Probing for Arithmetic Errors in Language Models

Yucheng Sun* **Alessandro Stolfo*** **Mrinmaya Sachan**
ETH Zürich
{yucsun, stolfoa}@ethz.ch

Abstract

We investigate whether internal activations in language models can be used to detect arithmetic errors. Starting with a controlled setting of 3-digit addition, we show that simple probes can accurately decode both the model's predicted output and the correct answer from hidden states, regardless of whether the model's output is correct. Building on this, we train lightweight error detectors that predict model correctness with over 90% accuracy. We then extend our analysis to structured chain-of-thought traces on addition-only GSM8K problems and find that probes trained on simple arithmetic generalize well to this more complex setting, revealing consistent internal representations. Finally, we demonstrate that these probes can guide selective re-prompting of erroneous reasoning steps, improving task accuracy with minimal disruption to correct outputs. Our findings suggest that arithmetic errors can be anticipated from internal activations alone, and that simple probes offer a viable path toward lightweight model self-correction.[1]

1 Introduction

Large language models have recently shown strong performance on mathematical problem solving and reasoning (OpenAI, 2024; Team, 2024a; Shao et al., 2024; DeepSeek-AI, 2025, *inter alia*), making mathematical ability an increasingly central axis for benchmarking model capabilities (Glazer et al., 2024). This surge in capability has sparked growing interest in understanding how these models internally process numerical information and execute arithmetic reasoning (Nanda et al., 2023; Stolfo et al., 2023a; Hanna et al., 2023; Zhong et al., 2023; Zhou et al., 2024b; Nikankin et al., 2025; Lindsey et al., 2025).

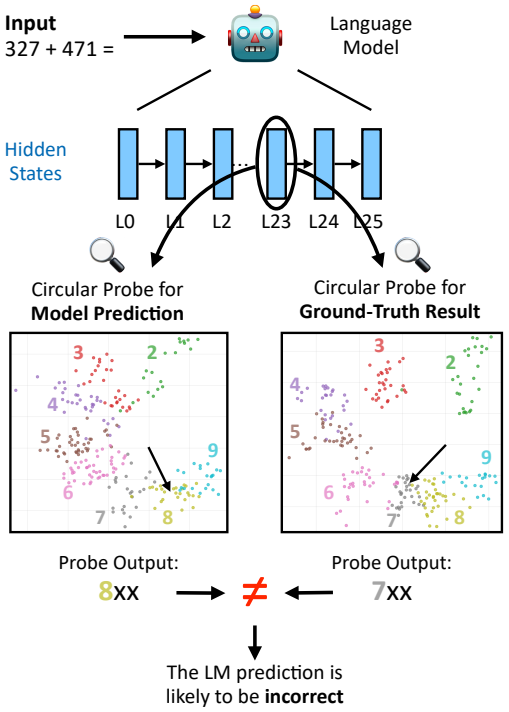

Figure 1: **Detecting Arithmetic Errors from Hidden States.** We investigate whether internal activations in a language model reveal when its arithmetic predictions are incorrect. We train simple probes to decode both the model's output and the correct answer. The probes' output can serve as a reliable signal of model error.

In particular, recent studies have investigated how pre-trained language models represent numerical quantities (Levy and Geva, 2025; Kantamneni and Tegmark, 2025; Zhu et al., 2025). However, despite increasing insight into the structure of these representations, their practical use for improving model behavior remains limited. At the same time, analyzing hidden representations has proven useful for identifying model failures and hallucinations in other domains (Kadavath et al., 2022; Azaria and Mitchell, 2023; Yuksekgonul et al., 2024; Chen et al., 2024; Orgad et al., 2025).

Motivated by these findings, we investigate whether internal model activations can be lever-

*Equal contribution.
[1]Our code and data are available at https://github.com/yuchen9-5un/arithmetic-error-probing

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 8111–8128
November 4-9, 2025 ©2025 Association for Computational Linguistics

aged to detect arithmetic errors in language models. We begin with a simple setting: the model is prompted with 3-digit addition queries (e.g., "327+471"), and we analyze its hidden activations using simple probes inspired by representational and circuit-level findings. We show that not only it is possible to recover the language model's prediction, but also to predict the ground-truth result from the model's hidden states, independently of whether the model's output is correct. Motivated by this observation, we adapt the probing method to predict whether the model's final answer will be correct (Figure 1), achieving over 90% accuracy in predicting model correctness on a balanced dataset.

We then extend this analysis to a more complex setting, where the model is asked to solve math word problems only requiring addition (Cobbe et al., 2021) using a structured chain-of-thought (CoT) format (Wei et al., 2022), in which intermediate steps are expressed as equations (e.g., <a+b=c>). Remarkably, we find that the same probes trained on simple arithmetic queries can be applied directly to this setting, maintaining over 80% accuracy in detecting whether the model is producing correct intermediate results.

Finally, we explore the practical benefits of these probes by using them as weak oracles to identify potentially erroneous steps within the model's reasoning traces. We design a re-prompting mechanism that selectively revisits flagged steps, leading to an overall correction of up to 11% of the wrong reasoning steps, without compromising any of the correct ones. These results suggest that internal representations of numerical quantities are robust across contexts, and can be exploited to detect and correct model errors with lightweight probing methods.

2 Probing for Numerical Representations

We aim to understand how information about arithmetic operations is internally represented in a language model. To this end, we consider a controlled setting in which the model is prompted with simple 3-digit addition queries and analyze its hidden activations at the point just before it produces an output. Specifically, we ask whether numerical quantities are encoded in a form that can be recovered by lightweight probes. We begin with a representation analysis (§2.1), then introduce a set of probing methods to decode this information (§2.2).

2.1 Representation Analysis

Prior work has shown that language models represent periodic concepts such as the days of the week in circular forms (Engels et al., 2025). In the mathematical domain, similar findings suggest that numerical quantities can be encoded in circular (Levy and Geva, 2025), linear (Zhu et al., 2025), or hybrid helix-like (Kantamneni and Tegmark, 2025) geometries. To better understand how numerical information is encoded in our setting, we conduct a representation analysis on the instruction-tuned version of Gemma 2 2B (Team, 2024b), focusing on its behavior when solving 3-digit addition problems (e.g., "327+471").[2]

We synthetically generate arithmetic queries by sampling 800 pairs of operands $(a, b) \in \{100, \ldots, 999\}^2$, such that $a + b < 1000$. Then, we feed the model the prompt "a+b=", and examine the model's internal activations at the position of the equals sign ("="). This token position immediately precedes the model's output and is expected to carry information about the computed result (Stolfo et al., 2023a; Nikankin et al., 2025).

We apply principal component analysis (PCA) to the residual stream activations at this position across all layers of the model, highlighting in different colors the hidden states associated with different hundreds digits in the ground-truth result. We observe that as depth increases, the representations of individual digits become increasingly structured and separable. In particular, deeper layers exhibit clearer clustering by digit number, with the top principal components revealing a circular layout similar to those observed in prior studies.

To illustrate this progression, we selected two representative layers for visualization. As shown in Figure 2, layer 15 shows no clear digit separation, whereas layer 25 exhibits both stronger clustering and a visible circular layout. This progression supports the hypothesis that the model gradually builds a more geometric and abstract encoding of numerical concepts over its depth, and is consistent with what previous work observed.

[2]The Gemma tokenizer splits numbers into individual digits. For our experiments, we focus on predicting the first digit of the result (i.e., the hundreds place). This choice captures the majority of the model's errors in this setting (evidence for this in Appendix F). Results for a model with a different tokenization scheme (Phi-3), which confirm our findings, are reported in Appendix I.

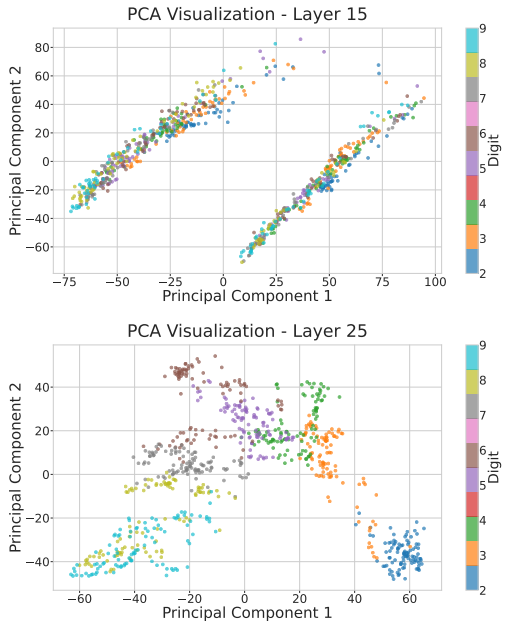

Figure 2: **PCA of Residual Stream Activations.** PC projections colored by the hundreds digit of the ground-truth result. Representations become more structured with depth, showing clear digit clusters and a circular layout in deeper layers.

2.2 Probing Methods

To quantitatively assess how numerical information is encoded in the model's internal representations, we design a set of lightweight probing methods. Building on the insights from our representational analysis, which indicated a circular structure in digit encodings at deeper layers, we treat circular probing as a natural baseline. However, we also explore alternative probing approaches to evaluate different ways of extracting numerical information from the same representations.

As in the previous section, we focus on the residual stream activations at the equals sign. Each probe is trained and evaluated independently at each layer of the model, allowing us to track how the accessibility of digit-level information evolves across the model's depth. We describe the implementation of each probe below.

Circular Probe. Denote the residual-stream hidden states of a language model at layer l by $\mathbf{x}_l \in \mathbb{R}^{d_{\text{model}}}$.[3] Let $y \in \{0, \ldots, 9\}$ be the label that the probe is tasked to predict. We design a circular probe that projects the model's hidden states onto a plane, then uses the angle that such projection point forms with the origin as its output. More formally, let the probe weights be two vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{d_{\text{model}}}$. We compute the probe's output $\hat{y}$ as

$$\theta = \text{atan2}\left(\mathbf{w}_1^\top \mathbf{x}_l, \mathbf{w}_2^\top \mathbf{x}_l\right) \in [0, 2\pi), \quad (1)$$

$$\hat{y} = \theta \cdot \frac{10}{2\pi}. \quad (2)$$

The probe is trained using smooth ℓ_1 loss (Girshick, 2015) between the prediction $\hat{y}$ and the label y. The probe is optimized using the AdamW optimizer (Loshchilov and Hutter, 2019).

Linear Probe. Motivated by prior work showing that numerical quantities can be linearly decoded from language model activations (Zhu et al., 2025; Kantamneni and Tegmark, 2025), we test whether any arithmetic information in our setting can be recovered by a simple linear probe:

$$\hat{y} = \mathbf{w}^\top \mathbf{x}_l + b, \quad (3)$$

where $\mathbf{w} \in \mathbb{R}^{d_{\text{model}}}$ and $b \in \mathbb{R}$ are the probe's parameters. The probe is trained using ℓ_2 regularization.[4] At inference time, the probe's output is rounded to the nearest integer to produce a discrete prediction.

Multi-Class Logistic Regression. We also consider a variant of the linear probe that treats digit prediction as a multi-class classification problem. Specifically, we associate a distinct weight vector $\mathbf{w}_i$ with each digit $i \in \{0, \ldots, 9\}$, and compute the probe's output as the digit corresponding to the maximum logit:

$$\hat{y} = \arg\max_i \left(\mathbf{w}_i^\top \mathbf{x}_l\right). \quad (4)$$

The probe is trained using cross-entropy loss and optimized with the Adam optimizer (Kingma and Ba, 2017).

MLP Probe. Finally, we experiment with a more expressive probe based on a multi-layer perceptron (MLP). The MLP consists of a single hidden layer with ReLU activation and a hidden dimensionality of 512. As in the logistic probe, the output layer produces 10 logits corresponding to the digit $i \in \{0, \ldots, 9\}$. The prediction is given by:

$$\hat{y} = \arg\max_i \left(\mathbf{W}_2^\top \text{ReLU}\left(\mathbf{W}_1^\top \mathbf{x}_l + \mathbf{b}_1\right) + \mathbf{b}_2\right),$$

[3] By "residual stream," we indicate per-token hidden state with dimensionality d_{model} consisting of the sum of all previous component outputs (Elhage et al., 2021).

[4] Following Zhu et al. (2025), we use regularization weight $\lambda = 0.1$.

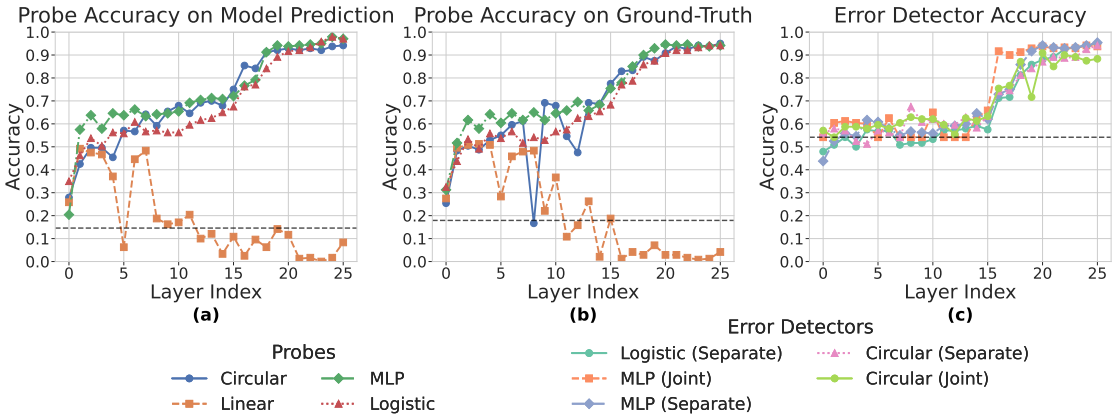

Figure 3: **Probing 3-Digit Arithmetic Queries.** (a) Probes recover the model's output with high accuracy in deeper layers; linear probes perform poorly. (b) Ground-truth digits are similarly decodable, suggesting correct answers are often internally represented. (c) Error detectors show that model correctness can be inferred from hidden states. The dashed lines indicate the accuracy of the majority class baseline.

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times 512}$, $\mathbf{W}_2 \in \mathbb{R}^{512 \times 10}$, $\mathbf{b}_1 \in \mathbb{R}^{512}$, and $\mathbf{b}_2 \in \mathbb{R}^{10}$ are the probe's parameters. This probe is trained using cross-entropy loss and optimized with Adam.

3 Probing 3-Digit Arithmetic Queries

We begin our analysis in the arithmetic setting described in §2, where the model is prompted to solve three-digit addition problems (e.g., "652+185"). We use a few-shot prompt containing two exemplars of addition (data details are provided in Appendix A). We organize our analysis in three stages: first, we test whether the probes can recover the model's predicted output (§3.1); second, we examine whether they can recover the ground-truth result (§3.2); and third, we test whether combining the two signals allows us to predict whether the model is correct on a given query (§3.3).

3.1 Can Probes Predict the Model's Output?

In this first experiment, we train probes to predict the first digit of the model's output, i.e., the digit that the model intends to produce as the result of the addition. We generate a dataset of queries of the form "a+b", where a and b are three-digit integers. For each query, we record the model's prediction and evaluate whether it matches the correct result. We construct a dataset with 800 queries, balanced across both output digits and model correctness, and split it into 70% training and 30% evaluation sets. Each probe is trained for 10,000 epochs. We apply and evaluate each probe independently across all 26 layers of Gemma 2B IT.

The results are shown in Figure 3a. The linear probe fails to recover meaningful information from the model's hidden states, maintaining low accuracy across all layers. In contrast, the circular, logistic, and MLP probes achieve significantly higher performance, with accuracy progressively improving across layers and plateauing around 92% in the final 4-5 layers.

Two key observations emerge. First, the accuracy gains across layers align with prior work showing that the final MLP blocks in transformer models are responsible for computing arithmetic outputs (Stolfo et al., 2023a; Nikankin et al., 2025). Second, despite its simplicity and small number of parameters, the circular probe performs on par with both the MLP and logistic probes. This supports the hypothesis–grounded in our representational analysis (§2)–that digit information is encoded in a circular manner, and that a probe matched to this geometry is sufficient to extract it.

3.2 Can Probes Recover the Ground-Truth Answer?

We next test whether probes can recover the correct answer to an arithmetic query, even when the model's own prediction is incorrect. Concretely, we train probes to predict the first digit of the ground-truth sum, using the same dataset and training procedure as in §3.1. The only difference is that the probe labels now correspond to the true answer rather than the model's prediction.

Accuracy results across all layers are shown in Figure 3b. Surprisingly, the trends follow those observed in the previous experiment: accuracy increases steadily with layer depth, and the MLP,

logistic, and circular probes all reach performance levels comparable to those observed when predicting the model's output (>90%). This suggests that the correct result is encoded in the model's internal representations and is accessible to simple probes, even when the model ultimately generates an incorrect answer. This raises a natural question: are probes (1) genuinely recovering a representation of the correct answer from the model's hidden states (despite the model's failure to output it), or are they (2) simply learning to perform the arithmetic themselves during training, by extracting operand information and learning to compute the result?

To disentangle these hypotheses, we conduct a follow-up experiment. We train MLP probes at an early layer (layer 5) to decode the input operands, both as complete numbers and as individual digits. If hypothesis (2) were correct (i.e., the probes were solving the arithmetic task themselves), then we would expect their performance on ground-truth prediction to match their performance on operand prediction. In other words, if the probes have access to the operands and are capable of computing the result, they should be able to predict the correct answer whenever the operand representations are available.

The results, summarized in Table 1, show that operand information is indeed recoverable at early layers, with probes achieving nearly perfect accuracy. However, as observed in Figure 3, the probes can only predict the ground-truth result with high accuracy in deeper layers. This discrepancy suggests that the probes are not simply performing the arithmetic task themselves, lending support to hypothesis (1) over (2). At the same time, the evidence does not fully justify the stronger claim that probes recover a clean representation of the correct answer. Instead, a more plausible interpretation is that probes learn to refine partial, noisy information about the result that is already present in the residual stream, and that the model itself may fail to fully decode through its output head. This interpretation is consistent with recent findings showing that language models often encode correct answers internally even when their output is incorrect (Orgad et al., 2025; Gekhman et al., 2025).

3.3 Can Probes Predict Model Correctness?

Having established that both the model's predicted output and the ground-truth answer can be decoded from its internal representations, we now ask whether this information can be combined to predict when the model is likely to make a mistake. In other words, can probes anticipate whether the model's answer will be correct or incorrect?

Operand Component	Accuracy
Operand #1 - Hundreds	0.9500
Operand #1 - Tens	0.8833
Operand #1 - Ones	0.9500
Operand #2 - Hundreds	0.9833
Operand #2 - Tens	0.9958
Operand #2 - Ones	1.0000

Table 1: **Operand Decoding Accuracy at Layer 5.** MLP probes can reliably recover operand values from early-layer activations, indicating that input numbers are explicitly represented in the model's residual stream.

To test this, we adapt our probing setup to a binary classification task: predicting whether the first digit of the model's output matches the ground-truth result. We experiment with five probing-based error detectors:

1. **Separate Circular Probes.** Two circular probes are trained independently–one to predict the model's output (as in Section 4.1), and one to predict the ground-truth answer (Section 4.2). The error predictor simply returns 1 if the two predictions disagree.

2. **Joint Circular Probe.** Two circular probes are trained jointly. The angular difference between their raw predictions is passed through a sigmoid function, and the resulting scalar is trained with binary cross-entropy to predict correctness.

3. **Separate MLP Probes.** Similar to the circular setup above, but using two independently trained MLP probes.

4. **Single MLP Classifier.** A single MLP is trained directly to predict correctness as a binary classification task.

5. **Separate Logistic Probes.** Two logistic probes are trained independently, and their predictions are compared via the same disagreement rule.

All probes are trained on the same data used in prior experiments, with the only change being the training label: whether the first digit of the model's output is correct. Additional implementation details for each method are provided in Appendix C.

The results are shown in Figure 3c. As expected, all detectors perform near chance level in early

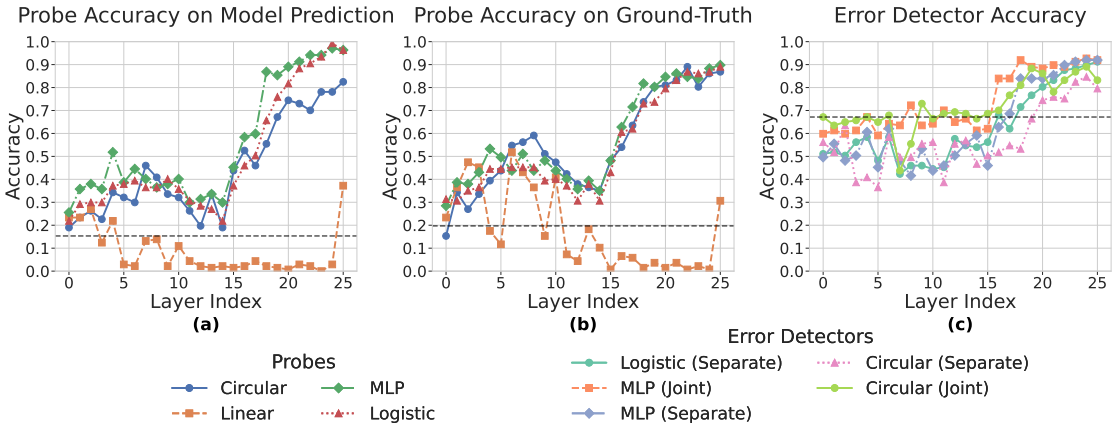

Figure 4: **Probing Structured Chain-of-Thought Reasoning.** (a) Probes recover the model's predicted digit with increasing accuracy across layers; non-linear probes reach over 90% in the final layers. (b) Ground-truth digits remain decodable, though slightly less accurately than in the pure arithmetic setting, likely due to added linguistic context. (c) Error detectors achieve 80-90% accuracy in later layers. The dashed lines indicate the accuracy of the majority class baseline.

layers but improve substantially in the later layers of the model. Most approaches reach accuracies above 90%, with some approaching 95%, far above the 50% majority baseline of the balanced dataset. Overall, these results demonstrate that probes can effectively detect model errors based solely on internal representations. The fact that high-accuracy error detection can be achieved using simple probe architectures reinforces the finding that LLMs encode useful information about both their predictions and the correct answers, even when the two diverge.

4 Probing Structured Chain-of-Thought Reasoning

So far, we have shown that simple probes can effectively recover both model predictions and ground-truth results in a direct arithmetic setting, where the language model is prompted with isolated addition queries. We now move to a more challenging and practically relevant scenario: arithmetic reasoning embedded within chain-of-thought (CoT) traces (Wei et al., 2022). Specifically, we focus on addition-only problems from the GSM8K dataset (Cobbe et al., 2021), prompting Gemma 2 2B IT to solve them using a structured CoT intermediate reasoning, with each intermediate step formatted as <a+b=c>.

To construct a suitable dataset, we follow the abstraction approach used in prior work (Patel et al., 2021; Stolfo et al., 2023b; Opedal et al., 2024; Mirzadeh et al., 2025). We first filter out problems that involve operations beyond addition. Then we augment this set of problems by abstracting away surface-level text by converting each question into a template form where numerical values are replaced by symbolic placeholders (e.g., x_1, x_2, ...), using Claude 3.7 Sonnet.[5] We then generate concrete problem instances by sampling new operand values and substituting them into the templates.We prompt the model to format each intermediate computation step as <a+b=c>. For each problem, we store the model's full reasoning trace and extract a set of intermediate computation steps (i.e., the input and all prior steps up to a given index). The final dataset contains 685 problem steps, roughly balanced in terms of model correctness and hundred-digit. We divide it into a 80/20% train-test split. Full dataset construction details are in Appendix D.

As in the previous experiments, we train probes on the residual stream activations at the equals-sign token (=) of each CoT step. We replicate our previous structure: first probing for the model's internal prediction (§4.1), then the ground-truth result (§4.2), and predicting model correctness (§4.3). We then evaluate whether probes trained on simple 3-digit queries generalize to the CoT setting (§4.4).

4.1 Can Probes Predict the Model's Output?

We begin by testing whether the model's internal representations at each CoT step encode the output it is about to produce. Accuracy results are shown in Figure 4a. As in the direct arithmetic setting, we observe that the linear probe performs poorly, while the MLP, logistic, and circular probes achieve

[5] https://www.anthropic.com/claude/sonnet

strong performance in deeper layers. In this CoT setting, accuracy improves sharply between layers 15 and 20, with the MLP and logistic probes reaching approximately 92% accuracy at the top layers. The circular probe performs slightly worse, plateauing at around 85%.

4.2 Can Probes Recover the Ground-Truth Answer?

We now turn to the task of predicting the ground-truth result of each intermediate step in the CoT reasoning chain. As before, we train probes to recover the correct answer from the residual stream activation at the equals-sign token. Results are shown in Figure 4b. Consistent with our earlier findings, probe accuracy increases with model depth for all probe types except the linear one. The strongest probes (MLP, logistic, and circular) plateau in the final 3-4 layers, indicating that the model progressively builds more explicit representations of the correct answer over its depth.

However, compared to the pure arithmetic setting (§3.2), we observe a modest drop in overall accuracy, with performance ranging between 80% and 90% in the best layers. We attribute this to the increased complexity of the CoT setting: the language model's input includes not only arithmetic expressions, but also natural language context describing the math word problem. This additional contextual information likely introduces noise that affects the clarity with which numerical information is encoded in the hidden states, making it slightly harder for the probes to recover the exact result.

4.3 Can Probes Predict Model Correctness?

Finally, we evaluate whether probes can predict whether the model's arithmetic predictions at intermediate CoT steps are correct. We adopt the same set of error detection strategies described in §3.3, applying them to the structured CoT setting.

The results are shown in Figure 4c. Although slightly noisier than in the pure arithmetic setting, the overall trend remains consistent. All error detectors perform at or near chance level until approximately layer 15, after which accuracy increases sharply and stabilizes between 80% and 90% in the final layers. These findings indicate that the ability of simple probes to detect arithmetic errors is not limited to isolated arithmetic queries. Instead, this capability transfers well to more realistic, structured settings involving chain-of-thought reasoning.

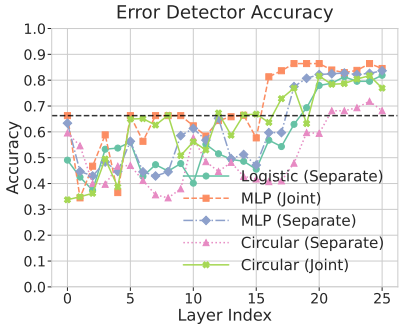

Figure 5: **Cross-Setting Error Detection.** Accuracy of probes trained on simple arithmetic queries evaluated on GSM8K problems. Probes generalize well to the structured CoT setting, reaching up to 85% accuracy and indicating consistent internal representations.

4.4 Do Probes Generalize Across Settings?

A natural question is whether probes trained in one setting (e.g., direct arithmetic queries) can generalize to another (e.g., arithmetic embedded in chain-of-thought reasoning). If the internal representations of numerical quantities are consistent across contexts, we would expect probes trained in one domain to retain predictive power when applied to the other. To evaluate this, we test the error detectors trained in the pure arithmetic setting on the structured CoT dataset, without any further training. Results are reported in Figure 5.

Remarkably, we find that these probes achieve nearly the same accuracy as those trained directly on the CoT data, with performance around 85%. This suggests that the language model encodes numerical information in a robust and consistent manner, independent of whether the arithmetic appears in isolation or within a reasoning chain. These findings point toward the potential for reusable probes that generalize across settings, offering a lightweight way to monitor model behavior in multi-step reasoning tasks.

5 Using Error Detectors for Self-Correction

So far, we have shown that simple probes can reliably identify when a language model is likely to make an arithmetic error. We now explore a practical application of these probes: using them to guide the model in revising erroneous reasoning steps during multi-step arithmetic problems in the GSM8K dataset.

All error detectors used in this section are trained

solely on data from the pure arithmetic setting, with supervision targeting only the hundreds digit of the result. As shown in §4.4, these detectors generalize well to reasoning steps in GSM8K.

For all experiments below, we use the best-performing configuration: a single MLP-based error detector applied to the residual stream at layer 25. When a reasoning step is flagged as likely incorrect, we append a follow-up prompt immediately after the model's output. This prompt takes the form of a brief corrective message (e.g., "That step looks suspicious. Let's re-do just this step:"), followed by the equation from the flagged step up to the equals sign (e.g., `<123+456=`). The message prompts the model to recompute the step without altering the rest of the chain. We experiment with several versions of the corrective message, ranging from neutral (e.g., "that step looks wrong") to stronger wording (e.g., "that's definitely wrong"). The full list of prompts is provided in Appendix E.

We evaluate this setup on a set of 685 intermediate steps from GSM8K, sampled to include both correct and incorrect predictions. Using full-number correctness as the evaluation criterion, the error detector flagged 178 true positives and 22 false positives. Table 2 reports the results across different prompting styles. We evaluate two metrics: (i) TP Correction, the proportion of model errors (true positives) that are corrected after re-prompting, and (ii) FP Preservation, the proportion of correct steps (false positives) that remain unchanged after re-prompting. We find that simple feedback prompts can correct up to 11.8% of model errors. At the same time, most prompts preserve all false positives, with a 100% preservation rate in most configurations.

These results demonstrate that error detectors trained on simple arithmetic data can serve as effective weak oracles for self-correction, enabling language models to revise specific reasoning steps with minimal risk of degrading correct ones.

Message	TP Correction	FP Preservation
suspicious	11.80%	100.00%
neutral	11.80%	100.00%
specific	10.11%	100.00%
stronger	8.99%	95.45%
detailed	6.18%	100.00%

Table 2: **Self-Correction via Re-Prompting.** Different re-prompting messages lead to varying correction rates, with up to 11.8% of flagged errors corrected and near-perfect preservation of correct answers.

6 Related Work

Arithmetic Capabilities in Language Models. Although several studies have highlighted the brittleness and inconsistency of language models' reasoning abilities (Razeghi et al., 2022; Stolfo et al., 2023b; Srivastava et al., 2024; Hong et al., 2024; Opedal et al., 2025; Mirzadeh et al., 2025), recent advancements have shown significant improvements in their performance on mathematical tasks (Lewkowycz et al., 2022; Azerbayev et al., 2024; Trinh et al., 2024; Shao et al., 2024). These gains have motivated a line of work aimed at enabling models to self-reflect or self-correct their reasoning (Weng et al., 2023; Madaan et al., 2023; Kim et al., 2023; Shinn et al., 2023; Zhou et al., 2024a; Pan et al., 2024). However, subsequent analysis has revealed that the improvements reported by some of these methods were partially due to the use of oracle labels during the correction process, rather than being fully model-internal (Huang et al., 2024). Our work aligns with these findings, demonstrating that even a weak oracle–here implemented as a simple probe–can effectively support model correction and improve arithmetic reasoning performance.

Interpretability of Arithmetic Reasoning. Several studies have investigated how language models encode and manipulate numerical information, both in small-scale transformers trained on synthetic tasks (Nanda et al., 2023; Zhong et al., 2023; Quirke and Barez, 2024; Ding et al., 2024; Maltoni and Ferrara, 2024), and in pre-trained language models (Hanna et al., 2023; Stolfo et al., 2023a; Zhang et al., 2024; Nikankin et al., 2025; Lindsey et al., 2025). A particular area of focus has been the structure of numerical representations in language models. Recent work has shown that such quantities may be encoded in circular, linear, or helical geometries (Levy and Geva, 2025; Kantamneni and Tegmark, 2025; Zhu et al., 2025), connecting to broader studies of non-linear representations in transformers (Yedidia, 2023a,b; Csordás et al., 2024; Shai et al., 2024; Engels et al., 2025).

7 Conclusion

We analyzed how language models encode arithmetic information and showed that simple probes can extract both predicted outputs and correct answers from internal activations. These probes can also anticipate errors, generalize to chain-of-

thought reasoning, and serve as weak oracles for self-correction. When used in a re-prompting setup, they improve accuracy with minimal risk of degrading correct outputs. These results are consistent with prior work showing that language models often encode the correct answer internally even when their output is incorrect (Orgad et al., 2025; Gekhman et al., 2025), and show that lightweight probing tools offer a practical path toward extracting and leveraging this latent knowledge.

Limitations

While our findings provide evidence that simple probes can detect arithmetic errors from internal activations, several limitations remain.

First, our analysis focuses exclusively on addition problems. In Appendix H.1, we extend the setup to include subtraction and observe consistent results. However, it remains an open question whether the same probing strategies are effective for other arithmetic operations such as multiplication, or division. Extending the methodology to these operators would help assess the robustness and generality of our approach.

Second, our experiments are primarily conducted on a single model, Gemma 2B IT. We additionally report consistent results on Phi-3 in Appendix I, but a broader evaluation across diverse architectures and scales would strengthen the generality of our conclusions.

Third, our chain-of-thought (CoT) analysis is limited to addition-only problems from the GSM8K dataset. While GSM8K provides a useful benchmark for arithmetic reasoning, future work should explore other datasets, including those with broader math coverage and more varied reasoning formats.

Finally, our CoT evaluation is conducted under a constrained setting where intermediate reasoning steps are explicitly formatted as structured equations (e.g., <a+b=c>). This structure simplifies probing and error detection. However, real-world model outputs are typically unstructured and expressed in natural language. An important direction for future work is to extend our approach to operate over general, unstructured CoT traces, with the goal of building lightweight tools that detect and correct reasoning errors in free-form outputs. Preliminary evidence in support of this extension is provided in Appendix H.3.

Acknowledgments

We would like to express our gratitude to Vilém Zouhar for useful discussions and comments on our work. AS acknowledges the support of armasuisse Science and Technology through a CYD Doctoral Fellowship.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. 2024. Recurrent neural networks learn to store and generate sequences using non-linear representations. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 248–262, Miami, Florida, US. Association for Computational Linguistics.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Preprint*, arXiv:2501.12948.

Xiaoman Delores Ding, Zifan Carl Guo, Eric J Michaud, Ziming Liu, and Max Tegmark. 2024. Survival of the fittest representation: A case study with modular addition. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. 2025. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*.

Zorik Gekhman, Eyal Ben David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. Inside-out: Hidden factual knowledge in LLMs. *Preprint*, arXiv:2503.15299.

Ross Girshick. 2015. Fast R-CNN. *Preprint*, arXiv:1504.08083.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, and 5 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in AI. *Preprint*, arXiv:2411.04872.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, and 7 others. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Pengfei Hong, Navonil Majumder, Deepanway Ghosal, Somak Aditya, Rada Mihalcea, and Soujanya Poria. 2024. Evaluating LLMs' mathematical and coding competency through ontology-guided interventions. *Preprint*, arXiv:2401.09395.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Subhash Kantamneni and Max Tegmark. 2025. Language models use trigonometry to do addition. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. 2023. Language models can solve computer tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Amit Arnold Levy and Mor Geva. 2025. Language models encode numbers using digit representations in base 10. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 385–395, Albuquerque, New Mexico. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Davide Maltoni and Matteo Ferrara. 2024. Arithmetic with language models: From memorization to computation. *Neural Networks*, 179:106550.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. 2025. MathGAP: Out-of-Distribution evaluation on problems with arbitrarily complex proofs. In *The Thirteenth International Conference on Learning Representations*.

Andreas Opedal, Alessandro Stolfo, Haruki Shirakami, Ying Jiao, Ryan Cotterell, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. 2024. Do language models exhibit the same cognitive biases in problem solving as human learners? In *Forty-first International Conference on Machine Learning*.

OpenAI. 2024. OpenAI o1 system card. *Preprint*, arXiv:2412.16720.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations*.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. 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. Association for Computational Linguistics.

Philip Quirke and Fazl Barez. 2024. Understanding addition in transformers. In *The Twelfth International Conference on Learning Representations*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Adam Shai, Paul M. Riechers, Lucas Teixeira, Alexander Gietelink Oldenziel, and Sarah Marzen. 2024. Transformers represent belief state geometry in their residual stream. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Saurabh Srivastava, Annarose M B, Anto P V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. 2024. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *Preprint*, arXiv:2402.19450.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023a. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052, Singapore. Association for Computational Linguistics.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schoelkopf, and Mrinmaya Sachan. 2023b. A causal framework to quantify the robustness of mathematical reasoning with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 545–561, Toronto, Canada. Association for Computational Linguistics.

Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Gemma Team. 2024b. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Trieu Trinh, Yuhuai Wu, Quoc Le, He He, and Luong Thang. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-Art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Adam Yedidia. 2023a. GPT-2's positional embedding matrix is a helix.

Adam Yedidia. 2023b. The positional embedding matrix and previous-token heads: How do they actually work?

Mert Yuksekgonul, Varun Chandrasekaran, Erik Jones, Suriya Gunasekar, Ranjita Naik, Hamid Palangi, Ece Kamar, and Besmira Nushi. 2024. Attention satisfies: A constraint-satisfaction lens on factual errors of language models. In *The Twelfth International Conference on Learning Representations*.

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. In *Forty-first International Conference on Machine Learning*.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. 2023. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2024a. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *The Twelfth International Conference on Learning Representations*.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. 2024b. Pre-trained large language models use fourier features to compute addition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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.

Vilém Zouhar. 2023. Ryanize bib.

A Pure Arithmetic Data Generation

To train and evaluate digit-level probes and error detectors in the pure arithmetic setting, we generate synthetic data using a two-shot prompting setup with an instruction-tuned language model. The prompting setup includes:

System Message:

```
You are a helpful assistant that
calculates the sum of two numbers.
Always provide your answer in the
format «x+y=z» where x is the
first number, y is the second
number, and z is their sum.
Do not provide any additional
explanation.
```

Few-shot Examples: Each prompt includes two demonstrations, where the user asks:

```
Calculate the sum of the
following two numbers:
First number: {i}
Second number: {j}
```

The expected assistant response is:

```
«i+j=z»
```

A new target question in the same format is then added to create a complete prompt. This setup is used to generate three-digit addition problems. We extract the residual stream activation at the equals sign (=) in the model's output to form input representations for probe training, with supervision targeting a specific digit (e.g., the hundreds digit) of the result.

We use a 2-shot prompt in the GSM8K setting mainly to ensure that the model adheres to the expected output format. To maintain consistency, we presented the results obtained with 2-shot prompting in the pure arithmetic setting as well. However, we additionally replicated our probing experiments for Gemma2-2b-it under 0-shot and 1-shot settings, and observed similarly strong probe performance across all configurations. Specifically, the best-performing error detectors achieved accuracies of 94.6% (0-shot), 95.8% (1-shot), and 95.4% (2-shot).

In the pure arithmetic setting, the accuracy of Gemma-2-2b-it under 0-shot, 1-shot, and 2-shot prompting is 98.3%, 98.5%, and 98.6%, respectively.

B Sampling Strategy

We construct the probing dataset from model outputs on all addition problems (a, b) where $a, b \in [100, 999]$ and $a + b < 1000$.

Samples are first filtered to include a mixture of both correct and incorrect model predictions. Specifically, the incorrect samples are those where the model's output differs from the true sum in the hundreds digit.

All samples are then grouped based on the hundreds digit of the model's predicted output. From each digit class (2-9), we randomly select up to 100 samples, ensuring that both correct and incorrect predictions are represented across digit classes. If a digit class contains fewer than 100 samples, all available samples from that class are included. For Gemma 2 2B IT, this procedure yields exactly 800 examples. The final dataset is then split into training (70%) and testing (30%) sets.

This sampling strategy achieves:

- A balanced distribution over digit classes (in terms of predicted hundreds digits);
- A representative mix of correct and incorrect predictions, supporting both value probing and error detection tasks.

C Error Detection Probes

Here we provide the formulation for the error-detecting probes discussed in §§ 3.3 and 4.3.

MLP Error Detector (Single). The MLP consists of a single hidden layer with ReLU activation and a hidden dimensionality of 512. As in the logistic probe, the output layer produces 2 logits corresponding to correct and incorrect predictions. The prediction about model correctness $\hat{y}_c$ is given by:

$$\hat{y}_c = \arg\max_i \left(\mathbf{W}_2^\top \, \mathrm{ReLU}\Big(\mathbf{W}_1^\top \mathbf{x}_l + \mathbf{b}_1 \Big) + \mathbf{b}_2 \right),$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times 512}, \mathbf{W}_2 \in \mathbb{R}^{512 \times 2}, \mathbf{b}_1 \in \mathbb{R}^{512}$, and $\mathbf{b}_2 \in \mathbb{R}^2$ are the probe's parameters. This probe is trained using **cross-entropy loss** on binary labels (1 = correct, 0 = incorrect), and optimized using Adam.

Circular Error Detector (Joint). We combine two circular probes by taking angular difference between their outputs and passing it through a sigmoid to get an output probability value. More

formally, let the probe weights be two vectors $\mathbf{w}_1^{(1)}, \mathbf{w}_2^{(1)}, \mathbf{w}_1^{(2)}, \mathbf{w}_2^{(2)} \in \mathbb{R}^{d_{\text{model}}}$. We compute the probe's output $\hat{y}$ as

$$\theta_1 = \text{atan2}\big(\mathbf{w}_1^{(1)\top}\mathbf{x}_l, \mathbf{w}_2^{(1)\top}\mathbf{x}_l\big), \quad (5)$$

$$\theta_2 = \text{atan2}\big(\mathbf{w}_1^{(2)\top}\mathbf{x}_l, \mathbf{w}_2^{(2)\top}\mathbf{x}_l\big), \quad (6)$$

$$\hat{y}_{\text{c}} = \sigma\big(\theta_1 - \theta_2\big). \quad (7)$$

The probe is trained with binary cross-entropy loss and the weights for the two circular probes are optimized jointly.

Separate Training Strategy. In this approach, two separate digit probes (e.g., logistic, MLP, or circular) are trained independently: one probe is trained to predict the model's actual output ($\hat{y}_{\text{model}}$), while the other is trained to predict the ground truth label ($\hat{y}_{\text{GT}}$). At inference, error detection is performed by checking whether the two probes agree:

$$\hat{y}_{\text{c}} = \mathbb{1}\big[\hat{y}_{\text{model}} = \hat{y}_{\text{GT}}\big] \quad (8)$$

This procedure does not require any supervision and can be applied post hoc to existing probes.

D GSM8K Dataset Construction for Controlled Reasoning

To enable consistent probing and self-correction experiments on symbolic arithmetic within chain-of-thought (CoT) reasoning, we construct a controlled subset of GSM8K problems involving only addition.

Step 1: Filtering addition-only problems. We first identify GSM8K questions whose solutions involve only addition operations. This ensures that intermediate steps are structurally aligned with the arithmetic patterns studied in the pure setting.

Step 2: Abstracting numerical structure. We use Claude 3.7 Sonnet in *extended thinking* mode to extract and abstract each problem's numerical content into symbolic variables (e.g., $\texttt{x}_1$, $\texttt{x}_2$, ..., $\texttt{x}_n$), creating reusable symbolic templates. We manually verified 20 examples produced by this pipeline and found the variable abstraction to be accurate in all cases.

Step 3: Sampling concrete values. We sample all variables independently and uniformly from the range $[100, 999]$. This process allows for consistent arithmetic structure while introducing variation across generated problems.

Example instance. One resulting example from our dataset is:

> **Question:** Sarah is planning to do some baking. She buys 771 pounds of rye flour, 611 pounds of whole-wheat bread flour, and 505 pounds of chickpea flour. Sarah already had 758 pounds of whole-wheat pastry flour at home. How many pounds of flour does she now have?
> **Symbolic variables:** $x_1 = 771$, $x_2 = 611$, $x_3 = 505$, $x_4 = 758$
> **Expected reasoning:**
>
> «$771 + 611 = 1382$»
> «$1382 + 505 = 1887$»
> «$1887 + 758 = 2645$»
>
> **Model response:**
>
> «$771 + 611 + 505 = 1987$»
> «$1987 + 758 = 2745$»
>
> **Predicted answer:** 2745 (incorrect)
> **Correct answer:** 2645

Model Performance of Gemma2-2b-it In the structured CoT setting, 83% of intermediate steps are correctly formatted (i.e., of the form «a+b=c»), and among these, 94.1% yield the correct result. The overall per-problem accuracy, accounting for all outputs including formatting errors, is 29.6%.

E Prompting and Sample Selection in GSM8K

For the self-correction experiments described in Section 5, we use a series of structured prompts to intervene when the error detector flags a potentially incorrect step.

Prompt format. Each prompt includes a short corrective message, followed by the previous equation up to the equals sign («`a + b =`). The following messages are used:

- **suspicious:** *"That step looks suspicious. Let's re-do just this step:"*
- **neutral:** *"That step looks incorrect. Let's re-do just this step:"*
- **specific:** *"The calculation in this step is incorrect. Let's recalculate:"*

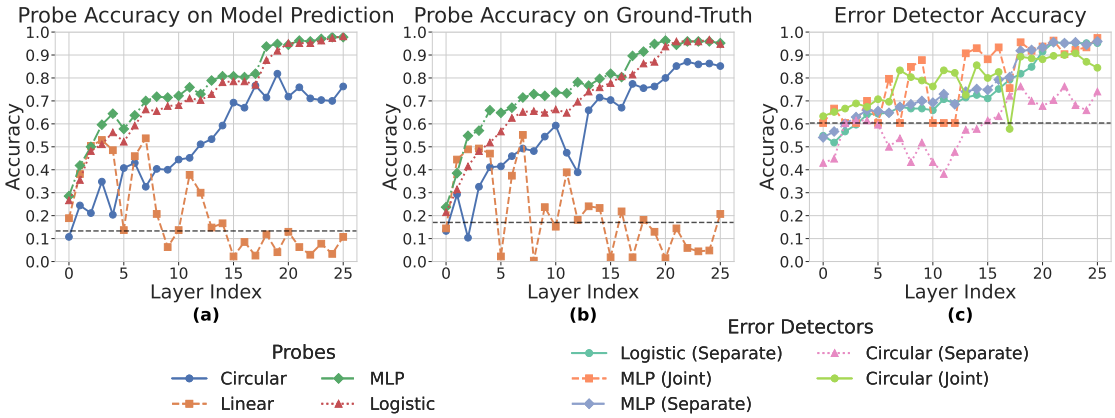

Figure 6: **Probing 3-Digit Arithmetic Queries (Subtraction).** Probing and error detection results on subtraction tasks. As with addition, probes reliably decode both model predictions (a) and ground-truth answers (b), and error detectors achieve high accuracy in predicting correctness (c), confirming the robustness of our findings.

- **stronger:** *"That's definitely wrong. The correct calculation should be:"*
- **detailed:** *"I made an error in adding these numbers. Let me compute the sum correctly step by step:"*

Sample selection. In both the probing and intervention experiments on GSM8K, we select a single step per response for evaluation:

- We first generate the full model response to a given question.
- We identify all equations in the format «`a + b = c`» and determine whether they are correct.
- If all equations are correct, we select the first one and extract the residual stream activation at the equals sign (=) as a **correct** sample.
- If any equation is incorrect, we select the first incorrect one and extract the activation at its equals sign as an **error** sample.

This means we test at most one equation per model response. The same procedure is used both for probe training and evaluation, and for the prompting-based self-correction experiments.

F Error Statistics

We evaluated Gemma 2 2B IT on 319,972 samples of the form (x, y) where $x + y < 1000$. The model produced 3,329 (1.04%) incorrect results. Among these errors: 537 (∼16% of the total errors) had incorrect ones digits, 2,203 (∼66%) had incorrect tens digits. 3,328 (∼100%) had incorrect hundreds digits.

G Computational Resources and Tools

All experiments were conducted on a single Nvidia RTX 4090 GPU. The longest training run for the probes took 1 hour. Generating model response for the experiments in §4 took 90 GPU hours. Our experiments were carried out using `PyTorch` (Paszke et al., 2019) and HuggingFace `transformers` (Wolf et al., 2020). All models were run with `bfloat16` precision for efficient memory usage. For some of our experiments, we used a subset of the GSM8K dataset (Cobbe et al., 2021), which is available through an MIT license.[6] We performed our data analysis using `NumPy` (Harris et al., 2020) and `Pandas` (Wes McKinney, 2010). The paper's bibliography was curated using `Ryanize-bib` (Zouhar, 2023).

H Additional Results

In this section, we extend our main findings by evaluating the generality of our probing methodology across a broader range of settings. In particular, we consider arithmetic tasks involving both addition and subtraction (Appendix H.1), problems with a wider operand range (up to 5-digit numbers; Appendix H.2), and free-form CoT reasoning with unconstrained natural language outputs (Appendix H.3).

H.1 Generalization to Subtraction

We construct a dataset with 800 addition and 800 subtraction queries, randomly mixing them. We train and evaluate digit-level probes and error detectors following the procedure in §3. Results in Fig-

[6] `https://choosealicense.com/licenses/mit/`

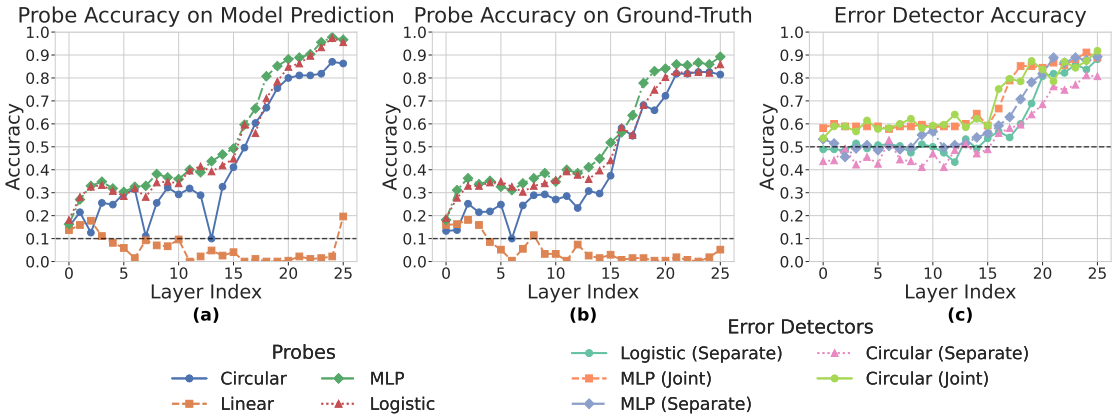

Figure 7: **Probing 3-Digit Arithmetic Queries (Wider Operand Range).** Probing and error detection results on an wider operand range. As with addition, probes reliably decode both model predictions (a) and ground-truth answers (b), and error detectors achieve high accuracy in predicting correctness (c), confirming the robustness of our findings.

ure 6 show that the probes reach accuracy levels comparable to the addition-only case, indicating that they remain effective even when multiple arithmetic operators are present.

H.2 Generalization to Wider Operand Range

We expand the operand range in the pure addition setting to include numbers from 100 to 99,999. We generate a total of 900 examples, ensuring balance across first-digit values. As shown in Figure 7, probes trained on this data continue to achieve strong performance on predicting both model outputs and ground-truth results, as well as detecting correctness, showing robustness to operand scale.

H.3 Generalization to Free-Form CoT

We let Gemma-2 2B IT generate responses to addition-only questions from our augmented version of GSM8K in a free format. Instead of enforcing a structured format (e.g., <a+b=c>), we use a traditional "let's think step by step" prompt and encourage the model to produce a bullet-pointed list of intermediate reasoning steps (e.g., "`1. [step 1]\n 2. [step 2]`"), where each step may contain a mix of natural language and numerical computations. To obtain evaluation data for our digit-level probes and error detectors, we use GPT-4.1-mini[7] to extract the operands and the model's predicted result at each reasoning step. We manually verify a randomly selected subset of 20 examples to confirm that the extraction is accurate and reliable.

We then evaluate the probes trained in the 3-to-5 digit "pure addition" setting (Appendix H.2) on this free-form CoT data. For each reasoning step, we locate the token immediately preceding the first digit of the model's prediction and extract the residual stream activation at that position for probing. The evaluation set is balanced such that a majority class baseline would yield 50% accuracy. Despite the increased variability of this free-form setting, our probes and error detectors maintain non-trivial performance (as shown in Tables 3 and 4). In particular, the best-performing error detector achieves 72% accuracy, demonstrating that our methodology generalizes beyond highly structured setups.

Probe Type	GT Accuracy	Output Acc.
Linear	0.0454	0.0306
MLP	0.6901	0.8547
Circular	0.4926	0.6629
Logistic	0.6413	0.8695

Table 3: **Performance of Different Probe Types on Free-Form CoT Data.** We report the accuracy in predicting both the ground-truth digit and the model's output digit at each reasoning step. Results correspond to the maximum accuracy achieved by each probe type across all layers.

I Generalization to Phi-3

To test the generality of our findings, we replicate our probing and error detection experiments on Phi-3 (Abdin et al., 2024) a 3.8B parameter language model with architecture and training data distinct from Gemma 2B IT.[8]

[7] https://openai.com/index/gpt-4-1

[8] HuggingFace ID: microsoft/Phi-3-mini-4k-instruct.

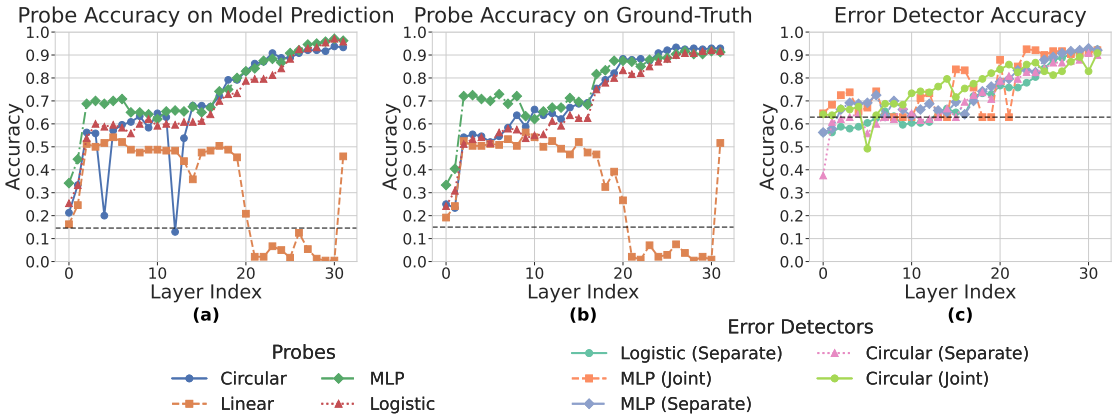

Figure 8: **Probing 3-Digit Arithmetic Queries (Phi-3).** Probing and error detection results on Phi-3. As with Gemma 2 2B IT, probes reliably decode both model predictions (a) and ground-truth answers (b), and error detectors achieve high accuracy in predicting correctness (c), confirming the robustness of our findings across models.

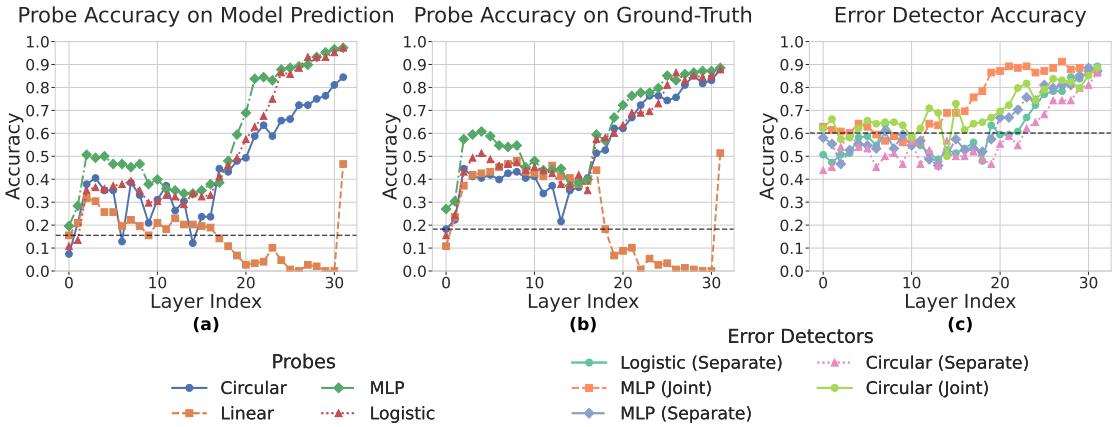

Figure 9: **Probing Structured Chain-of-Thought Reasoning (Phi-3).** (a) Probes accurately recover the model's prediction in deeper layers. (b) Ground-truth digits are similarly decodable. (c) Error detectors achieve strong performance, confirming that findings generalize to structured CoT reasoning with Phi-3.

Error Detector Type	Accuracy
Logistic Separately	0.6754
MLP	0.7208
MLP Separately	0.7026
Circular Separately	0.6129
Circular Jointly	0.6901

Table 4: **Accuracy of Different Error Detector Configurations Applied to Free-Form CoT Data.** Despite the increased variability in reasoning steps, detectors trained on structured arithmetic still generalize well. Results correspond to the maximum accuracy achieved by each probe type across all layers.

We follow the same prompting and evaluation procedures as in §§ 2 and 4. Probes and error detectors are trained using residual stream activations at the equals-sign token in both the pure arithmetic and GSM8K settings.

Overall, we observe results on Phi-3 that are consistent with those observed on Gemma 2 2B IT. In the pure arithmetic setting (Figure 8), probes successfully recover both the model's predicted digit and the ground-truth result, with MLP, logistic, and circular probes reaching over 90% accuracy in the final layers. In addition, error detectors trained on these signals achieve high accuracy in predicting model correctness, surpassing 90% in deeper layers. Overall, these findings reinforce the generality of our main results. Despite differences in architecture, training, and tokenization, both models exhibit similar representational trends.

The same takeaways apply also for the structured CoT setting (Figure 9), where probes recover both the model's prediction and the ground-truth answer with high accuracy in deeper layers, and error detectors reliably identify incorrect steps.

J Ethical Considerations

This work explores how simple probing techniques can be used to detect arithmetic errors from the internal activations of language models. While primarily intended to improve model reliability and transparency, this capability may introduce potential risks.

One concern is that probing tools could be used to reverse-engineer or extract sensitive information from model internals in settings where the model has been fine-tuned on proprietary or confidential data. Although our experiments are limited to synthetic arithmetic tasks, similar techniques might be adapted to more sensitive domains.

Another consideration is the use of probing-based feedback mechanisms to automatically modify or steer model behavior. While we focus on error correction in a controlled setting, improperly validated corrective feedback could introduce bias or reinforce incorrect reasoning patterns in more open-ended tasks.

Lastly, while our methods are lightweight and interpretable, they might be incorrectly interpreted as offering guarantees of correctness or safety. We emphasize that probes are statistical tools, and any system incorporating them should be evaluated rigorously before deployment in high-stakes settings.