we examine the overlap of member neurons across digit position circuits. We find that neuron sets for different digit positions are largely distinct (Figure 18). In fact, for higher thresholds (>= 0.7) neural overlap between circuits is largely negligible (<2%). The distinctiveness of digit-position specific arithmetic circuits provides a first indication of digit-positional modularity in LLM arithmetic.

Circuit Sufficiency. To evaluate whether Fisher-identified neuron groups are sufficient to represent digit-specific arithmetic subtasks, we perform a Linear Discriminant Analysis (LDA)-based classification test. For each layer l and digit position d, we train two LDA classifiers to predict the digit-pair label c: one using the full MLP activation vector, and one using a reduced representation containing only neurons above a Fisher importance threshold.

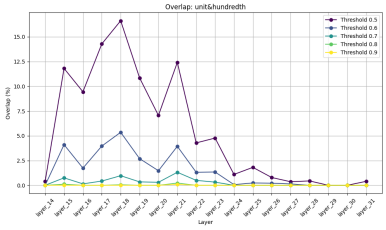
Figure 18 compares classification accuracy between the full and reduced settings. We find that performance remains high in the reduced setting—especially in middle layers (15–24)—indicating that the selected neurons are sufficient to support digit-position classification.

Sufficiency is strongest for the unit and hundreds digits, and at lower thresholds, also for the tens digit. Stable accuracy across thresholds suggests that unit and hundreds information is more redundantly or compactly encoded, while the tens digit may require more neurons. A drop in later layers suggests that digit-specific information becomes less localized in deeper MLP layers.

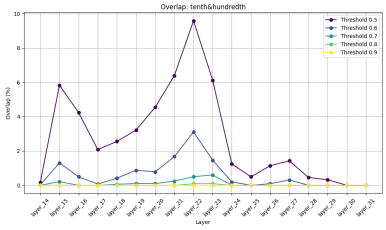
[5] See Section 3.2: 0.5 for unit, 0.6 for tens, 0.9 for hundreds



(a) Unit and Tens

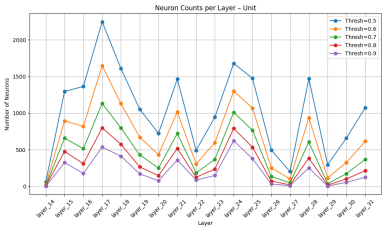


(b) Unit and Hundreds

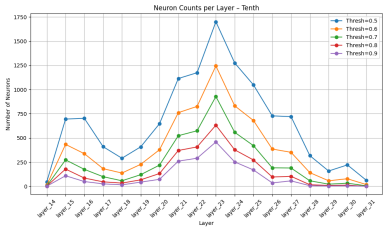


(c) Tens and Hundreds

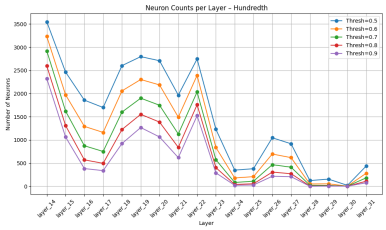
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

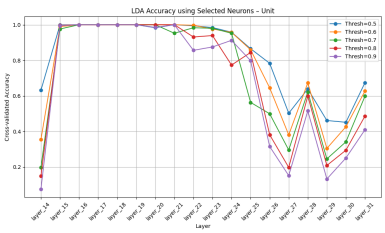


(f) Tens

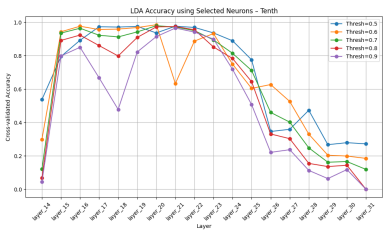


(g) Hundreds

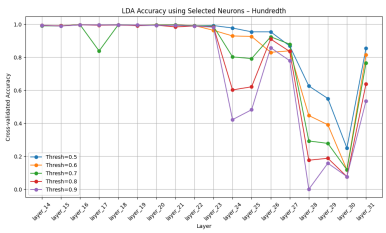
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 4096).



(i) Unit



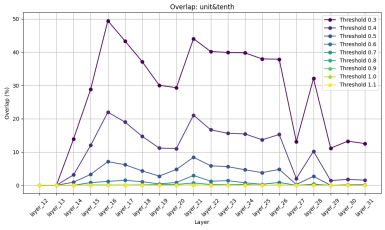
(j) Tens



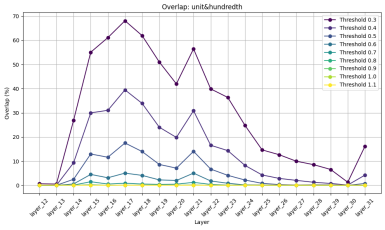
(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

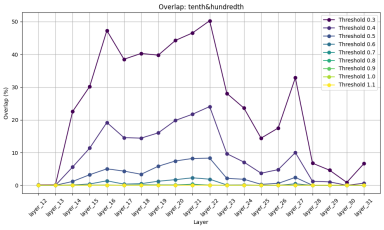
Figure 18: Llama 3 8B, D_{add}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

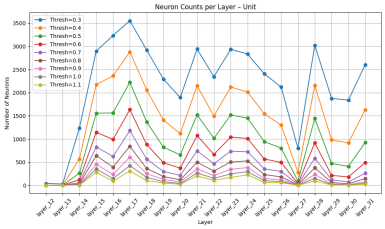


(b) Unit and Hundreds

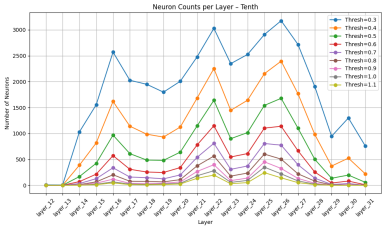


(c) Tens and Hundreds

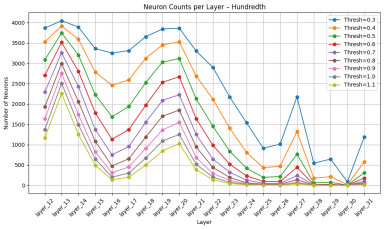
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

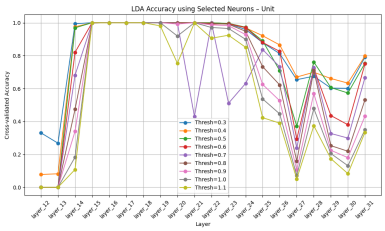


(f) Tens

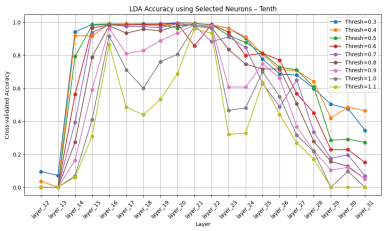


(g) Hundreds

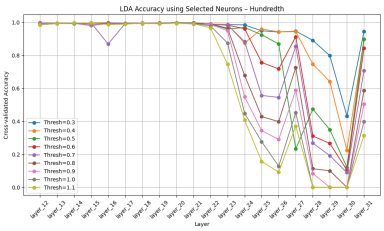
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 4096).



(i) Unit



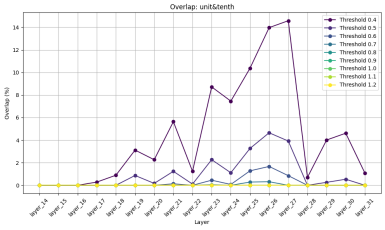
(j) Tens



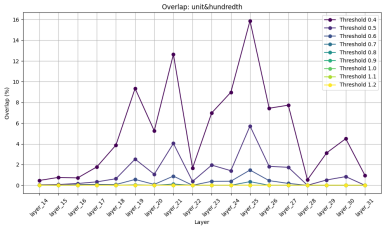
(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

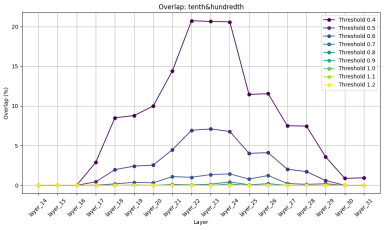
Figure 19: Llama 3 8B, D_{sub}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

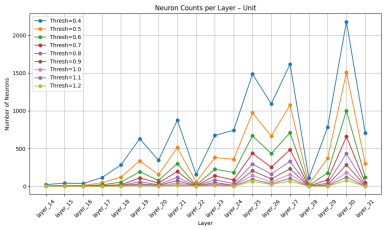


(b) Unit and Hundreds

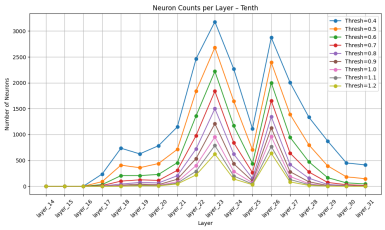


(c) Tens and Hundreds

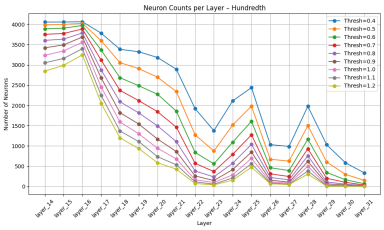
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

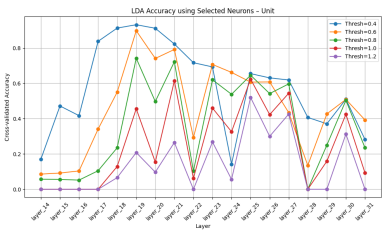


(f) Tens

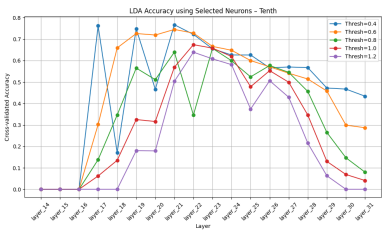


(g) Hundreds

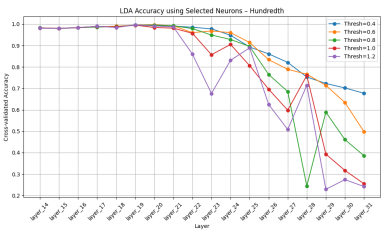
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 4096).



(i) Unit



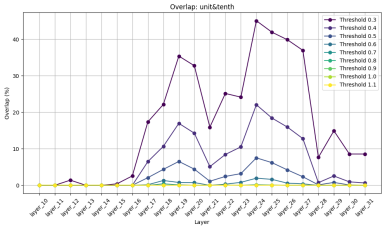
(j) Tens



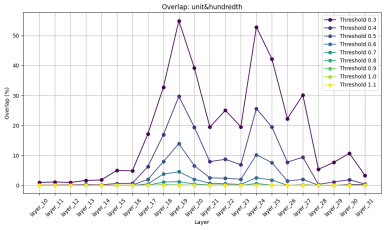
(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

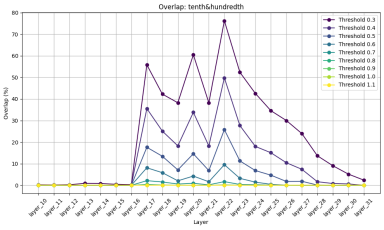
Figure 20: Olmo 2 7B, D_{add}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

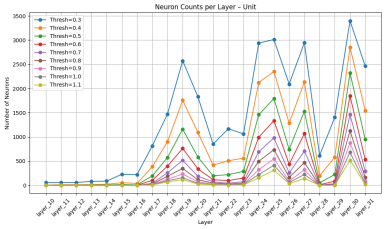


(b) Unit and Hundreds

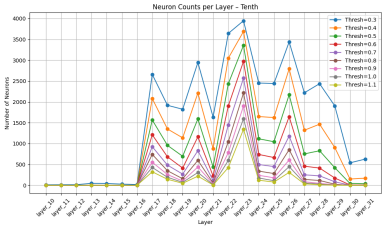


(c) Tens and Hundreds

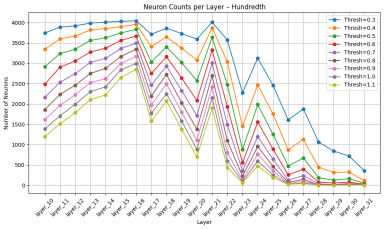
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

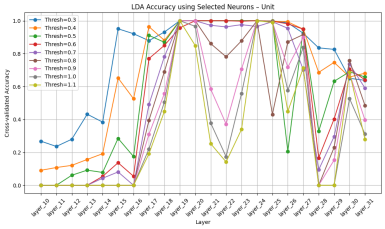


(f) Tens

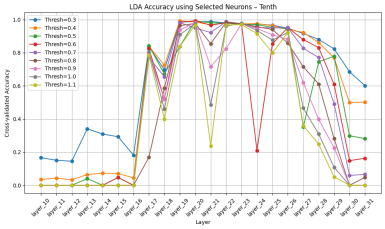


(g) Hundreds

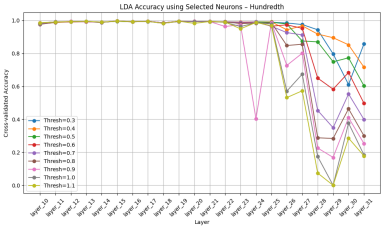
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 4096).



(i) Unit



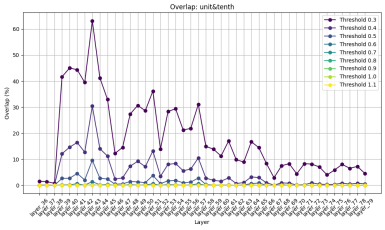
(j) Tens



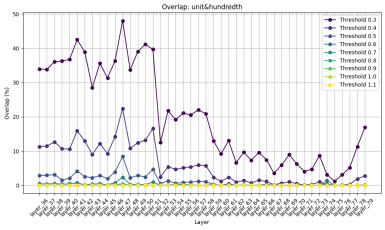
(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

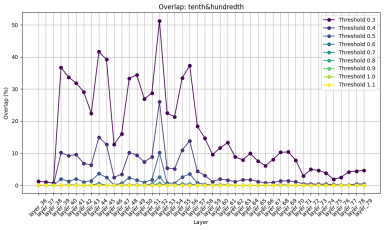
Figure 21: Olmo 2 7B, D_{sub}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

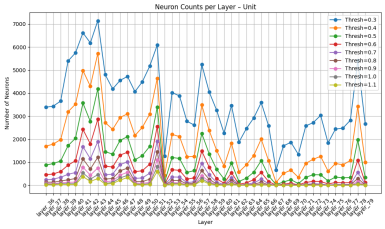


(b) Unit and Hundreds

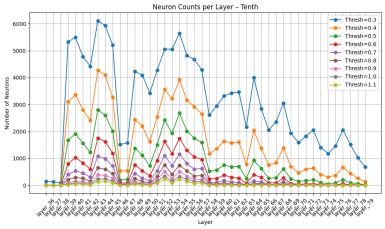


(c) Tens and Hundreds

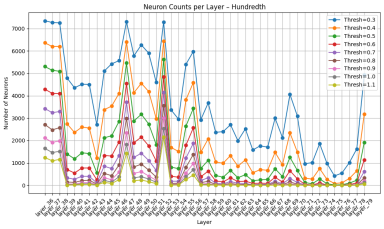
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

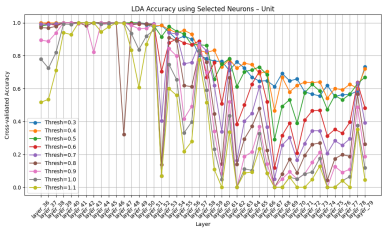


(f) Tens

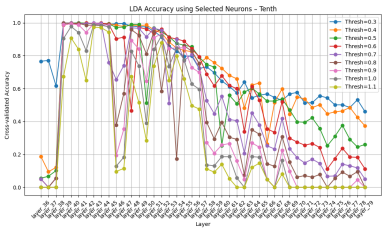


(g) Hundreds

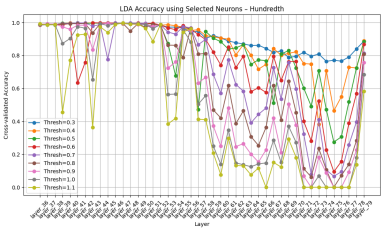
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 8192).



(i) Unit



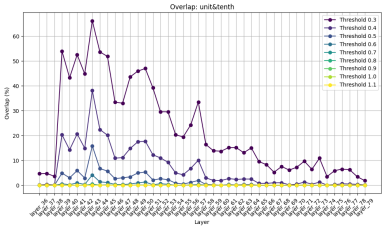
(j) Tens



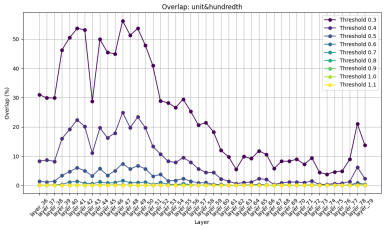
(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

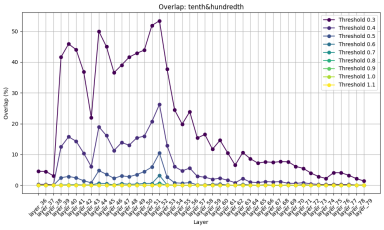
Figure 22: Llama 3 70B, D_{add}: Circuit statistics across digit positions and thresholds.



(a) Unit and Tens

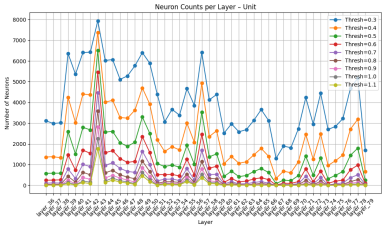


(b) Unit and Hundreds

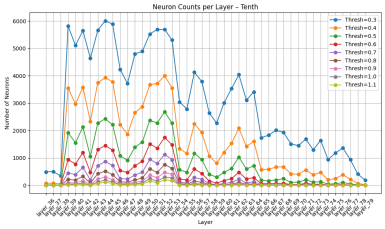


(c) Tens and Hundreds

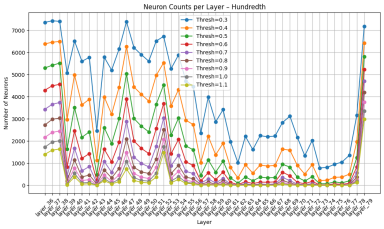
(d) *Circuit Overlap*: Overlap in neurons (%) between digit-position circuits.



(e) Unit

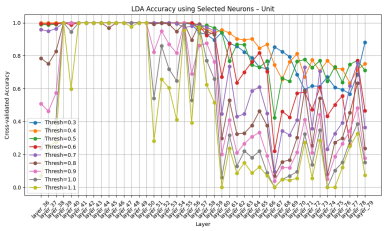


(f) Tens

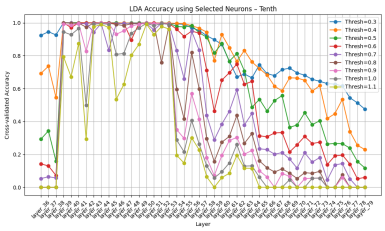


(g) Hundreds

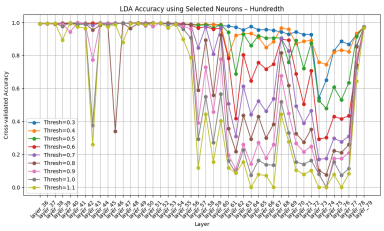
(h) *Circuit Size*: Number of neurons per layer in digit-position circuits (Hidden size = 8192).



(i) Unit



(j) Tens

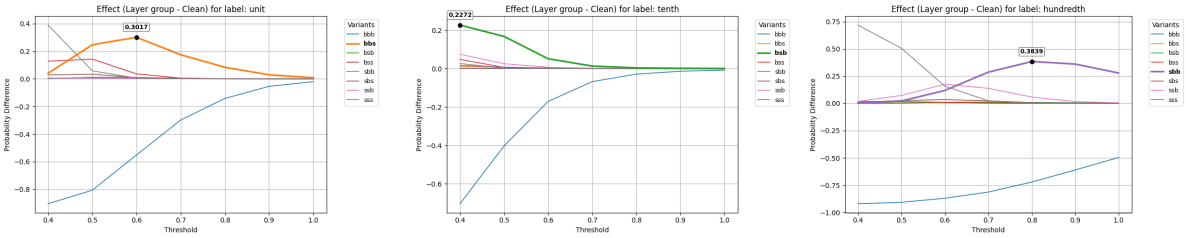


(k) Hundreds

(l) *Circuit Sufficiency*: Sufficiency of digit-position circuit vs. full LDA

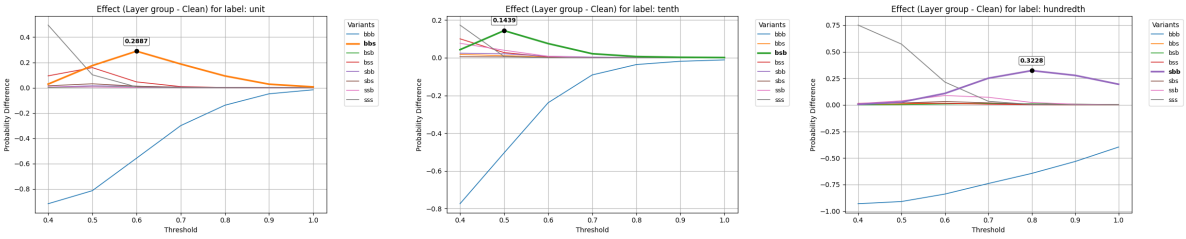
Figure 23: Llama 3 70B, D_{sub}: Circuit statistics across digit positions and thresholds.

F Effect of chosen Threshold on Digit-Circuit Intervention



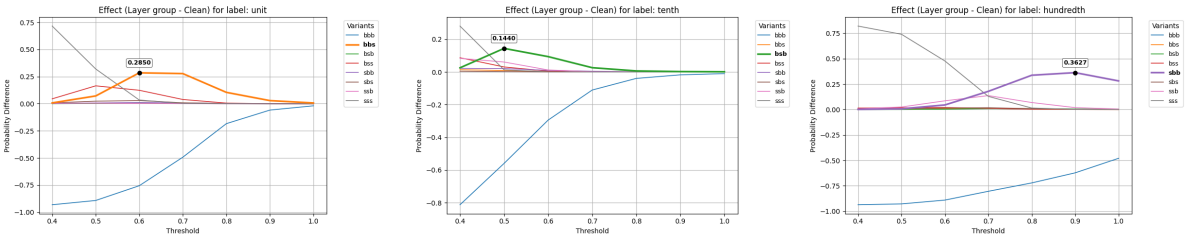
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 24: **Llama 3 8B**, $D_{add,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{16, ..., 24\}$.



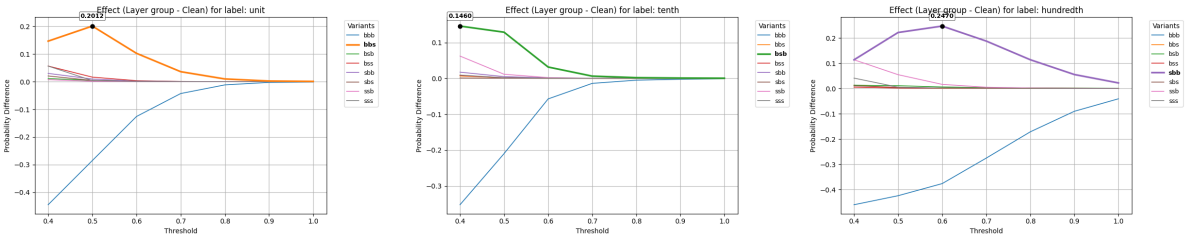
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 25: **Llama 3 8B**, $D_{sub,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{16, ..., 28\}$.



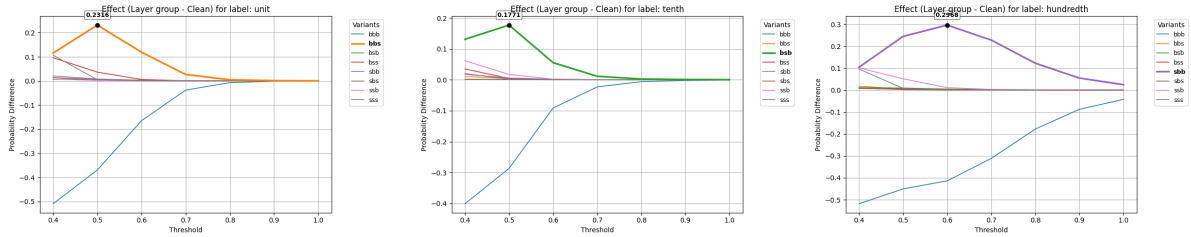
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 26: **Llama 3 8B**, $D_{sub,op2}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{15, ..., 28\}$.



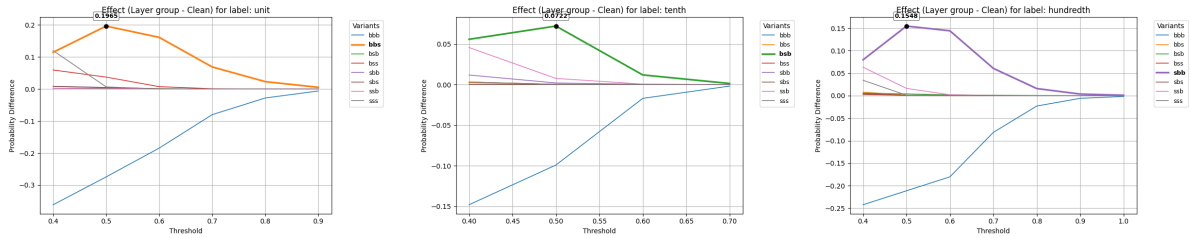
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 27: **Llama3-70B**, $D_{add,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{39, ..., 56\}$.



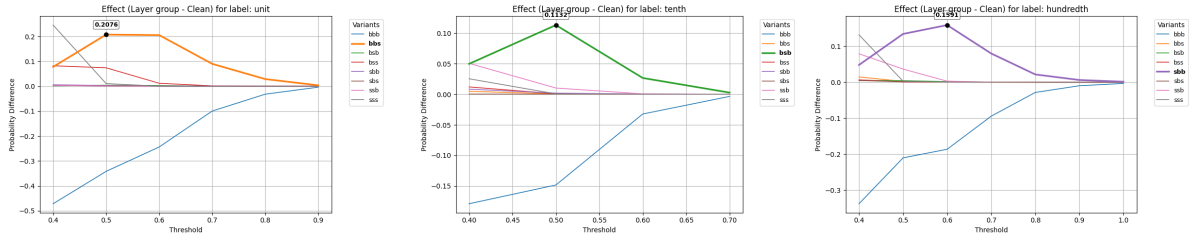
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 28: **Llama3-70B**, $D_{add,op2}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{39, ..., 56\}$.



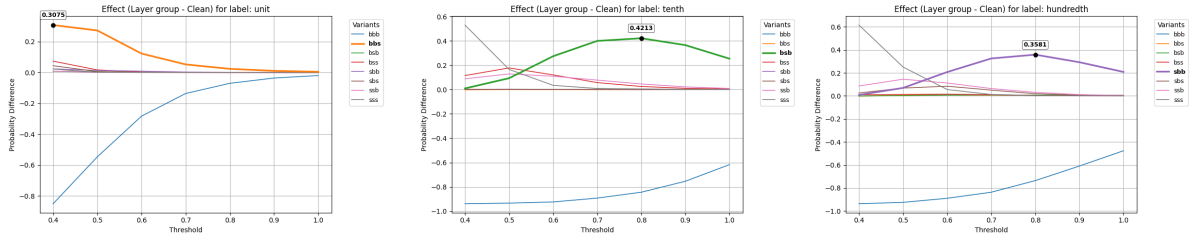
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 29: **Llama3-70B**, $D_{sub,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{39, ..., 58\}$.



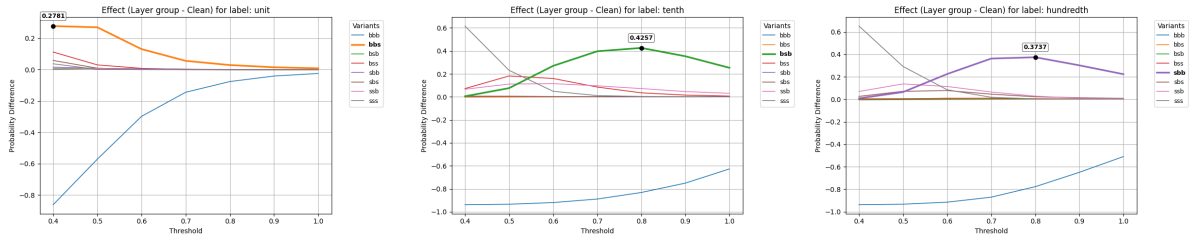
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 30: **Llama3-70B**, $D_{sub,op2}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{39, ..., 58\}$.



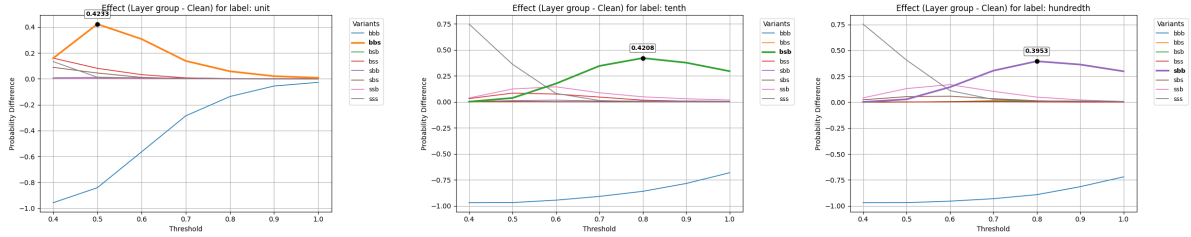
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 31: **Olmo 2 7B**, $D_{add,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{17, ..., 30\}$.



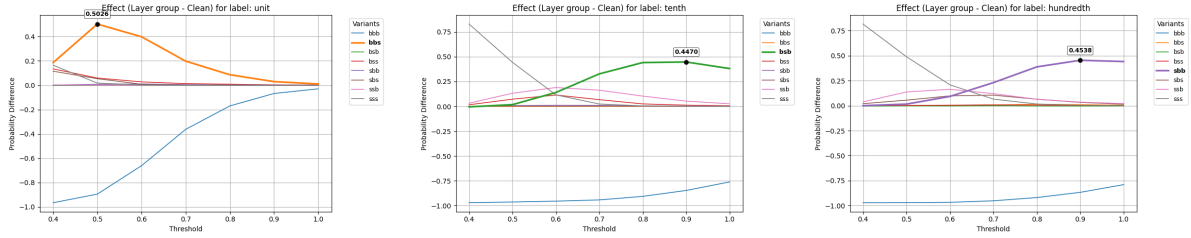
(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 32: **Olmo 2 7B**, $D_{add,op2}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{17, ..., 30\}$.



(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 33: **Olmo 2 7B**, $D_{sub,op1}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{19, ..., 27\}$.



(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 34: **Olmo 2 7B**, $D_{sub,op2}$: Effect size of circuit specific interventions with different thresholds for neuron circuit membership, on circuit neurons in layers $L = \{17, ..., 27\}$.

G Intervention Results on $D_{add,op1}$ and $D_{sub,op1}$

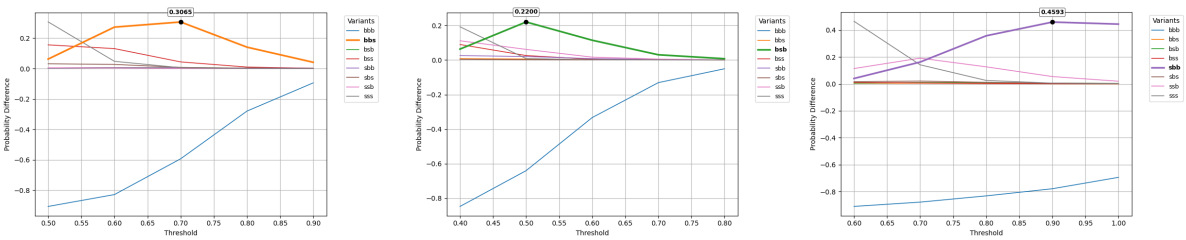
m	o	d	$t*$	bbb	**bbs**	**bsb**	**sbb**	bss	sbs	ssb	sss
				Absolute change in prediction probability Δp in percentage points (after - before).							
Llama 3 8B	+	unit	0.6	-55.74%	**+30.17%**	+1.01%	+0.41%	+3.65%	+0.87%	+0.30%	+0.54%
		tens	0.4	-70.44%	+1.27%	**+22.72%**	+1.71%	+4.82%	+0.28%	+7.52%	+2.67%
		hun.s	0.8	-72.17%	+0.25%	+0.87%	**+38.39%**	+0.13%	+0.71%	+5.83%	+0.40%
	−	unit	0.6	-55.75%	**+28.87%**	+0.46%	+1.12%	+4.57%	+1.20%	+0.19%	+0.63%
		tens	0.5	-50.48%	+1.12%	**+14.39%**	+2.11%	+2.76%	+0.48%	+3.97%	+0.69%
		hun.s	0.8	-64.48%	+0.46%	+1.07%	**+32.28%**	+1.77%	+0.62%	+2.26%	+0.41%
Llama 3 70B	+	unit	0.5	-28.51%	**+20.12%**	+0.48%	+0.97%	+1.66%	+0.45%	+0.15%	+0.40%
		tens	0.4	-35.24%	+0.67%	**+14.60%**	+1.75%	+0.84%	+0.12%	+6.24%	+0.93%
		hun.s	0.6	-37.65%	+0.08%	+0.61%	**+24.70%**	+0.12%	+0.20%	+1.63%	+0.18%
	−	unit	0.5	-27.45%	**+19.65%**	+0.22%	+0.49%	+3.72%	+0.43%	+0.07%	+0.66%
		tens	0.5	-9.52%	+0.41%	**+7.22%**	+0.21%	+0.04%	+0.18%	+0.76%	+0.03%
		hun.s	0.5	-21.14%	+0.13%	+0.40%	**+15.47%**	+0.08%	+0.09%	+1.62%	+0.11%
Olmo 2 7B	+	unit	0.4	-85.04%	**+30.75%**	+0.83%	+7.34%	+2.07%	+4.39%	+0.90%	+2.28%
		tens	0.8	-84.41%	-0.06%	**+42.13%**	+2.51%	-0.05%	+0.04%	+4.56%	+0.47%
		hun.s	0.8	-73.61%	+0.55%	+0.69%	**+35.81%**	+0.30%	+1.88%	+3.04%	+0%
	−	unit	0.5	-84.73%	**+42.33%**	+0.42%	+1.13%	+8.14%	+4.50%	+0.13%	+1.30%
		tens	0.8	-86.03%	+0.14%	**+42.08%**	+0.73%	+1.58%	+0.03%	+4.80%	+0.30%
		hun.s	0.8	-89.26%	+1.29%	+1.11%	**+39.53%**	+0.25%	+1.40%	+0.49%	+1.22%

Table 6: Main Results: For all detected circuits (across models, operators, and digit positions) we report the change in prediction probabilities for result variants after interventions on digit-position-specific arithmetic circuits for the best threshold $t*$ in each circuit, on datasets $D_{add,op1}$ and $D_{sub,op1}$. The increase in prediction probability for the correct digit-specific result variant is shown in bold.

m	o	d	$t*$	Flip Rate
Llama 3 8B	+	unit	0.6	42.5%
		tens	0.4	38.5%
		hun.s	0.8	58.5%
	−	unit	0.6	36.5%
		tens	0.5	19%
		hun.s	0.8	47.5%
Llama 3 70B	+	unit	0.5	17.5%
		tens	0.4	13%
		hun.s	0.6	24.5%
	−	unit	0.5	16.5%
		tens	0.5	3.5%
		hun.s	0.5	11%
Olmo 2 7B	+	unit	0.4	50.5%
		tens	0.8	53%
		hun.s	0.8	45%
	−	unit	0.5	56%
		tens	0.8	53%
		hun.s	0.8	48%

Table 7: Flip rate from bbb result to the intended digit-specific result variant (Unit: bbb → bbs,Tens: bbb → bsb, Hundreds: bbb → sbb), results given for the best threshold $t*$ for each circuit, on datasets $D_{add,op1}$ and $D_{sub,op1}$.

H Ablation: Deeper Circuits



(a) Unit Circuit intervention (b) Tens Circuit intervention (c) Hundreds Circuit Intervention

Figure 35: Effect size of circuit specific interventions on Llama 3 8B and $D_{add,op2}$. We intervene on a deeper circuit ($L = \{15, ..., 28\}$) and find no significant difference to the shallower circuit depth ($L = \{15, ..., 24\}$) chosen based on the statistics on circuit sufficiency and size.

J Examples of Neuron Heuristics

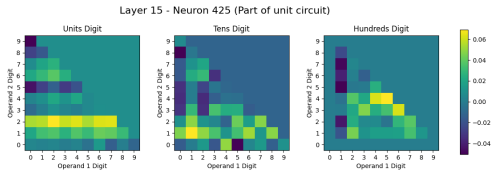


Figure 36: MLP neuron $N_{15,425}$ is part of unit circuit - Heuristic: Operand 2 is 2 in unit digit position.

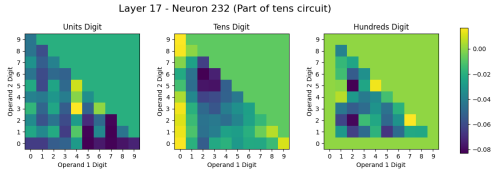


Figure 37: MLP neuron $N_{17,232}$ is part of tens circuit - Heuristic: Operand 1 is 0 in tens digit position.

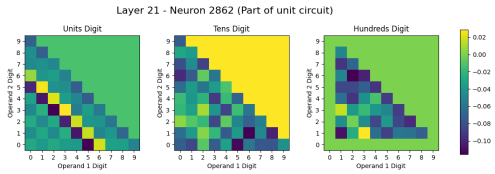


Figure 38: MLP neuron $N_{21,2862}$ is part of unit circuit - Heuristic: Result is 6 in unit digit position.

I Similarity of Addition and Subtraction Circuits

Layer	Digit Pos	Top-50	Top-100	Top-250	Top-500	Top-1000
15	unit	12.0%	17.0%	12.8%	7.2%	3.6%
	tens	4.0%	2.0%	0.8%	0.4%	0.2%
	hundreds	10.0%	10.0%	15.6%	18.2%	16.0%
16	unit	16.0%	14.0%	5.6%	2.8%	1.4%
	tens	8.0%	4.0%	1.6%	0.8%	0.4%
	hundreds	16.0%	15.0%	11.6%	5.8%	2.9%
17	unit	36.0%	32.0%	36.4%	26.0%	13.0%
	tens	2.0%	1.0%	0.4%	0.2%	0.1%
	hundreds	18.0%	18.0%	20.0%	10.0%	5.0%
18	unit	24.0%	25.0%	28.0%	15.6%	7.8%
	tens	6.0%	3.0%	1.2%	0.6%	0.3%
	hundreds	40.0%	32.0%	28.4%	31.0%	22.4%
19	unit	16.0%	18.0%	8.0%	4.0%	2.0%
	tens	6.0%	3.0%	1.2%	0.6%	0.3%
	hundreds	48.0%	44.0%	44.0%	42.8%	44.8%
20	unit	14.0%	7.0%	2.8%	1.4%	0.7%
	tens	6.0%	3.0%	1.2%	0.6%	0.3%
	hundreds	44.0%	45.0%	42.0%	40.0%	36.8%
21	unit	28.0%	29.0%	29.2%	14.6%	7.3%
	tens	26.0%	34.0%	21.6%	10.8%	5.4%
	hundreds	28.0%	26.0%	29.2%	25.0%	12.5%
22	unit	4.0%	7.0%	3.6%	1.8%	0.9%
	tens	26.0%	26.0%	20.4%	10.2%	5.1%
	hundreds	4.0%	3.0%	6.0%	5.6%	5.5%
23	unit	18.0%	17.0%	12.4%	6.2%	3.1%
	tens	4.0%	4.0%	5.6%	3.6%	1.8%
	hundreds	2.0%	5.0%	2.8%	1.4%	0.7%
24	unit	26.0%	24.0%	24.4%	24.2%	12.1%
	tens	12.0%	12.0%	7.6%	3.8%	1.9%
	hundreds	0.0%	0.0%	0.0%	0.0%	0.0%

Table 8: Overlap of Top-K Fisher Score neurons between addition and subtraction circuits across layers and digit positions.