Language Models Understand Numbers, at Least Partially

Anonymous ACL submission

Abstract

Large language models (LLMs) have exhibited impressive competence in various tasks, but their opaque internal mechanisms hinder their use in mathematical problems. In this paper, we study a fundamental question: whether language models understand numbers, a basic element in math. Based on an assumption that LLMs should be capable of compressing numbers in their hidden states to solve mathematical problems, 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 compressed numbers in LLMs. However, it is difficult to precisely reconstruct the original numbers, indicating that the compression process may not be lossless. Further experiments show that LLMs can utilize encoded numbers to perform arithmetic computations, and the computational ability scales up with the model size. Our preliminary research suggests that LLMs exhibit a partial understanding of numbers, offering insights for future investigations about the models' mathematical capability.

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), as the ability of drafting solutions is not explicitly learned under the pretraining objective. Large language models like GPT series (OpenAI, 2023) and PaLM (Anil et al., 2023) have achieved satisfactory results on various mathematical benchmarks. However, smaller models encounter challenges in mathematical problems, and even fail on simple questions. Due to the opaque internal mechanisms of LLMs, the underlying cause for this limitation still remains unknown.

Numbers are fundamental elements in math, and the way how LLMs understand numbers can essentially impact the final outcome. In order to accurately answer mathematical problems, LLMs need the ability of **understanding** the numbers in the input text and **utilizing** them for calculations.

If LLMs function as pure text generators that memorize textual correlations, they will not really understand or utilize numbers for mathematical problems. However, if LLMs can learn the compact generative process underlying the training data (Gurnee and Tegmark, 2023) and compress input numbers, they will have potential to utilize the compressed numbers for math reasoning. Figuring out whether LLMs are able to understand and utilize numbers can provide critical insights for advancing research on mathematical problems.

In this paper, we explore whether LLMs are capable of understanding numbers and compressing these numbers in their hidden states. In order to prove the existence of compressed numbers, we construct a synthetic dataset comprising simple addition problems, whose numbers range from 2 to 10 digits to cover different magnitudes of numbers. LLaMA-2 models (Touvron et al., 2023b) and Mistral-7B (Jiang et al., 2023) are selected as backbone models, and the questions are fed into the models without fine-tuning to obtain the hidden states of each layer.

Following previous probing work (Alain and Bengio, 2016; Gurnee and Tegmark, 2023), we train linear probes on the hidden states of LLMs to predict the numbers in the input text. Afterwards, we also compare the sum of probed input numbers with the output of language models, which could reflect whether language models utilize the compressed numbers to perform calculations.

We conduct a series of experiments on the synthetic dataset, and the probing results demonstrate that language models are aware of the value of input numbers from early layers, while the precision of prediction seems to be gradually decreasing on deeper layers. However, we find it difficult to precisely reconstruct original numbers from the probes. We hypothesize that language models have learned their own approach to compress numbers in the input text, while the process is not yet lossless.

We also discover that language models are capable of utilizing the compressed numbers to perform arithmetic calculations, and the ability scales up with the model size. Moreover, how well a model understands and utilizes numbers seems to be positively related to its mathematical capability. These discoveries may reveal future directions for utilizing the encoded numbers, 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 have the ability to understand and utilize numbers to accurately answer mathematical problems, and construct a synthetic dataset to analyze the language models. (2) We utilize linear probes to probe the existence of compressed numbers in hidden states and discover that language models do understand the value of numbers, but can not guarantee a lossless compression. (3) We discover that language models can utilize compressed numbers to perform arithmetic calculations, and the ability is enhanced as the model scales up.

2 Probing Numbers in Language Models

2.1 Problem Formulation

Given that there is a number x in the input text t, we assume that a language model LM can compress 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:

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

where f_i refers to the process of compression regarding layer i, and $t - x$ refers to the remaining part in t that is independent of x.

If f_i is lossless, there exists 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:

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

2.2 Dataset Construction

To investigate whether LLMs understand numbers, we construct a dataset containing addition problems with different magnitudes of numbers. The dataset contains problems whose numbers range from 2 digits to 10 digits, with each digit corresponding to 1000 problems. We split the dataset into training, validation, and test sets at a ratio of 80%/10%/10%. Let a and b be randomly generated numbers, and each question is formulated as follows:

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

2.3 Probing

Obtaining Hidden States. We choose a series of LLaMA-2 models (Touvron et al., 2023b) and Mistral-7B (Jiang et al., 2023) to investigate. Among them, Mistral-7B demonstrates stronger math ability. We feed the question text in Section 2.2 into the models, and save the hidden states at the end of input tokens of all layers. For each layer, we obtain a set $\mathbf{H} \in \mathbb{R}^{n \times d_{model}}$ of hidden states, 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 $\mathbf{W} \in \mathbb{R}^{d_{model} \times 1}$ 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 control the size difference, which guarantees the numerical stability of probes.

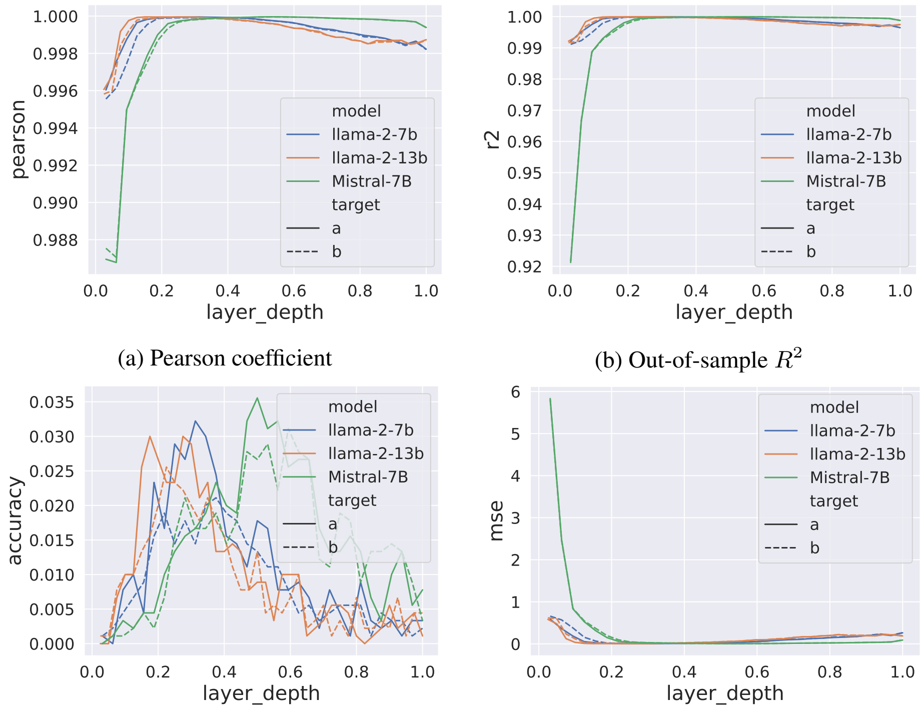
We utilize Ridge regression, which adds L2 regularization to the vanilla linear regression model, to construct the probes:

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

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.

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



(a) Pearson coefficient

(b) Out-of-sample R^2

(c) Exact accuracy

(d) Mean square error

Figure 1: Pearson coefficient, out-of-sample R^2, exact accuracy, and MSE of probes on different layers. a and b refer to the two input numbers denoted in Section 2.2 respectively.

variable that can be explained by the independent variable, and the Pearson coefficient ρ which measures the linear correlation between two variables.

Meanwhile, as mathematical problems require a precise understanding of numbers, we introduce two additional metrics to examine whether a model can compress the number losslessly. **Exact accuracy (eAcc)** evaluates whether the predicted number is exactly the same as the original number. High eAcc indicates that the compression of numbers is more likely to be lossless. **Mean square error (MSE)** is the average squared difference between predictions and actual values. Smaller MSE means lower loss during the compression.

$$\mathrm{eAcc}(\mathbf{P}^*, \mathbf{X}) = \frac{|[2^{\mathbf{P}^*}] == \mathbf{X}|}{|\mathbf{X}|} \tag{4}$$

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

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 a 4-beam beam search with a max new token restriction of 50 during decoding. The regularization strength is set to $\lambda = 0.1$ for all probes.

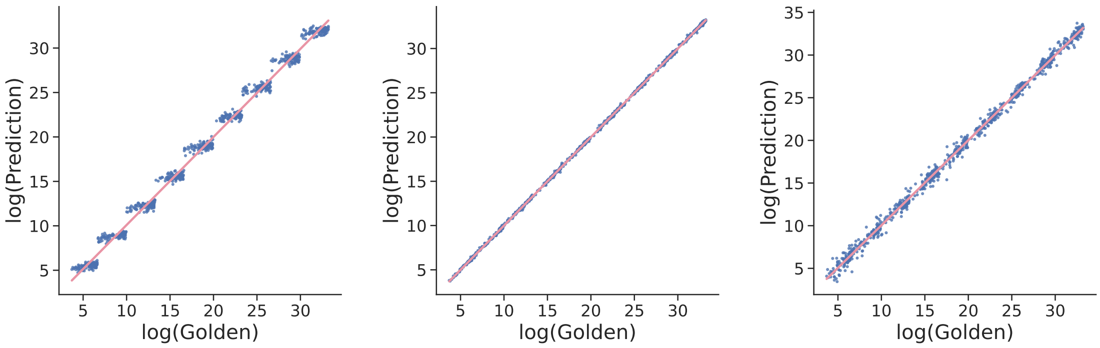
3 Do LLMs Understand Numbers?

3.1 The Existence of Compressed Numbers

LLMs do understand numbers. We first inspect into 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 compress numbers in their hidden states. As illustrated in Figure 1, the probes achieve surprisingly high ρ and R^2 on all layers, proving that the hidden states of LLMs contain the compressed form of input numbers, and the compression starts from even the first layer.

The compression may not be lossless. It is interesting to notice that both scores of LLaMA-2 models gradually drop when the layer gets deeper, which may be a reminder that language models would "forget" the precise value of numbers. To verify the hypothesis, we calculated the eAcc and MSE of different probes, whose results are shown in Figures 1c and 1d.

In contrast to high correlation coefficients, the eAcc is below 5% on all layers, which means that the linear probes struggle with precisely reconstructing the input numbers. The trends in eAcc and MSE are consistent with the Pearson coefficient,



(a) Layer 2 (b) Layer 10 (c) Layer 30

Figure 2: How the prediction results of probes on the second input number b change as the layer gets deeper.

indicating that LLaMA-2 models achieve the most precise number encoding in intermediate layers, but fail to maintain it in deeper layers. In contrast, Mistral-7B maintains the precision in deep layers.

Model family rather than scale matters more. In the LLaMA-2 family, the 13B model does not show any advantage over the 7B model, and their score curves are almost identical. In contrast, Mistral-7B achieves generally higher scores on ρ and R^2, especially on deep layers, which is consistent with its outstanding math ability. The difference implies that the ability to compress numbers is consistent across different model scales, but vary between different model families. Meanwhile, the ability of understanding numbers show a positive correlation with the math ability of LLMs.

In summary, LLMs clearly demonstrate an ability to understand and compress input numbers. The compression show the highest precision at intermediate layers, but reconstructing the accurate numbers with linear probes is still difficult.

3.2 The Process of Compression

To better analyze how language models compress numbers, we pick distinct layers in the LLaMA-2-7B model and observe how the prediction of probes change as the layer gets deeper. Layer 2, 10, and 30 are selected to represent early, intermediate, and late layers respectively. The trend of change on the second input number b is shown in Figure 2.

On early layers like layer 2, the predictions of probes are distorted to some extent: for original numbers with the same digit, 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 of passing compressed numbers to later layers may not be lossless, which leads to errors in the final number encoding of language models.

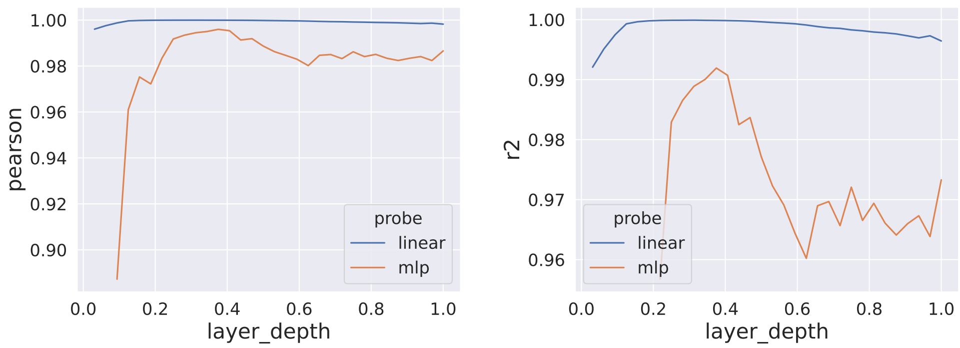
3.3 LLMs Compress Numbers Linearly

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 in a linear manner, we follow the example of Gurnee and Tegmark (2023) to train two-layer MLP probes and compare their performance with linear probes. The MLP probes have an intermediate hidden state of 256 dimensions and can be formulated as:

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

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

Figure 3 demonstrates the results of MLP probes compared with linear probes. We find that nonlinear MLP probes do not show clear advantages over



(a) Pearson coefficient

(b) Out of sample R^2

Figure 3: The comparison between linear probes and MLP probes.

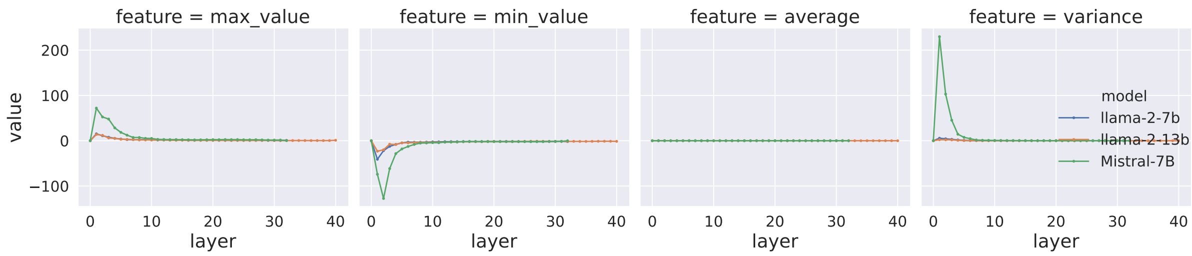


Figure 4: Features of probe coefficients.

linear probes, and even seem to be constantly performing worse across different layers. We think the phenomenon proves that the encoding of numbers can be represented linearly, or at least near-linearly.

3.4 No "Math Dimensions" in LLMs

Aside from the existence of encoded numbers, the features of probes are also worth attention: Are the encoded numbers stored mainly on a few specific "math dimensions"? Or do the probes have extremely large or small values in their weights?

Figure 4 demonstrates the minimum value, maximum value, average value, and variance of probe weights on different layers. Extreme minimum and maximum values appear in probes on very early layers, but rapidly converge to near 0 on subsequent layers, and so is the variance. Meanwhile, the average value always stays around 0, which indicates that the information of numbers is uniformly distributed across all dimensions of hidden states, and there do not exist specific "math dimensions".

4 Do LLMs Utilize Numbers?

Aside from probing numbers in the input text, we are also interested in whether models utilize the compressed numbers to get their "calculation results". As shown in Figure 5a, while the models achieve satisfying accuracy on small numbers, the accuracy faces a obvious decline from 5-digit addition problems, and the LLaMA-2-7B model even nearly fails to answer every 10-digit problem.

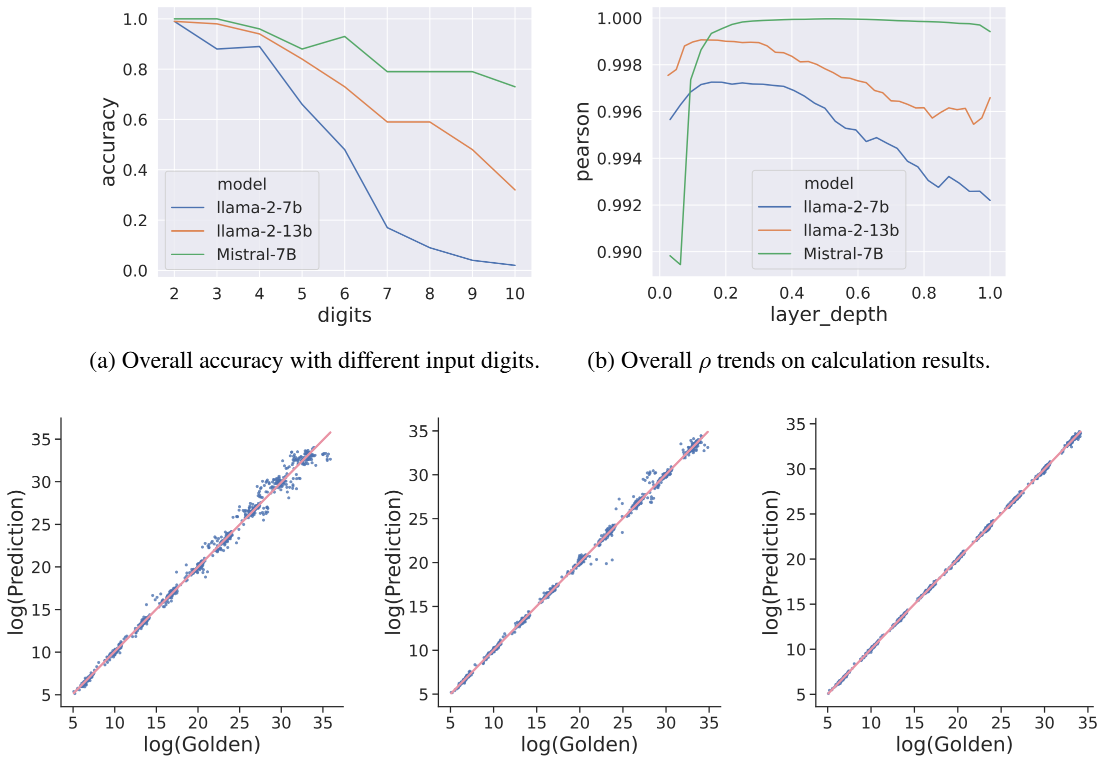
In this section, we perform further analysis to investigate what leads to the errors in large number addition problems, and whether a model can utilize compressed numbers for arithmetic calculations.

4.1 The Ability of Calculation

We believe that a model can be claimed to be capable of performing arithmetic calculations only if its output can be probed. In fact, LLMs seem to be conscious of their calculation results, regardless of the correctness of the results. As demonstrated in Figure 5b, the Pearson coefficients are still high enough to prove the existence of calculation results.

While the ability to encode numbers is consistent across models of different scales (See Section 3.1), they show a greater difference in the ability of calculation. The LLaMA-2-7B model is clearly falling behind the 13B model from early layers, and the gap continuously widens as the layer gets deeper. However, Mistral-7B still surpasses both LLaMA-2 models, displaying a better understanding of its own calculation results.

Figure 5c and Figure 5d give a more detailed



(a) Overall accuracy with different input digits.

(b) Overall ρ trends on calculation results.

(c) Layer 10 of LLaMA-2-7B

(d) Layer 10 of LLaMA-2-13B

(e) Layer 10 of Mistral-7B

Figure 5: Probes on the calculation results of LLMs.

depiction of the gap. The probes fail to read out calculation results of large numbers in the LLaMA-2-7B model, where the outputs of large number additions seem to be going random, while the outputs of the 13B model can still be roughly probed. From this perspective, the calculation ability of the LLaMA-2-7B model is limited to small numbers, but the upper bound of a model's calculation ability displays a positive correlation with its scale. The outputs of Mistral-7B aligns near-perfectly with the probes, proving that a well-curated model could also enhance the calculation ability, and the ability in utilizing numbers is also positively related to mathematical competency.

4.2 Computational Error

The error of a wrong prediction could be rooted in either number encoding or computation. To seek the root of errors, we construct an extra dataset consisting of 10000 8-digit addition problems, which are difficult for all models. We take a closer look at computational errors by observing only questions that the models answer incorrectly.

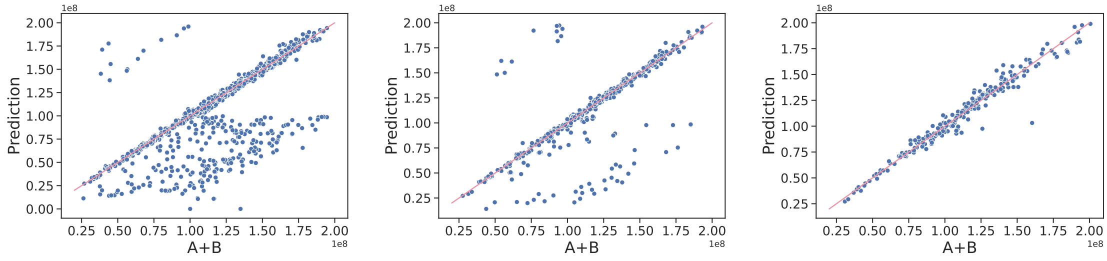
First, we try to analyze the correlation between the sum of probed input numbers $a + b$ and the prediction generated by language models. It can be clearly observed in Figure 6 that there exist outlier points, which refer to situations where language model outputs deviate far from the sum of probed input numbers.

This phenomenon happens across both LLaMA-2 models, but there are fewer outliers in the LLaMA-2-13B model than the 7B model. There are even fewer outliers in Mistral-7B, and its maximum deviation range is significantly smaller than LLaMA-2 models. The observation matches the findings in Section 4.1, indicating that language models have difficulty performing arithmetic calculations on large numbers, but the calculation ability will enhance as the model scales up.

Second, we are curious about whether the probed number encoding 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.

$$\text{logMSE}(\mathbf{S}, \mathbf{G}) = \text{avg}((\log_2 \mathbf{S} - \log_2 \mathbf{G})^2) \quad (7)$$

$$\text{margin}(\mathbf{S}, \mathbf{G}) = \min(\frac{\max(|\mathbf{S} - \mathbf{G}|}{\mathbf{G}}), 1) \quad (8)$$



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

Figure 6: Relation between probed $a + b$ and language model outputs, where language models answer the question incorrectly. AB means the sum of probed $a + b$ and LM means language model predictions.

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 7, 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.

5 Discussion and Future Directions

In previous sections, we find that LLMs are able to understand the value of numbers and utilize the compressed numbers to perform calculation. However, the compression may not be lossless, and the calculation ability scales with model size. Moreover, the ability to understand and utilize numbers are positively correlated to the 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 exactly 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. It seems that LLMs are currently not able to losslessly compress numbers, and the compression loss will inevitably bring errors to subsequent computation, especially when the input numbers are large. 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.

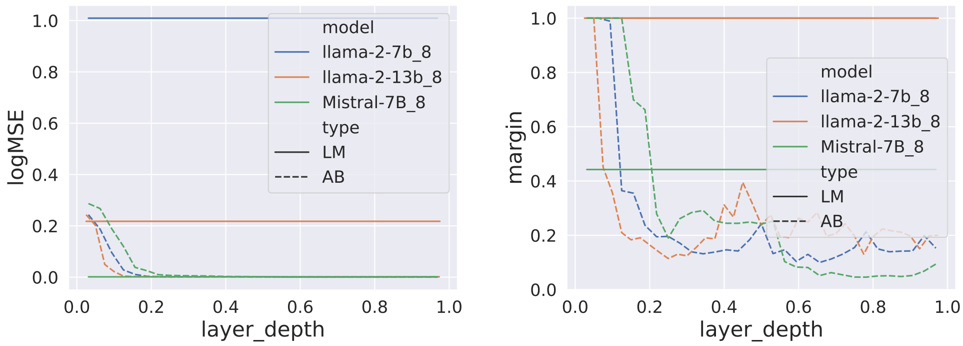
Mitigating computational errors with compressed numbers. In Section 4.2, we reveal the potential of utilizing probed numbers to control computational errors. By adding modules that directly utilize the compressed numbers in language models, the computational errors may be further reduced, especially on large-number calculations.

6 Related Work

6.1 LLMs 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), multiple methods have been explored to help LLMs better solve these questions.

Wei et al. (2022) proposes chain-of-thought reasoning to enhance the reasoning ability of LLMs by breaking the reasoning process into substeps. The self-consistency (Wang et al., 2022) technique samples multiple reasoning paths and then performs majority voting to select the final answer. MetaMath (Yu et al., 2023) rewrites the original question from multiple perspectives to enable the transfer of meta-knowledge. Math-Shepherd (Wang et al., 2023) assigns a reward score to each step in multi-



(a) logMSE

(b) Error margin

Figure 7: Comparison between probed $a+b$ and language model predictions. AB means the result of probed $a+b$ and LM means language model predictions.

step mathematical problems, outperforming other verification models without manual annotation.

6.2 Probing Language Models

Prior research has unveiled that language models are able to understand certain information and store the information in their hidden states. 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 explicitly represented with a linear projector. Li et al. (2021) claims that language models are able to encode the properties and relations of entities in a defined scenario, and these representations can be linearly decoded. Gurnee and Tegmark (2023) reveals evidence that large language models build spatial and temporal representations about a certain entity from early layers.

To better interpret the inner structures of language models, probing (Alain and Bengio, 2016; Belinkov, 2022) is a common technique that attempts to reconstruct features from the hidden states, and multiple pieces of work (Alain and Bengio, 2016; Nanda et al., 2023; Li et al., 2021; Gurnee and Tegmark, 2023) indicate that the features can be read out with simple linear projectors.

7 Conclusion

In this paper, we dive into the question of whether large language models understand numbers. Based on the premise that LLMs have to precisely represent a number to perform accurate mathematical calculations, we assume that LLMs are capable of understanding numbers and compressing numbers in their hidden states via a low-loss approach. We construct a dataset consisting of simple addition problems and introduce linear probes to investigate whether language models understand numbers.

Experimental results prove that LLMs do have a rough estimation of input numbers, but the compression process may not be lossless. The ability to encode numbers is consistent across different model scales, and the encoding seems to be the most precise in intermediate layers, while the error gets larger in deeper layers. Further experiments show that LLMs exhibit the ability to utilize compressed numbers to perform arithmetic calculations, and the ability scales up with model size. By comparing the LLaMA-2 models with Mistral-7B, we also find that how well a model understands and utilizes numbers is positively related to its mathematical capability.

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.

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 open-sourced LLMs, while close-sourced LLMs like the GPT series have the probability of exhibiting 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*.

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*.

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*.

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.

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.

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*.

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*.

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 of digit. For questions whose number has 4 or less 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 20 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 follow the term of use of all models, and use them only for research.

C Prompt Robustness

Previous sections have proved the existence of compressed numbers in language models, whereas whether the encoding is consistent across different prompt contexts remains a mystery.

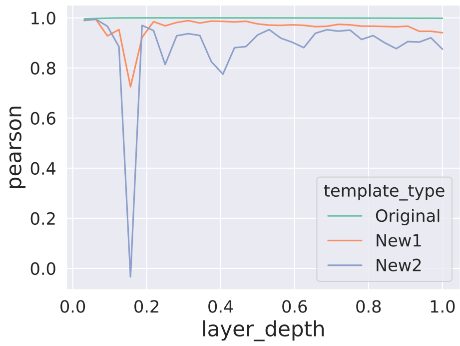
To investigate this issue, we first formulate the question text with 2 more different templates:

$$\{a\}\ +\ \{b\} = \dots$$
$$\{a\}\ adds\ \{b\}\ equals \dots$$

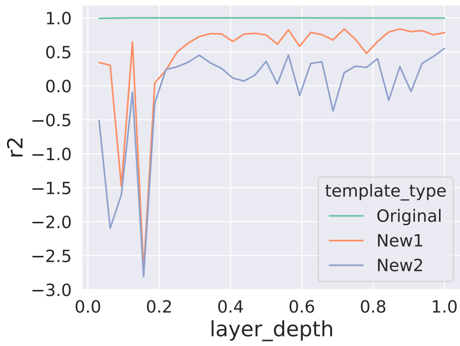
Afterwards, we feed the new prompts into the LLaMA-2-7B model to obtain new hidden states $\mathbf{H}^{'}$, and test whether the probes trained on the original dataset could detect the first number a in $\mathbf{H}^{'}$.

Figure 8 reports the overall probing results on the 2 new templates. Despite not trained on the new prompts, the original probes still shows a degree of competence, maintaining the Pearson coefficient above 0.8 on most layers. However, the probes still significantly deteriorate compared with their performance on the original prompt, especially on the early layers. One possible explanation for this phenomenon is that number may be a high-level feature in language models. The compressed numbers are only detectable when the model has finished collating the low-level representations, which is identical to the trend in Figure 1.

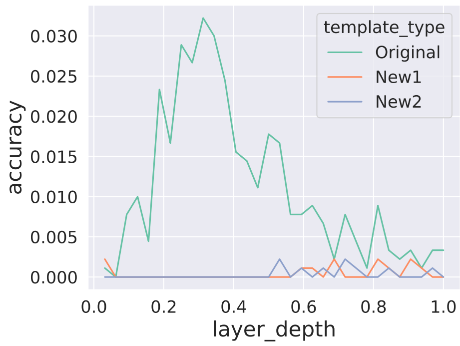
Figure 9 proposes a more detailed view about the prediction results on Layer 10, where the metrics begin to stabilise. Aside from the apparent noise, a more interesting discovery is that despite the overall linear relationship between predicted and golden values, the slopes of the fitted linear functions are not equal to 1, and their intercepts are not equal to 0 either. We conjecture that the compressed numbers may be projected into different spaces depending on their different context, which could explain the phenomenon.



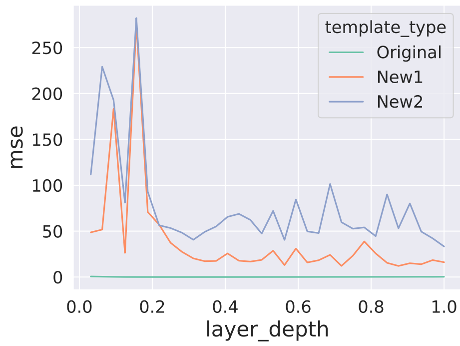
(a) Pearson coefficient



(b) Out-of-sample R^2

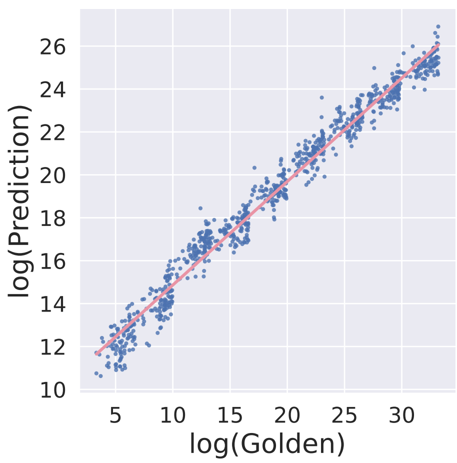


(c) Exact accuracy

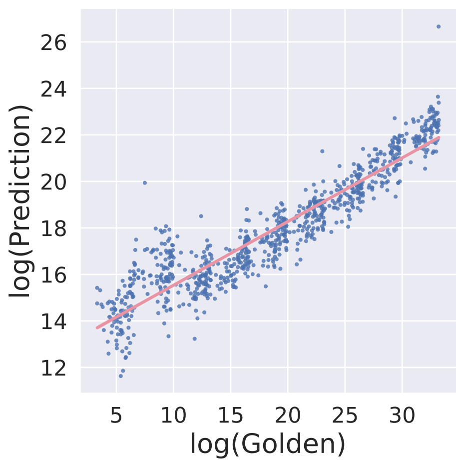


(d) Mean Square Error

Figure 8: Pearson coefficient, out-of-sample R^2, exact accuracy, and MSE of probes on the first input number a on different layers for alternative prompts.



(a) Prediction results for new prompt 1.



(b) Prediction results for new prompt 2.

Figure 9: Prediction results of probes on the first input number a on layer 10 of LLaMA-2-7B with 2 different new prompts as input.