Language Models Encode the Value of Numbers Linearly

Fangwei Zhu, Damai Dai, Zhifang Sui
School of Computer Science,
State Key Laboratory of Multimedia Information Processing,
Peking University
zhufangwei2022@stu.pku.edu.cn
{daidamai, szf}@pku.edu.cn

Abstract

Large language models (LLMs) have exhibited impressive competence in various tasks, but their internal mechanisms on mathematical problems are still under-explored. In this paper, we study a fundamental question: how language models encode the value of numbers, a basic element in math. To study the question, we construct a synthetic dataset comprising addition problems and utilize linear probes to read out input numbers from the hidden states. Experimental results support the existence of encoded number values in LLMs on different layers, and these values can be extracted via linear probes. Further experiments show that LLMs store their calculation results in a similar manner, and we can intervene the output via simple vector additions, proving the causal connection between encoded numbers and language model outputs. Our research provides evidence that LLMs encode the value of numbers linearly, offering insights for better exploring, designing, and utilizing numeric information in LLMs. The code and data are available at https://github.com/solitaryzero/NumProbe.

1 Introduction

Large language models (LLMs) have demonstrated excellent ability in various scenarios like question answering (Zhao et al., 2023; Li et al., 2023b), instruction following (Brown et al., 2020; Ouyang et al., 2022; Taori et al., 2023), and code generation (Chen et al., 2021; Nijkamp et al., 2022; Li et al., 2023a). Solving mathematical problems is generally viewed to be more difficult (Yu et al., 2023), and language models even struggle to solve simple arithmetic problems (Dziri et al., 2024).

Numbers are fundamental elements in math. In order to accurately answer mathematical problems, LLMs should be able to precisely encode value of numbers in the input text. Currently, the way how LLMs process numbers is still not fully explored. While previous studies (Stolfo et al., 2023; Hanna et al., 2024) have explored the inner mechanisms of language models on mathematical problems, they focus on small numbers or modular arithmetic (Engels et al., 2024; Zhong et al., 2024), and how LLMs utilize numbers in a larger, unconstrained range remains largely unknown.

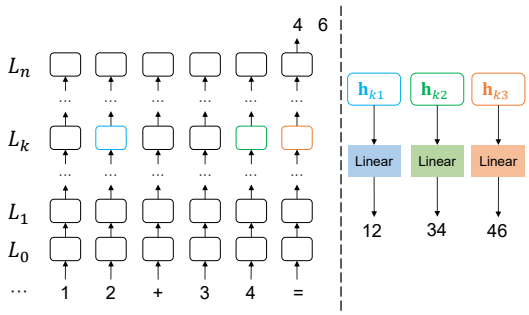


Figure 1: Encoded number values in the hidden state of language models. We find that both the value of input numbers (blue and green) and calculation results (red) can be read out from the hidden state of language models via linear probes.

In this paper, we explore the question whether and how LLMs encode the value of numbers through extracting numerical information from their internal representations. To be specific, we construct a synthetic dataset comprising simple addition questions, and train linear probes (Nanda et al., 2023; Gurnee and Tegmark, 2023) on the hidden states of LLMs to predict the number values provided in the input text. Experimental results on the dataset demonstrate that the value of input numbers can be probed from the hidden states of language models from early layers, as illustrated in Figure 1. Both input values and calculation results can be read out, and encoded values can be found at different token positions. These results support that language models do encode numerical information, and possibly in a linear manner.

To further verify the fact that the encoded number values are utilized by language models, we

Proceedings of the 31st International Conference on Computational Linguistics, pages 693–709
January 19–24, 2025. ©2025 Association for Computational Linguistics

study the causal connection between numeric information and model outputs. To be specific, we discover that we can influence the calculation result of language models by performing interventions like activation patching or adding linear vectors.

The above discoveries may reveal future directions for utilizing the encoded numerical information, for example, specialized encoding systems and error mitigation modules.

To sum up, our contributions can be listed as: (1) We study the question of whether language models are able to encode the value of numbers in the input text and construct a synthetic dataset to analyze the language models. (2) We discover that language models encode the value of numbers linearly by utilizing linear probes to probe encoded number values in the hidden states of language models. (3) We further prove that language models utilize the encoded numerical information by revealing the causal connection between encoded number values and the final output of language models.

2 Probing Numbers in Language Models

2.1 The Goal of Probing

Given that there is a number x in the input text t, we assume that a language model LM can encode the number in its hidden state $\mathbf{h}_i \in \mathbb{R}^{d_{model}}$ of a specific layer i, where d_{model} is the hidden dimension. We denote the mapping as:

$$\mathbf{h}_i = f_i(x, t - x) \tag{1}$$

where f_i refers to the encoding process on layer i, and $t - x$ refers to the tokens in t excluding x.

If the mapping function f is a bijective function, there will exist an inverse function f_i^{-1} that reconstructs the original number x from the hidden state $\mathbf{h}_i$. For each layer i, we aim to find a optimal predictor $\mathcal{P}_i^*$ that imitates f_i^{-1}, whose prediction best fits the original number x:

$$\mathcal{P}_i^* = \arg\min_{\mathcal{P}_i} |x - \mathcal{P}_i(\mathbf{h}_i)| \tag{2}$$

Considering the numerical stability, we probe the logarithmic value $\log_2(x)$ instead of the original number x in all our experiments.

We can assess the existence of encoded number values by observing how much the probing result $\mathcal{P}_i^*((\mathbf{h}_i))$ resembles the original number x.

2.2 Dataset Construction

To investigate whether LLMs encode numbers, we construct a synthetic dataset containing different magnitudes of numbers. The dataset contains numbers ranging from 2 digits to 10 digits, with each digit corresponding to 1000 entries[1]. We split the dataset into training, validation, and test sets at a ratio of 80%/10%/10%.

To observe how LLMs encode and utilize numbers, we adopt addition problems as our prompt[2]. Let a and b be two randomly generated numbers, each question is formulated as:

```
Question: What is the sum of {a} and {b}?
Answer: {a + b}
```

2.3 Probing Method

Obtaining Hidden States. We choose the LLaMA-2 model family (Touvron et al., 2023b) and Mistral-7B (Jiang et al., 2023) as base models to be investigated. We feed the question text in Section 2.2 into the models, and save the hidden states of all layers. For each layer, we obtain a set of hidden states (i.e. the residual stream) $\mathbf{H} \in \mathbb{R}^{n \times d_{model}}$ at every token position, where n is the number of samples in the dataset.

Training Probes. Following previous work, we adopt the widely acknowledged linear probing technique to reconstruct numbers from the hidden states. To be specific, for each layer, given a set of hidden states $\mathbf{H}$ and their corresponding original numbers $\mathbf{X} = \{x\}$, we train a linear regressor $\mathcal{P}$ that yields best predictions $\mathbf{P} = \mathbf{HW} + b$, where $\mathbf{W} \in \mathbb{R}^{d_{model}}$ and b are the weights of $\mathcal{P}$.

In practice, directly performing linear regression could give erroneous results, as the value of numbers varies over a wide range. We do a logarithmic operation on input numbers $\mathbf{X}$ with a base of 2 to guarantee the numerical stability of probes.

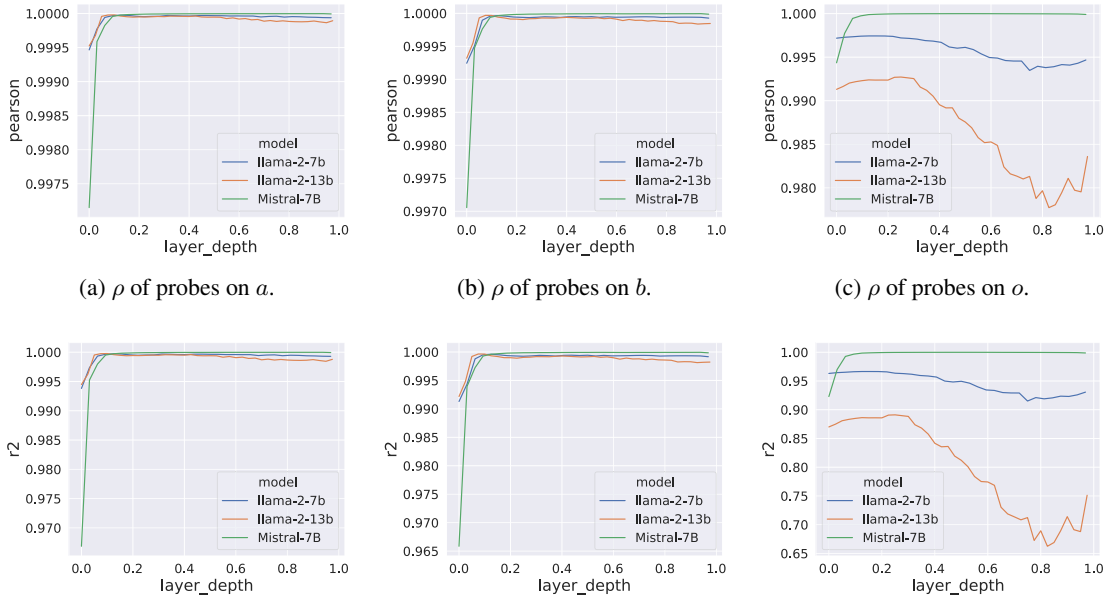
We utilize Ridge regression, which adds L2 regularization to the vanilla linear regression model, to construct the probes:

$$\mathbf{W}^*, b^* = \arg\min_{\mathbf{W}, b} || \log_2(\mathbf{X}) - \mathbf{HW} - b||_2^2 + \lambda ||\mathbf{W}||_2^2 \tag{3}$$

where $\mathbf{W}^*, b^*$ are the weights of regressors, and λ is a hyperparameter that controls regularization strength. In this way, we can predict logarithmic results $\mathbf{P}^* = \mathbf{HW}^* + b^*$ based on the hidden states.

[1] See Appendix A for more details.

[2] The experimental results on subtraction problems are similar to the results on addition problems (see Appendix C). We do not include multiplication and division problems either, as LLMs perform poorly on these problems (even 5-digit multiplication yields an accuracy of about 0%).



(a) ρ of probes on a. (b) ρ of probes on b. (c) ρ of probes on o.

(d) R^2 of probes on a. (e) R^2 of probes on b. (f) R^2 of probes on o.

Figure 2: Pearson coefficient (ρ) and out-of-sample R^2 of probes on different layers. a and b refer to the two input numbers denoted in Section 2.2, and o refers to the prediction of language models respectively. High ρ and R^2 indicate the existence of encoded number values in the hidden states.

2.4 Evaluation Metrics

We use two standard regression metrics on the probing task to evaluate the probes: R^2 which determines the proportion of variance in the dependent variable that can be explained by the independent variable, and the Pearson coefficient ρ which measures the linear correlation between two variables.

As mathematical problems require a precise understanding of numbers, we introduce two additional metrics to examine how well can a language model encode numbers:

Approximate accuracy (AAcc) evaluates whether the predicted number is approximately the same as the original number, namely with an error margin of $< 1\%$. Higher AAcc indicates that the number encoding is more likely to be precise.

Mean square error (MSE) is the average squared difference between probe predictions and actual values. Smaller MSE means lower loss during the encoding process.

$$\text{AAcc}(\mathbf{P}^*, \mathbf{X}) = \frac{|(2^{\mathbf{P}^*} - \mathbf{X}) < 0.01X|}{|\mathbf{X}|} \quad (4)$$

$$\text{MSE}(\mathbf{P}^*, \mathbf{X}) = \text{avg}((\mathbf{P}^* - \log_2 \mathbf{X})^2) \quad (5)$$

2.5 Experimental Setup

We use the original LLaMA-2-7B, LLaMA-2-13B, and Mistral-7B models without fine-tuning for all experiments. The outputs are obtained by performing greedy search with a max new token restriction of 30 during decoding. The regularization strength is set to $\lambda = 0.1$ for all probes.[3].

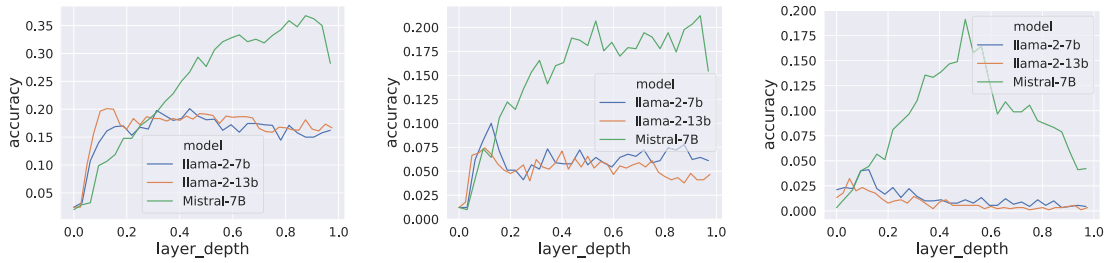
In main experiments, we probe 3 distinct values at different positions: the first number a at the last digit of a (for example, 3 for 123), the second number b at the last digit of b, and the prediction of language models o at the last token of the entire input text. We report the accuracy of o, i.e. the ratio of $o = a + b$, in Appendix D.

3 Do LLMs Encode Number Values?

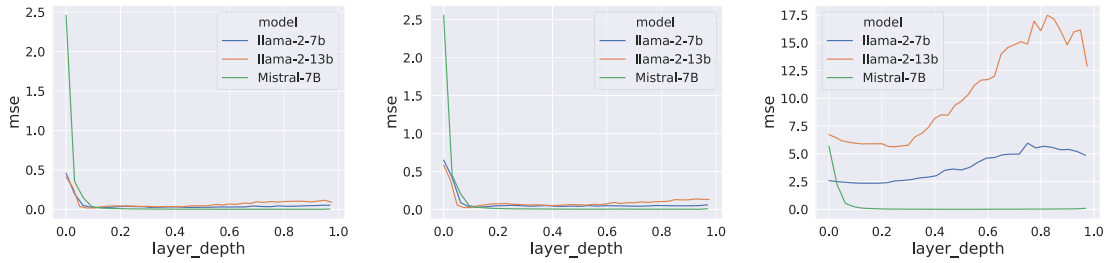
3.1 The Existence of Encoded Number Values

LLMs do encode number values. We first inspect the overall Pearson coefficient (ρ) and out-of-sample R^2 on all layers. High ρ and R^2 indicate that LLMs are likely to be able to encode number values in their hidden states. As illustrated in Figure 2, the probes achieve surprisingly high ρ and R^2 on all layers, proving that the hidden states of LLMs contain the encoded value of input numbers, and the encoding process starts from even the first layer. Meanwhile, notice that both ρ and R^2 slightly drop on late layers, which may indicate that intermediate layers better encode number values.

[3]See Appendix B for more details.



(a) AAcc of probes on a. (b) AAcc of probes on b. (c) AAcc of probes on o.



(d) MSE of probes on a. (e) MSE of probes on b. (f) MSE of probes on o.

Figure 3: Approximate accuracy (AAcc) and mean square error (MSE) of probes on different layers. a and b refer to the two input numbers denoted in Section 2.2, and o refers to the prediction of language models respectively. High AAcc and low MSE indicate precise number encoding.

Linear probes cannot reconstruct the precise value. Aside from the existence of encoded number values, we are also interested in their precision, which is depicted by AAcc and MSE in Figure 3.

In contrast to high correlation coefficients, the AAcc is below 50% on all layers, which means that the linear probes have difficulty in precisely reconstructing the input numbers. The trends in AAcc and MSE are consistent with ρ and R^2, indicating that LLaMA-2 models achieve the most precise number encoding in intermediate layers, but the encoding faces more error in deeper layers.

This phenomenon may indicate that language models use stronger non-linear encoding systems, which we will further explore in Section 3.4; Or it may be a hint that the number encoding in language models is not precise[4].

3.2 Number Encoding Patterns are Different across Layers

To better analyze how language models encode numbers, we pick distinct layers in LLaMA-2-7B and observe how the pattern of probe predictions changes as the layer gets deeper. Layer 0 (i.e. the first transformer block after embedding layer), 10, and 30 are selected to represent early, intermediate, and late layers respectively. The trend of change on the first input number a is shown in Figure 4.

On early layers like layer 0, the predictions of probes are distorted to some extent: for original numbers with the same length, their corresponding predictions in the figure display a pattern of horizontal lines. This phenomenon indicates that early layers focus on the length of numbers, which corresponds to the number of input digit tokens.

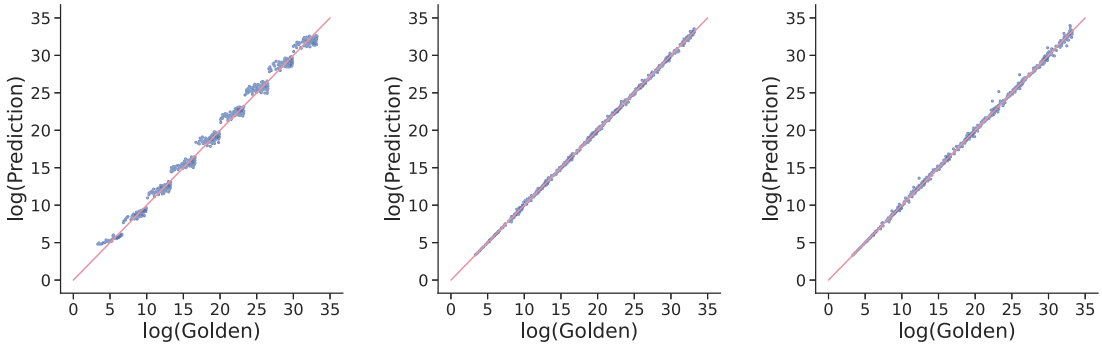
As the layer gets deeper, probes on intermediate layers show the best performance. On layer 10, the predicted results are very close to the actual answers, yielding a near-perfect linear probe for original numbers. However, noise emerges in the prediction results again in late layers, with the form of uniformly distributed errors.

The trend of change leads us to a conjecture that language models first roughly estimate the value of a number with its token length, and then refine the estimation in subsequent layers. The process may not be lossless, which leads to errors in the final number encoding of language models.

3.3 Numeric Information Persist at Subsequent Positions

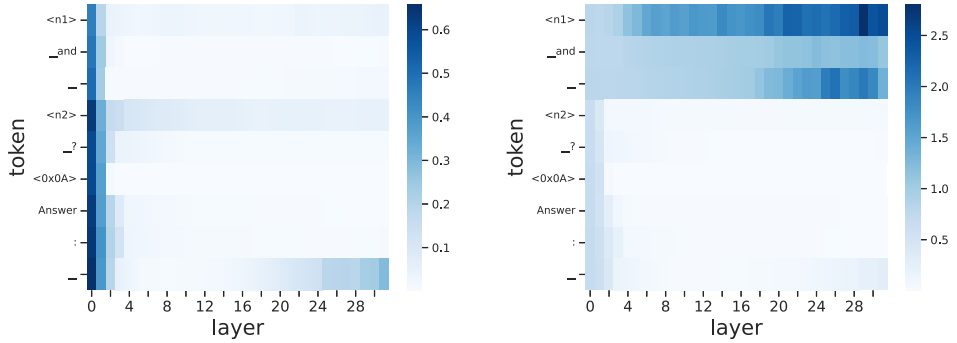
Another question is whether these encoded values are only stored at certain positions, or are they persist at subsequent positions. For input number values a, b, we train probes at every individual

[4]See Appendix F for more detailed experiments.



(a) Layer 0 (b) Layer 10 (c) Layer 30

Figure 4: How the pattern of probe predictions on the first input number a changes as the layer gets deeper. Probe predictions on different layers of LLaMA-2-7B show different patterns.



(a) MSE of probes on a. (b) MSE of of probes on b.

Figure 5: The mean square error (MSE) of probes at different token positions on LLaMA-2-7B. <n1> represents the last token of the first input number a, and <n2> represents the last token of the second input number b, respectively. The rectangular pattern indicates that the value of an input number can be read out at any subsequent position.

token position to examine where these values exist. Figure 5 shows the mean square error of probes on the LLaMA-2-7B model.

The results demonstrate a clear rectangular pattern, indicating that the value of an input number can be read out at any subsequent position. In other words, the number values would persist at subsequent positions. It is also worth noticing that the probing accuracy on the last token is lower than other positions, which may be interpreted as language models do not continue to remember input numbers after computing the final outcome.

3.4 LLMs Encode Numbers Linearly

Previous work (Nanda et al., 2023; Gurnee and Tegmark, 2023) on probing neural networks propose the linear representation hypothesis: the presence of features of a neural network can be proved by training a linear projector which projects the activation vector to the feature space, and complex structures are unnecessary. To verify whether the numbers can be represented linearly, we follow the method of Gurnee and Tegmark (2023) which trains two-layer MLP probes and compares their performance with linear probes. The MLP probes have an intermediate hidden state of 256 dimensions and can be formulated as:

$$\mathbf{P} = \mathbf{W}_2 \text{ReLU}(\mathbf{W}_1 \mathbf{H} + b_1) + b_2 \quad (6)$$

where $\mathbf{W}_1, \mathbf{W}_2, b_1$ and b_2 are trainable weights.

Figure 6 demonstrates the comparison between MLP probes and linear probes on mean square error. We find that nonlinear MLP probes do not show any clear advantage over linear probes, proving that the encoded number values can be represented linearly, or at least near-linearly.

4 Do LLMs Utilize Number Values?

The previous section has proved the existence of encoded number values in language models. How-

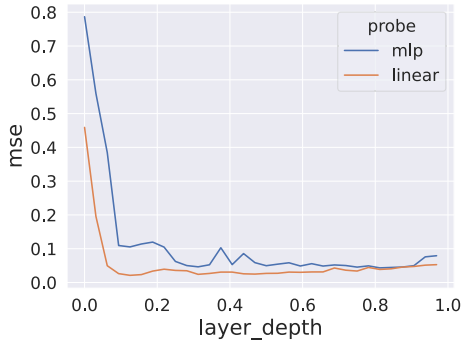


Figure 6: The comparison between linear probes and MLP probes on mean square error (MSE). The MLP probes do not show advantage over linear probes. More detailed experiments are reported in Appendix E.

ever, an inherent issue is that the probed information is only correlational to the output of models, and no causal effects can be directly claimed (Belinkov, 2022).

In this section, we will try to verify the hypothesis that language models do use the encoded number values to get their calculation results by performing a set of intervention experiments. Given an input question Q with an expected result of o, we intervene in the internal activation of language models to make it believe in an altered question Q', and observe how the new result o' changes.

To ensure the effectiveness of the intervention, we conduct the experiments on Mistral-7B with 4-digit addition questions as input, where the model could correctly answer most of the questions.

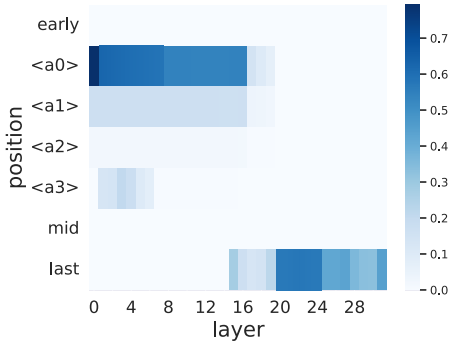


Figure 7: The effect of patching on different components. Early and mid refer to the non-number tokens before and after the first input number a, and last refers to the last token of the input text.

4.1 Patching Encoded Numbers

Firstly, we study the influence of number encoding at different positions by changing the activation of language models. We adopt the activation patching technique proposed by Stolfo et al. (2023) to quantify the importance of encoded number values $\mathbf{h}_i$ at different layers i and different token positions.

To be specific, given an input addition problem consisting of input numbers a and b, we will conduct the following procedure:

1. Obtain the clean output of the language model $o = LM(a, b)$.

2. Replace a with another number a' to get a new output $o' = LM(a', b)$, and record the hidden states $\mathbf{h}'$ at certain position t during the forward pass;

3. Perform an additional forward pass with a and b as input numbers, where we substitute the hidden state $\mathbf{h}_i$ of layer i with $\mathbf{h}'_i$. This would lead to an intervened result o^*.

We set $a' = 9999$ in our experiments, and evaluate the effect of intervention as:

$$E(i,t) = \frac{|o^* - o|}{|o' - o|} \tag{7}$$

which measures how much a specific layer i at position t affects the final result. Note that the metric is intended for qualitative rather than quantitative analysis.

Figure 7 demonstrates the effect of activation patching on different components, from which we can draw multiple observations:

Patching	Result	Explanation
None	6912	5678+1234=6912
Full	11233	9999+1234=11233
$5 \rightarrow 9$	10912	9678+1234=10912
$6 \rightarrow 9$	7212	5978+1234=7212
$7 \rightarrow 9$	6932	5698+1234=6932
$8 \rightarrow 9$	6913	5679+1234=6913

Table 1: Patching results on the question "Question: What is the sum of 5678 and 1234 ?" by patching the activation on layer 8.

Each digit affects the result independently. The effect of patching on different number digits displays a clear pattern: the earlier a digit appears, the more patching it changes the final output value. While the latter digits encode the values of partial number sequences (See Appendix F for details), activation patching seems to only change the final

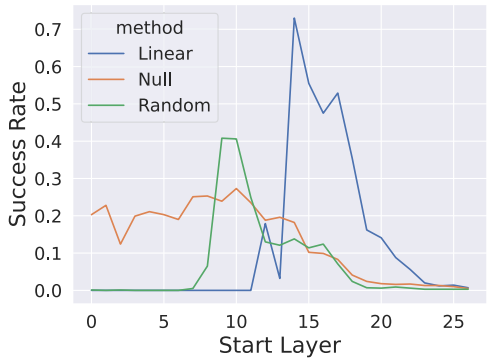


Figure 8: The success rate of performing a linear intervention on 6 consecutive layers. More detailed experiments are reported in Appendix I.

result by the value of the digit itself. For example, although the activation at digit "3" in "1234" encodes the value of 123, patching it equals changing the input number to "1294" rather than "9994", as demonstrated in Table 1. More detailed experiments are reported in Appendix H.

Language models concern only certain tokens during calculation. Despite our finding in Section 3.3 that encoded number values would persist in subsequent tokens, patching non-number tokens has almost zero effect on the final outcome. This pattern indicates that the encoded number values at most positions are simply "memorized" rather than "used" by the language model. An exception is the last token, where language models seem to store their calculation results.

Early and late layers play different roles. The effect of activation patching can be divided into two parts: on early layers before layer 14, patching the number tokens greatly influences the final outcome, while patching the last token is mostly ineffective; but in late layers after layer 20 it is just the opposite. We assume that early layers perform the task of processing the value of input number token sequences, while late layers use encoded values to calculate the final outcome, which is similar to the findings in Stolfo et al. (2023).

4.2 Linearly Intervening Encoded Numbers

To determine whether the encoded computational results causally affect the outcome of language models, we linearly intervene the hidden states and see whether the output changes as expected.

Method. Following the method of Nanda et al. (2023), for each intervened layer i, we add the number encoding direction vector $\mathbf{d}_i$ to the residual stream $\mathbf{h}_i$:

$$\mathbf{h}_i^{'} = \mathbf{h}_i + \alpha \mathbf{d}_i \tag{8}$$

where $\alpha > 0$ is a scaling factor and the direction vector $\mathbf{d}_i$ is obtained by normalizing the probe coefficients:

$$\mathbf{d}_i = \frac{\mathbf{W}_i}{|\mathbf{W}_i|} \tag{9}$$

Considering that the probed number value is the projection of $\mathbf{h}_i$ along the direction $\mathbf{d}_i$, the effect of our intervention is to "push" the residual stream towards a larger encoded number. We set $\alpha = 2$ in our experiments, and intervened language models outputting a larger number $o' > o$ than the original prediction o is viewed as a success.

In the linear intervention experiment, we choose probes for language model predictions o at the last input token to obtain the direction vector $\mathbf{d}_i$, and perform an intervention on every newly generated token. We use a test set of 1,000 entries and measure the efficacy of our intervention by observing the ratio of successful interventions.

We also use two alternative directions as baselines: normalized $\mathbf{h}_i$ as null intervention, and a random unit vector $\mathbf{I}$ as random intervention.

Result and Findings. Figure 8 shows the success rate of intervening on 6 consecutive layers. Linear intervention achieves the highest success rate of 0.73 when intervening layer 14 to layer 19, outperforming the null intervention baseline by a large margin. This suggests that the linearly encoded number values are causal to model predictions.

It is also worth noticing that intervening on mid-late layers is significantly more effective than on early layers and late layers. We hypothesize that this phenomenon is related to the findings of Stolfo et al. (2023): language models use mid-late layers to perform arithmetic computations, while the late layers are responsible for converting the computational result to output tokens.

5 Discussion and Future Directions

In previous sections, we find that LLMs know the value of numbers and utilize the encoded number values to perform calculations. However, the compression may not be lossless, and the calculation ability scales with model size. Moreover, the ability to understand and utilize numbers is positively correlated to mathematical competency. These findings reveal some future research directions that are potentially promising.

The exact way that LLMs encode numbers. While our experiments show that the original input number cannot be reconstructed from the hidden state via linear probes, there exists a possibility that the LLMs encode numbers in a way that is close to a linear projection but not identical, such as the floating-point system (Muller et al., 2018). Finding out the exact encoding, if possible, will give us a better insight into how LLMs function.

Specialized number encoding systems. The loss of encoded number values in LLMs will inevitably bring errors to subsequent computation, especially for large input numbers. Developing specialized encoding systems that could give precise presentations for numbers (Golkar et al., 2023) could eliminate errors at the root, thus helping LLMs better solve mathematical problems.

Mitigating computational errors with encoded numbers. By adding modules that directly utilize the encoded numbers in language models, the computational errors may be further reduced, especially on large-number calculations. We conduct a pioneer experiment in Appendix J to reveal the potential of controlling computational errors with probed numbers.

6 Related Work

Large Language Models on Mathematical Problems. Large language models (LLMs) like the GPT series (OpenAI, 2023), PaLM (Anil et al., 2023) and LLaMA (Touvron et al., 2023a,b) have demonstrated their impressive ability in various fields (Zhao et al., 2023; Li et al., 2023b; Taori et al., 2023; Chen et al., 2021; Nijkamp et al., 2022; Li et al., 2023a). On mathematical datasets like GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), there have been methods like chain-of-thought reasoning (Wei et al., 2022) and self-consistency (Wang et al., 2022) to help LLMs better solve these questions. Specialized large language models like MetaMath (Yu et al., 2023) and Math-Shepherd (Wang et al., 2023) also show great competency.

Interpreting Internal Representations in Language Models. Prior research has unveiled that language models are able to store certain information in their hidden states, for example, passive voice (Shi et al., 2016) and sentence structure (Tenney et al., 2018). By adopting the probing technique (Alain and Bengio, 2016; Belinkov, 2022), complex representations have also been detected in language models: Li et al. (2022) shows that language models are capable of memorizing the state of an Othello game, and Nanda et al. (2023) further proves that the states can be linearly represented; Li et al. (2021) claims that language models are able to encode the properties and relations of entities; Gurnee and Tegmark (2023) reveals evidence that large language models build spatial and temporal representations about an entity from early layers.

Explaining Numbers and Arithmetic in Language Models. How language models process numbers has been studied by multiple researchers. Wallace et al. (2019) detects the existence of numeracy in static pre-trained word embeddings. Hanna et al. (2024) finds a critical circuit that performs greater-than comparisions in GPT-2. Stolfo et al. (2023) studies how language models process arithmetic information by intervening on specific modules of the model. Zhong et al. (2024); Engels et al. (2024) discover evidence that numbers on modular arithmetic may be circularly encoded.

7 Conclusion

In this paper, we study the question of whether and how large language models encode the value of numbers. If number values can be extracted from the internal representations of LLMs, we can assume that LLMs encode the value of numbers in their hidden states. We construct a dataset consisting of simple addition problems and introduce linear probes to investigate whether language models encode number values.

Experimental results prove that LLMs do encode the value of input numbers, and the representation could be linearly read out. The ability to linearly encode numbers is consistent across different model scales, and the encoding seems to be the most precise on intermediate layers. Further experiments show that LLMs utilize the encoded number values to perform arithmetic calculations, and the behavior of language models can be controlled via simple linear interventions, proving the causal connection between encoded numbers and model outputs.

Our work shows a glimpse of the internal mechanisms of how language models solve mathematical questions. Future works on the internal representations of numbers, for example, better probes and specialized number encoders, may enhance the mathematical competence of language models in an explainable way.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This paper is supported by NSFC project 62476009. The contact author is Zhifang Sui.

Limitations and Risks

While we explore the inner mechanisms of how language models understand numbers, the probes trained in our current method are only approximations of the encoded numbers rather than exact internal presentations. Directly performing calculations with probes would lead to undesired results. Meanwhile, our experiments are conducted on LLMs whose parameters are openly available, while other LLMs the ChatGPT or GPT-4 may exhibit different behaviors.

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Joshua Engels, Isaac Liao, Eric J Michaud, Wes Gurnee, and Max Tegmark. 2024. Not all language model features are linear. *arXiv preprint arXiv:2405.14860*.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, et al. 2023. xval: A continuous number encoding for large language models. In *NeurIPS 2023 AI for Science Workshop*.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Belinda Z Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2022. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023a. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. 2023. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*.

Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, Serge Torres, et al. 2018. *Handbook of floating-point arithmetic*. Springer.

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1526–1534.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. 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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2018. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. *arXiv preprint arXiv:2305.03268*.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. 2024. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in Neural Information Processing Systems*, 36.

A Dataset Details

The dataset in Section 2.2 contains 9000 addition problems. For each number of digits between 2 and 10, 1000 problems are generated, and two numbers in the same problem share the same digit. For questions whose number has 4 or fewer digits, we list all possible combinations of numbers and randomly sample 1,000 of them to generate the questions. For questions whose number has 5 or more digits, we randomly sample both numbers to generate the 1000 questions.

B Experiment Implementation

The experiments are conducted on 4 NVIDIA GTX 3090 GPUs. Acquiring the hidden states of LLMs

on our synthetic dataset requires 10$\tilde{2}$0 GPU hours per model.

We obtain the LLaMA-2 models and Mistral-7B model from the huggingface model hub, and implement the experiments with the huggingface transformers Python library. The probes are trained with the scikit-learn Python library. We use the TransformerLens library[5] for intervention experiments. We follow the terms of use of all models and use them only for research.

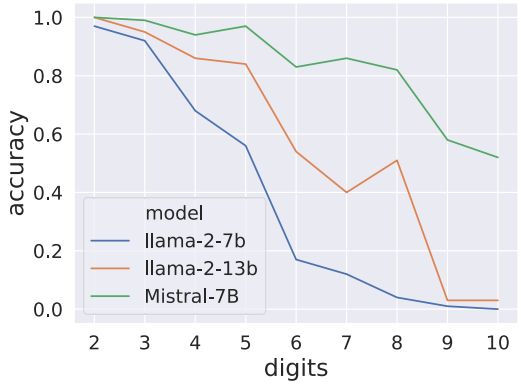


Figure 9: The overall accuracy of language model predictions on addition problems.

C Experiments on Subtraction Problems

In the main paper, we only show the results of probing on addition problems. We also conduct experiments on subtraction problems with the form of:

```
Question: What is the result of {a}
minus {b}?
Answer: {a - b}
```

where we assert $a > b$ to ensure the result being a positive number.

Figure 10 demonstrates the result of probing on subtraction problems. We can clearly observe that the trends of different metrics are similar to those on addition problems. In other words, the behaviour of language models on subtraction problems are similar to the behaviour on addition problems.

D Overall Accuracy

Figure 9 shows the overall accuracy of different language models on addition problems. We can see that the accuracy of all models, especially LLaMA-2 models, faces a sharp decline at 6-digit problems, which may have a possible correlation with the partial number encoding accuracy demonstrated in Figure 12.

In the LLaMA-2 family, the 13B model does not show any advantage over the 7B model on probing metrics. In contrast, Mistral-7B displays better performance on all probing metrics, which is consistent with its outstanding math ability. The difference implies that the ability to encode numbers is consistent across different model scales, but varies between different model families. Meanwhile, the ability to understand numbers show a positive correlation with the math ability of LLMs.

E Detailed Experiments on Linearity

Figure 11 shows the comparison between linear probes and MLP probes on ρ, R^2 and MSE. We can observe that MLP probes generally perform no better than linear probes.

F Experimental Results on Partial Number Encoding

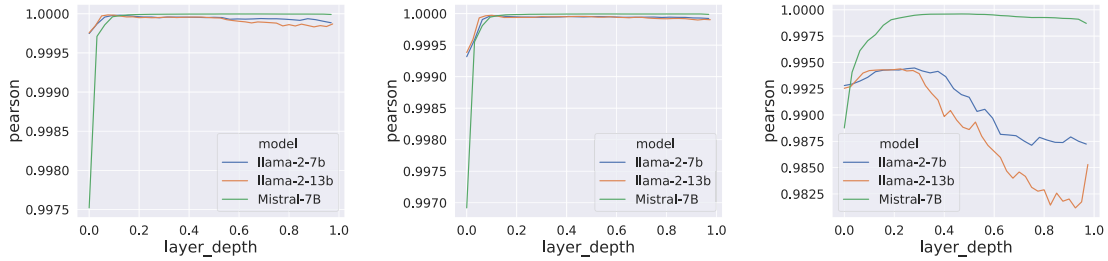
In large language models like LLaMA-2, large numbers are split into multiple tokens, where each token represents a certain digit of the original number. This raises a question: whether the encoding process will proceed from token to token, or will it only happen at the end of number token sequences?

To investigate the problem, we choose addition problems consisting of 8-digit numbers and probe the value of the partial number sequence at every token position. For example, given a number token sequence "12345678", we will probe the value 12 at the position of token "2", and probe the value 123 at the position of token "3".

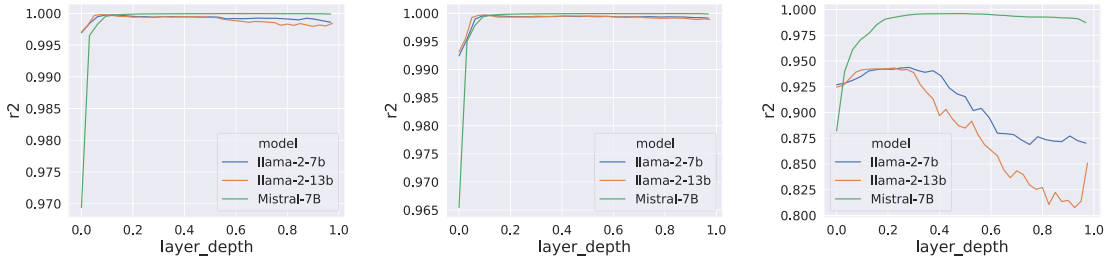
Figure 12 shows the probing accuracy of 3 models. It can be observed that the value of the partial number sequence can be read out at every token position. In other words, language models encode the number token sequence incrementally.

Meanwhile, the accuracy significantly declines as the token sequence gets longer, which means that language models face increasing difficulty in capturing the precise value as the number gets larger in scale. Notice that Mistral-7B suffers less from accuracy decay, we can assume that the ability to precisely encode long number token sequences is positively correlated to the mathematical ability of language models.

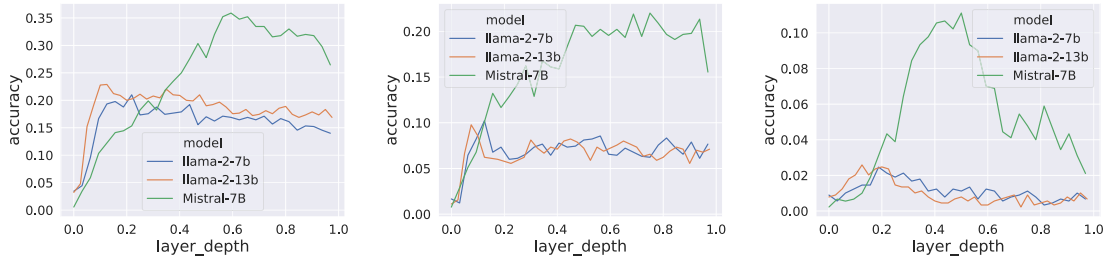
[5] https://github.com/neelnanda-io/TransformerLens



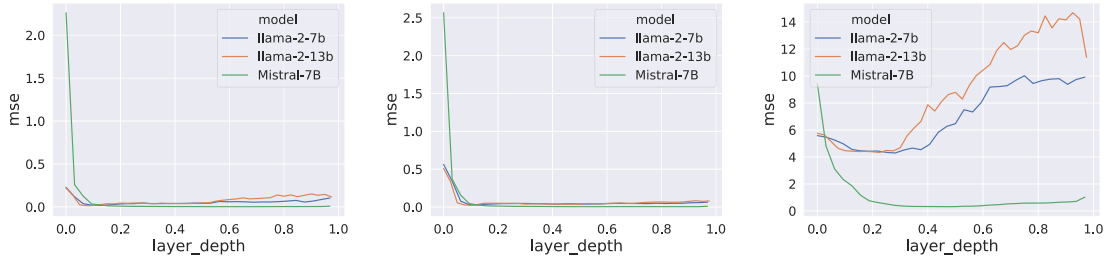
(a) ρ of probes on a. (b) ρ of probes on b. (c) ρ of probes on o.



(d) R^2 of probes on a. (e) R^2 of probes on b. (f) R^2 of probes on o.

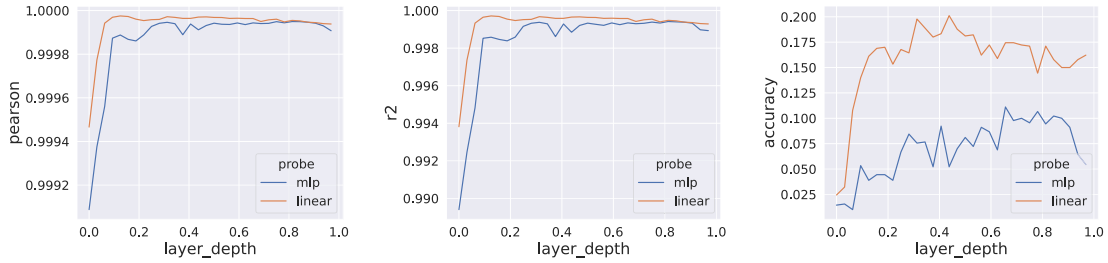


(g) AAcc of probes on a. (h) AAcc of probes on b. (i) AAcc of probes on o.



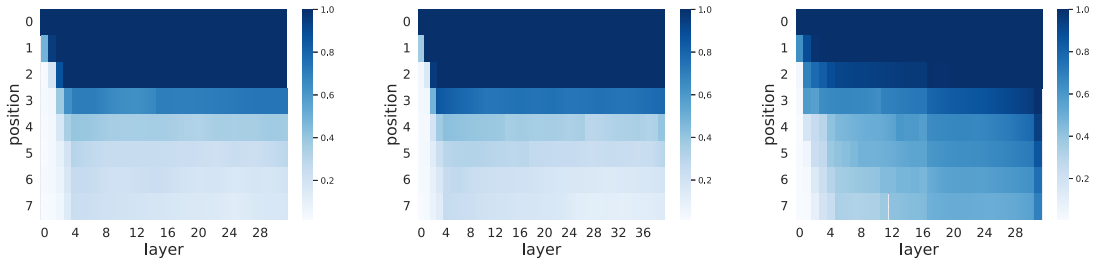
(j) MSE of probes on a. (k) MSE of probes on b. (l) MSE of probes on o.

Figure 10: Pearson coefficient (ρ), out-of-sample R^2, approximate accuracy (AAcc), and mean square error (MSE) of probes on different layers for subtraction problems. a and b refer to the two input numbers denoted in Section 2.2, and o refers to the prediction of language models respectively. High ρ and R^2 indicate the existence of encoded number values in the hidden states.



(a) Pearson coefficient (b) Out of sample R^2 (c) Mean square error

Figure 11: Comparison between linear probes and non-linear MLP probes. Pearson coefficient, out-of-sample R^2, and AAcc of probes on the first input number a on different layers are shown in the figure.



(a) AAcc of LLaMA-2-7B. (b) AAcc of LLaMA-2-13B. (c) AAcc of Mistral-7B.

Figure 12: The approximate accuracy (AAcc) of probes on partial number sequence of 8-digit numbers. The y-axis represents the index of number tokens in the token sequence.

Figure 13 shows the Pearson coefficient, out-of-sample R^2, and mean square error of probes on partial sequence of 8-digit numbers. These metrics remain stable as the length of number token sequence gets longer, indicating that language models do have the ability to incrementally encode number values, but there would be more error when the number gets larger in scale.

G Probing With Control Tasks

There exists the risk that probes may learn to extract values that language models do not encode. In Figure 5, we can see that probing on the second input number b at positions before it appears would lead to extremely large mean square errors, which acts as a piece of preliminary evidence that the probe performance does not solely come from probe strength.

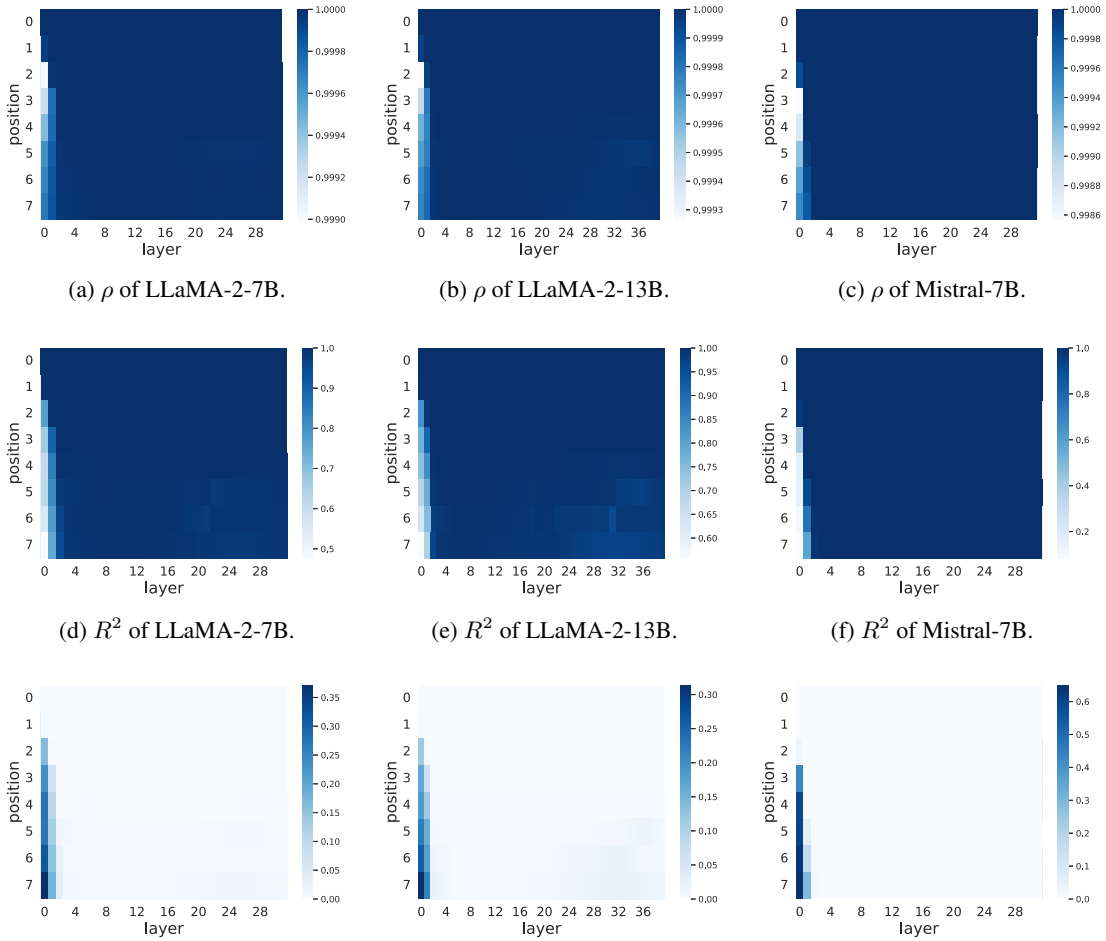
To quantify the influence of probe strength, we conduct an experiment that probes with control tasks. For each question, we generate a random number c that shares the same digit with a and b as the control signal. If the probing performance comes from the encoded number values rather than probe strength, there would be a clear gap between the probing performance on c and a, b.

Figure 14 shows the difference between probe performances. It can be observed that probing on input numbers constantly yields better performance than probing on random control signals, proving that language models do encode number values in their hidden states.

Meanwhile, probing b on positions before b shows performance similar to probing c, which corresponds to the fact that b is unknown to the model at these positions.

Patching	Result	Explanation
None	6912	5678+1234=6912
Full	11233	9999+1234=11233
$5 \rightarrow 9$	10912	9678+1234=10912
$6 \rightarrow 9$	7212	5978+1234=7212
$7 \rightarrow 9$	6932	5698+1234=6932
$8 \rightarrow 9$	6913	5679+1234=6913

Table 2: Patching results on the question "Question: What is the sum of 5678 and 1234 ?" by patching the activation on layer 8.



(a) ρ of LLaMA-2-7B. (b) ρ of LLaMA-2-13B. (c) ρ of Mistral-7B.

(d) R^2 of LLaMA-2-7B. (e) R^2 of LLaMA-2-13B. (f) R^2 of Mistral-7B.

(g) MSE of LLaMA-2-7B. (h) MSE of LLaMA-2-13B. (i) MSE of Mistral-7B.

Figure 13: The Pearson coefficient (ρ), out-of-sample R^2, and mean square error (MSE) of probes on partial number sequence of 8-digit numbers. The y-axis represents the index of number tokens in the token sequence.

H Detailed Experiments on Activation Patching

Table 3 shows the results of patching on layer 8 of Mistral-7B on the question "Question: What is the sum of 5678 and 1234 ?"

We can clearly see that patching a digit will only influence the value of the digit itself, rather than the value of the partial token sequence: patching the last digit 8 in 5678 equals changing the number to 5679 rather than 9999, although the encoded value of 9999 can be found in the activation. We hypothesize that language models encode the number values from scratch at every new position, rather than using previous encoded values.

We also notice that patching the last number digit on early layers shows a higher effect than expected, but the reason why the last digit is more special is still unknown.

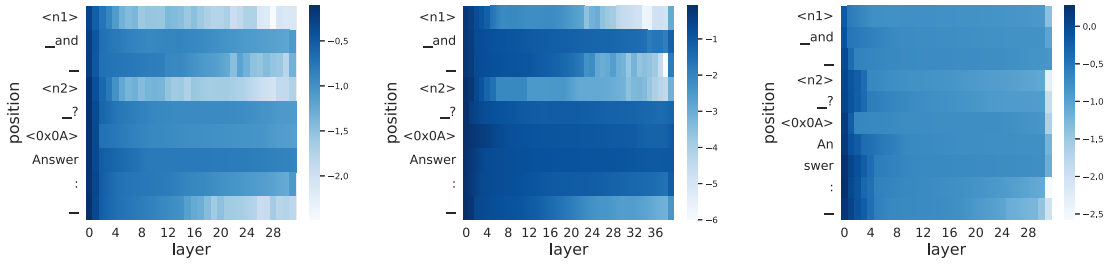
I Detailed Experiments on Linear Intervention

I.1 Success Rate

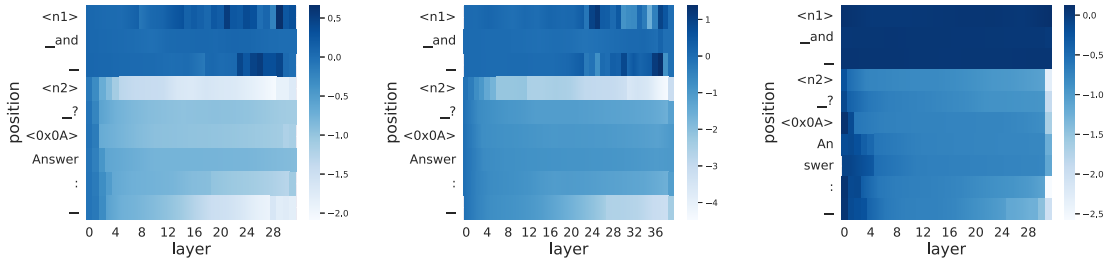
Figure 15 shows the success rate of intervening on 5 consecutive layers with a maximum success rate of 0.698, and Figure 16 shows the success rate of intervening on a series of layers starting from layer 14. It can be observed that a sufficient number of layers need to be intervened for language models to successfully change their predictions. Nanda et al. (2023) observed a similar phenomenon in OthelloGPT, and a related hypothesis is that language models demonstrate the Hydra effect (McGrath et al., 2023), where other layers would self-repair the intervention on certain layers.

I.2 Output Patterns

We also observe that while intervening on early or late layers both lead to poor success rates, they dis-



(a) a versus c of LLaMA-2-7B. (b) a versus c of LLaMA-2-13B. (c) a versus c of Mistral-7B.



(d) b versus c of LLaMA-2-7B. (e) b versus c of LLaMA-2-13B. (f) b versus c of Mistral-7B.

Figure 14: The difference in mean square error (MSE) between probes on input numbers and control signals. A lighter color indicates a greater performance gap.

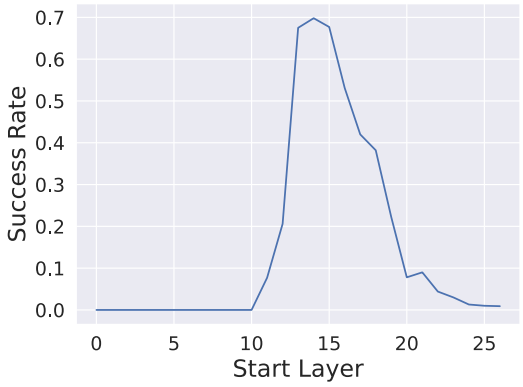


Figure 15: The success rate of performing a linear intervention on 5 consecutive layers.

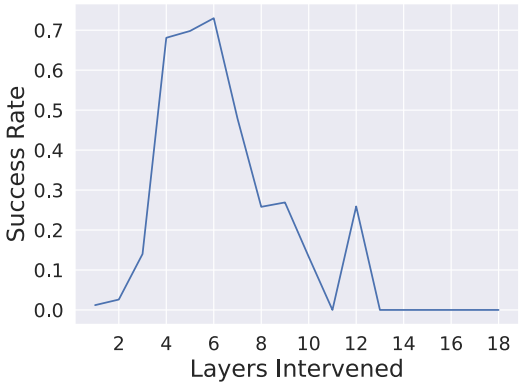


Figure 16: The success rate of performing a linear intervention on layers starting from layer 14.

play different patterns of output. Table 3 shows the result of intervening on different layers of Mistral-7B. It can be seen that performing a linear intervention on early layers would completely destroy the final outcome, while intervening on late layers will not change the result at all. We hypothesize that the number encoding in early layers has not fully developed yet, and intervening in it would lead to unexpected results; In late layers, the number encoding is simply remembered but not used, and the language models rely on other subspace to decode the final outcome.

I.3 Additional Experiments

We have also tried to change the probed number from the original value o to a new value $o + o'$:

$$\mathbf{h}_i \mathbf{W}_i + b_i = o \tag{10}$$

$$\mathbf{d}_i = o' \frac{\mathbf{W}_i}{|\mathbf{W}_i|^2} \tag{11}$$

$$(\mathbf{h}_i + \mathbf{d}_i)\mathbf{W}_i + b_i = o + o' \tag{12}$$

However, the intervention does not yield results as expected: the intervened model continues to predict o rather than $o + o'$.

A possible hypothesis is that the probed number value is the projection of $\mathbf{h}_i$ along the direction

Layer	Generation Result
0-5	Answer: gainedcnt I I I I I I I I I C C C
14-19	Answer: 12515
25-30	Answer: 6455

Table 3: Intervention results on the question "Question: What is the sum of 2936 and 3519 ?". Running Mistral-7B without intervention would lead to the result of 6455.

$\mathbf{W}_i$, and simply adding vectors to $\mathbf{h}_i$ would draw it away from its valid subspace. To maintain intervened $\mathbf{h}_i$ in its valid subspace, it should be rotated along certain direction. The method of precisely changing the encoded number values in language models still remains to be explored.

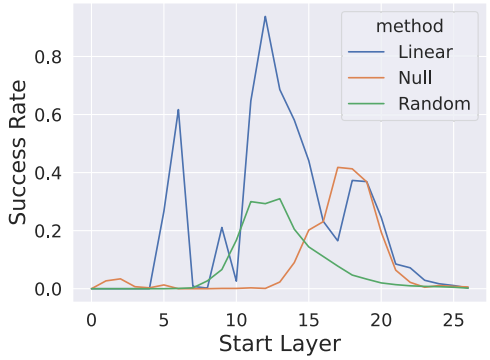


Figure 17: The success rate of performing a linear intervention on 6 consecutive layers, with a negative $\alpha = -2.0$

We also experimented on negative α values, which will "push" the residual stream towards a smaller encoded number. The results are demonstrated in Figure 17. We can see that the trend of success rate is similar to the trend in Figure 8, further proving that the value of calculation result can be linearly intervened.

J Directly Calculate with Encoded Number Values

We are curious about whether the probed number values could help LLMs better perform calculations. Considering that adding the probed input numbers does not yield precise answers (Section 3.1), we evaluate the sum of probed numbers with two new metrics: logMSE and error margin.

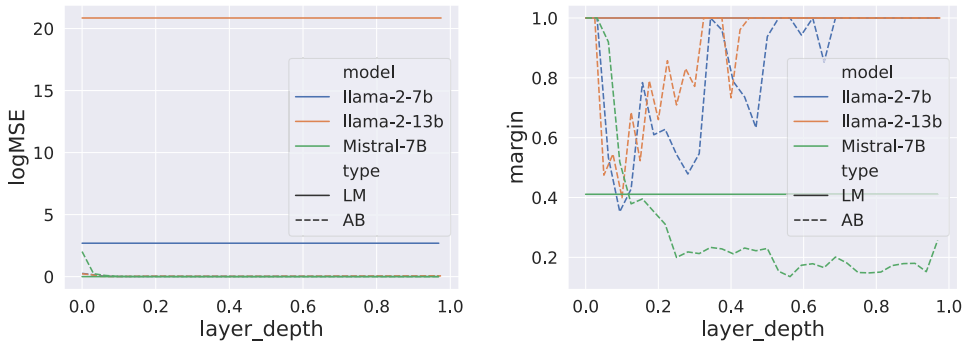
$$\text{logMSE}(\mathbf{S}, \mathbf{G}) = \text{avg}((\log_2 \mathbf{S} - \log_2 \mathbf{G})^2) \quad (13)$$

$$\text{margin}(\mathbf{S}, \mathbf{G}) = \min(\frac{\max(|\mathbf{S} - \mathbf{G}|}{\mathbf{G}}), 1) \quad (14)$$

where $\mathbf{S}$ and $\mathbf{G}$ represent predicted answers and golden answers respectively. Both metrics indicate how much the calculated results deviate from the golden answers.

In Figure 18, despite failing to generate accurate answers, all three models could keep their logMSE and error margin at a very low level by adding probed a and b, while directly accepting the output of language models would lead to results that deviate far away from the golden answers. We think that this reveals a possibility to control the computational error of language models within a reasonable range, and will not produce results that are far too unreasonable.

We also notice that for LLaMA-2 models, adding the probed number on late layers will result in a high error margin, which may be a result of the findings in Section 4.1: number encoding on late layers is not used by the model.



(a) logMSE

(b) Error margin

Figure 18: Comparison between the sum of probed (a, b) and language model predictions. AB means the sum of probed (a, b) and LM means language model predictions.