Benford's Curse: Tracing Digit Bias to Numerical Hallucination in LLMs

Jiandong Shao[1]*, **Yao Lu**[2], **Jianfei Yang**[1]
[1]Nanyang Technological University, [2]University College London
shamyshao@gmail.com, yao.lu@cs.ucl.ac.uk,
jianfei.yang@ntu.edu.sg

Abstract

Large Language Models (LLMs) exhibit impressive performance on complex reasoning tasks, yet they frequently fail on basic numerical problems, producing incorrect outputs. Inspired by Benford's Law, a statistical pattern in which lower digits occur more frequently as leading digits, we hypothesize that the skewed digit distributions in web-collected corpora may be learned by LLMs during pretraining, leading to biased numerical generation. To investigate the hypothesis, we first examine whether digits frequencies in pretraining corpus (OLMo2) follows Benford's law. We then construct an evaluation benchmark in which the ground-truth digits are uniformly distributed within each of the seven numerical reasoning tasks. Our evaluation results demonstrate that leading open-source LLMs show a consistent pattern of digit bias that resembles Benford's law. Through logit-lens tracing and neuron-level dissection, we identify that this bias arises predominantly from a small subset of highly digit-selective feed-forward network (FFN) neurons in the deeper layers. Finally, we demonstrate that pruning these neurons mitigates imbalanced overgeneration and partially corrects erroneous outputs, providing causal evidence that fine-grained pretraining digit bias can propagate into model behavior. Our findings reveal a fundamental connection between corpus-level statistics and symbolic failure modes in LLMs, offering a new lens for diagnosing and mitigating hallucinations in numerical tasks. Code and data is made available here: https://github.com/shamy28/Benford-Curse.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive performance not only on daily reasoning tasks but also on challenging domains such as mathematical competition problems, showcasing strong symbolic and logical capabilities [1, 2, 3, 4, 5, 6, 7]. Yet, despite these amazing success, LLMs frequently fail on surprisingly simple numerical reasoning—often producing mathematically incorrect or logically inconsistent outputs [8, 9, 10]. This discrepancy prompts a fundamental and pressing question: why are LLMs so prone to hallucinations when dealing with basic numbers?

Previous studies [11, 12, 13, 14] have identified duplication bias in pretraining corpus as a major contributor to hallucinations in natural language tasks. For instance, when prompted with "list some red fruits, excluding apples", LLMs frequently still mention "red apples" likely due to the high frequency of co-occurring phrases such as "red apples, watermelon, cherries, and strawberries" in the training data. This over-reliance on memorized patterns causes the model to overlook explicit instructions in the prompt [11]. Given this strong pattern-completion tendency in language, a natural question arises: might similar memorization-driven biases emerge in numerical reasoning tasks, potentially leading to systematic numerical hallucinations?

*This work was conducted during the author's research internship at MARS Lab, NTU.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Task: Identifying the Last Term in a Sequence

Prompt 1: What is the result of multiplying the sequence's last term by two? [..., 2.95, 1.81, 8.26]
Model Generation: To find the result of multiplying the sequence's last term by two, we need to identify the last term in the sequence and then multiply it by 2.
The last term in the sequence is **1.81** ✗
(Incorrect: the model mistakenly identifies the second-to-last number as the last term.)

Prompt 2: What is the result of multiplying the sequence's last term by two? [..., 7.43, 8.51, 1.05]
Model Generation: To find the result of multiplying the sequence's last term by two, we need to identify the last term in the sequence and then multiply it by 2.
The last term in the sequence is **1.05** ✓
(Correct: the model correctly identifies the last number in the sequence.)

Figure 1: Illustration of digit bias in identifying the last term of a numerical sequence. LLaMA3.1-8B-Instruct is asked to multiply the final term of a sequence by two, but first must identify the last item. In the first case, the model incorrectly selects a smaller intermediate value (1.81) instead of the actual final term (8.26). In the second, it correctly selects 1.05. This asymmetry suggests a bias toward smaller digits when the model is uncertain, revealing a subtle form of numerical hallucination. Detailed accuracy comparisons are shown in Figure 2.

Benford's Law [15], a striking statistical phenomenon, was first observed in 1881 by astronomer Simon Newcomb, who noticed that the earlier pages of logarithmic tables—those starting with digit `1`—were more worn than others [16]. This empirical observation was later formalized by physicist Frank Benford in 1938, who verified the pattern across over 20 diverse datasets [17]. Benford's Law states that the probability that a number begins with digit $d \in \{1, \ldots, 9\}$ is given by $P(d) = \log_{10}(1 + \frac{1}{d})$. This implies that digit `1` occurs as the leading digit in about 30% of cases, while digit `9` appears less than 5%. This law has since been observed across various domains such as economic records, scientific measurements, and population statistics, and thus naturally emerges in human-written text [15, 18, 19]. Given this ubiquity, it is natural to ask whether the pretraining corpora of LLMs, which are largely composed of web-scraped data, also reflect this skewed digit distribution. If so, could LLMs internalize such patterns during training, and might this lead to a form of *digit bias* that contributes to *numerical hallucinations*, even on otherwise simple numerical reasoning tasks?

To investigate this question, we begin by examining the digit distribution in the Olmo-mix-1124 pretraining corpus [20] and confirm a strong digit skew consistent with Benford's Law. To test whether this corpus-level skew propagates into model behavior, we construct a diagnostic benchmark with uniformly distributed ground-truth digits, eliminating task-induced priors. Empirical results show that LLMs consistently overgenerate smaller digits, despite uniform targets. Further analysis reveals that when the model generates incorrect answers, the distribution of first error digits(e.g., for ground truth 758 and prediction 714, the first error digit is 1) exhibits an even stronger skew toward smaller values. To better understand the underlying mechanisms, we analyse the model's internal representations. First, we apply the Logit Lens [21] to trace how digit preferences evolve across layers and find that the preference for smaller digits often emerges strongly in the later layers. Second, we disentangle the contributions of FFN and self-attention, finding that the bias is primarily driven by the FFN. Third, by quantifying the digit selectivity of FFN neurons, we find that their aggregated preferences form a skewed distribution favouring smaller digits, closely mirroring the statistics of the pretraining corpus. Motivated by these insights, we explore a lightweight neuron-level intervention that prunes a small set of biased neurons. This intervention can partially mitigate overgeneration and correct certain hallucinated outputs, offering further evidence of the causal role digit bias plays in numerical hallucination.

The contributions of this work are summarized as follows: (1) We formulate and investigate a fundamental question chain: does the skewed digit distribution in LLMs' pretraining corpus induce digit generation bias, leading to numerical hallucination? (2) To explore this, we construct a diagnostic benchmark with uniformly distributed target digits and observe that LLMs consistently overgenerate smaller digits. Notably, the first error digits exhibit an even stronger skewed distribution, suggesting a bias-driven mechanism behind numerical hallucination. (3) Using logit lens tracing and neuron-level analysis, we identify that this digit bias primarily originates from a subset of highly digit-selective feedforward (FFN) neurons, particularly concentrated in the later layers of the model. (4) Finally, we

show that pruning these neurons can partially mitigate hallucination, offering causal evidence that digit bias is a contributing factor to numerical hallucination. By identifying a fine-grained statistical artifact as a mechanistic failure point, our work highlights a critical yet underexplored source of errors in numerical reasoning and underscores the importance of addressing pretraining-induced biases to develop more reliable language models.

2 Related Works

Numerical Hallucinations in LLMs. While there is no universally accepted definition of numerical hallucination, the term broadly refers to a language model's tendency to generate numerically incorrect, inconsistent, or implausible outputs, despite syntactically or semantically coherent completions. These include phenomena such as miscounting, incorrect number reproduction, inappropriate number substitution, or generating fabricated values. Prior work has largely attributed these numerical hallucinations to reasoning deficiencies [22], suboptimal tokenization schemes [23], and misalignment [24]. Razeghi et al. [25] further showed that model accuracy in numerical tasks correlates with token frequency in pretraining data, suggesting a statistical basis for some failures. However, these studies do not examine how digit-level statistical patterns manifest during generation. We highlight this as a distinct and measurable form of numerical hallucination overlooked in prior work.

Dataset Bias as a Source of Hallucination. Prior studies have increasingly identified dataset bias as a key driver of hallucination in LLMs [14, 11, 13, 25, 12], suggesting that many hallucinations stem not from architectural flaws but from imbalances in the pretraining corpus. For example, overrepresented entities or linguistic patterns in large-scale web data can lead models to repeat them even in inappropriate contexts [11]. However, existing work [26, 27, 28] has primarily focused on semantic-level biases, such as spurious facts or misattributions, while overlooking lower-level statistical regularities. In particular, the extent to which digit-level statistical pattern in the training data influences the model's numerical generation remains largely unexplored. Our work fills this gap by showing that frequency pattern over digits in the pretraining corpus is systematically reflected in model generations, providing a concrete statistical basis for a subset of numerical hallucinations.

3 Investigating Digit Bias in LLMs

Benford's Law describes a logarithmic distribution over leading digits in naturally occurring numerical data, where smaller digits appear with higher frequency than larger ones. This prompts a critical question: does the pretraining corpus, largely sourced from real-world Internet text, also exhibit a similar skewed digit distribution? If so, how might such a skewed distribution influence digit generation in LLMs? To motivate this investigation, we begin with a simple diagnostic experiment illustrated in Figure 1. In this basic sequence identification task, the model demonstrates noticeably higher accuracy when the final digit to be predicted is small (1 or 2) compared to when it is large (8 or 9), as shown in Figure 2. This asymmetry suggests a preference for generating small digits, hinting at a potential internal digit bias that could underlie numerical hallucinations.

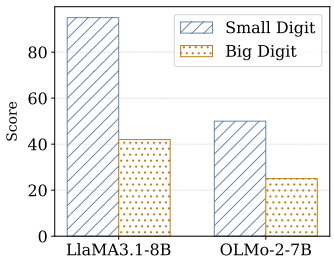


Figure 2: Accuracy in the sequence last term identification task shown in Figure 1. Digits in the range 1-3 (low-value) are recognized with significantly higher accuracy compared to digits in the range 8-9 (high-value), indicating a generation bias toward smaller digits.

Digit Distribution in Pretraining Corpus. To empirically assess whether the digit distribution in pretraining corpus aligns with Benford's Law, we analyze the Olmo-Mix-1124 corpus [20], a widely-used 22.4TB data collection curated for training open-source LLMs. As shown in Figure 4a, the digit frequencies strongly align with Benford's Law, exhibiting a pronounced logarithmic distribution that favors smaller digits. These findings suggest that LLMs are likely to encounter digit distributions that are heavily biased toward smaller values during training.

Table 1: Examples of tasks in the Digit Bias Benchmark, Covering seven numerical reasoning tasks.

Task Category	Examples
Add or Sub	1. What is -0.9121789 minus -6? 2. Add 4292 and 597069553.5.
Multiplication	1. Multiply 13862 and 0.5. 2. What is -464 times -3?
Division	1. Solve 81190 divided by 2. 2. Calculate the division of 40380 by 5.
Evaluate	1. Let c(a) = -2*a - 10. What is c(-1)? 2. Let l(f) = f**3 - 3*f**2 + f + 3. Give l(2).
Nearest Integer Root	1. What is the fifth root of 29889504 to the nearest integer? 2. What is 4528941 to the power of 1/9, to the nearest integer?
Linear_1d	1. Solve 33*r = 8*r - 137 + 712 for r. 2. Solve 309*j + 23940 = -356*j for j.
Sequence Next Term	1. What comes next: 532118, 1064232, 1596346? 2. What is next in 704, 506, 334, 188, 68?

A Benchmark for Measuring Digit Bias. To investigate whether skewed digit distribution in pretraining data leads to generation bias, we introduce the **Digit Bias Benchmark**, a suite of seven numerical reasoning tasks designed to yield uniformly distributed ground-truth digits. Tasks include *add or sub*, *multiplication*, *division*, *evaluate*, *nearest integer root*, *linear_1d*, and *sequence next term*, primarily adapted from DeepMind's Mathematics Dataset [29]. Each task contains over 1,000 examples, with answer sets carefully constructed to ensure uniform digit distribution: when pooling all digits from all positions across all answers within a task (e.g., the answer "132" contributes three digits: 1, 3, and 2), each digit 0-9 appears approximately 10% of the time. This design enables us to disentangle generation bias from task-induced digit distribution effects, allowing a more controlled evaluation of digit-level preferences in LLMs. Representative examples for each task are listed in Table 1.

Empirical Evidence of Digit Bias. We evaluate six open-source LLMs on the proposed Digit Bias Benchmark, including models from LLaMA [30], Qwen [31], Gemma [32], OLMo [33] and Mistral [34] families. By design, the benchmark enforces a uniform digit distribution in its ground truth, so any deviation in the model's output distribution directly reflects inherent generation bias. As shown in Figure 3a, Mistral-7B exhibits a strong and consistent over-generation of smaller digits. For example, digit `1` often appears over 12% of the time, while digits such as `8` and `9` are severely underrepresented. This trend closely parallels the skew found in the pretraining corpus, reinforcing the hypothesis that the bias originates from corpus-level statistics. To further probe the behavioral impact of this bias, we conduct a fine-grained error analysis. For each incorrect output answer, we identify the first digit where the model's output diverges from the ground truth and record its value (e.g., for ground truth 758 and prediction 714, the first error digit is 1). As shown in Figure 3b, the

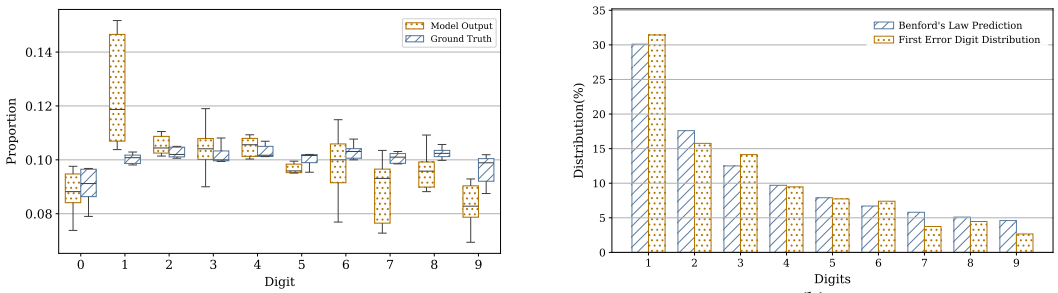


Figure 3: Digit bias observed in the Digit Bias Benchmark with Mistral-7B. (a) The boxplot shows the distribution of digits in generated answers across all tasks, revealing a significant overrepresentation of smaller digits despite the benchmark's uniform ground-truth distribution. (b) The distribution of digits at the first error position exhibits an even stronger skew toward smaller values, closely following Benford's Law. Together, these results suggest that digit bias shapes not just preferences but also the numerical hallucination.

distribution of these "first error digits" exhibits an even stronger skew toward smaller values, closely following Benford's Law. This finding suggests that digit bias not only affects overall preferences but also distorts the model's generation trajectory when it deviates from the correct answer, implicating it as a potential driver of numerical hallucination. More results are shown in Figure 9.

4 Analyzing the Mechanisms of Digit Bias

4.1 Probing Layerwise Behavior with Logit Lens

Method: Logit Lens. The LLMs used in this paper are based on a decoder-only architecture composed of stacked Transformer layers [35] with residual connections [36]. During a standard forward pass, the hidden representation $h_n^{(0)} \in \mathbb{R}^d$ for the final input token x_n is retrieved from a learned embedding table. This representation is then iteratively updated through each Transformer block, incorporating outputs from both self-attention and feed-forward submodules via residual addition. After the final layer, the hidden state $h_n^{(L)}$ (with L denoting the total number of layers) is normalized and projected via the unembedding matrix $U \in \mathbb{R}^{v \times d}$, yielding logits over the vocabulary of size v. Since hidden states across all layers share the same dimensionality, intermediate representations can also be projected via U to obtain layer-wise token distributions. This technique, known as the *logit lens* [21], allows us to visualize how the model's token predictions evolve across layers, providing interpretable insights into the generation process. To explore the origins of digit bias, we apply the logit lens to trace layer-wise changes in predicted digits.

Datasets. As observed in Section 3, LLMs consistently tend to favor smaller digits in numerical reasoning tasks. This suggests that, under conditions of uncertainty or hallucination, the model is more likely to generate smaller digits. Therefore, to trace the internal origins of this bias, hallucinated outputs should be the focus. However, hallucinations in multi-step reasoning tasks are often hard to localize, as the precise point of failure is difficult to identify. To simplify this analysis, we begin with single-step reasoning tasks (basic arithmetic tasks from the benchmark), where hallucination can be precisely localized by identifying the first incorrectly generated digit. Then, to expand this analysis beyond a limited set of hand-selected cases, we develop an entropy-based automated sampling strategy grounded in observations from these hand-analyzed examples to identify uncertain samples. Specifically, we compute the entropy of the model's output digit token distribution at each layer. Samples with entropy exceeding a threshold (e.g., >3.0 at layer 26) are flagged as potentially biased. Applying this criterion to the *Evaluate* task yields a larger set of high-uncertainty samples, enabling a more systematic investigation of digit bias within the model's internal dynamics.

Layerwise Visualization of Digit Bias. By applying the logit lens [21] to high-uncertainty samples that ultimately generate different digits, we obtain the layer-wise digit preference heatmap shown in Figure 4b. The visualization reveals a consistent pattern: under similar levels of uncertainty, smaller digits tend to exhibit stronger generation signals only in the later layers, whereas larger digits

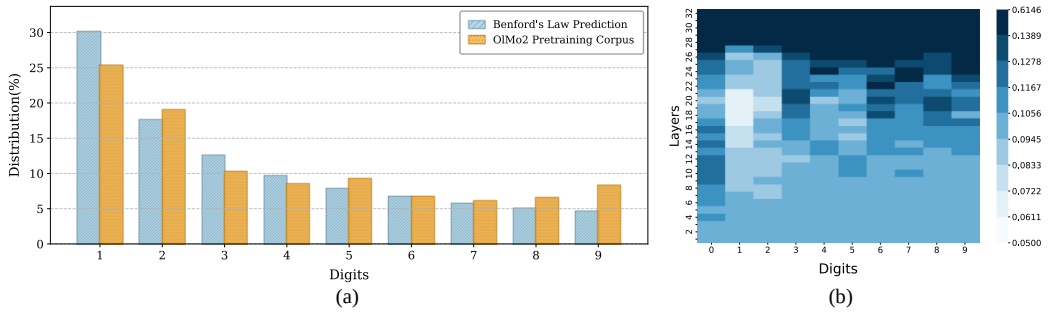


Figure 4: (a) The histogram compares the digit distribution predicted by Benford's Law with that of the OlMo-mix-1124 corpus, showing their degree of similarity. (b) The heatmap of digit probabilities across layers obtained via Logit Lens. Smaller digits remain relatively undistinguished in early and middle layers, whereas larger digits show sharper activations earlier. This indicates that the overgeneration of small digits is driven by preferences formed in the final layers.

show earlier and more gradual emergence. For instance, digit 1 typically shows little preference in intermediate layers but becomes strongly favored in the final few. These results indicate that digit bias is not uniformly distributed across the network, but instead predominantly arises in the final layers.

Layerwise Bias Trends. To gain deeper insight into how digit bias emerges in the later layers, we conduct a layer-wise analysis focusing on how the model's digit preferences evolve across later layers. Specifically, we examine samples where digit token probabilities are approximately equal at an intermediate layer and then trace how these probabilities shift in subsequent layers. As shown in Figure 5, the model begins to exhibit a clear preference for smaller digits in later layers, with their probabilities increasing more sharply compared to larger digits. This further supports our observation that digit bias is primarily concentrated and amplified in the final layers of token generation.

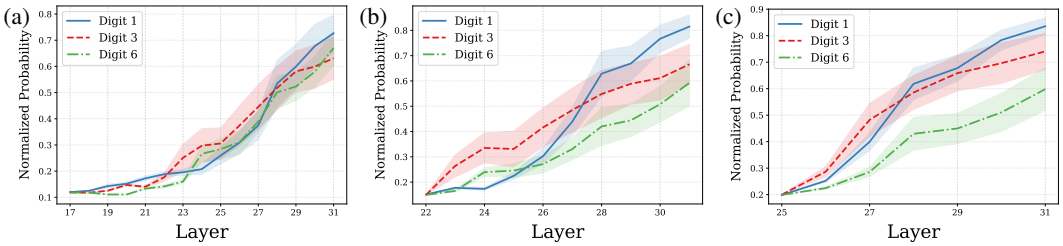


Figure 5: **Digit probability trajectories in later layers.** Starting from layers where digits 1, 3, and 6 have equal probabilities (layers 17, 22, and 25 respectively), we trace their subsequent evolution. Digit 1 consistently gains probability more rapidly than the others, suggesting that the model amplifies its bias toward smaller digits during final token prediction stages.

4.2 Digit Selectivity Score

To quantify how strongly a hidden vector favors a specific digit more precisely, we propose the **Digit Selectivity Score (DSC)**. Given a hidden vector h^d, we first project it into vocabulary logits via the model's unembedding matrix. For each digit token $i \in \{0, \ldots, 9\}$, we compute its rank within the full vocabulary. Let S denote the sum of ranks of all digit tokens. Then, the DSC for digit i is defined as $\text{DSC}_i = S/\text{rank}(i)$. Higher values indicate stronger selectivity for that digit. While normalized probabilities can indicate digit preference to some degree, they often fail to provide meaningful differentiation when all digit probabilities are low, leading to misleading or noisy interpretations. In contrast, DSC offers a more sensitive and robust measurement of digit selectivity for any vector in the model's hidden space.

4.3 Self-Attention vs. FFN

Our goal in this section is to determine whether digit bias primarily originates from the self-attention, the FFN, or their combined effect. Prior work [37, 38, 39] using causal mediation analysis has shown that, in arithmetic tasks, digit generation is predominantly driven by the FFN, while only a small number of attention heads contribute by storing simple arithmetic facts or attending to operands and operators. To examine whether digit bias follows a similar pattern, we compare the correlation between the DSCs of each layer's residual stream and the DSCs of its self-attention and FFN. Let $\mathbf{d} = (d^0, d^1, \ldots, d^L)$ denote the DSCs of the residual stream across all layers, $\mathbf{d}_{\text{ffn}} = (d^0_{\text{ffn}}, d^1_{\text{ffn}}, \ldots, d^L_{\text{ffn}})$ the DSCs computed from FFN outputs, and $\mathbf{d}_{\text{attn}} = (d^0_{\text{attn}}, d^1_{\text{attn}}, \ldots, d^L_{\text{attn}})$ those from self-attention outputs. We then compute the Spearman correlation between $\mathbf{d}$ and each of $\mathbf{d}_{\text{ffn}}$ and $\mathbf{d}_{\text{attn}}$ to assess which component more closely aligns with the overall digit bias in the model's hidden representations.

Comparison Results. Figure 6 presents the Spearman correlation results across layers. In intermediate layers, the residual DSC is strongly correlated with the self-attention output, while in the later layers, the residual DSC shows a much stronger correlation with the FFN output and a very weak correlation with the self-attention output. Given that digit bias emerges most prominently in later layers, this suggests that the FFN module in the later layers is the main contributor to such bias.

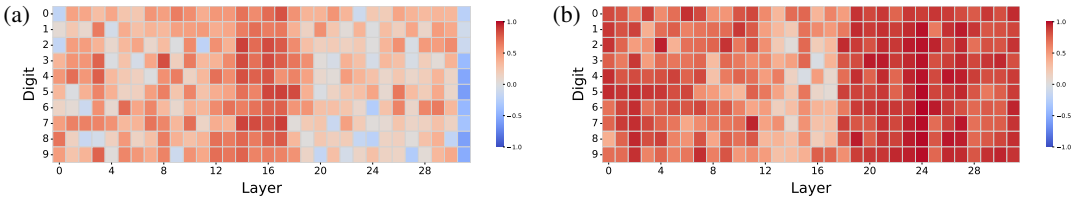


Figure 6: (a) Self-attention DSC vs. residual DSC. (b) FFN DSC vs. residual DSC. Self-attention exhibits moderate correlation in middle layers, while the FFN shows a stronger correlation in the later layers, highlighting the FFN's role in driving digit bias.

4.4 FFN-level Analysis

Having established that digit bias primarily emerges from the FFN in later layers, we now investigate why FFNs in these layers tend to induce such bias.

Interpreting the FFN. The feed-forward network (FFN) at layer m produces its output as:

$$\mathbf{f}_{\text{out}}^{(m)} = \text{FFN}_{\text{in}}(\mathbf{f}_{\text{in}}^{(m)}) \cdot \mathbf{W}_{\text{out}}^{(m)} = \mathbf{f}_{\text{int}}^{(m)} \cdot \mathbf{W}_{\text{out}}^{(m)} = \sum_{n=1}^{d_{\text{int}}} f_{\text{int}}^{m,n} \cdot \mathbf{w}_{\text{out}}^{m,n} \tag{1}$$

Here, $\mathbf{f}_{\text{in}}^{(m)} \in \mathbb{R}^d$ and $\mathbf{f}_{\text{out}}^{(m)} \in \mathbb{R}^d$ are the input and output vectors of the FFN at layer m, $\mathbf{f}_{\text{int}}^{(m)} \in \mathbb{R}^{d_{\text{int}}}$ is the post-activation intermediate vector, and $\mathbf{W}_{\text{out}}^{(m)} \in \mathbb{R}^{d_{\text{int}} \times d}$ is the down-projection matrix. The pair $(f_{\text{int}}^{m,n}, \mathbf{w}_{\text{out}}^{m,n})$ represents the activation and output direction of the n-th neuron in the layer. This formulation aligns with the view of Geva et al. [40], who interpret each FFN neuron as a "key-value memory" unit, where the output direction $\mathbf{w}_{\text{out}}^{m,n}$ encodes semantic information. When projected through the unembedding matrix U, it reveals a token preference distribution that can be interpreted as the neuron's functional role.

What Drives Digit Bias in the FFN? The output of an FFN layer is shaped not only by the input representation but also by the directional structure of its learned parameters. While inputs are often complex and difficult to interpret, the output directions of individual neurons offer a more interpretable handle. Building on the interpretability analysis above, we examine whether the skewed digit distribution from the pretraining corpus is reflected in the FFN's parameters. To this end, we compute the DSC of each neuron with respect to each digit, and then aggregate these scores to obtain a model-wide selectivity profile. We find a strong correlation between this profile and the digit frequency distribution in the pretraining data ($r = 0.949$, Pearson), suggesting that the FFN not only encodes general numerical knowledge but also internalizes corpus-level frequency biases. Figure 7 shows the DSC distributions of the top 1000 most selective neurons for digit 1 and digit 7. Neurons associated with digit 1 consistently exhibit higher selectivity scores than those for digit 7, indicating that the model dedicates a larger portion of its representational space to more frequent digits. This uneven allocation likely contributes directly to the emergence of digit bias in generation.

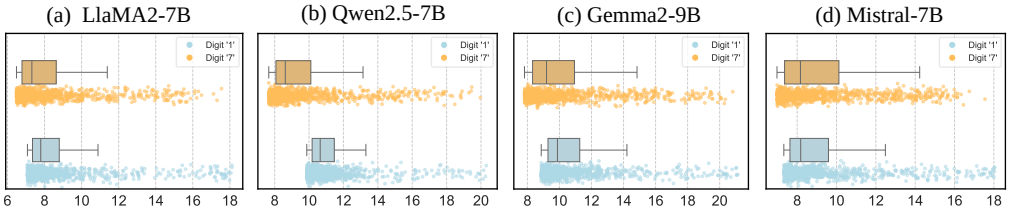


Figure 7: FFN neuron-level selectivity distributions for digit '1' (blue) and digit '7' (orange) in four open-source LLMs: (a) LLaMA2-7B, (b) Qwen2.5-7B, (c) Gemma2-9B, and (d) Mistral-7B. For each digit, we independently select the top 1000 neurons with the highest selectivity. Across all models, neurons most selective for digit '1' exhibit higher selectivity scores than those most selective for digit '7', revealing a stronger model-internal bias toward lower digits.

Table 2: Effect of pruning top 0.01% most digit-1-selective neurons. **Coreccted - Prop.** denotes the proportion of all test samples that were originally incorrect but become correct after pruning. **Original - Prop.** denotes the original frequency of digit 1 in model outputs. **Pruned - Prop.** denotes the digit 1 frequency after pruning.

Model		Multistep Reasoning Tasks			
		Evaluate	Linear_1d	Nearest Integer root	Sequence Next term
Llama2-7B	Coreccted - Prop.	1.36 %	0.76 %	0.19 %	0.18 %
	Original - Prop.	16.26 %	16.21 %	11.91 %	11.70 %
	Pruned - Prop.	11.17 %	13.10 %	6.6 %	9.74 %
Mistral-7B	Coreccted - Prop.	1.22 %	1.94 %	0.35 %	0.54 %
	Original - Prop.	15.63 %	14.49 %	21.64 %	11.90 %
	Pruned - Prop.	11.85 %	10.92 %	10.71 %	10.61 %
Qwen2.5-7B	Coreccted - Prop.	3.49 %	5.05 %	4.96 %	4.73 %
	Original - Prop.	16.45 %	15.85 %	16.25 %	14.06 %
	Pruned - Prop.	14.72 %	13.86 %	13.13 %	12.29 %
Gemma2-9B	Coreccted - Prop.	1.69 %	2.14 %	1.94 %	1.16 %
	Original - Prop.	13.66 %	11.04 %	17.49 %	11.36 %
	Pruned - Prop.	12.53 %	10.91 %	14.22 %	11.23 %

5 Debiasing Method: Probing the Causal Role of Digit Bias

Our analysis shows that LLMs not only overgenerate smaller digits but also exhibit a stronger tendency to generate them when first deviating from the correct answer, following a pattern consistent with Benford's Law. This suggests that digit bias may shape not just generation preferences but also the trajectory of numerical hallucination. However, statistical alignment alone does not establish whether digit bias plays a direct causal role in numerical hallucination. To probe this link, we introduce a lightweight neuron pruning intervention. Rather than aiming to improve overall accuracy, this approach tests whether selectively suppressing neurons most biased toward digit 1 can reliably correct erroneous outputs. If successful, such targeted correction would provide concrete evidence of a causal link between digit bias and numerical hallucination.

Table 3: Corrected - Prop. on arithmetic benchmarks after pruning.

Model	GSM8k	SVAMP
Llama2-7B	2.35 %	1.60 %
Mistral-7B	1.97 %	1.40 %
Qwen2.5-7B	2.12 %	0.70 %
Gemma2-9B	0.08 %	0.50 %

5.1 Locating and Pruning Biased Neurons

Each FFN neuron's output contributes directly to the residual stream and thus influences final token prediction. We estimate a neuron's contribution to digit bias by isolating its standalone output and computing its Digit Selectivity Score toward each digit.

Formally, for the n-th neuron in layer m, we define its bias score toward digit i as:

$$\textbf{biscore}^{m}_{n,i} = \textbf{DSC}_i\left(\mathrm{f}^{m,n}_{\text{int}} \cdot \mathbf{w}^{m,n}_{\text{out}}\right) \tag{2}$$

Here, $\mathrm{f}^{m,n}_{\text{int}}$ is the neuron's activation, and $\mathbf{w}^{m,n}_{\text{out}}$ is its corresponding output direction. This scalar product represents the neuron's individual contribution to the final output, which can be interpreted in terms of digit selectivity via the DSC. Given that individual neurons may encode multiple features or functions [41], indiscriminate pruning across all decoding steps may disrupt general model behavior. Therefore, we prune only the top 0.01% most digit-1-selective neurons, and activate this intervention only during the generation of digit tokens. Furthermore, to minimize the disruption to the model's reasoning path, the generation token after pruning is restricted to digit token only.

5.2 Debiasing Results and Causal Insight

Table 2 reports two key effects of pruning the top 0.01% most digit-1-selective neurons: a notable reduction in the generation frequency of digit 1, and a measurable correction rate among previously

Comparing Model Behavior With and Without Pruning

Prompt 1: Solve -2005 = -1453*p + 1977*p + 3235 for p.
Generation (no pruning): ... Simplify the left side: `\[-5240 = 524p\] 3. **Solve for` p`**` by by dividing both sides by 524: `\[p = \frac{-5241}{524}\]`... ✗
Generation (pruning): ... Simplify the left side: `\[-5240 = 524p\] 3. **Solve for` p`**` by by dividing both sides by 524: `\[p = \frac{-5240}{524}\]`... ✓

Prompt 2: Solve 366*k + 9029 = 14885 for k.
Generation (no pruning): To solve for k in the equation 366k + 9029 = 14885, we can use the following steps: 1.Subtract 9029 from both sides of the equation to get: 366k = 5856 2. Divide both sides of the equation by 366 to get: k = 161 Therefore, the value of k is 161. ✗
Generation (pruning): To solve for k in the equation 366k + 9029 = 14885, we can use the following steps: Step 1: Subtract 9029 from both sides of the equation to get: 366k = 5856 Step 2: Divide both sides of the equation by 366 to get: k = 5856 / 366 k = 16 Therefore, the value of k is 16. ✓

Prompt 3: Solve -47613*w = -47331*w + 14100 for w.
Generation (no pruning): To solve the equation $-47613w = -47331w + 14110$ for w... ✗
Generation (pruning): To solve the equation $-47613w = -47331w + 14100$ for w...✓

Figure 8: Examples showing model responses before and after pruning the top 0.01% of neurons biased toward digit 1. In each cases, pruning corrects an originally erroneous sample by rectifying an intermediate step, demonstrating a causal relationship between digit bias and numerical hallucination.

erroneous outputs. Manual inspection confirms that most of these corrected outputs originally featured small-digit hallucinations as shown in Figure 8. Consistent with these findings, Table 3 shows that similar correction effects also emerge on real-world mathematical reasoning benchmarks (GSM8k [42] and SVAMP [43]). Given the Complexity of multi-step numerical reasoning, such corrections are unlikely to be coincidental. Rather, these corrections suggest that digit bias plays an active role in driving the model off its correct reasoning path by skewing generation toward over-frequent digits like 1. If digit bias were not a causal factor in hallucination, such targeted suppression would not yield consistent improvements on biased outputs. This result provides empirical support for a causal link: **digit bias is not merely a statistical artifact, but a mechanistic contributor to numerical hallucination**.

Takeaway. While our pruning method is coarse and may negatively affect some previously correct predictions, its ability to reliably correct biased errors underscores the causal link between digit bias and hallucination. Rather than being a general-purpose debiasing tool, this method serves as a probing mechanism to deepen our understanding on how low-level statistical priors manifest as systematic reasoning failures in LLMs.

6 Conclusions

This work investigates the overlooked role of digit bias in LLMs. We begin by showing that pretraining corpus exhibit a logarithmic digit distribution consistent with Benford's Law. LLMs internalize this skew, as evidenced by their tendency to overgenerate smaller digits during numerical generation. Notably, in incorrect answers, the first erroneous digit often falls among the smaller digits, suggesting that this bias not only shapes output preferences but also contributes to numerical hallucination. Mechanistically, we trace this bias to the final feed-forward layers of LLMs, where a subset of highly digit-selective neurons encode preferences aligned with corpus statistics. Finally, we propose a lightweight neuron pruning strategy that corrects a portion of biased errors, offering causal evidence that fine-grained digit biases can directly cause numerical hallucination. These findings highlight how low-level statistical priors from pretraining data can affect high-level behavior in LLMs.

Limitations and future work. While our work reveals a compelling link between digit bias in the pretraining corpus and numerical hallucinations in LLMs, it has several limitations. Most importantly, although we observe a strong correlation between digit frequencies in the training data and biased generation behavior, we do not claim a causal relationship; establishing causality would require controlled interventions during training, which we leave for future work. In addition, our experiments

are conducted on relatively small-scale decoder-only LLMs (7B–9B parameters) with standard MLP architectures. Whether similar patterns of bias and internal activation dynamics persist in larger models or those employing Mixture-of-Experts (MoE) remains an open question. Finally, our pruning strategy offers causal insights into digit-selective neurons but is coarse, potentially disrupting correct generations and limiting accuracy gains. We believe that more fine-grained or adaptive debiasing methods could yield stronger performance improvements. However, developing such techniques is beyond the scope of this work and is left for future investigation.

7 Acknowledgements

This work is supported by a Start-up Grant from Nanyang Technological University and jointly funded by the Singapore Ministry of Education (MOE) under a Tier-1 research grant.

References

[1] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *ArXiv*, abs/2402.00157, 2024.

[2] Simon Frieder, Julius Berner, Philipp Petersen, and Thomas Lukasiewicz. Large language models for mathematicians. *ArXiv*, abs/2312.04556, 2023.

[3] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *ArXiv*, abs/2501.04519, 2025.

[4] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *ArXiv*, abs/2310.10631, 2023.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916, 2022.

[7] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[8] Safal Shrestha, Minwu Kim, and Keith Ross. Mathematical reasoning in large language models: Assessing logical and arithmetic errors across wide numerical ranges. *ArXiv*, abs/2502.08680, 2025.

[9] Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *ArXiv*, abs/2410.05229, 2024.

[10] Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *ArXiv*, abs/2402.19255, 2024.

[11] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv*, abs/2311.05232, 2023.

[12] Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. Sources of hallucination by large language models on inference tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[13] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models. *ArXiv*, abs/2309.01219, 2023.

[14] Katja Filippova. Controlled hallucinations:learning to generate faithfully from noisy data. In *Findings of EMNLP 2020*, 2020.

[15] Theodore P. Hill. A statistical derivation of the significant-digit law. *Statistical Science*, 10:354–363, 1995.

[16] Simon Newcomb. Note on the frequency of use of the different digits in natural numbers. *American Journal of Mathematics*, 4:39, 1881.

[17] F. Benford and I. Langmuir. *The Law of Anomalous Numbers*. Proceedings of the American Philosophical Society. American Philosophical Society, 1938.

[18] Mark J Nigrini. *Benford's Law: Applications for forensic accounting, auditing, and fraud detection*. John Wiley & Sons, 2012.

[19] Joseph Deckert, Mikhail Myagkov, and Peter C. Ordeshook. Benford's law and the detection of election fraud. *Political Analysis*, 19(3):245–268, 2011.

[20] AllenAI. Olmo-mix-1124 dataset. `https://huggingface.co/datasets/allenai/olmo-mix-1124`, 2024.

[21] Nostalgebraist. Interpreting gpt: The logit lens. LessWrong, 2020.

[22] Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *ArXiv*, abs/2502.06453, 2025.

[23] Aaditya K. Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *ArXiv*, abs/2402.14903, 2024.

[24] Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. *ArXiv*, abs/2405.17399, 2024.

[25] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *ArXiv*, abs/2202.07206, 2022.

[26] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36:3580–3599, 2023.

[27] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[28] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[29] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *ArXiv*, abs/1904.01557, 2019.

[30] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cris tian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aur'elien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[31] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[32] Gemma Team Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stańczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Boxi Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Christoper A. Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozi'nska, D. Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost R. van Amersfoort, Josh Gordon, Josh Lipschultz, Joshua Newlan, Junsong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, L. Sifre, Lena Heuermann, Leti cia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Gorner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nen shad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Peng chong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Marie Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Se bastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomás Kociský, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Daniel Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118, 2024.

[33] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2024.

[34] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *ArXiv*, abs/2310.06825, 2023.

[35] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[36] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[37] Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *ArXiv*, abs/2410.21272, 2024.

[38] Zeping Yu and Sophia Ananiadou. Interpreting arithmetic mechanism in large language models through comparative neuron analysis. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

[39] Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. *ArXiv*, abs/2409.01659, 2024.

[40] Mor Geva, R. Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *ArXiv*, abs/2012.14913, 2020.

[41] Anthropic. On the biology of a large language model. `https://transformer-circuits.pub/2025/attribution-graphs/biology.html`, March 2025. Accessed: May 7, 2025.

[42] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[43] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly summarize the key contributions and scope of the paper, aligning with the detailed methodology and results presented in later sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly addresses limitations, discussing potential constraints, ensuring transparency about the work's boundaries.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not have any theoretical result, but do provide a complete mathematics proof for our method in appendix.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient methodological details, code and data to allow reproduction of key results, supporting the validity of its claims.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper clearly states open access to data and code in supplemental material, along with sufficient instructions for replication.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper clearly specifies all key experimental details needed to interpret the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper appropriately reports error bars and statistical significance where relevant to support the experimental claims.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the relvant information in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research adheres to the NeurIPS Code of Ethics by ensuring transparency in methodology, using publicly available datasets, and avoiding any biased or harmful applications.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators and original owners of all assets (e.g., code, data, models) used in the paper are properly credited with appropriate citations, and the license and terms of use are explicitly mentioned and respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All new assets introduced in the paper are thoroughly documented with detailed descriptions, usage instructions, and metadata.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

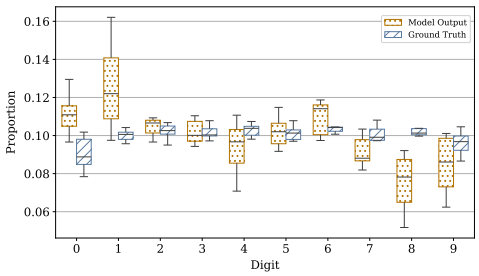
Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

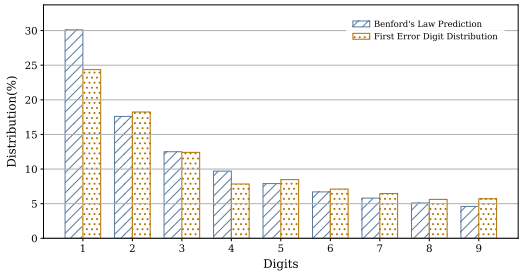
A Additional Experiment Results

A.1 Result on Digit Bias Benchmark of across Models

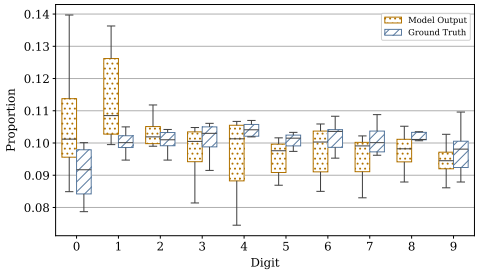
We plot the digit distribution in generated outputs across all models on the Digit Bias Benchmark as shwon in Figure 9. All models show significant overgeneration of small digits, with digit 1 being especially dominant. This consistent trend highlights the generality of digit-level generation bias across open-source LLMs.



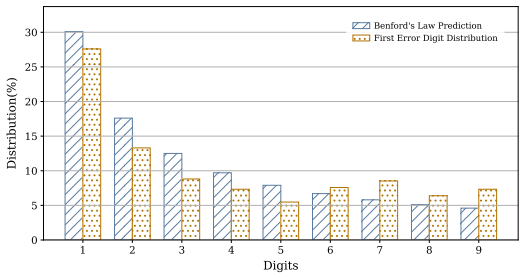
llama2-7b-chat-hf



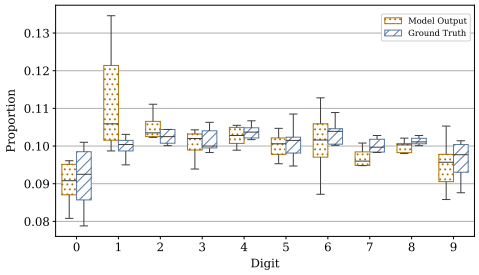
llama2-7b-chat-hf



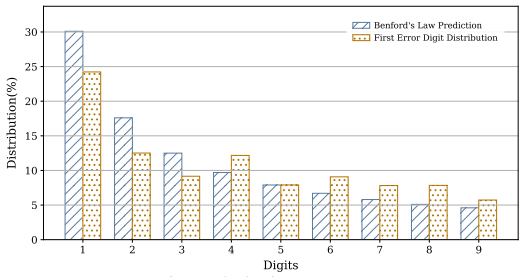
gemma2-9b-it



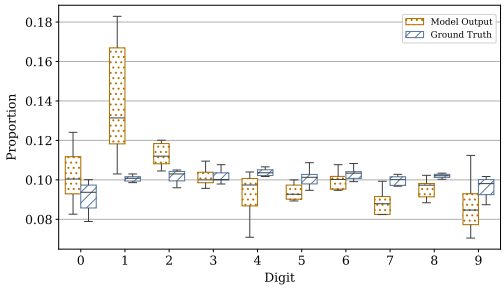
gemma2-9b-it



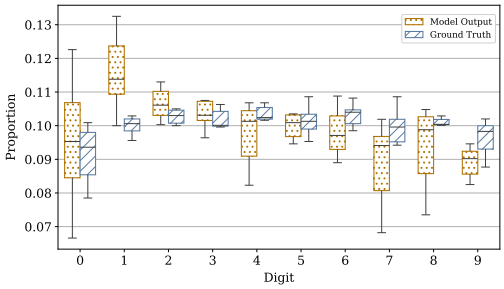
Qwen2.5-7b-Instruct



Qwen2.5-7b-Instruct



OLMo2-2-1124-7B-Instruct



llama3.1-8b-Instruct

Figure 9: Digit generation bias across models on the Digit Bias Benchmark. Since OLMo and LLaMA3.18B employ multi-digit tokenization schemes, first-error-digit distribution analyses are not applicable and thus omitted for these models.

A.2 Digit Selectivity

Figure 10 visualizes the selectivity of FFN neurons across all digit tokens. The distributions are clearly skewed, indicating that more neurons are specialized toward frequent digits like '1', suggesting an uneven allocation of model capacity that may underlie observed generation biases.

Figure 10: FFN neuron-level selectivity distributions for digit '1' (blue) versus all other digits (orange) across four open-source LLMs: (a) LLaMA2-7B, (b) Qwen2.5-7B, (c) Gemma2-9B, and (d) Mistral-7B. For each digit, the top 1000 neurons with the highest selectivity scores are selected independently.

A.3 Additional Pruning Examples and Results

Figure 11 presents additional examples of samples corrected after pruning. Many of the original errors occur in surprisingly simple scenarios, such as repeated or trivial arithmetic, highlighting how digit bias can subtly but significantly affect digit generation and lead to failure even in straightforward cases.

Comparing Model Behavior With and Without Pruning

Prompt 1: Solve 0 = -90*w – 12 – 708 for w.
Generation (no pruning): To solve for w, we need to isolate the variable w on one side of the equation. First, we can simplify the equation by combining like terms: $$0 = -90w - 12 - 708$$. $$0 = -102w - 720$$. Next, we can divide both sides of the equation by -102 to get: $$w = -720/102$$. ✗
Generation (pruning): To solve for w, we need to isolate the variable w on one side of the equation. First, we can simplify the equation by combining like terms: $$0 = -90w - 12 - 708$$. $$0 = -90w - 720$$. Next, we can divide both sides of the equation by -90 to get: $$w = frac{720}{90}$$. ✓

Prompt 2: Solve 38*w = 75 - 75 - 1026 for w.
Generation (no pruning): Let the equation given is: `\[ 38w = 75 - 75 - 1126 \]`... ✗
Generation (pruning): Let the equation given is: `\[ 38w = 75 - 75 - 1026 \]`... ✓

Prompt 3: Solve 78 + 491 + 817 = 28*x + 266 for x.
Generation (no pruning): ... 4. **Perform the division:** `\[x = 40 \]` So, the solution for x is is 41. ✗
Generation (pruning): ... 4. **Perform the division:** `\[x = 40 \]` So, the solution for x is is 40. ✓

Prompt 4: Solve 70 = -1160*m - 6852 - 3518 for m.
Generation (no pruning): To solve the equation $71 = -1161m - 6852 - 3518$ for m ... ✗
Generation (pruning): To solve the equation $71 = -1160m - 6852 - 3518$ for m ... ✓

Prompt 5: Solve 38*a - 1664 = -36*a - 69*a + 39*a for a.
Generation (no pruning): ... Simplify the left side: `\[ 104a - 1664 = 0\]` 3. **Isolate the term with a:** Add 1664 to both sides to move the constant to the right side: `\[104a = 1164\]`... ✗
Generation (pruning): ... Simplify the left side: `\[ 104a - 1664 = 0\]` 3. **Isolate the term with a:** Add 1664 to both sides to move the constant to the right side: `\[104a = 1664\]`...✓

Prompt 6: What is the next term in -106527, -106453, -106379?
Generation (no pruning): The pattern is adding 104 to each term. * -106527 + 104 = -106453 * -106453 + 104 = -106379 * -106379 + 104 = **-106275** Therefore, the next term is **-106275**. ✗
Generation (pruning): The pattern is adding 74 to the previous term. * -106527 + 74 = -106453 * -106453 + 74 = -106379 * -106379 + 74 = **-106305** Therefore, the next term is **-106305**. ✓

Figure 11: Examples showing model responses before and after pruning the top 0.01% of neurons biased toward digit 1. In each cases, pruning corrects an originally erroneous sample by rectifying an intermediate step, demonstrating a causal relationship between digit bias and numerical hallucination.

B Additional Implementation Details

B.1 Experiment Setup

To ensure the accuracy and reproducibility of all results, we employed greedy decoding for generation. Additionally, to achieve fully accurate statistical outcomes, we utilized the DeepSeek API to extract the answers model generated for each sample, thereby preventing potential omissions in script-based statistics due to variations in the position of answers within the responses.

B.2 Tokenization Method

LLMs divide numbers into segmented tokens rather than representing the entire number as a single token. Different LLMs employ various tokenization methods, including *one-digit tokenizers* and *multi-digit tokenizers*. Benford's Law states that in many real-life sets of numerical data, the leading digit is likely to be small. In other words, regardless of the type of tokenizer, the distribution of number tokens in pretraining data is likely to be skewed. Therefore, in this paper, we use six models with two different tokenizers to investigate this phenomenon of digit bias as shwon in Figure 12. LLaMA2-7B, Mistral-7B, Qwen2.5-7B, and Gemma2-9B employ single-digit tokenizers, whereas LLaMA3.1-8B and OLMo2-7B utilize multi-digit tokenization schemes.

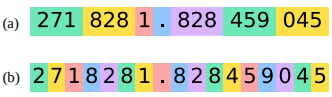


Figure 12: (a) multi-digit tokenizer. (b) single-digit tokenizer.

B.3 Prompt Templates

We provide the exact prompt templates used for Identification task and Digit Bias Benchmark in table 4 and table 5.

Table 4: Prompt Templates Used in Identification Task

Identification Prompt Template
1. What is the result when the last term of the sequence is multiplied by two? [...] 2. What is the outcome when the final term of the sequence is doubled? [...] 3. What is the product of the sequence's last term and two? [...] 4. What is the result of multiplying the sequence's last term by two? [...]

C Use of existing assets

C.1 Models

Table 6: The list of models used in this work.

Model	**Accessed via**	**License**
Qwen2.5-7B-Instruct	Link	Apache license 2.0
gemma-2-9b-it	Link	Gemma Terms of Use
Mistral-7B-Instruct-v0.3	Link	Apache license 2.0
Llama-3.1-8B-Instruct	Link	Llama 3.1 Community License Agreement
Llama-2-7b-chat-hf	Link	Llama 2 Community License Agreement
OLMo-2-1124-7B-Instruct	Link	Apache license 2.0

Table 5: Prompt Templates Used in Digit Bias Benchmark

Addition	**Division**
{p} + {q} {p}+{q} Work out {p} + {q}. Add {p} and {q}. Put together {p} and {q}. Sum {p} and {q}. Total of {p} and {q}. Add together {p} and {q}. What is {p} plus {q}? Calculate {p} + {q}. What is {p} + {q}?	Calculate the division of {q} by {p}. Divide {q} by {p}. What is the quotient of {q} divided by {p}? What is {q} divided by {p}? Find {q} divided by {p}. Compute {q} ÷ {p}. Solve {q} divided by {p}.
Subtraction	**Multiplication**
{p} - {q} Work out {p} - {q}. What is {p} minus {q}? What is {p} take away {q}? What is {q} less than {p}? Subtract {q} from {p}. Calculate {p} - {q}. What is {p} - {q}?	{p} × {q} Calculate {p} × {q}. Work out {p} × {q}. Multiply {p} and {q}. Product of {p} and {q}. What is the product of {p} and {q}? {p} times {q} What is {p} times {q}?

Evaluate
Let {c(x) = f(x)}. What is {c(a)}?
Let {c(x) = f(x)}. Determine {c(a)}.
Let {c(x) = f(x)}. Give {c(a)}.
Let {c(x) = f(x)}. Calculate {c(a)}.
Nearest Integer Root
What is the {num-th} root of {p} to the nearest integer?
What is {p} to the power of {1/num-th}, to the nearest integer?
Linear_1d
Solve {eqution} for {r}.
Sequence Next Term
What comes next: {sequence}?
What is next in {sequence}?
What is the next term in {sequence}?

C.2 Dataset

Table 7: The list of datasets used in this work.

Dataset	**Accessed via**	**License**
olmo-mix-1124	Link	Open Data Commons License Attribution
mathematics_dataset	Link	Apache license 2.0

D Compute statement

All experiments presented in this paper were run on a cluster of four NVIDIA GeForce RTX 3090 GPUs with 24GB of memory and using a single 24GB memory GPU. Each model requires an average of 50 hours to complete a full run across the entire benchmark.