Eliciting Numerical Predictive Distributions of LLMs Without Auto-Regression

Julianna Piskorz [*1] **Katarzyna Kobalczyk** [*1] **Mihaela van der Schaar** [1]

Abstract

Large Language Models (LLMs) have recently been successfully applied to regression tasks—such as time series forecasting and tabular prediction—by leveraging their in-context learning abilities. However, their autoregressive decoding process is ill-suited to continuous-valued outputs, and obtaining numerical predictive distributions typically requires repeated sampling, leading to high computational cost. In this work, we investigate whether distributional properties of LLM predictions (e.g., mean, median, quantiles) can be recovered directly from LLM's internal representations, without explicit autoregressive generation. Our results suggest that LLM embeddings carry informative signals about numerical uncertainty, and that summary statistics of their predictive distributions can be approximated with reduced computational overhead.

1. Introduction

LLMs are increasingly used for structured prediction tasks such as tabular regression and time series forecasting (e.g. Requeima et al., 2024; Hegselmann et al., 2023; Gruver et al., 2024). However, issuing numerical predictions with LLMs remains computationally expensive due to their autoregressive nature: real-valued numbers typically span multiple tokens, and decoding them requires sequential autoregressive generation. This is particularly problematic when one would like to quantify the prediction uncertainty, which requires repeated sampling from the model's output distribution or auto-regressive computation of token logits (Gruver et al., 2024; Requeima et al., 2024).

This motivates our central question: *Do the LLM's hidden states already encode the full numerical prediction, including its uncertainty, before any decoding occurs?* While predicting a complete number requires resolving its magnitude—a decision typically made late in decoding—we investigate whether this information is recoverable directly from the model's internal representations. Focusing on time series forecasting, we probe the LLM's predictive distribution from the frozen embeddings from a pre-trained model. Specifically, we ask:

Do LLMs encode the next number they intend to generate? (Section 2) We train magnitude-aware regression probes to recover the greedy output, mean, and median of the LLM's predictive distribution. We show that *our probe accurately predicts numerical targets across data with varying orders of magnitude.*

Can we elicit the uncertainty of the LLM's predictive distribution? (Section 3) We then ask whether uncertainty information is also captured in LLM's hidden states. Using quantile regression, we show that *probes can accurately recover distributional properties such as the interquartile range and calibrated confidence intervals*.

Do these results generalise to other settings? (Section 4) We test whether a single probe generalises beyond the specific conditions under which they were trained. Specifically, we investigate whether probes trained on real-world data generalise across different sub-domains and whether probes trained on synthetic data generalise to real-world data. Our results show strong generalisation, with only modest degradation in out-of-distribution settings (Section 4).

Taken together, these findings suggest that much of the computation behind numerical prediction is already embedded in LLM representations—offering a path toward efficient, sampling-free regression with uncertainty estimation.[1]

2. Do LLMs Encode the Next Number They Intend to Generate?

2.1. Method of Investigation

We provide an overview of our methodology below. For more details, see Appendix.

[*]Equal contribution [1]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK. Correspondence to: Julianna Piskorz <jp2048@cam.ac.uk>, Katarzyna Kobalczyk <knk25@cam.ac.uk>.

Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, Canada. 2025. Copyright 2025 by the author(s).

[1]Code to reproduce our experiments: https://github.com/kasia-kobalczyk/guess_llm_official.

Objective. We ask: To what extent is the full predicted number already encoded in the LLM's internal representation, prior to decoding? Let $\mathbf{x} = [x_1, \ldots, x_n]$ be a sequence of numbers. Given $\mathbf{x}$, a language model induces a predictive distribution $p_{\text{LLM}}(\cdot \mid \mathbf{x})$ over the next value x_{n+1}. In this section, we aim to train independent probing models to recover: (a) the LLM's greedy prediction, (b) the mean of p_{LLM}, and (c) the median of p_{LLM}. We approximate the mean and median using 100 samples $y^j \sim p_{\text{LLM}}(\cdot \mid \mathbf{x})$.

LLM Representation. Following Gruver et al. (2024), we serialise $\mathbf{x}$ to text as "$x_1, x_2, x_3, \ldots, x_n,$". We then extract the final token's hidden state (denoted $\mathbf{h}_\ell[-1]$) from a preselected set of N layers $\mathcal{H}$. We concatenate these vectors to form a single input embedding for the probe:

$$\mathbf{e} := \text{concat}\left(\mathbf{h}_\ell[-1]\right)_{\ell \in \mathcal{H}} \in \mathbb{R}^{d_{\text{input}}}, \tag{1}$$

where $d_{\text{input}} = d_\ell \times |\mathcal{H}|$. We use LLama-2-8B model for generating the embeddings ($d_\ell = 4096$).

Datasets. We construct synthetic time series sequences from diverse function classes (e.g., sinusoids, Gaussians, noise) with varying length, noise, and scale. To probe magnitude robustness, we rescale each sequence across several ranges ($[-1, 1]$ to $[-10^4, 10^4]$), and aggregate these into a dataset of $\approx$ 80k sequences. Each training example includes $\mathbf{x}_i$, its embedding $\mathbf{e}_i$, the greedy prediction y_i^{greedy}, and 100 samples y_i^j from the model's predictive distribution.

Probing Model. The primary challenge in training regression probes for LLM numerical predictions lies in the wide spread of target magnitudes. Standard regression losses such as MSE or transformation techniques like log-scaling fail to provide stable gradients across this scale variability. To address this, we introduce a *magnitude-factorised regression model* that decomposes the prediction into a magnitude classification and a scale-invariant regression.

Let y^* be a target scalar (greedy, mean, or median prediction). We define its order of magnitude as:

$$m(y^*) := \lfloor \log_{10}\left(|y^*|\right) \rfloor. \tag{2}$$

Our model architecture consists of the following modules:

- $g : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^{d_{\text{hidden}}}$: an encoder mapping the input $\mathbf{e}$ to a latent representation.
- $f_{\text{order}} : \mathbb{R}^{d_{\text{hidden}}} \to \mathbb{R}^M$: a classifier predicting logits over M magnitude bins.
- $f_{\text{val}} : \mathbb{R}^{d_{\text{hidden}}} \to \mathbb{R}$: a regressor predicting the scaled value.

The final prediction is reconstructed by taking the expectation over the top-K predicted orders:

$$\hat{y}_i = \hat{r}_i \cdot 10^{\hat{m}_i}, \quad \text{where} \quad \hat{r}_i = f_{\text{val}}(g(\mathbf{e}_i)), \tag{3}$$

$$\hat{m}_i := 10^{\sum_{k \in \mathcal{K}_i} k \cdot p_k(g(\mathbf{e}_i))}, \tag{4}$$

where $\mathcal{K}_i$ is a set of K exponents with the largest logit values as predicted by $f_{\text{order}}(g(\mathbf{e}_i))$ and $p_k(g(\mathbf{e}_i))$ is derived from $f_{\text{order}}(g(\mathbf{e}_i))$ using the softmax over the top-K logits.

Training Objective. To decouple magnitude errors from value regression during early training, we use the ground-truth magnitude $m(y^*)$ to compute $\hat{y}$ and define the training objective as:

$$\mathcal{L} = \mathcal{L}_{\text{order}} + \beta \cdot \mathcal{L}_{\text{val}}, \quad \text{where} \tag{5}$$

$$\mathcal{L}_{\text{order}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \text{CrossEntropyLoss}(\hat{m}_i, m(y_i^*)), \tag{6}$$

$$\mathcal{L}_{\text{val}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(\hat{r}_i - \frac{y_i^*}{10^{m(y_i^*)}} \right)^2. \tag{7}$$

In the above N_b is the batch size and the hyperparameter β balances the magnitude and value objectives. This formulation of the loss allows the model to learn scale-invariant value predictions and as we found out, it improves stability during optimization.

2.2. Results

We train three separate models, for each of the mean, median and greedy predictions. As visualised on Figure 1, we find strong correlation between the $\log_{10}$ of the predicted number and $\log_{10}$ of the true value, for all three statistics. Further, the bar chart on the right hand side of Figure 1 visualises that our model achieves above 80% accuracy in predicting the exponent of the generated number. To further assess whether the LLM's internal representations encode fine-grained information beyond the order of magnitude, in the Appendix we show results on the dataset with time series values in the interval $[-1, 1]$.

> The internal representations of a pre-trained LLM encode detailed information about its intended numerical output–even before any tokens are generated. This demonstrates that much of the numerical reasoning performed by the LLM is already present in its hidden states, and may not require the autoregressive decoding process.

3. Can We Elicit the Uncertainty of the LLM's Predictive Distribution?

Encouraged by the above findings, we now investigate whether we can go beyond point estimates to recover the *uncertainty* of p_{LLM} by approximating its distributional shape. Specifically, we attempt to recover multiple quantiles of p_{LLM}, enabling a coarse-grained reconstruction of its distribution function as an easy way of estimating the confidence intervals for the LLM's predictions.

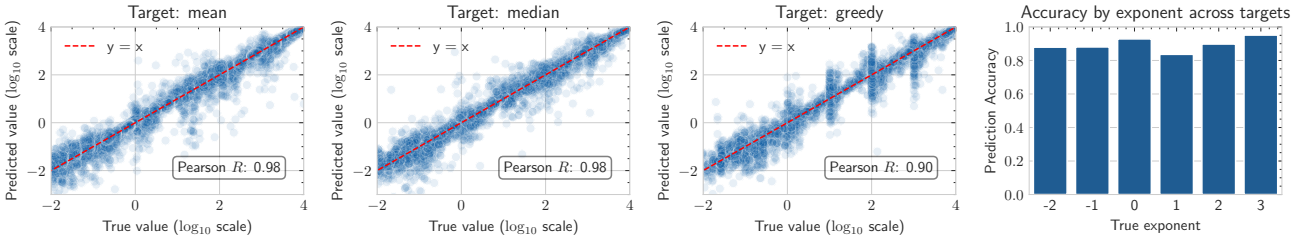

Figure 1. Predicted vs. true values of mean, median and greedy prediction, presented on $\log_{10}$ scale. The probing model accurately recovers the number that the LLM intends to predict, indicating that the internal representations encode the order of magnitude of prediction.

Table 1. Coverage of the predicted confidence intervals. Values denote empirical coverage (%) ± standard error.

α	50%	90%	95%
1.0	49.2 ± 0.4	89.2 ± 0.3	94.1 ± 0.3
10.0	49.8 ± 0.4	90.2 ± 0.3	94.1 ± 0.3
1000.0	50.4 ± 0.5	89.0 ± 0.3	93.7 ± 0.3
10000.0	51.2 ± 0.5	88.2 ± 0.4	92.7 ± 0.3

3.1. Method of Investigation

Quantile Regression. We adopt a *quantile regression* (QR) model, which enables direct estimation of the distributional shape without strong parametric assumptions. We train the quantile predictor using the *pinball loss* (Koenker & Hallock, 2001), computed with respect to LLM samples. As in Section 2, for each quantile predictor we use a magnitude-factorised model to address the challenge of scale variance in numerical outputs. For a detailed formulation, see Appendix. To train the QR models, we use the same datasets as in section 2, considered separately rather than concatenated.

3.2. Results

IQR Prediction. To investigate whether the LLM's internal representations encode information about the spread of its predictive distribution, we estimate the interquartile range (IQR) using the predicted 25th and 75th percentiles. As shown in Figure 2, we find a strong correlation between predicted and sample-based IQRs, suggesting that the model is able to infer distributional spread from internal LLM activations.

Confidence Interval Coverage. We next evaluate whether the predicted quantiles yield calibrated confidence intervals. Given a desired confidence level α and its associated interval $\mathcal{C}(\alpha)$ predicted by the probe, we compute the empirical coverage by checking what fraction of LLM samples fall within the predicted interval. We expect that:

$$\alpha = \mathbb{E}_{y\sim p(\cdot|\mathbf{x})}\left[\mathbb{1}\{y \in \mathcal{C}(\alpha)\}\right] \approx \frac{1}{N_{sa}}\sum_{j=1}^{N_{sa}} \mathbb{1}\{y^j \in \mathcal{C}(\alpha)\}, \quad \text{where } y^j \sim p_{\text{LLM}}(\cdot|\mathbf{x}).$$

Table 1 reports the empirical coverage for 50%, 90%, and 95% intervals across datasets with different scaling ℓ. In all cases, empirical coverage closely matches the target level, indicating that the quantile probe is well-calibrated.

> **Key Insights.** Our findings provide strong evidence that the uncertainty of an LLM's predictive distribution is encoded in its internal activations and can be effectively elicited using a quantile regression probe.

4. Generalisation Properties

In this section, we assess the generalisation of our approach to real-world data and across datasets. Since training probes can be costly, generalisation properties are key for practical use—allowing pre-trained probes to replace repeated autoregressive sampling on new data with differing distributions. Throughout this section, we use the same quantile regression model as used in Section 3.

4.1. Method of Investigation

Datasets. We construct a real-world dataset using time series from the Darts (Herzen et al., 2022) and Monash (Godahewa et al., 2021) collections. Following the same format as in our synthetic experiments, we generate LLM embeddings and samples from their predictive distribution for $\approx$ 45k distinct sequences across 32 sub-datasets (e.g., US Births, Bitcoin, Air Passengers). Furthermore, we also investigate an even stronger form of generalisation: from a model trained on synthetic data only to testing on real-world data. Thus, we train the following models:

- **Real (all):** Trained on a random 80% of all sequences across all sub-datasets. The remaining 20% is held out for testing.
- **Real (5 fold):** We partition the dataset into 5 folds such that, in each fold, one model is trained on 80% of the sub-datasets and evaluated on the remaining 20%. This ensures that each sub-dataset appears in the test fold of exactly one out of 5 models trained.
- **Synth:** A model trained on the combination of the 4 synthetic dataset from the previous sections with scales 1.0, 10.0, 1000.0 and 10000.0.

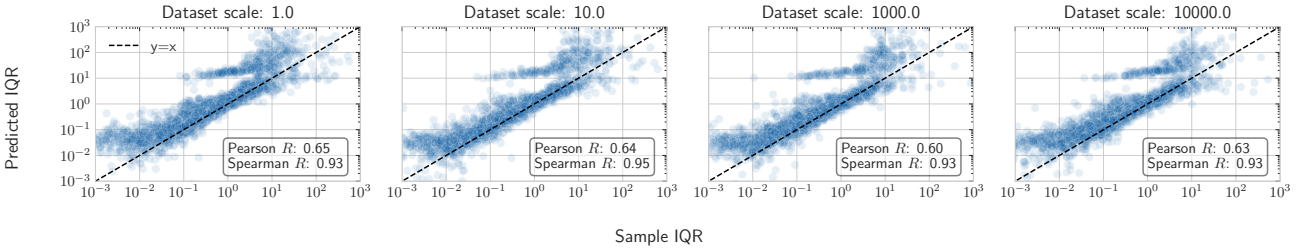

Figure 2. Predicted vs. sample-based IQR (median-normalised). The model accurately tracks the spread of the LLM's output distribution.

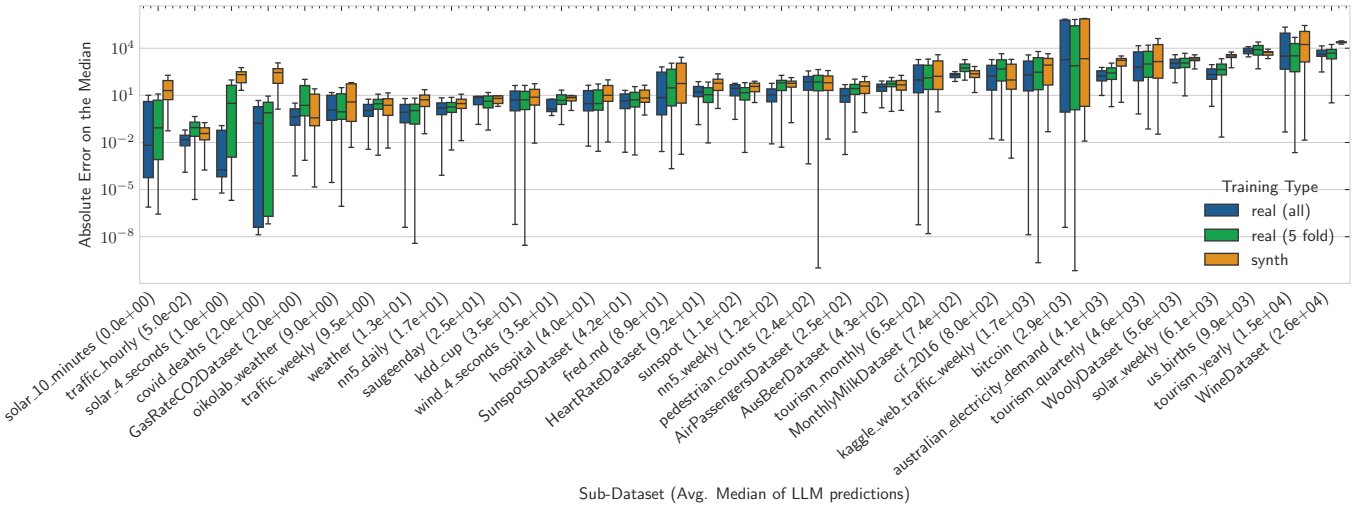

Figure 3. Absolute Error on the Median across different sub-dataset. Comparison of generalisation across models trained on different data.

At test time, the above models face increasingly stronger distribution shifts. In terms of generalisation performance to previously unseen data distributions, we can view the Real (all) model as a baseline for Real (5 fold) and Synth.

4.2. Results

Figure 3 shows the distribution of the Absolute Error of the predicted vs. sample median across all sub-datasets. The x-axis is sorted by increasing order of magnitude of the datasets. We note that the sub-datasets in our collection cover widely varying ranges of values (with the magnitudes varying from 10^{-3} to 10^{13}). We suspect that this is the main reason for our probing model struggling to generalise across some datasets. Interestingly, while the Synth model underperforms, it still demonstrates good generalisation for some of the sub-datasets.

> **Key Insights.** When applied to real-world datasets, the model achieves accurate empirical coverage and demonstrates partial transferability to unseen data distributions. Cross-dataset generalisation is possible, but challenged by large variation in scale and distribution.

5. Discussion, Limitations and Further Work

Discussion. Our findings demonstrate that LLMs internally encode rich numerical information about their intended predictions, well before any autoregressive decoding occurs. This suggests that much of the LLM's "reasoning" over numeric outputs is already complete at the point of processing the input sequence, and that decoding primarily serves as a mechanism for surfacing the LLM's predictions. Beyond shedding light on the internal mechanics of LLMs in regression settings, these results open up a practical direction: enabling uncertainty-aware numerical prediction without incurring the high cost of repeated sampling.

Limitations. Despite these promising findings, several limitations remain. While our probing models exhibit some generalisation abilities, they are still trained per-LLM and require retraining for new architectures or tokenization schemes. Further, for training and evaluation purposes, we approximate the LLM's predictive distribution using empirical sampling, which is itself a noisy and computationally costly proxy.

Further Work. Future research could explore extending this framework to more diverse prediction tasks. A deeper investigation into the mechanistic basis of numerical encoding could also reveal connections to known computational circuits or arithmetic operations within the model. Finally, motivated by our generalisation results, an important next step is the development of universal probing models which could be applied off-the-shelf across diverse tasks and data domains, eliminating the need for costly retraining.

References

Akhtar, M., Shankarampeta, A., Gupta, V., Patil, A., Cocarascu, O., and Simperl, E. Exploring the Numerical Reasoning Capabilities of Language Models: A Comprehensive Analysis on Tabular Data. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 15391–15405, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.1028. URL https://aclanthology.org/2023.findings-emnlp.1028/.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. Monash Time Series Forecasting Archive, May 2021. URL http://arxiv.org/abs/2105.06643. arXiv:2105.06643 [cs].

Golkar, S., Pettee, M., Eickenberg, M., Bietti, A., Cranmer, M., Krawezik, G., Lanusse, F., McCabe, M., Ohana, R., Parker, L., Blancard, B. R.-S., Tesileanu, T., Cho, K., and Ho, S. xVal: A Continuous Numerical Tokenization for Scientific Language Models, December 2024. URL http://arxiv.org/abs/2310.02989. arXiv:2310.02989 [stat].

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large Language Models Are Zero-Shot Time Series Forecasters, August 2024. URL http://arxiv.org/abs/2310.07820. arXiv:2310.07820 [cs].

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. TabLLM: Few-shot Classification of Tabular Data with Large Language Models, March 2023. URL http://arxiv.org/abs/2210.10723. arXiv:2210.10723 [cs].

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Pottelbergh, T. V., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kościsz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M., and Grosch, G. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research*, 23(124):1–6, 2022. ISSN 1533-7928. URL http://jmlr.org/papers/v23/21-1177.html.

Koenker, R. and Hallock, K. F. Quantile Regression. *Journal of Economic Perspectives*, 15(4):143–156, December 2001. ISSN 0895-3309. doi: 10.1257/jep.15.4.143. URL https://www.aeaweb.org/articles?id=10.1257/jep.15.4.143.

Koloski, B., Margeloiu, A., Jiang, X., Škrlj, B., Simidjievski, N., and Jamnik, M. LLM Embeddings for Deep Learning on Tabular Data, February 2025. URL http://arxiv.org/abs/2502.11596. arXiv:2502.11596 [cs] version: 1.

Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the Biology of a Large Language Model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Requeima, J., Bronskill, J., Choi, D., Turner, R. E., and Duvenaud, D. LLM Processes: Numerical Predictive Distributions Conditioned on Natural Language, December 2024. URL http://arxiv.org/abs/2405.12856. arXiv:2405.12856 [stat].

Schwartz, E., Choshen, L., Shtok, J., Doveh, S., Karlinsky, L., and Arbelle, A. NumeroLogic: Number Encoding for Enhanced LLMs' Numerical Reasoning, March 2024. URL http://arxiv.org/abs/2404.00459. arXiv:2404.00459 [cs] version: 1.

Singh, A. K. and Strouse, D. J. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs, February 2024. URL http://arxiv.org/abs/2402.14903. arXiv:2402.14903 [cs].

Stolfo, A., Belinkov, Y., and Sachan, M. A Mechanistic Interpretation of Arithmetic Reasoning in Language Models using Causal Mediation Analysis, October 2023. URL http://arxiv.org/abs/2305.15054. arXiv:2305.15054 [cs].

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288. arXiv:2307.09288 [cs].

Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. Do NLP Models Know Numbers? Probing Numeracy in

Embeddings, September 2019. URL http://arxiv.org/abs/1909.07940. arXiv:1909.07940 [cs].

Yang, S., Wang, L., and Ding, P. Causal inference with confounders missing not at random, February 2019. URL http://arxiv.org/abs/1702.03951. arXiv:1702.03951 [stat].

Zhou, Y., Alon, U., Chen, X., Wang, X., Agarwal, R., and Zhou, D. Transformers Can Achieve Length Generalization But Not Robustly, February 2024. URL http://arxiv.org/abs/2402.09371. arXiv:2402.09371 [cs].

Zhu, A. Y., Mitra, N., and Roy, J. Addressing positivity violations in causal effect estimation using Gaussian process priors. *Statistics in Medicine*, 42(1):33–51, 2023. ISSN 1097-0258. doi: 10.1002/sim.9600. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.9600. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.9600.

A. Related Works

Numerical Predictive Distributions of LLMs. When used as regressors, LLMs can provide not only point estimates but also full predictive distributions, reflecting their stochastic nature. To elicit continuous distributions over numerical outputs, Gruver et al. (2024) and Requeima et al. (2024) propose an autoregressive approach that generates logit values over discretised numeric bins, which are then scaled to form a valid probability distribution. Access to such distributions is crucial for downstream tasks requiring uncertainty quantification, including decision-making under uncertainty and Bayesian optimisation. However, these methods are computationally intensive, as they require multiple sequential queries to the LLM to construct a single distribution (e.g., $p(123.4) = p(1)p(2|1)p(3|12)p(.|123)p(4|123.)$). This motivates us to explore alternative approaches to eliciting numerical predictive distributions from LLMs.

Discrepancy between number generation and auto-regression. As next-token predictors, LLMs are not explicitly trained to understand the value of numbers. Due to their autoregressive nature, early tokens encode digits before key decisions like decimal placement (that determine a number's magnitude) are made. This can lead to surprisingly poor performance on simple numerical tasks (Yang et al., 2019; Akhtar et al., 2023; Zhou et al., 2024; Schwartz et al., 2024). To address these limitations, several works have proposed alternatives to standard autoregressive decoding for numerical predictions. For instance, Golkar et al. (2024) introduce a special `[NUM]` token, replaced post-hoc with a continuous value predicted by a learned regression head–though this requires retraining the model. Others (Singh & Strouse, 2024; Schwartz et al., 2024) investigate number-specific tokenizations to improve numerical accuracy of LLMs. In contrast, we ask whether one can bypass autoregressive decoding in *pre-trained* LLMs by directly reading out the predictive distribution from the internal representations.

Probing numeracy in LLM embeddings. A number of prior works give evidence that simple probing models can be used to learn numerical values encoded in the LLM embeddings. Wallace et al. (2019) has shown that the value of a number can be successfully decoded from its encoded word embedding (e.g., "71" $\rightarrow$ 71.0.). Stolfo et al. (2023) identified specific layers in LLMs that store numerical content, recoverable via simple linear probes, while Zhu et al. (2023) demonstrated that intervening on these layers alters generated outputs. More recently, Koloski et al. (2025) showed that LLM embeddings can serve as effective covariates in downstream regression models. Complementary findings from mechanistic interpretability suggest that, even in purely textual settings, LLM hidden states encode representations of tokens that the model is most likely to generate (Lindsey et al., 2025). Taken together, these results support the hypothesis that it should be possible to train probes that approximate the numerical *predictive distribution* of the LLM, motivating our work.

B. Additional Experimental Results

B.1. Precision in the generated digits of the point estimates

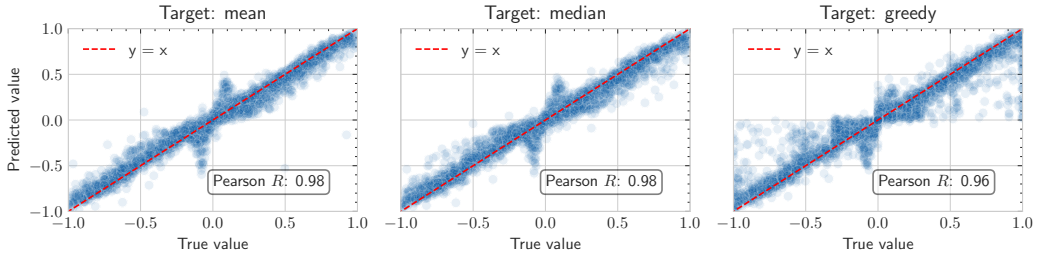

Figure 4. Predicted vs. true values of mean, median and greedy prediction. The probing model accurately recovers the number the LLM intends to predict, with the precision surpassing just order of magnitude estimation.

Precision in generated digits. To further assess whether the LLM's internal representations encode fine-grained information beyond the order of magnitude of the mean, median and greedy prediction, we focus on the dataset with time series values in the interval $[-1, 1]$. We report the mean squared error (MSE) of our predictions in Figure 5 and compare them against three baselines: $\bar{\mathbf{x}}$ (predicting the average value in the whole training dataset), $\bar{\mathbf{x}}_i$ (predicting

Figure 5. MSE of the predicted values (no scaling).

target	$\hat{y}$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
mean	0.009	0.256	0.035	0.085
median	0.009	0.260	0.041	0.087
greedy	0.024	0.273	0.065	0.109

the average of each time series) and $x_{i,n}$ (predicting the last value from each time series). We further plot the obtained predictions in Figure 4. Interestingly, among the three targets considered (mean, median, greedy), the model performs worst when predicting the greedy output. As shown in Figure 4, the probe captures the sign of the greedy prediction reliably but exhibits larger errors in the decimal digits. We hypothesise that this is because the greedy prediction is not an explicit function of the model's predictive distribution, but rather a byproduct of the autoregressive decoding process, making it harder to recover precisely from internal states.

B.2. Sample efficiency of the median estimates

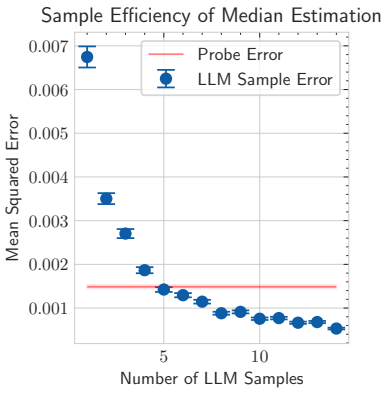

Figure 6. The probe (horizontal line) achieves lower MSE than sampling for $n \leq 5$.

Sample Efficiency. We further examine whether the probe can outperform direct sampling from the LLM in terms of sample efficiency. Let S denote a target statistic (e.g., median or a quantile). We define $S(n)$ as the estimate from $n < 100$ samples and we let $S^* := S(100)$ as a proxy for the ground-truth. We compute the LLM sample error as $\text{MSE}(S(n), S^*)$ and compare it to the probe error $\text{MSE}(\hat{S}, S^*)$.

Figure 6 illustrates this comparison for the median on a dataset with scale 1.0. The probe outperforms empirical sampling for all $n \leq 5$, demonstrating that our approach can be more sample-efficient. Results for additional quantiles and larger-scale datasets are reported in the Appendix. A probe of this kind can serve as a computationally efficient surrogate for estimating statistics of the LLM output distribution which can help in cost and compute time reduction.

C. Details of the Quantile Regression Model

Architecture. As in Section 2, we use a magnitude-factorised model to address the challenge of scale variance in numerical outputs. The model is defined as follows:

- $g : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^{d_{\text{hidden}}}$: a shared encoder that maps the input representation $\mathbf{e}$ to a latent space.
- For each quantile index $s \in \{1, \ldots, S\}$:
  - $f^s_{\text{order}} : \mathbb{R}^{d_{\text{hidden}}} \to \mathbb{R}^M$: a classifier predicting the order of magnitude m^s of quantile q^s.
  - $f^s_{\text{val}} : \mathbb{R}^{d_{\text{hidden}}} \to \mathbb{R}$: a regressor predicting a scale-invariant value $\hat{r}^s$.

Similarly as before, each quantile is reconstructed from the predicted components as:

$$\hat{q}^s_i = \hat{r}^s_i \cdot 10^{\hat{m}^s}, \quad \text{where} \quad \hat{r}^s_i = f^s_{\text{val}}(g(\mathbf{e}_i)), \quad \hat{m}^s_i = 10^{\sum_{k \in \mathcal{K}^s} k \cdot p^s_k(g(\mathbf{e}_i))}. \tag{8}$$

where $\mathcal{K}^s_i$ is a set of K exponents with the largest logit values as predicted by $f^s_{\text{order}}(g(\mathbf{e}_i))$ and $p^s_k(g(\mathbf{e}_i))$ is derived from $f^s_{\text{order}}(g(\mathbf{e}_i))$ using the softmax over the top-K logits.

Training Objective. As before, we use the true order of magnitude $m(y^j_i)$ for each target value during training to enable stable learning. The total loss is the sum of the cross-entropy losses for magnitude prediction and pinball losses for quantile regression:

$$\mathcal{L} = \sum_{s=1}^{S} w_s \left(\mathcal{L}^s_{\text{order}} + \beta \cdot \mathcal{L}^s_{\text{val}} \right), \tag{9}$$

$$\mathcal{L}^s_{\text{order}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \text{CrossEntropyLoss}(f^s_{\text{order}}(g(\mathbf{e}_i)), m(y^*_i)), \tag{10}$$

$$\mathcal{L}^s_{\text{val}} = \frac{1}{N_b N_{sa}} \sum_{i=1}^{N_b} \text{PinballLoss}\left(\tau^s, \hat{r}^s_i, \frac{y^j_i}{10^{m(y^j_i)}} \right), \tag{11}$$

where N_b is the batch size, N_{sa} is the number of LLM samples per input, S is the number of quantiles, and $[w_1, \ldots, w_S]$ a set of weights per each quantile, which is a hyperparameter of our model. In our experiments we have $N_{sa} = 100$ and $S = 7$, with the corresponding quantile list $\mathcal{Q} = [0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975]$. This choice of quantiles allows us to estimate: the median, the interquartile range (IQR), as well as the 90% and 95% confidence intervals.

D. Details of the Experimental Setup

D.0.1. Assets and Licensing Information

The following existing assets were used to produce the experimental results:

- **Monash dataset** (Godahewa et al., 2021)
- **Darts dataset** (Herzen et al., 2022)
- **Llama-2-7B model** (Touvron et al., 2023)

D.1. Computer infrastructure used

Hardware. All experiments were conducted using 2 separate NC24rs_v3 instances and one NC80adis_H100_v5 instance on the Microsoft Azure cloud platform. This instances are a part of Azure's GPU-optimised virtual machine series, with their hardware specifications summarised in Table 2.

Table 2. Azure Virtual Machine Specifications

Specification	**NC24rs_v3**	**NC80adis_H100_v5**
vCPUs	24	80
System Memory (GiB)	448	640
GPU Model	4× NVIDIA Tesla V100	2× NVIDIA H100 NVL
GPU Memory (per GPU)	16 GiB	94 GiB
Total GPU Memory	64 GiB	188 GiB
GPU Architecture	Volta	Hopper
CUDA Version	11.x	12.x
CPU Model	Intel Xeon E5-2690 v4	AMD EPYC Genoa
Local Storage	2.9 TB	7.1 TB

Generating the synthetic dataset for one scaling factor $\ell \in \{1, 10, 1000, 10000\}$ took no more than 10h. Training one probe model took no more than 4h.

D.2. Details of the datasets

D.2.1. Details of the synthetic time series dataset

We generate a synthetic dataset comprising time series derived from a family of parametric functions, each evaluated over a fixed domain and perturbed with controlled noise. The purpose is to simulate diverse temporal patterns, inducing varying levels of uncertainty in the LLM's predictions.

We use a set of base functions defined over the interval $x \in [0, 60]$, discretized into 120 equidistant points. The functions are summarised in Table 3. For each function and value of a, we generate a clean series $y = f(a \cdot x)$, and then apply:

- Additive Gaussian noise with variance $\sigma^2 \in \{0.0, 0.01, 0.05, 0.1\}$.
- Vertical scaling by $b \sim \mathcal{U}(0, \ell)$
- Vertical translation by $d \sim \mathcal{U}(-\ell, \ell)$

From each transformed series, we sample 10 different subsequences for each of the lengths $n \in \{3, 5, 7, 10, 13, 15, 17, 20, 25, 30, 35, 40\}$, with each subsequence starting at a random offset. Each sequence becomes

a training input. Inputs are serialized as floating-point strings with a user-defined number of decimal places p (we use $p = 4$ for $\ell = 1.0$, $p = 3$ for $\ell = 10.0$, $p = 2$ for $\ell = 1000.0$ and $p = 1$ for $\ell = 10000.0$). This results in 33600 generated time series for each value of ℓ.

Concatenated dataset. Having constructed the individual dataset for each scaling factor $\ell \in \{1, 10, 1000, 10000\}$, we also construct one concatenated dataset. In doing that, we limit the number of datapoints to 80000 and ensure that the y_{test} values of the generated time series are equally distributed on the log scale, from 10^{-2} to 10^4. This is to ensure a balanced distribution of the train and test examples.

Dataset filtering. Before using the generated datasets for training the probing models, we apply dataset filtering to exclude any potential outliers. Namely, we ensure that the mean, median and greedy LLM prediction lie in $[-\ell, \ell]$.

Function name	**Formula**	**a-range**
`sin`	$\sin(x)$	[0.5, 6.0]
`linear_sin`	$0.2 \cdot \sin(x) + \frac{x}{450}$	[0.5, 6.0]
`sinc`	$\text{sinc}(x)$	[0.05, 0.2]
`xsine`	$\frac{x-30}{50} \cdot \sin(x - 30)$	[0.5, 1.3]
`beat`	$\sin(x) \cdot \sin\left(\frac{x}{2}\right)$	[0.1, 6.0]
`gaussian_wave`	$e^{-\frac{(x-2)^2}{2}} \cdot \cos(10\pi(x-2))$	[0.01, 0.1]
`random`	$\mathcal{U}(-1, 1)$	[0.0, 1.0]

Table 3. Functions used to generate time series data, their mathematical forms, and the range of the time-scaling parameter a.

D.2.2. MONASH DATASET

- **Data Loading**: We use the data from the Monash dataset, preprocessed by (Gruver et al., 2024) and available from https://drive.google.com/file/d/1sKrpWbD3LvLQ_e5lWgX3wJqT50sTd1aZ/view?usp=sharing. Each sub-dataset file contains tuples of the form `(train,test)`, which are concatenated to form full univariate time series.

- **Resampling**: To ensure computational tractability, each series is subsampled (via strided slicing) to contain at most 1000 time steps.

- **Series Selection**: For each dataset, a maximum of 50 time series are selected at random to control the number of examples used during training.

- **Subsequence Generation**: From each selected series, we extract multiple training subsequences of varying lengths $n \in \{3, 5, 7, 10, 13, 15, 17, 20, 25, 30, 35, 40\}$. For each length, we generate up to 10 training subsequences, sampled at different offsets.

D.2.3. DARTS DATASET

- **Data Loading**: We use the data from the Darts dataset, available from the `darts` python package. We use the following sub-datasets: AirPassengersDataset, AusBeerDataset, GasRateCO2Dataset, MonthlyMilkDataset, SunspotsDataset, WineDataset, WoolyDataset, HeartRateDataset.

- **Resampling**: To ensure computational tractability, the series for the datasets SunspotsDataset and HeartRateDataset are subsampled (via strided slicing).

- **Series Selection**: For each dataset, all available time series are selected.

- **Subsequence Generation**: From each selected series, we extract multiple training subsequences of varying lengths $n \in \{3, 5, 7, 10, 13, 15, 17, 20, 25, 30, 35, 40\}$. For each length, we generate up to 10 training subsequences, sampled at different offsets.

D.2.4. LLM GENERATION SETTINGS

We generate the LLM hidden states from a Llama-2-7B model, available through the `huggingface` library. Each of the generated time series, we obtain 100 samples from the LLM, generated auto-regressively, as well as the greedy generation. As the Llama-2 tokenizer encodes each digit separately, during generation we narrow down the generated tokens to digits, decimal point and $+/-$ signs. For obtaining the random samples, we use `temperature=1.0` and `top_p=0.95`. We exclude from the final dataset samples for which generation failed at least once (i.e. the obtained generation was not a valid number), such that each time series in the final dataset has exactly 100 LLM samples.

D.2.5. TRAIN-VALIDATION-TEST SPLIT

Before training, we split each of the datasets in 80% training dataset, 10% validation dataset and 10% test dataset. Unless otherwise stated (in the generalisation experiments), these splits are random. We do not apply any scaling or transformation to either the LLM embeddings (which are inputs to our model) or the outputs.

D.3. Details of the magnitude-factorised regression model

Our magnitude-factorised regression models, used both for the purpose of point prediction and for the purpose of quantile regression, has the following hyperparameters. We report the default values of the hyperparameters used in Table 4 and Table 5, and then report any deviations from these values for specific experiments below. We train the model using the ADAM optimiser. For a detailed implementation of the magnitude-factorised regression models, see the provided code.

Hyperparameter	**Description**	**Default Value**
`min_mag`	Minimum exponent for base-10 magnitude scaling (as used by f_{order})	-3
`max_mag`	Maximum exponent for base-10 magnitude scaling	$\log_{10} \ell$
`use_arctan`	Apply $10 \cdot \arctan(0.5 \cdot x)$ to bound output of f_{val}	`True`
`beta`	Weight for regression loss component	10.0
`K`	Top-K exponents taken into consideration (see Equation 4)	3
`hidden_layers`	Number of hidden layers in feature extractor	1
`hidden_dim`	Dimensionality of hidden feature representation	512
`hidden_states_list`	A list of the hidden states $\mathcal{H}$ to use as input	$[25, \ldots, 32]$
`quantile_weights`	Weights of each of the quantiles in the quantile regression loss function	$[1, 1, 2, 5, 2, 1, 1]$

Table 4. Model-specific hyperparameters for the magnitude-factorised regression model.

Hyperparameter	**Description**	**Default Value**
`learning_rate`	Learning rate for the optimizer	10^{-4}
`weight_decay`	L2 regularization weight	0.1
`scheduler_step_size`	Learning rate scheduler step size	100
`scheduler_gamma`	Learning rate scheduler step size	0.5
`batch_size`	Number of samples per training batch	1024
`max_epochs`	Number of training epochs	500
`patience`	Patience for the early stopping	200

Table 5. Optimizer and training-related hyperparameters.

D.3.1. EXPERIMENT-SPECIFIC HYPERPARAMETER SETTINGS

Figure 1. We use `max_mag` $= 4$.

Figure 4 and Figure 5. We use `lr` $= 10^{-4}$, `max_epochs` $= 2000$.

Figure 2 and Table 1. We use `max_mag` $= 13$.

Figure 3. We use `batch_size` $= 2048$, `lr` $= 10^{-5}$ and `max_mag` $= 13$.