# LLMs Know More About Numbers than They Can Say

**Fengting Yuchi**[1] **Li Du**[2] **Jason Eisner**[2]
[1]Shanghai Jiao Tong University and [2]Johns Hopkins University
yc1114@sjtu.edu.cn and {leodu,jason}@cs.jhu.edu

## Abstract

Although state-of-the-art LLMs can solve math problems, we find that they make errors on numerical comparisons with mixed notation: "Which is larger, $5.7 \times 10^2$ or 580?" This raises a fundamental question: Do LLMs even know how big these numbers are? We probe the hidden states of several smaller open-source LLMs. A single linear projection of an appropriate hidden layer encodes the *log-magnitudes* of both kinds of numerals, allowing us to recover the numbers with relative error of about 2.3% (on restricted synthetic text) or 19.06% (on scientific papers). Furthermore, the hidden state after reading a *pair* of numerals encodes their *ranking*, with a linear classifier achieving over 90% accuracy. Yet surprisingly, when explicitly asked to rank the same pairs of numerals, these LLMs achieve only 50–70% accuracy, with worse performance for models whose probes are less effective. Finally, we show that incorporating the classifier probe's log-loss as an auxiliary objective during finetuning brings an additional 3.22% improvement in verbalized accuracy over base models, demonstrating that improving models' internal magnitude representations can enhance their numerical reasoning capabilities. Our code is available at github.com/VCY019/Numeracy-Probing.

## 1 Introduction

Large language models (LLMs) are increasingly used in mathematical (Jiang et al., 2023), scientific (Taylor et al., 2022; Zhang et al., 2025), financial (Wu et al., 2023), and engineering (Jimenez et al., 2024) domains. In these domains, numeracy—the ability to understand and reason about *numbers*—is an essential basic skill.

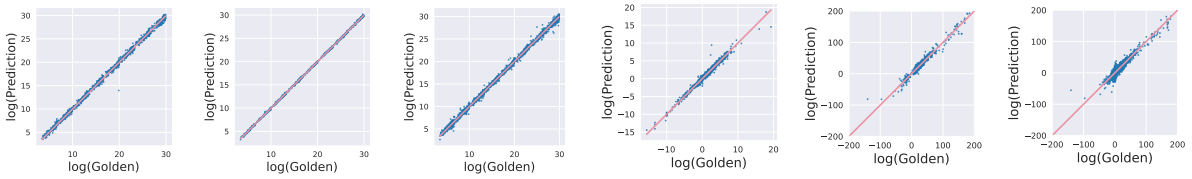
While LLMs can correctly answer questions such as "What is $5.7 \times 10^2$?" and "Which is larger, 570 or 580?", we find that even leading non-reasoning models like GPT-4.1 make many mistakes on cross-notation comparisons like "Which is larger, $5.7 \times 10^2$ or 580?" This discrepancy raises questions: Do LLMs know how big these numbers are, representing them internally in a way that supports intelligent discussion of scientific text? Or do they lack number sense and are specialized to specific manipulations of *numerals*—the textual representations of numbers—in specific notations?

| Model | Verbalization Accuracy | Probing Accuracy |
|---|---|---|
| GPT-4.1 | 94.19 | — |
| GPT-4.1-mini | 92.12 | — |
| DeepSeek-R1-Distill-Llama-8B | 57.81 | 95.62 |
| DeepSeek-R1-Distill-Qwen-7B | 64.19 | 97.38 |
| Llama-2-7b | 50.81 | 98.44 |
| Llama-3.1-8B-Instruct | 55.06 | 94.81 |
| Mistral-7B-v0.1 | 50.00 | 96.44 |
| OLMo-2-1124-7B | 53.44 | 93.50 |
| Qwen3-8B | 70.00 | 98.88 |

Table 1: Cross-notation comparison task on held-out data (see App. B.1 for data details). **Verbalization accuracy** evaluates verbal responses using a one-shot prompt and no training, while **probing accuracy** measures performance of a linear classifier trained on hidden states. All values are percentages without the % sign. Despite hidden states containing rather accurate comparison information extractable via linear probes, verbalization performs substantially worse.

To study these questions, we first probe the internal numerical knowledge of smaller LLMs (7B–8B parameters) by training linear probes on hidden states of numeral tokens, following Zhu et al. (2025). We train and test on two kinds of text: synthetic prompts and real-world arXiv papers. We find that standard linear regression can successfully predict log-magnitudes (§3.1). For example, Mistral-7B has layers that linearly encode numerals to ~2.3% median relative error on synthetic data and ~19.06% on scientific papers.

Second, we show that these smaller LLMs also internally *compare* numbers (§3.2). A linear binary classifier (trained with logistic regression) reveals which of the two numbers in the question is bigger, even when the numbers are so close that their predicted magnitudes (using the previous method) are too noisy for reliable comparison.




(a) Synth, dec (18) (b) Synth, sci (31) (c) Synth, mixed (30) (d) ArXiv, dec (5) (e) ArXiv, sci (5) (f) ArXiv, mixed (10)

Figure 1: Scatterplots of predicted vs. true ("golden") log-magnitudes for Mistral-7B (32 layers) across different datasets and notation types. "Synth" refers to the synthetic cross-notation data we constructed in §2. Dec/sci/mixed conditions train and test on decimals only, scientific notation numerals only, and both, respectively. We train and test a probe at each layer, and then plot performance on held-out test data for only the probe that achieved the highest $R^2$ on held-out validation data. The parenthesized number is the layer index of that probe.

Despite having done these computations internally, these LLMs struggle significantly at *verbalizing* the cross-notation comparison (§3.3). When prompted to say which of two cross-notation numerals is larger, the models achieve only 50–70% accuracy—far closer to the 50% random baseline than their internal knowledge would predict. This sharp performance gap indicates that LLMs possess numerical knowledge that they cannot reliably verbalize through language generation.

Still, probing effectiveness in early layers does *correlate* well with verbalized performance (§3.4). For a probe of either type (applied to the first 3 layers), probe accuracy predicts the accuracy of verbal output (produced by the last layer) across diverse 7B–8B LLMs (Figs. 5, 6 and 9).

To test whether this connection is causal, we experiment with adding the classifier probe's training loss to the LLM's log-loss training objective on the task of verbalized cross-notation comparison (§3.5). Most LLMs do achieve higher verbalized comparison accuracy when fine-tuned with this augmented objective rather than the standard objective (Table 2). This finding suggests that targeting internal representations is a promising approach to improving numeracy in LLMs. That is, models know more than they can say—but improving what they know improves what they can learn to say.

To summarize our key findings:

- Numerical log-magnitudes and magnitude comparisons are encoded linearly, for various LLMs, datasets, and numeral notations.
- Even so, LLMs verbalize cross-notation comparisons poorly.
- Tuning a model's internal representations to better encode numeric comparisons helps the model better answer such comparison questions verbally.

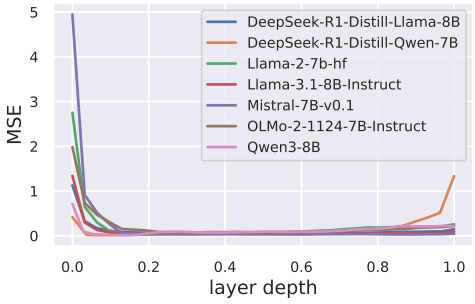



Figure 2: MSEs in log-space of regression probes on cross-notation data of each LLM across layers.

## 2 Experimental Setup

**Data** We construct a synthetic `cross-notation` dataset to support our studies of both regression and comparison. To study regression probing on realistic scientific text, we derive an `arXiv` dataset from the peS2o dataset (Soldaini and Lo, 2023). See App. B for details of both datasets.

In `cross-notation`, numeral pairs are constructed so that each pair consists of one numeral in scientific notation and the other in normal form (e.g., $5.7 \times 10^2$ vs. 560).[1]

For verbalization on `cross-notation`, we prompt with a one-shot demonstration since the model has not been trained for the task (see App. C for more details):

```
Q: Which is larger, 9.9 × 10^2 or 100? A: 9.9 × 10^2
Q: Which is larger, {a} or {b}? A:
```

In contrast, we train and test our `cross-notation` probes on a zero-shot prompt (no demonstration), to better isolate the model's intrinsic encoding:

```
Q: Which is larger, {a} or {b}? A:
```

[1]We found LLMs can compare numerals of the same notation with 100% accuracy.

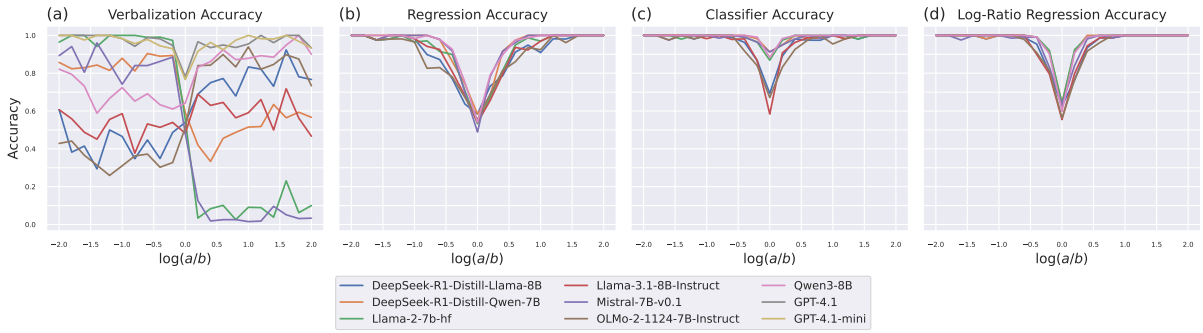



Figure 3: Accuracy of cross-notation comparison ($a \overset{?}{>} b$) versus the relative magnitude of the two numbers ($\log_2(a/b)$). (a) Verbalized comparison using one-shot prompting; (b) Comparison of the two values predicted by regression; (c) Comparison via a classification probe; (d) Comparison via the log-ratio predicted by regression. Note that (b)–(d) require access to the hidden states, so they do not include the large closed-source models GPT-4.1 and GPT-4.1-mini. See Fig. 10 for individual models' results.

**Probing** For each layer of the transformer LLM, we extract hidden states from the zero-shot prompt and train separate probes. Linear regression probes are trained on the last tokens of the numerals to predict their $\log_2$ magnitudes and the last token of the prompt to predict $\log_2(a/b)$. Linear classification probes (using logistic regression) are trained on the last token of the prompt to predict whether `{a}` $>$ `{b}`. See App. D for details.

**Finetuning** To investigate whether internal magnitude representations can be leveraged to improve numerical reasoning, we incorporate auxiliary probing loss during finetuning on `cross-notation`. We add probe heads to a single layer and finetune on the standard language modeling loss *plus* the classification probing loss,[2] $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \beta\mathcal{L}_{\text{cls}}$, where the hyperparameter $\beta$ controls the strength of auxiliary supervision. See App. E for details.

**Models** To ensure that the experimental results are not specific to a certain model's architecture or training procedure, we select a set of widely used open-weight 7B–8B LLMs, including base pretrained, instruction-tuned, and distilled models. We additionally evaluate GPT-4.1 and GPT-4.1-mini (OpenAI's non-reasoning frontier models) to assess whether verbalization difficulties persist in larger, more capable models.

## 3 Experimental Results

### 3.1 Magnitude Probing

To investigate whether LLMs internally encode numerical magnitude information, we first train linear regression probes on hidden states to predict the $\log_2$ values of numerals. Fig. 1 shows scatterplots of predicted versus true log-magnitudes for Mistral-7B across different datasets and notation types.

The strong correlation observed in Fig. 1 suggests that, on the `cross-notation` dataset, log-magnitudes of decimals and scientific notation numerals are encoded linearly in the LLM's hidden representations. Even in mixed probing, where probes are trained and tested on both decimal and scientific notation tokens, linear regressors achieve high performance on `cross-notation` with $R^2 > 0.998$ (Fig. 1c).

On `arXiv` (Fig. 1f), the linear probe is not as precise, though it still achieves a correlation of $\rho = 71\%$ ($R^2 = 0.51$). Numbers in `arXiv` play many more different *semantic roles* (Lo et al., 2020, App. F). Perhaps these are represented in different subspaces, so that a single linear probe is not powerful enough.

Even so, this latter model can predict magnitudes on held-out `arXiv` data to 19.06% median relative error. Thus, this layer of Mistral-7B contains a noisy invariant linear representation of each numeral's magnitude—which the next layer's attention mechanism could learn to extract through linear query/key/value projections, providing "number sense". Note that the better 2.3% median relative error on `cross-notation` (Fig. 1c) is possible only because the models are overfitted to very specific prompt formats.[3]

[2] We experimented with adding regression probing losses but found no improvement in performance (see App. E).

[3] These models are also underdetermined because they have only seen one kind of data. They do not generalize well even within the `cross-notation` dataset: the Fig. 1a model performs much worse on the Fig. 1b data ($R^2 = 0.56$), and vice-versa ($R^2 = 0.34$).

### 3.2 Comparison Probing

Despite the regression probes' low error in log-space, their predictions are too noisy to compare close-by values. In Fig. 3b, we use the regression probe's predicted magnitudes to compare the two numbers in the prompt. We observe across all models that this method degrades when the absolute $\log_2$-ratio of the two numbers drops below 1 (i.e., when the two numbers differ by less than a factor of 2). When $|\log_2(a/b)| < 0.2$, the regression probes' predictions are too noisy to meaningfully compare the numbers, resulting in random chance accuracy.

However, the LLM's internal states also seem to include a comparison of the two numbers. We additionally train a binary classifier on the `cross-notation` dataset. Fig. 4 plots accuracy as a function of layer depth. Higher layers consistently exceed 95% accuracy. Fig. 3c shows the accuracy with respect to the log-ratio of two numbers. We observe that, while classifier probes also suffer when the two numbers are close, they generally perform better than regression-based comparison—and much better for 4 of the 7 open-source models.

The internal states encode not only which number is bigger, but how much bigger it is. We trained additional regression probes to predict $\log_2(a/b)$ from the last hidden state of the prompt. These achieved overall predictive accuracy similar to the regression probes of §3.1 (see Fig. 8 in App. G). Their accuracy may still degrade when $a \approx b$, but as $a$ approaches $b$, comparison based on predicted $\log_2(a/b) > 0$ (Fig. 3d) does not degrade quite as quickly as comparison based on predicted $\log_2(a) >$ predicted $\log_2(b)$ (Fig. 3b). Fig. 10 clearly shows that for each model, the log-ratio comparison method is more accurate at all $a/b$ values. Indeed, it behaves similarly to direct binary classification (Fig. 3c)—except for the 4 models whose binary classifiers are much more robust. Presumably those models are internally performing the actual comparison task requested by our prompt.

### 3.3 Verbalization

Table 1 shows each LLM's verbalized response on `cross-notation` using a one-shot prompt. Across a variety of 7B–8B open-weight models, the best model reaches only 70% accuracy; the worst is at chance (50%). While few-shot prompting improves performance (cf. App. H), we emphasize one-shot performance because numeracy *should be* an innate ability of LLMs and such basic questions should require as few examples as possible.

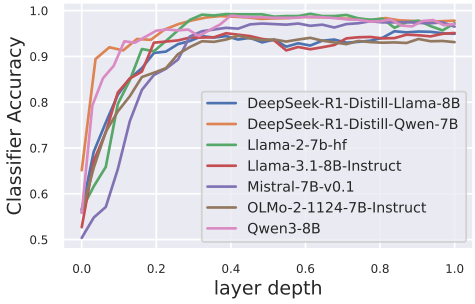



Figure 4: Logistic classifier accuracies on `cross-notation` of each LLM across layers.

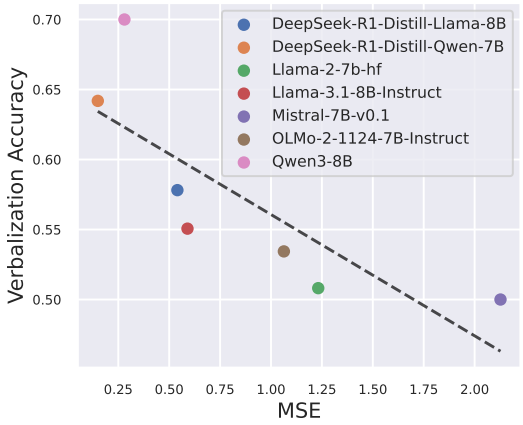



Figure 5: Average performance of linear regression probes at the first 3 layers correlates with model's verbalization accuracy on `cross-notation`.

To visualize the pattern of errors, Fig. 3a shows the verbalized comparison accuracy as $\log_2(a/b)$ changes. Though GPT-4.1 and GPT-4.1-mini achieve over 92% accuracy overall, they have errors at all ratios and especially when the two numbers are close. Interestingly, we observe that Llama-2-7B and Mistral-7B disregard the values of the numbers and almost always answer with the second number. We conduct further experiments in App. H and show that, in the one-shot setting, those two models lack awareness of the values of numbers and superficially imitate the example in the one-shot prompt. We present few-shot results and a more thorough error analysis in App. H.

### 3.4 Correlation

To explore how internal representations relate to verbalized performance, we investigate the correlation between probe performance and verbal accuracy across models.

We observe that for scientific notation numerals, the best probes (Figs. 1b–1c) are near the top of

| Model | Base | Finetuned | Error Rate Reduction | Finetuned with Probing Loss | Further Error Rate Reduction |
|---|---|---|---|---|---|
| DeepSeek-R1-Distill-Llama-8B | 57.81 | 86.69 | 68.5% | 89.69 | 22.5% |
| DeepSeek-R1-Distill-Qwen-7B | 64.19 | 84.06 | 55.5% | 95.19 | 69.8% |
| Llama-2-7B | 50.81 | 94.44 | 88.7% | 94.56 | 2.2% |
| Llama-3.1-8B-Instruct | 55.06 | 96.44 | 92.1% | 95.81 | −17.7% |
| Mistral-7B-v0.1 | 50.00 | 95.69 | 91.4% | 99.31 | 84.0% |
| OLMo-2-1124-7B | 53.44 | 87.75 | 73.7% | 90.25 | 20.4% |
| Qwen3-8B | 70.00 | 96.19 | 87.3% | 99.00 | 73.8% |
| Average | 57.33 | 91.61 | 80.3% | 94.83 | 38.4% |

Table 2: Cross-notation comparison accuracy of verbalized responses. Accuracy numbers are percentages without the % sign. Most errors of the base model are fixed by finetuning. Incorporating probing loss into the finetuning objective often fixes many of the *remaining* errors (more than $\frac{2}{3}$ of them for 3 of the models).

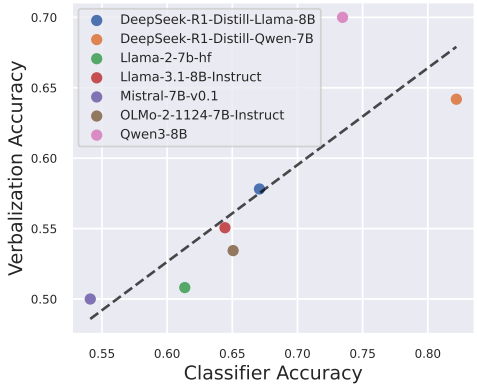



Figure 6: Average performance of logistic classification probes at the first 3 layers predicts the model's verbalization accuracy on `cross-notation`.

the transformer. These representations presumably arrive too late in processing to support the comparisons required by the verbalization task. We further observe that representations that support accurate linear regression are only available after several layers (Fig. 2), and representations that support accurate classification arrive even later (Fig. 4).

We speculate that earlier representations are more useful for downstream tasks. Indeed, Figs. 5 and 6 reveal a strong correlation *across diverse 7B–8B models* between the average performance of probes from the first three layers (even though that performance is still low) and verbalized comparison accuracy on the `cross-notation` dataset.

This correlation suggests that higher quality of internal numerical representations is linked with improved verbalization performance.

### 3.5 Finetuning

To test whether the link is causal, we incorporate probing loss as an auxiliary objective during finetuning, i.e., we jointly train the LLM and the probe. Table 2 compares three conditions: base model, standard finetuning ($\beta = 0$), and finetuning with probing loss ($\beta = 0.02$).[4] Through hyperparameter tuning, we select the probe layer depth to be 90%. $\beta > 0$ encourages the representations to better support the comparison task that will be needed by verbalization. Through backpropagation, this affects the representations at earlier layers as well.

Ordinary finetuning fixes $> \frac{4}{5}$ of the total errors made by these 7 models (in the sense of reducing error rate). But augmenting the finetuning with probing loss fixes $> \frac{1}{3}$ of the total *remaining* errors. For the 3 Mistral- and Qwen-based models, augmentation fixes $> \frac{2}{3}$ of remaining errors, bringing 2 of them to $\geq 99\%$ accuracy. This suggests the possibility of improving LLMs' verbal numeracy by explicitly improving their internal representations.

## 4 Conclusion

We study a fundamental question in LLM numeracy (see App. A for related work): Do language models know how big numbers are? We found a striking fact: While LLMs do have strong internal numerical representations that can be exposed by both regression and classification probes, they struggle to answer cross-notation comparisons, performing only marginally above chance.

We demonstrate that early-layer representations strongly predict verbalized performance, and auxiliary probing loss enhances finetuning with an additional 3.22% improvement. These results suggest that LLMs possess sophisticated numerical knowledge they cannot reliably access during generation unless fine-tuned for a task, but also that better knowledge causally leads to better generation.

Our work highlights both limitations and opportunities. Current LLMs know more about numbers than they can say; and targeted tuning on these two goals (representation, verbalization) is synergistic.

[4] See hyperparameter tuning in App. E.

## Limitations

- **Probing reveals representation, not usage.** Unlike causal tracing approaches (Lindsey et al., 2025), we did not investigate how the representations are actually used during inference. Our intervention trained the model to produce both better representations ($\mathcal{L}_{\text{cls}}$) and better text output ($\mathcal{L}_{\text{LM}}$). We found a synergy between these two objectives, but we did not intervene on the representations directly at inference time to see how this changed the text output.

- **Synthetic-real gap.** Most of our core findings—such as probing performance, verbalization failure, and probe-based finetuning gains—are demonstrated on controlled synthetic data. We do not systematically validate these results on real scientific text such as arXiv papers. The extent to which our conclusions generalize to such text (and to other aspects of numeracy) remains to be established.

- **Linear probes may not be enough.** Our tests on arXiv showed that a single linear probe will not necessarily be enough to work beyond our simple synthetic data setting. On real data, we might try using a two-layer neural network, or using the minimum loss of several linear probes that look at different subspaces. Or perhaps these nonlinearities are unnecessary if through finetuning a linear probe, we can successfully change the internal representations so that they do support linear probing after all, even on arXiv. That is, finetuning might recruit a dimension to encode numeric log-magnitude, which might improve the model's abilities in downstream verbal numeracy tasks. We leave these questions to future work.

- **Probing findings are limited to 7B–8B LLMs.** Our probing results are limited to 7B–8B open-weight models where we can access internal representations. We cannot probe GPT-4.1 and GPT-4.1-mini, so it remains unknown whether larger models encode numerical information similarly, or whether the links between internal numeracy and verbal performance are preserved.

- **Limited diversity in test examples.** Our synthetic cross-notation dataset has several constraints. First, all exponents are positive, but negative exponents like $10^{-3}$ are common in scientific literature. Second, our few-shot demonstration examples (App. C) did not include examples where numbers are very close (e.g., 1234 vs. $1.241 \times 10^3$); perhaps targeted demonstrations on such cases would reduce their error rate. Third, the same prompt template and few-shot examples are used for all test problems, so our results may depend on those specific ordering and selection patterns. In retrospect, we should have randomly varied the prompt format and few-shot examples within the test set.

- **Semantic context is not considered.** Our probing methods focus on the magnitudes of numerals in isolation, without considering context such as units. In practice, numeracy requires contextual understanding—for instance, 5 miles is greater than 50 meters, even though $5 < 50$, and 50 meters and 50 kg are incomparable. While LLMs likely represent such contextual information internally, we leave this aspect for future work.

## Acknowledgments

This work was supported by DOE Award No. DE-SC0025653 from the U.S. Department of Energy. Johns Hopkins University provided use of the Rockfish compute cluster (App. I). We thank Patrick Emami, Jared Willard, and members of the Argo Lab for questions and feedback during the project.

## References

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *Preprint*, arXiv:2310.06825.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Haoyang Li, Xuejia Chen, Zhanchao XU, Darian Li, Nicole Hu, Fei Teng, Yiming Li, Luyu Qiu, Chen Jason Zhang, Qing Li, and Lei Chen. 2025. Exposing numeracy gaps: A benchmark to evaluate fundamental numerical abilities in large language models. *Preprint*, arXiv:2502.11075.

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

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Luca Soldaini and Kyle Lo. 2023. peS2o (Pretraining Efficiently on S2ORC) Dataset. Technical report, Allen Institute for AI. ODC-By, `https://github.com/allenai/pes2o`.

Xingyou Song and Dara Bahri. 2025. Decoding-based regression. *Computing Research Repository*, arXiv:2501.19383.

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 (EMNLP)*.

Eric Tang, Bangding Yang, and Xingyou Song. 2025. Understanding LLM embeddings for regression. *Transactions on Machine Learning Research*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *Preprint*, arXiv:2211.09085.

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, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A large language model for finance. *Preprint*, arXiv:2303.17564.

Haotong Yang, Yi Hu, Shijia Kang, Zhouchen Lin, and Muhan Zhang. 2025. Number cookbook: Number understanding of language models and how to improve it. In *The Thirteenth International Conference on Learning Representations*.

Qiang Zhang, Keyan Ding, Tianwen Lv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, Xiang Zhuang, Zeyuan Wang, Ming Qin, Mengyao Zhang, Jinlu Zhang, Jiyu Cui, Renjun Xu, Hongyang Chen, Xiaohui Fan, and 2 others. 2025. Scientific large language models: A survey on biological & chemical domains. *ACM Comput. Surv.*, 57(6).

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

## A Related Work

There is rich recent literature on internal numerical representations in large language models. Wallace et al. (2019) demonstrate that numerical information can be extracted from contextual or static word embeddings. Stolfo et al. (2023) and Hanna et al. (2023) study internal mechanisms of LLMs' mathematical abilities. Closely related to our work is Zhu et al. (2025), who provide evidence that language models encode the value of integers linearly in log-space. Most recently, Lindsey et al. (2025) attempted to recover the internal procedure through which a specific LLM performs additions of two-digit numbers.

Our work also overlaps with recent LLM numeracy benchmark efforts. Recently, Yang et al. (2025) designed a comprehensive benchmark for evaluating LLMs' black-box responses to basic synthetic numerical questions. However, they did not investigate internal numerical representations. Similarly, Li et al. (2025) designed a basic numerical benchmark based on real-world datasets, using a similar blackbox approach. In a similar effort to extract precise numbers from LLM encodings, Tang et al. (2025) and Song and Bahri (2025) studied whether LLM encodings can be used to tackle classical regression tasks.

## B Data Setup

### B.1 Synthetic Data

The `cross-notation` dataset in §2 contains 11,200 comparison problems, with 8,000 samples for training, 1,600 for validation and 1,600 for evaluation. We generate 1,400 numeral pairs for each digit length between 2 and 9 digits, randomly converting one numeral in each pair to scientific notation while keeping the other in standard notation.

We construct two variants of this dataset: `int-sci` (integer vs. scientific notation) and `dec-sci` (decimal vs. scientific notation). In `int-sci`, both numbers start as integers before one is converted to scientific notation (e.g., 570 vs. $580 \rightarrow 5.7 \times 10^2$ vs. 580). In `dec-sci`, we first append 0–4 random decimal digits independently to each number before conversion (e.g., 570.23 vs. $871.6 \rightarrow 570.23$ vs. $8.716 \times 10^2$). When zero decimal digits are selected, we append ".0" to maintain consistent formatting (e.g., 342.0). These two datasets are functionally interchangeable for our experiments. We report results using `dec-sci` in Figs. 3, 10, 11, 13 and 14, with `int-sci` used for all other experiments on `cross-notation`.

### B.2 Scientific Data

We utilize the `peS2o` dataset (Soldaini and Lo, 2023), which contains approximately 40 million open-access academic papers from arXiv that have been cleaned, filtered, and formatted for language model pre-training. `peS2o` consists of two subsets: `s2orc`, which provides full-text papers (Lo et al., 2020), and `s2ag`, which contains titles and abstracts only.

To build `arXiv` data, we only use the `s2orc` subset. We first shuffle the corpus using `terashuf` and select the first 100k articles from `s2orc`. We then employ regular expressions to extract numerals in standard decimal notation (`((?<!\d\.)\d+\.\d+(?!\.\d))`) and in scientific notation (`((?:\d+(?:\.\d+)?)\s*×\s*10\s+[-+]?\d+`). To avoid overflowing GPU memory, we filter out papers exceeding 30,000 tokens. We process complete documents through the language models and extract hidden states at the final token position of each identified numeral. The resulting processed `arXiv` dataset contains 5,000 samples, divided into 4,000 training samples and 500 validation samples and 500 evaluation samples.

## C Inference and Prompting Setup

For verbalization tasks, all models are evaluated with temperature 0 (greedy decoding) to ensure deterministic and reproducible outputs.

### C.1 Open-weight Models

For the 7B–8B open-weight models, we use the Hugging Face `Transformers` library for decoding. Prompts are formatted as plain text with "Q:" and "A:" markers. The one-shot demonstration for `int-sci` is shown in §2. For `dec-sci`, we use:

```
Q: Which is larger, 9.9 × 10^2 or 899.9? A: 9.9 × 10^2
Q: Which is larger, {a} or {b}? A:
```

For $k$-shot experiments using `int-sci` (App. H), where $k \in [1, 5]$, we prepend the first $k$ of these examples:

```
Q: Which is larger, 9.9 × 10^2 or 100? A: 9.9 × 10^2
Q: Which is larger, 161230 or 7.182 × 10^5? A: 7.182 × 10^5
Q: Which is larger, 713 or 4.78 × 10^2? A: 713
Q: Which is larger, 1.354 × 10^6 or 4906723? A: 4906723
Q: Which is larger, 20834 or 6.5 × 10^3? A: 20834
```

followed by the usual test question:

```
Q: Which is larger, {a} or {b}? A:
```

### C.2 GPT Models

GPT-4.1 and GPT-4.1-mini are called via the OpenAI Chat Completions API with the system prompt "Just answer with a number." The questions and answers are provided as messages with the `user` and `assistant` roles respectively, rather than being marked with `Q:` and `A:` strings. The content of the few-shot examples is identical to that above.

## D Probing Setup

### D.1 Obtaining Hidden States

We feed each input into the LLMs and extract the hidden states at every layer. For each layer, we identify specific token positions corresponding to our target representations and save their hidden states $\boldsymbol{H} \in \mathbb{R}^{n \times d_{\text{model}}}$ for subsequent probing experiments, where $n$ is the number of samples and $d_{\text{model}}$ is the hidden dimension.

### D.2 Training Regression Probes

For regression probes, we extract hidden states from the last token of each numeral. Given hidden states $\boldsymbol{H}$ and their corresponding numerical values $\boldsymbol{x} \in \mathbb{R}^n$, we train a linear regressor $\mathcal{R}$ that predicts $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{w} + b\,\mathbf{1}$, where $\boldsymbol{w} \in \mathbb{R}^{d_{\text{model}}}$ and $b \in \mathbb{R}$ are the learned parameters.

We apply $\log_2$ transformation to compress the numerical range and emphasize relative magnitude relationships, using an $\ell_2$ regularizer ($\lambda = 1$) for training (i.e., ridge regression):

$$\boldsymbol{w}^*, b^* = \arg\min_{\boldsymbol{w}, b} ||\log_2(\boldsymbol{x}) - \boldsymbol{H}\boldsymbol{w} - b\,\mathbf{1}||_2^2 + \lambda||\boldsymbol{w}||_2^2$$

The trained probe predicts logarithmic magnitudes as $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{w}^* + b^*\,\mathbf{1}$.

### D.3 Training Classification Probes

For classification probes, we extract hidden states from the last token of the entire prompt after processing both numerals. We frame the task as binary classification to predict which numeral is larger in a given pair $(a, b)$.

Given hidden states $\boldsymbol{H} = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\} \in \mathbb{R}^{n \times d_{\text{model}}}$ and binary labels $\mathbf{y} \in \{0, 1\}^n$ (where $y_i = 1$ if the first numeral is larger), we train a logistic regression model with an $\ell_2$ regularizer ($\gamma = 1$):

$$\boldsymbol{w}^*, b^* = \arg\max_{\boldsymbol{w}, b} \left\{ \sum_{i=1}^{n} y_i \log(\sigma(\boldsymbol{h}_i^\top \boldsymbol{w} + b)) + (1 - y_i) \log(1 - \sigma(\boldsymbol{h}_i^\top \boldsymbol{w} + b)) - \frac{1}{2\gamma}\|\boldsymbol{w}\|^2 \right\}$$

where $\sigma$ is the sigmoid function. Given a hidden state $\boldsymbol{h} \in \mathbb{R}^{d_{\text{model}}}$, the probe estimates $P(a > b) = \sigma(\boldsymbol{h}^\top \boldsymbol{w}^* + b^*)$. Our binary classifier predicts $a > b$ iff this probability exceeds $\frac{1}{2}$, i.e., iff $\boldsymbol{h}^\top \boldsymbol{w}^* + b^* > 0$.

## E Finetuning Setup

The total loss is initially comprised of three terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \alpha\mathcal{L}_{\text{reg}} + \beta\mathcal{L}_{\text{cls}}$$

We finetune on the training set from App. B.1 using LoRA (Hu et al., 2022) with settings `lora_r=16`, `lora_alpha=32`, `lora_dropout=0.1`, `target_modules=[q_proj, v_proj]`. We use AdamW (Loshchilov and Hutter, 2019) as our optimizer, finetuning for 3 epochs with batch size 16.

We first initialize the probe parameters by fitting them in our usual way (App. D) on the same training data, before proceeding with finetuning of all parameters as above.

Using the validation set from App. B.1, we perform grid-search on probe and finetuning hyperparameters: $\alpha, \beta \in \{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}$, learning rate $\in$ {`2e-6`, `5e-6`, `1e-5`, `2e-5`, `5e-5`, `1e-4`, `2e-4`}, and probing layer depth $\in$ {`10%`, `20%`, `30%`, `40%`, `50%`, `60%`, `70%`, `80%`, `90%`, `100%`}. We select the hyperparameters that achieve the highest verbalization accuracy on validation data.

These optimal hyperparameters are $\alpha = 0$, $\beta = 0.02$, learning rate = 5e-5, probing layer depth = 90%. Thus the regression probing loss term is omitted in §2 since including it is suboptimal. In fact, we found that $\alpha > 0$ was suboptimal even when $\beta = 0$, indicating that attempting to improve regression loss did not help with the verbalized comparison. Presumably that was because regression on the `cross-notation` dataset was already extremely accurate. On a dataset like `arXiv`, however, there would be more room to improve regression, and this might benefit the downstream task of verbalized comparison.

## F Evaluation Metrics

**Mean squared error (MSE)** is the average squared difference between probe predictions and actual values in log space.

**Relative error (RE)** (reported in our paper abstract) converts the median absolute error in log space to a relative error as shown below.

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

$$\text{MSE}(\boldsymbol{y}, \boldsymbol{x}) = \text{mean}((\boldsymbol{y} - \log_2 \boldsymbol{x})^2) \tag{1}$$

$$\text{RE}(\boldsymbol{y}, \boldsymbol{x}) = 2^{\text{median}(|\boldsymbol{y} - \log_2 \boldsymbol{x}|)} - 1 = \text{median}\left(\max\left(\frac{2^{\boldsymbol{y}}}{\boldsymbol{x}} - 1, \frac{\boldsymbol{x}}{2^{\boldsymbol{y}}} - 1\right)\right) \tag{2}$$

$$\text{AAcc}(\boldsymbol{y}, \boldsymbol{x}) = \frac{\left|\, |2^{\boldsymbol{y}} - \boldsymbol{x}| < 0.01\boldsymbol{x} \,\right|}{|\boldsymbol{x}|} \tag{3}$$

## G Additional Probing Results

For a more detailed regression evaluation, we consider the Pearson correlation $\rho$, the coefficient of determination $R^2$ (which $= \rho^2$ for linear regression), mean square error (MSE), and approximate accuracy (AAcc).[5] Fig. 7 and Fig. 9 are expanded versions of Fig. 2 and Fig. 5.

The individual models' results for Fig. 3 can be found in Fig. 10.

## H Additional Error Analysis

Here, we present additional experimental results and analysis to complement our discussion in §3.3.

[5] See App. F for more details about metrics.

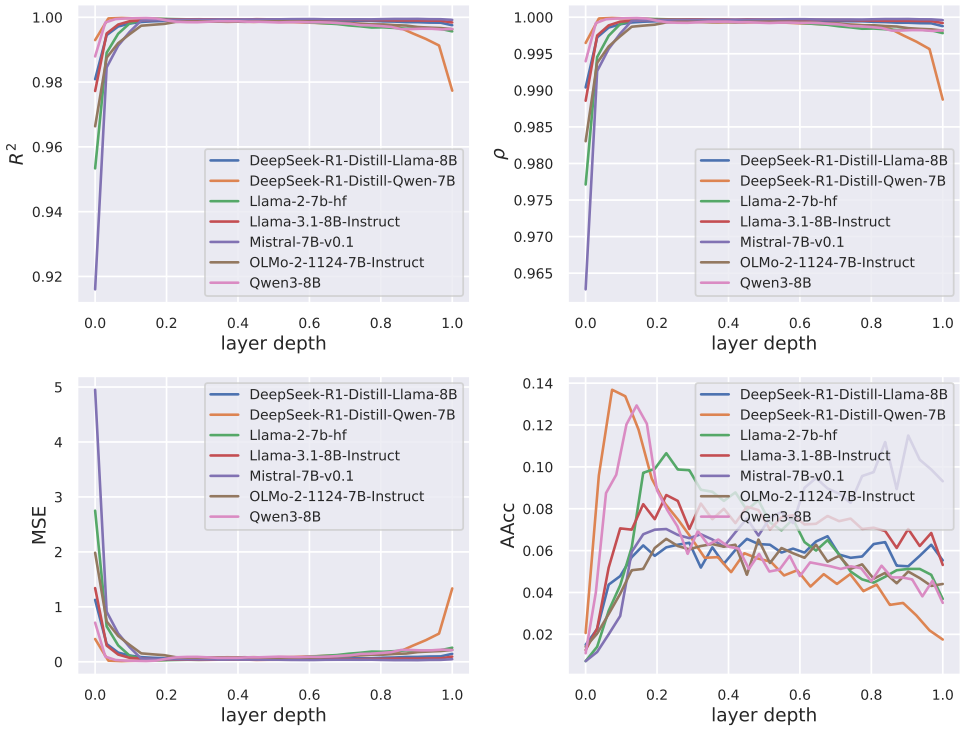



Figure 7: Regression probe performance on cross-notation data of each LLM across layers. Here, we report the Pearson correlation $\rho$, the coefficient of determination $R^2$ (which $= \rho^2$ for linear regression), mean square error (MSE), and approximate accuracy (AAcc). Definitions of MSE and AAcc can be found in App. F. This is the expanded version of Fig. 2.

**One-shot Prompting.** We observed in §3.3 and Fig. 3 that Llama-2-7B and Mistral-7B ignore the values of the numbers and answer only with the second number. This may be because the example that we provided in the one-shot prompt (see §2 and App. C) placed the correct answer in the second position. Thus, we tried exchanging the two numbers in the example. Fig. 11 displays the results of the altered prompt next to the original.

We observe that, out of all models, Llama-2's response to the change in prompt is the most extreme. In both prompts, Llama-2 answered exclusively with the same position in which the answer was presented in the one-shot example. Other models (DeepSeek-R1-Distill-Llama-8B, Mistral) show similar tendencies, but to a lesser extent. OLMo-2 shows a slight though consistent preference for the second number, regardless of the one-shot prompt. In contrast, GPT-4.1 and GPT-4.1-mini show minimal sensitivity to the ordering of the two options in the prompt, maintaining consistent performance across both variations.

**Few-shot Prompting.** Throughout this work, we have focused on one-shot prompting. We take the view that numeracy should be an innate capability: models should already exercise basic numeracy skills—cross-notation comparison being just one example—whenever they read scientific documents. In lieu of instructions, a single demonstration should be enough to illustrate the intended task and output format. Nevertheless, we evaluate few-shot performance here for completeness.

Fig. 12 shows the few-shot performance on cross-notation comparison using up to 5 examples (see specific few-shot prompts in App. C). Overall, all models benefit from additional demonstrations in the few-shot prompt, with performance improvement plateauing around 5 examples. Still, all models struggle when the two numbers are close ($|\log_2(a/b)| < 0.1$). While the smaller models (7B–8B) perform around random chance in this regime, GPT-4.1 and GPT-4.1-mini maintain 75–85% accuracy, with additional demonstrations providing marginal further improvement.

We designed our 5 examples so that the correct answer's position alternates between the first place and the second place, in an attempt to eliminate positional bias we observed in Fig. 11. While Llama-2, OLMo-2 and Mistral's positional biases are remedied by additional examples, the positional bias of

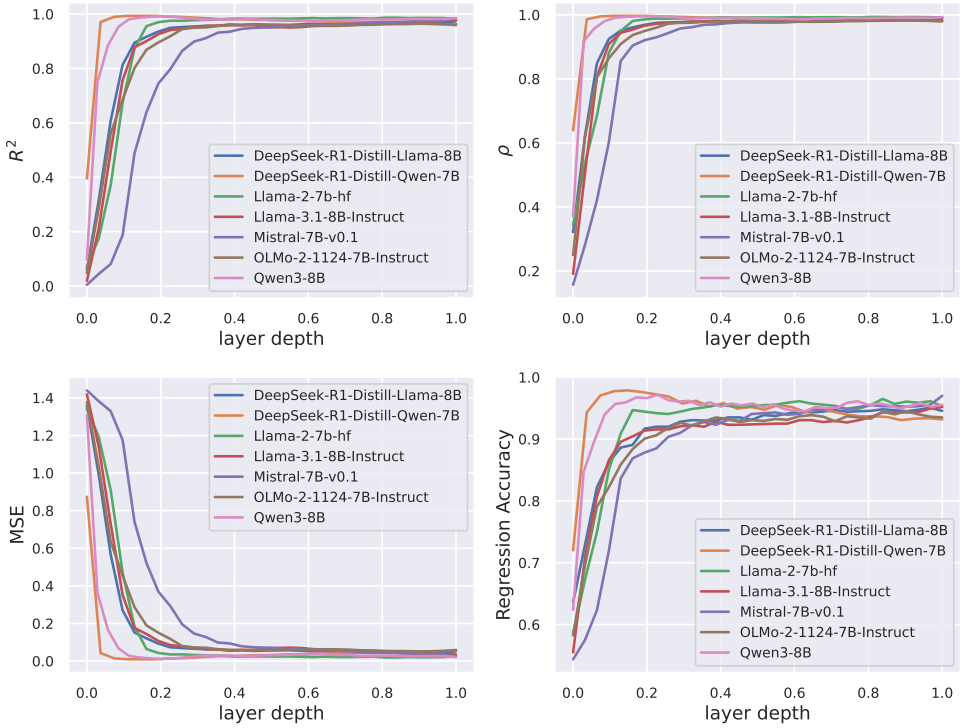



Figure 8: *Log-ratio* regression probe performance on cross-notation data of each LLM across layers. Here, we report the Pearson correlation $\rho$, the coefficient of determination $R^2$ (which $= \rho^2$ for linear regression), mean square error (MSE), and approximate accuracy (AAcc). Definitions of MSE and AAcc can be found in App. F.

Deepseek-R1-Distill-Llama-8B remains.

**Number of digits and magnitudes of numbers.** Finally, we examine the effect of the number of digits and magnitudes of numbers on cross-notation comparison via either verbalization or probes. Fig. 13 and Fig. 14 plot accuracy with respect to number of digits and magnitude (measured by $\log_2(a+b)$), respectively. We observe that verbalized comparison accuracy generally deteriorates for numbers that are larger or have more digits in their surface form. In contrast, comparison accuracy via regression or logistic classifier remains much higher and does not deteriorate in either case. These observations suggest that LLMs implicitly understand larger and longer numerals but fail to verbalize their relationship.

## I Computational Budget

All our experiments used a single NVIDIA A100 GPU (40GB or 80GB). Extracting hidden states for the full dataset takes a few hours, training and evaluating probes for one model takes about 40 minutes, and verbalization evaluation on 1,600 examples takes roughly 25 minutes per model. Finetuning a single model under one hyperparameter setting takes around 1 hour.

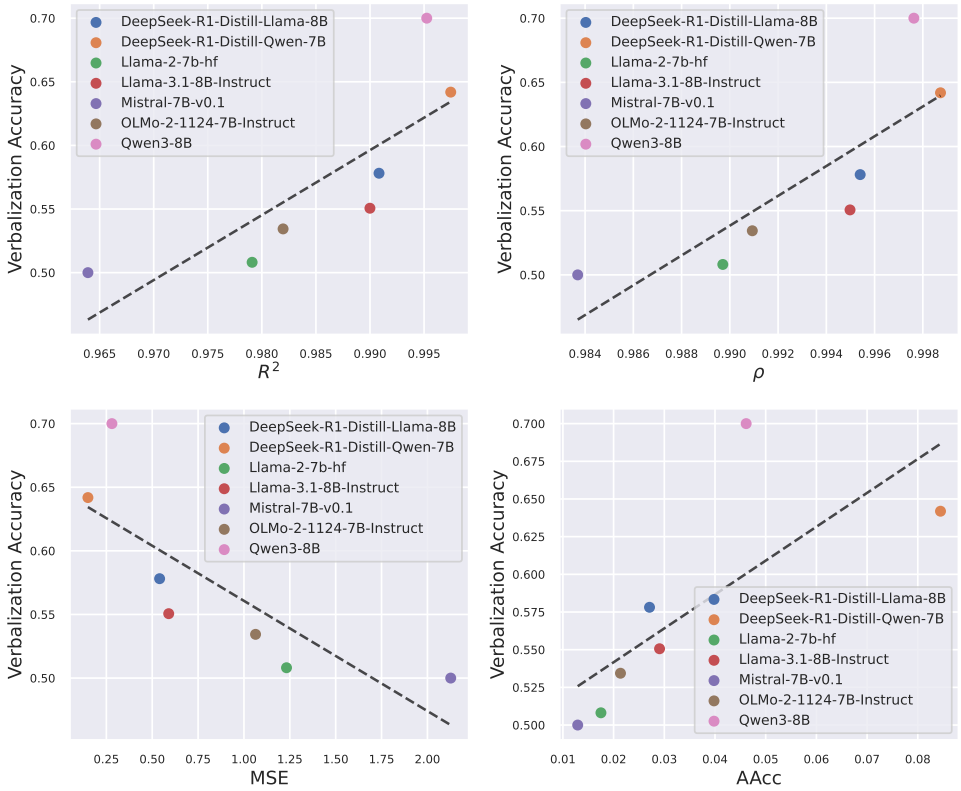



Figure 9: Average performance of linear regression probes at the first 3 layers correlates with model's verbalization accuracy on `cross-notation`. This is an expanded version of Fig. 5.

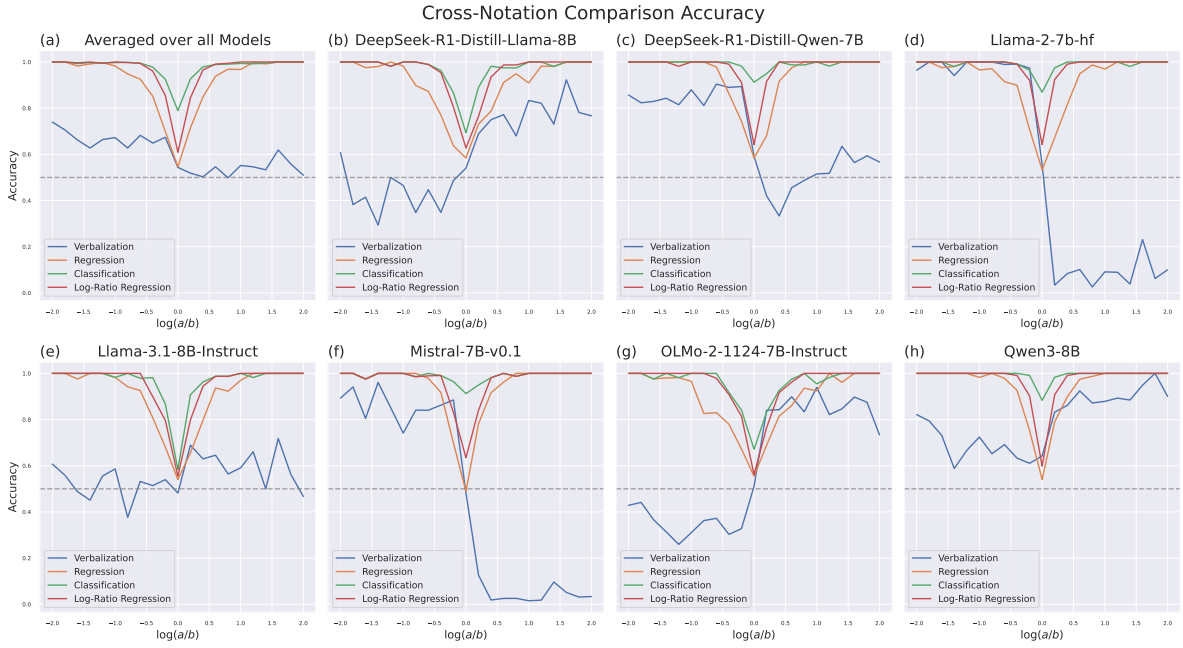



Figure 10: Cross-notation comparison accuracy with varying relative magnitude of two numbers (measured by $\log_2(a/b)$) using different methods. Same as Fig. 3, but with each models' results displayed separately.

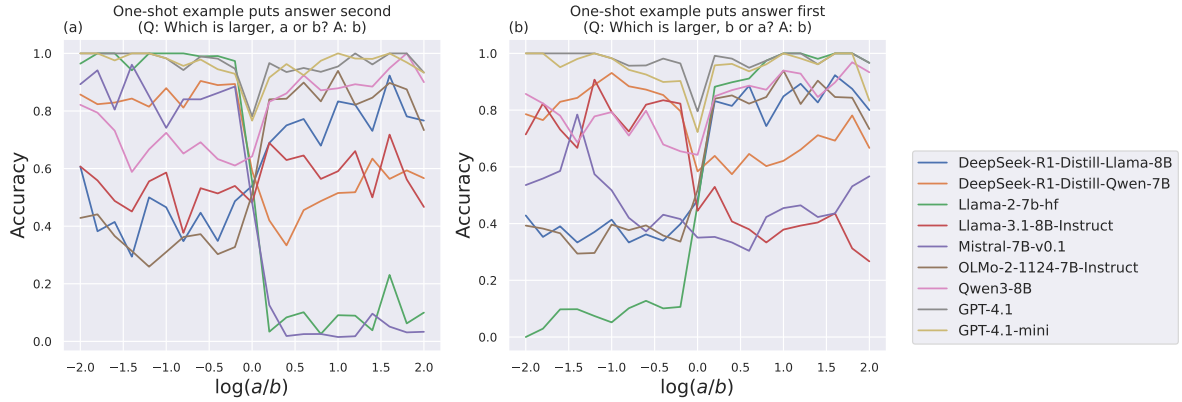



Figure 11: One-shot verbalized cross-notation comparison accuracy using different examples. (a) Same as Fig. 3(a); (b) Models' verbalized response when the numbers in the example are exchanged in positions.


Figure 12: Few-shot verbalized cross-notation comparison accuracy for different models. In each panel, color-matched dashed lines indicate the average accuracy of each shot count.


Figure 13: Cross-notation comparison accuracy vs. number of digits using different comparison methods. (a) Verbalized comparison using one-shot prompting; (b) Comparison via the predicted value of the regression probe trained on the hidden states; (c) Comparison using a logistic classification probe trained on the hidden states;

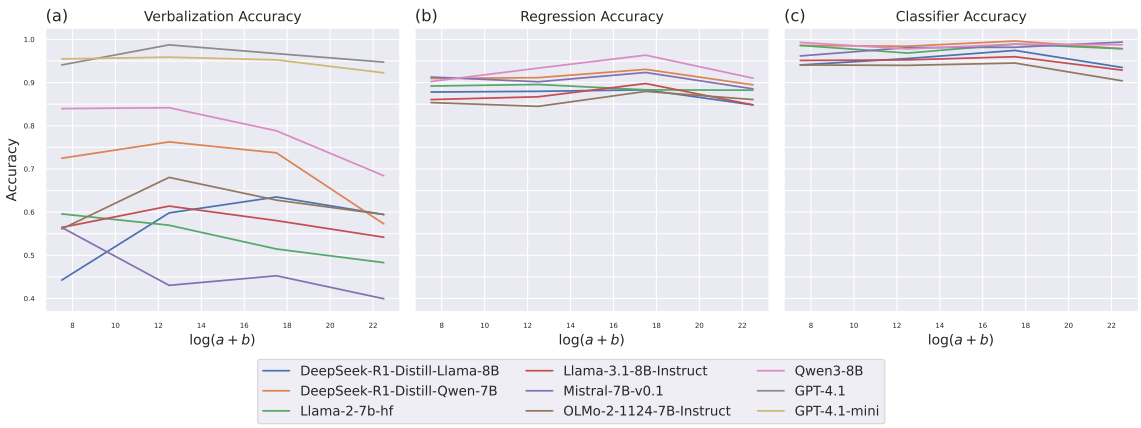



Figure 14: Cross-notation comparison accuracy vs. $\log_2(a + b)$ using different comparison methods. (a) Verbalized comparison using one-shot prompting; (b) Comparison via the predicted value of the regression probe trained on the hidden states; (c) Comparison using a logistic classification probe trained on the hidden states.