ELICITING NUMERICAL PREDICTIVE DISTRIBUTIONS OF LLMS WITHOUT AUTO-REGRESSION

Anonymous authors
Paper under double-blind review

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 may be ill-suited to continuous-valued outputs, where obtaining predictive distributions over numerical targets requires repeated sampling, leading to high computational cost and inference time. In this work, we investigate whether distributional properties of LLM predictions can be recovered *without* explicit autoregressive generation. To this end, we study a set of regression probes trained to predict statistical functionals (e.g., mean, median, quantiles) of the LLM's numerical output distribution directly from its internal representations. Our results suggest that LLM embeddings carry informative signals about summary statistics of their predictive distributions, including the numerical uncertainty. This investigation opens up new questions about how LLMs internally encode uncertainty in numerical tasks, and about the feasibility of lightweight alternatives to sampling-based approaches for uncertainty-aware numerical predictions. Code to reproduce our experiments can be found at `https://anonymous.4open.science/r/guess_llm-811B/`.

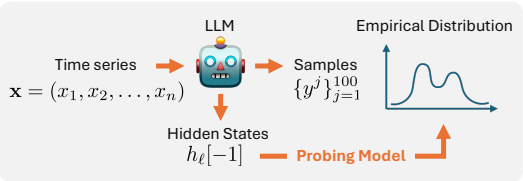
1 INTRODUCTION

With the increasing capabilities of LLMs, a growing body of work has explored their use for structured data prediction–most notably for tabular data regression (e.g. Requeima et al., 2024; Hegselmann et al., 2023; Shysheya et al., 2025; Vacareanu et al., 2024) and time series forecasting (e.g. Gruver et al., 2024; Xue & Salim, 2023). These studies demonstrate that LLMs can act as competitive regressors, even without task-specific fine-tuning. This advantage is especially pronounced in low-data regimes, where LLMs can leverage their pre-training, prior knowledge, and capacity to condition on auxiliary textual context to match or outperform specialised models. We provide a more detailed overview of related works in Appendix A.

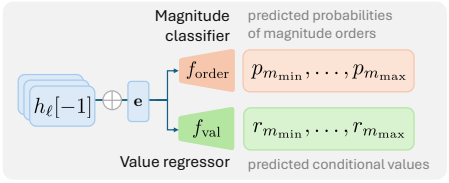
However, issuing numerical predictions with LLMs requires sequential autoregressive generation–real-valued numbers typically span multiple tokens requiring multiple forward passes of the input through the LLM to generate a single prediction. This makes inference costly and time-consuming, especially when many samples are needed–for example, to quantify the uncertainty (Gruver et al., 2024; Requeima et al., 2024) or improve prediction accuracy (Requeima et al., 2024).

This raises a natural question: is there a way of eliciting the LLM's predictive distribution without performing costly autoregressive number generation? This question is non-trivial: generating a real number requires determining its order of magnitude, including decisions about decimal placement and termination–choices that are made only after several tokens have already been generated. Our work therefore contributes to exploring the broader question of how LLMs 'plan' their outputs before generation begins (Lindsey et al., 2025).

In this work, focusing on the problem of time series forecasting specifically, we explore to what extent the LLM's internal representation of the input sequence can be used to reconstruct its numerical predictive distribution of the next number, requiring just a single pass through the LLM. Concretely, we explore the following three questions:



(a) The goal: can we recover the LLM's numerical predictive distribution from its internal representation of the input?



(b) Schematic diagram of the magnitude-factorised probing model used in section 2.

Figure 1: Illustration of this paper's goals and methodology.

Do LLMs encode the next number they intend to generate? (Section 2) We begin by examining whether LLM's internal representations of the input series encode sufficient information to recover point predictions–specifically, the greedy output, mean, and median of the predictive distribution. To test this, we develop a magnitude-factorised regression probe that separates prediction into two components: a coarse magnitude classification and a scale-invariant value regression, such that our model can effectively learn to predict numbers of varying orders of magnitude. Trained on LLM embeddings from synthetic time series data, *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 a magnitude-factorised quantile regression model, we train probes to predict the quantiles of the LLM's output distribution, approximated via sampling. *The resulting models accurately recover the interquartile range and produce well-calibrated confidence intervals*.

What is the practical value of our findings? (Sections 4 and 5) The ability to recover numerical predictions directly from LLM embeddings holds the potential to bypass auto-regressive sampling—offering savings in inference time and computational costs, which we demonstrate empirically. However, for such probes to be practically useful, they must generalise beyond the specific conditions under which they were trained. We therefore evaluate whether a single probing model can be deployed across varied settings without retraining. First, we test generalisation to unseen time series lengths. Second, we assess generalisability of our previous results to real-world data. 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. *We demonstrate that, while some drop in calibration occurs on out-of-distribution datasets, our probes demonstrate encouraging generalisation abilities, showing the universality of LLM's representations of numerical quantities.*

Our findings provide new insights into the numerical capabilities of LLMs: much of the "reasoning" underlying numerical predictions appears to be encoded in the model's internal representations of the input, prior to token-level decoding. This raises questions whether auto-regressive sampling is necessary to extract real-valued outputs from LLMs, and opens the door to developing more efficient, single-pass approaches. By showing that both point estimates and uncertainty can be reliably extracted from hidden states, our work suggests a lightweight, general-purpose strategy for deploying LLMs in regression tasks–particularly in settings where computational efficiency and uncertainty estimation are essential. We hope these results motivate further study of how LLMs internally represent numerical quantities, and how this information can be surfaced for practical downstream use.

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

LLMs are trained for next-token-predictions. Thus, as a single number typically spans multiple tokens, obtaining a complete numerical prediction from the LLM requires repeated auto-regressive sampling. This can be computationally expensive in number-heavy tasks, particularly when one would like to obtain repeated samples for the purpose of uncertainty estimation. To mitigate this overhead, we ask: *to what extent is the full predicted number–beyond just its leading digit–already encoded in the LLM's internal representation, prior to any token-by-token generation?* If such information can be reliably extracted, one could sidestep autoregressive generation altogether. However, this possibility

is not trivial: critical aspects of number generation, such as the placement of the decimal point or number termination, which determine the order of magnitude of the number, often occur late in the decoding process, particularly for large magnitudes.

2.1 METHOD

Objective. Let $\mathbf{x} = [x_1, \ldots, x_n]$ be a sequence of numbers (e.g., an equally-spaced time series). 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 investigate whether the internal representations of the LLM encode sufficient information to predict this distribution's key statistics. Specifically, we aim to train independent probing models to recover: (a) the LLM's greedy prediction, (b) the mean, and (c) the median of p_{LLM}. The empirically estimated scalar statistics of the LLM's predictive distribution are our targets for prediction based on the LLM's hidden representation of the input series $\mathbf{x}$.

LLM Representation. Following Gruver et al. (2024), we serialise an input series $\mathbf{x}$ to text as "$x_1, x_2, x_3, \ldots, x_n$,". We *do not* apply any scaling to the time series before serializing the inputs. This is important, as LLMs often contain contextual prior knowledge and scaling of the original time series may prohibit the LLM from using this prior knowledge effectively. From a pre-selected set of N transformer layers denoted by $\mathcal{H}$, we extract the final token's hidden state from each layer, obtaining a set of embeddings $\left\{\mathbf{h}_\ell[-1] \in \mathbb{R}^{d_{\text{model}}} : \ell \in \mathcal{H}\right\}$. We concatenate these embedding vectors to form a single input 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 $d_{\text{input}} = d_{\text{model}} \times |\mathcal{H}|$. The choice of the hidden layers $\mathcal{H}$ is a hyperparameter of our model. Throughout the main body of this paper we use the Llama-2-7B model, for which $d_{\text{model}} = 4096$, and we set $\mathcal{H}$ as the last 8 layers. Experiments with other LLMs can be found in Appendix C, alongside an ablation study on the choice of layers.

Datasets. We use synthetically generated datasets to evaluate probing performance. Each sequence $\mathbf{x}$ is sampled from a set of functions exhibiting varied dynamics, including sinusoidal patterns, Gaussians, beat functions, and random noise (see Appendix B.3 for details). The generated time series also vary in the length, n, and the level of noise, to ensure diversity of the input embeddings and target distributions. We generate variants of the dataset by scaling the value range from $[-1, 1]$ to $[-10, 10]$, $[-1000, 1000]$, and $[-10000, 10000]$. We also combine the datasets of different scales to obtain a larger dataset of approximately 70k unique sequences, balanced across the different orders of magnitude. In this section, our training datasets take the following structure: $\left\{\left(\mathbf{x}_i, \mathbf{e}_i, y_i^{\text{greedy}}, y_i^{\text{mean}}, y_i^{\text{median}}\right)\right\}_{i=1}^{N}$, where N is the total number of examples in a dataset, y_i^{greedy} the LLM's greedy prediction given $\mathbf{x}_i$, and $y_i^{\text{mean}}, y_i^{\text{median}}$ the empirical mean and median, respectively, estimated based on N_{sa} samples from the LLM's predictive distribution, $\{y_i^j\}_{j=1}^{N_{sa}} \sim p_{\text{LLM}}(\cdot|\mathbf{x}_i)$. In our experiments we set $N_{sa} = 100$.

Probing Model. A significant challenge in training regression probes for LLM numerical predictions is the wide spread of target magnitudes. Standard regression losses such as the MSE or transformation techniques like log-scaling fail to provide stable gradients, prioritising optimisation of the largest values only. To address this, we introduce a *magnitude-factorised regression model* that consists of two components (both initialised as MLPs, see Appendix B.4 for hyperparameter details):

- $f_{\text{order}} : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^M$: a classifier predicting the order of magnitude of the target number.
- $f_{\text{val}} : \mathbb{R}^{d_{\text{input}}+1} \to \mathbb{R}$: a regressor predicting the target value scaled by the predicted order.

The magnitude classification component predicts the order of magnitude of a target value y^*, where $* \in \{\text{greedy}, \text{mean}, \text{median}\}$. Formally, we define the order of magnitude of a scalar y as: $m(y) = \lfloor \log_{10}(|y|) \rfloor$. For a given input $\mathbf{x}$ and its corresponding representation $\mathbf{e}$, $f_{\text{order}}(\mathbf{e})$ outputs a vector of logits of a set of pre-defined magnitude classes $m_k \in \{m_{\min}, \ldots, m_{\max}\}$. In our experiments, we let $m_{\min}$ and $m_{\max}$ be the minimum and maximum orders of magnitude in the training data. The predicted vector of magnitude class probabilities (after softmax) is denoted as $\text{softmax}(f_{\text{val}}(\mathbf{e})) = \mathbf{p}(\mathbf{x}) = [p_{m_{\min}}, \ldots, p_{m_{\max}}]$.

The regression component predicts a scaled version of the target value, conditioned on all possible magnitude classes (to allow the model to adjust its prediction based on the predicted magnitude). For each magnitude class $m_k \in \{m_{\min}, \ldots, m_{\max}\}$, the corresponding scale is equal $s_k = 10^{m_k}$. The regression head takes as input the concatenation of the feature representation $\mathbf{e}$ and the scale factor s_k outputting $r_k = f_{\text{val}}([\mathbf{e}; s_k])$ for each k. This produces regression outputs $\mathbf{r} = [r_{m_{\min}}, \ldots, r_{m_{\max}}]$ for all magnitude classes. The conditional prediction for each magnitude order m_k is then computed as: $\hat{y}_k = r_k \cdot 10^{m_k}$.

Loss Function and Training. Our model supports a two-phase training procedure:

- **Phase 1:** Train only the classification head while freezing the regression head, using the classification loss—standard cross-entropy loss for magnitude prediction:

$$\mathcal{L}_{\text{order}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \text{CrossEntropyLoss}(\mathbf{p}(\mathbf{x}_i), m(y_i^*)), \tag{2}$$

  where N_b is the batch size.

- **Phase 2:** Train only the regression head while freezing the classification head, using the regression loss, i.e. the mean squared error between the predicted scaled value and the scaled target value:

$$\mathcal{L}_{\text{val}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(r_{\hat{m}_i} - \frac{y_i^*}{10^{m(y_i^*)}} \right)^2, \quad \text{where} \quad \hat{m}_i = \arg\max_{m_k} p_{m_k}(\mathbf{x}_i). \tag{3}$$

Empirically, we find that the 2-stage training procedure performs better than joint training of the order and value heads. To further provide justification for our approach, in Appendix C.1 we compare the performance of our magnitude-factorised probe against a vanilla MLP probe, showing significant performance gains.

Expected Prediction. During evaluation of our model, we compute the *expected prediction* by marginalising over the top-K magnitude classes: $\mathbb{E}_K[\hat{y}] = \sum_{k \in \text{top-}K} p_{m_k} \hat{y}_k$, where top-$K$ refers to the K magnitude classes with highest predicted probabilities (we set $K = 3$).

2.2 Results

Order of magnitude. We first investigate to what extent our probing model can correctly recover the order of magnitude of the number the LLM intends to generate. We train three separate models, one for each of the mean, median and greedy targets. In this experiment, we use the combined dataset consisting of time series with varying scales. The bar chart on the right hand side of Figure 2 visualises that the classifier part of our magnitude-factorised model achieves above 90% accuracy in predicting the exponent of the target values. Further, as visualised with the scatter plots on Figure 2, we find strong correlation between our probes' final predictions and the target statistics.

Precision in generated digits. To further assess whether the LLM's internal representations encode fine-grained information beyond the order of magnitude, 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 Table 1 and compare them against three simple baselines, using as the prediction: a) the average value of the entire training dataset ($\bar{\mathbf{x}}$), b) the average of the series $\mathbf{x}_i$ ($\bar{\mathbf{x}}_i$), or c) the las value of the series $\mathbf{x}_i$ ($x_{i,n}$). We also provide scatter plots for this dataset in Appendix C. Interestingly, among the three targets considered (mean, median, greedy), the model performs worst when predicting the greedy output. We hypothesise that this is because the greedy prediction is not an explicit function of the model's predictive distribution, but rather a by-product of the autoregressive decoding process, making it harder to recover precisely from internal states.

Table 1: MSE obtained when predicting the statistics of the LLM's predictive distribution.

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$*$ = mean	0.006	0.256	0.035	0.085
$*$ = median	0.006	0.260	0.041	0.087
$*$ = greedy	0.015	0.273	0.065	0.109

These results show that **the internal representations of a pre-trained LLM encode detailed information about its intended numerical output**—even before any tokens are generated. Our

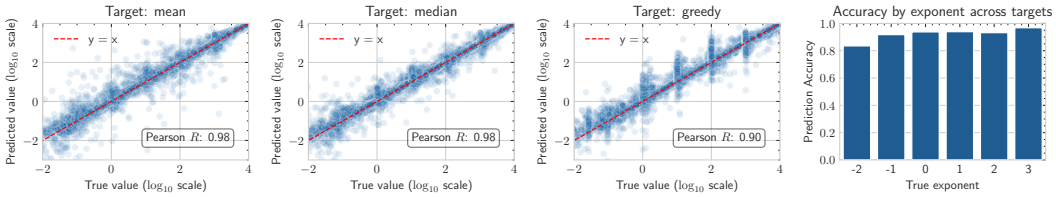


Figure 2: *Predicted vs. true values of mean, median and greedy prediction, presented on* $\log_{10}$ *scale.* The probing model accurately recovers the number that the LLM intends to predict, indicating that the internal representations encode the order of magnitude of prediction.

probing model accurately recovers not only the order of magnitude, but also fine-grained point estimates of the mean, median, and the greedy output. 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?

In the previous section, we demonstrated that point estimates—such as the greedy prediction, mean, and median—of an LLM's predictive distribution $p_{\text{LLM}}(\cdot \mid \mathbf{x})$ can be recovered from its internal representations without the need for performing auto-regressive sampling. Encouraged by these 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 and providing an easy way of estimating the confidence intervals for the LLM's predictions.

3.1 Method

Quantile Regression. Since the distribution p_{LLM} may be multi-modal and non-Gaussian, we rule out parametric approximations. Instead, we adopt *quantile regression*, which enables direct estimation of distributional shape without strong assumptions about its form. Let $\mathcal{Q} = [\tau^1, \ldots, \tau^S]$ be a list of target quantile levels. For each $\tau^s \in [0, 1]$, we denote the predicted quantile value as $\hat{q}^s$. We train the quantile predictor using the *pinball loss* (Koenker & Hallock, 2001), computed with respect to LLM samples $y_i^j \sim p_{\text{LLM}}(\cdot \mid \mathbf{x}_i)$. For a single quantile level τ, predicted quantile value $\hat{q}$ and a single LLM sample y_i^j, this loss function is defined as:

$$\text{PinballLoss}(\tau, \hat{q}, y_i^j) := \max\left(\tau(y_i^j - \hat{q}), (1-\tau)(\hat{q} - y_i^j)\right). \tag{4}$$

Probing Model. As in section 2, we use a magnitude-factorised model to address the challenge of scale variance in numerical outputs. The quantile model takes an equivalent form to the one in section 2, except we introduce S classification and regression heads, one for each quantile level, denoted by f^s_{order} and f^s_{val}, respectively. As previously, each classification head outputs a vector of magnitude class probabilities $\mathbf{p}^s = [p^s_{\min}, \ldots, p^s_{\max}]$. The regression heads output vectors of conditional scaled values $\mathbf{r}^s = [r^s_{\min}, \ldots, r^s_{\max}]$. For an order m_k the predicted conditional quantile value is computed as $\hat{q}^s = r^s_k \cdot 10^{m_k}$.

Datasets. We use the same datasets as in section 2, but with target values being the raw LLM samples $\{y_i^j\}_{j=1}^{N_{sa}}$, instead of the aggregate statistics.

Training. Unlike in the previous section, we use a joint training approach (2-phase training did not yield any significant improvements in performance). We construct the total loss as the sum of the cross-entropy losses for magnitude prediction and pinball losses for quantile regression:

$$\mathcal{L} = \sum_{s=1}^{S} \left(\mathcal{L}^s_{\text{order}} + \beta \cdot \mathcal{L}^s_{\text{val}}\right), \tag{5}$$

$$\mathcal{L}^s_{\text{order}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \text{CrossEntropyLoss}\left(\mathbf{p}^s(\mathbf{x}_i), m(\tilde{q}_i^s)\right), \tag{6}$$

$$\mathcal{L}^s_{\text{val}} = \frac{1}{N_b N_{sa}} \sum_{i=1}^{N_b} \sum_{j=1}^{N_{sa}} \text{PinballLoss}\left(\tau^s, r_i^s, \frac{y_i^j}{10^{m(\tilde{q}_i^s)}}\right). \tag{7}$$

In the above, $\tilde{q}_i^s$ denotes the empirical quantile value derived from the LLM samples $\{y_i^j\}_{j=1}^{N_{sa}}$. In our experiments, we use a set of $S = 7$ quantile levels: $\mathcal{Q} = [0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975]$. This choice of allows us to easily estimate: the median, the interquartile range (IQR), as well as the 90% and 95% confidence intervals.

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 the IQR is sensitive to scale, we normalise it by the predicted median, and similarly normalise the empirical IQR from LLM samples using the sample median. If the probe captures uncertainty faithfully, we should observe a monotonic relationship between the predicted and empirical (normalised) IQRs. Scatter plots in Figure 3 show a strong correlation between predicted and sample-based IQRs, with points aligning well on the $x = y$. This demonstrates that our probing model is able to infer distributional spread from the LLM's hidden states.

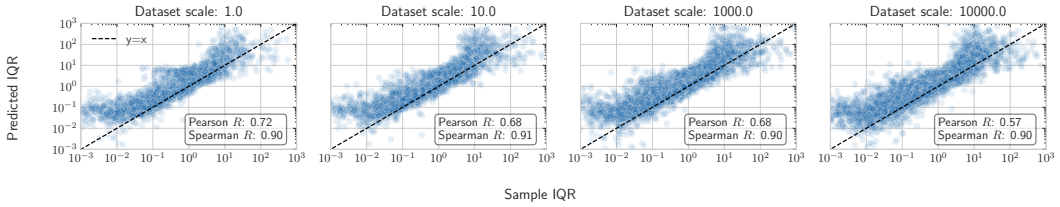


Figure 3: Predicted vs. sample-based IQR (both median-normalised). The model accurately tracks the variability of the LLM's output distribution.

Confidence Interval Coverage. Next, we 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 2: Coverage of the predicted confidence intervals. Values denote empirical coverage (%) ± standard error.

α	50%	90%	95%
1.0	52.0 ± 0.4	90.9 ± 0.3	95.5 ± 0.2
10.0	52.7 ± 0.5	91.3 ± 0.3	96.1 ± 0.2
1000.0	51.4 ± 0.3	90.7 ± 0.3	95.7 ± 0.2
10000.0	48.2 ± 0.3	90.5 ± 0.2	95.4 ± 0.2

Table 2 reports the empirical coverage for 50%, 90%, and 95% intervals across datasets of varying scale. In all cases, empirical coverage closely matches the target level, indicating that the quantile probe is well-calibrated.

Our findings provide strong evidence that **the uncertainty of the LLM's predictive distribution is encoded in its internal activations and can be effectively elicited using a quantile regression probe**. The probe is capable of predicting meaningful spread measures, producing well-calibrated confidence intervals that match the empirical coverage. These results suggest that LLMs internalise rich distributional information during generation, which can be accessed and approximated efficiently via probing. This opens up new opportunities for downstream applications that rely on uncertainty quantification—such as safe decision-making, model-based control, and probabilistic reasoning—while avoiding the overhead of autoregressive sampling.

4 EFFICIENCY AND ACCURACY

In this section, we further investigate the practical applicability of probing models from the perspective of computational and inference time efficiency as well as accuracy with respect to the ground truth time series value. Throughout this section, we focus on the scalar probing models from section 2.

4.1 ERRORS WITH RESPECT TO THE GROUND-TRUTH

Table 3: MSE of predicted values vs. the ground truth $x_{i,n+1}$.

predicted value	probe ($\hat{y}_i^*$)	LLM (y_i^*)
$*$ = greedy	0.0652	0.0668
$*$ = mean	0.0562	0.0555
$*$ = median	0.0561	0.0553

$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$	GP
0.3454	0.0878	0.1226	0.0717

In this experiment, we compare the error of our probes' predictions with respect to the ground-truth next value $x_{i,n+1}$ of the input time series $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})$. We present the comparison between the error achieved by using the probe trained to predict the mean, median and greedy statistics of the LLM distribution ($\epsilon^*_{\text{probe}} = \mathbb{E}\left[(\hat{y}_i^* - x_{i,n+1})^2\right]$) vs. by using these statistics directly ($\epsilon^*_{\text{LLM}} = \mathbb{E}\left[(y_i^* - x_{i,n+1})^2\right]$). For a better sense of scale, we additionally present the errors using the simple baselines introduced in section 2.2, as well as the error of the mean prediction of a GP model fit to the input time series $\mathbf{x}_i$. The MSE values of the predictions with respect to the ground-truth next value of the time series are presented in Table 3. Results were obtained for the probing models from section 2.2 and the dataset with scale 1.0. We make two key observations. First, consistent with prior work (Requeima et al., 2024), the LLM's median prediction achieves the lowest error relative to the ground truth. Second, our probe attains errors comparable to those obtained by sampling directly from the LLM. This places its performance in context: **the probe captures enough information from hidden states to match the LLM's own accuracy on the one-step-ahead prediction task.**

4.2 SAMPLE EFFICIENCY AND COMPUTATIONAL COSTS

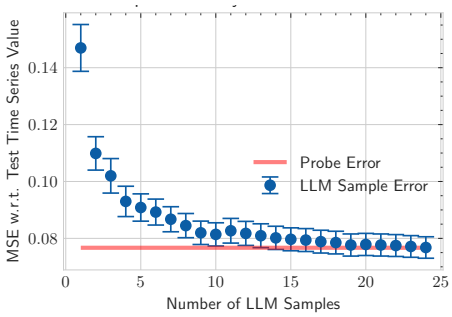


Figure 4: The probe achieves comparable error to using around 20-25 LLM samples on the one step ahead prediction task.

We further compare the error of the probe in predicting $x_{i,n+1}$ vs. using the mean prediction of the LLM using N empirical samples. Figure 4 illustrates this on a dataset with scale 1.0. The horizontal line shows the error attained by our probe, $\epsilon^{\text{mean}}_{\text{probe}} = \mathbb{E}\left[(\hat{y}_i^{\text{mean}} - x_{i,n+1})^2\right]$ and the blue points with error bars show the error attained when using N LLM samples, $\epsilon^{\text{mean}}_{\text{LLM}}(N) = \mathbb{E}\left[(y_i^{\text{mean},N} - x_{i,n+1})^2\right]$, where $y_i^{\text{mean},N}$ is the mean of N LLM samples: $\{y_i^j\}_{j=1}^N \sim p_{\text{LLM}}(\cdot|\mathbf{x}_i)$. Our probe outperforms empirical sampling for all N up to **20-25 samples**, demonstrating that a probe of this kind can serve as a **computationally efficient surrogate for making numerical predictions**.

For a detailed discussion regarding the computational costs comparisons of LLM sampling and inference with our probe, see Appendix C.3.

5 GENERALISATION

Finally, we investigate the generalisation capabilities of our approach along several axes including context length generalisation, applicability to real-world data, and cross-dataset generalisation. As the process of training a probe can be costly, such generalisation capabilities are important for real-world applications, if we would like to use a pre-trained probe on new datasets with different distributional properties. Throughout this section, we focus on the quantile probing models from section 3.

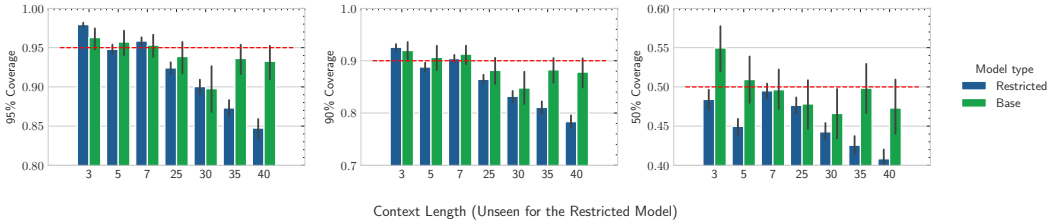


Figure 5: *Generalisation to unseen context lengths.* A probe trained on a restricted context length range (Restricted) exhibits greater deviation in empirical coverage outside its training range.

5.1 Generalisation to unseen context lengths

First, we ask whether a probe trained on a fixed range of input sequence lengths generalises to longer or shorter contexts. We train and compare against each other two models:

- **Base:** Trained on input sequences $\mathbf{x}$ with lengths in range $[3, 40]$.
- **Restricted:** Trained only on input sequences $\mathbf{x}$ with lengths in range $[10, 20]$.

At test time, we evaluate both models on contexts shorter than 10 and longer than 20. We assess generalisation by measuring the empirical coverage of predicted confidence intervals, as defined in section 3.2. Figure 5 shows results on the dataset with the scale factor 1.0.

We observe that while both models achieve reasonable calibration, the Restricted model exhibits slightly greater deviations from the nominal coverage, particularly for context lengths further from the training distribution. These results suggest that the probe generalises to novel context lengths, but training on a wider context ranges should be beneficial for a more robust generalisation.

5.2 Applicability to real-world data

Thus far, our analysis has focused on synthetic data. In this section, we evaluate whether our probing model can be trained successfully on real-world datasets, and how well predictions can generalise across different types of input series.

Table 4: Coverage of the CI intervals on previously unseen testing inputs.

Model \ α	50%	90%	95%
Real (all)	48.8 ± 0.1	88.5 ± 0.1	94.3 ± 0.1
Real (5 fold)	43.4 ± 0.2	82.1 ± 0.2	89.4 ± 0.2
Synth	30.2 ± 0.3	67.7 ± 0.4	77.3 ± 0.4

To assess this, we construct a 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 for approximately 45,000 distinct sequences across 31 sub-datasets (e.g., US Births, 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. For this purpose, 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 datasets with scales 1.0, 10.0, 1000.0 and 10000.0.

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 an upper-bound baseline for Real (5 fold) and Synth.

In Table 4, we report the average coverage of the CI across all training types. We observe that the Real (all) model demonstrates good performance, with the empirical coverage of LLM samples closely matching the expected coverage. The Real (5 fold) model demonstrates a slight downgrade in performance. Interestingly, while the Synth model underperforms, it still demonstrates good generalisation for some of the sub-datasets as we can see on Figure 6. This figure shows the distribution of the absolute error of the predicted median vs. the median of LLM samples across

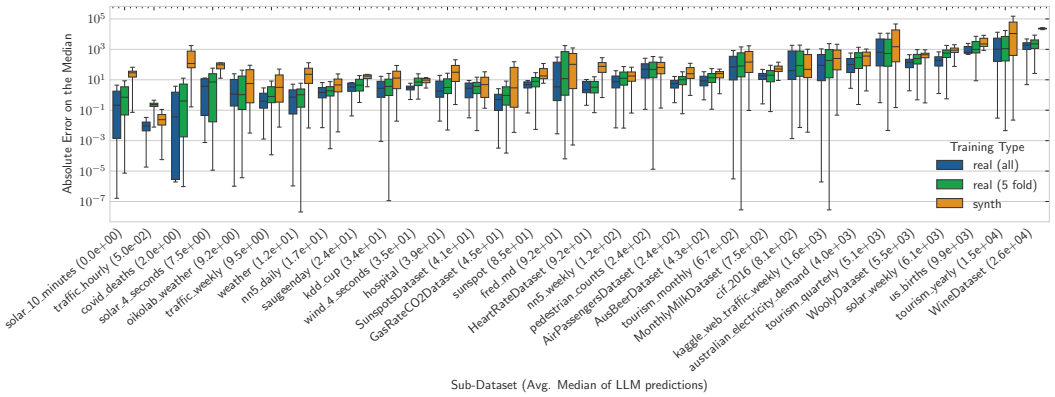


Figure 6: Absolute error on the median across different sub-dataset. Comparison of generalisation across models trained on different data.

all sub-datasets. The x-axis is sorted by increasing order of magnitude of the datasets defined by the average of the median of LLM samples. We note that the sub-datasets in our collection cover different ranges of values (with individual LLM samples varying in magnitude from 10^{-3} to 10^{13}). We suspect that this is the key reason why the probing model struggles to generalise across some datasets.

Key Insights. The probing model exhibit some, albeit limited, generalisation across context lengths. 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.

6 Discussion, Limitations and Further Work

While most probing studies of LLMs focus on predicting discriminative behaviours in natural language tasks, we instead target numerical prediction, which is particularly challenging due to high variance in output magnitudes. To address this, we introduce novel probes that decompose the task into magnitude classification and scaled value regression. Our findings show that LLMs encode rich numerical information about their predictions *before* autoregressive decoding. Training lightweight probes on hidden states allows us to recover both point estimates (mean, median, greedy outputs) and uncertainty. This suggests that much of the LLM's "reasoning" over numerical outputs occurs during input processing, with autoregressive decoding primarily simply surfacing the predictions. Beyond offering insight into how LLMs handle regression, our results also open a practical path: enabling uncertainty-aware numerical prediction without the computational cost of repeated sampling.

Despite these promising results, several limitations remain. First, our approach requires access to internal model activations, even though it does not involve fine-tuning the LLM itself. Second, while our probing models exhibit some generalisation ability, they are still model-specific, requiring retraining for each architecture or tokenization scheme. Third, training and evaluation requires approximating the LLM's predictive distribution via empirical sampling, which is an imperfect and computationally expensive proxy.

Future work could extend this framework to a broader range of structured data and prediction tasks, including univariate or multivariate regression and time-series forecasting, as well as multi-step ahead prediction tasks. While in this work we focused on quantifying the spread of the LLM distribution using quantile regression (which allowed us to obtain the estimates of the confidence intervals), future models can consider using alternative methods for modelling LLM's predictive distributions (e.g. Bayesian neural networks) which might offer a more fine-grained description of the distribution. Further, a deeper investigation into the mechanistic basis of numerical encoding– that is, how and where numerical quantities are represented across LLM layers–may also uncover connections to known computational circuits or arithmetic operations. Finally, motivated by our generalisation results, a key next step is the development of a universal probing model that can be applied off-the-shelf to a given LLM across diverse tasks and domains.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone, August 2024. URL http://arxiv.org/abs/2404.14219. arXiv:2404.14219 [cs].

Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. Exploring the Numerical Reasoning Capabilities of Language Models: A Comprehensive Analysis on Tabular Data. In Houda Bouamor, Juan Pino, and Kalika Bali (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/.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, November 2018. URL http://arxiv.org/abs/1610.01644. arXiv:1610.01644 [stat].

Amos Azaria and Tom Mitchell. The Internal State of an LLM Knows When It's Lying, October 2023. URL http://arxiv.org/abs/2304.13734. arXiv:2304.13734 [cs].

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting Latent Predictions from Transformers with the Tuned Lens, November 2025. URL http://arxiv.org/abs/2303.08112. arXiv:2303.08112 [cs].

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash Time Series Forecasting Archive, May 2021. URL http://arxiv.org/abs/2105.06643. arXiv:2105.06643 [cs].

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A Continuous Numerical Tokenization for Scientific Language Models, December 2024. URL http://arxiv.org/abs/2310.02989. arXiv:2310.02989 [stat].

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya

Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran

Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024. URL http://arxiv.org/abs/2407.21783. arXiv:2407.21783 [cs].

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large Language Models Are Zero-Shot Time Series Forecasters, August 2024. URL http://arxiv.org/abs/2310.07820. arXiv:2310.07820 [cs].

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot Classification of Tabular Data with Large Language Models, March 2023. URL http://arxiv.org/abs/2210.10723. arXiv:2210.10723 [cs].

Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Kościsz, Dennis Bader, Frédérick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gaël Grosch. 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.

Roger Koenker and Kevin F. Hallock. 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.

Boshko Koloski, Andrei Margeloiu, Xiangjian Jiang, Blaž Škrlj, Nikola Simidjievski, and Mateja Jamnik. LLM Embeddings for Deep Learning on Tabular Data, February 2025. URL http://arxiv.org/abs/2502.11596. arXiv:2502.11596 [cs] version: 1.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs, June 2024. URL http://arxiv.org/abs/2406.15927. arXiv:2406.15927 [cs].

Dmitrii Krasheninnikov, Richard E. Turner, and David Krueger. Fresh in memory: Training-order recency is linearly encoded in language model activations, September 2025. URL http://arxiv.org/abs/2509.14223. arXiv:2509.14223 [cs].

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, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the Biology of a Large Language Model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Oscar Obeso, Andy Arditi, Javier Ferrando, Joshua Freeman, Cameron Holmes, and Neel Nanda. Real-Time Detection of Hallucinated Entities in Long-Form Generation, September 2025. URL http://arxiv.org/abs/2509.03531. arXiv:2509.03531 [cs].

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations, May 2025. URL http://arxiv.org/abs/2410.02707. arXiv:2410.02707 [cs].

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future Lens: Anticipating Subsequent Tokens from a Single Hidden State. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560, 2023. doi: 10.18653/v1/2023.conll-1.37. URL http://arxiv.org/abs/2311.04897. arXiv:2311.04897 [cs].

Dario Di Palma, Alessandro De Bellis, Giovanni Servedio, Vito Walter Anelli, Fedelucio Narducci, and Tommaso Di Noia. LLaMAs Have Feelings Too: Unveiling Sentiment and Emotion Representations in LLaMA Models Through Probing, June 2025. URL http://arxiv.org/abs/2505.16491. arXiv:2505.16491 [cs].

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

Sarthak Roy, Ashish Harshavardhan, Animesh Mukherjee, and Punyajoy Saha. Probing LLMs for hate speech detection: strengths and vulnerabilities, October 2023. URL http://arxiv.org/abs/2310.12860. arXiv:2310.12860 [cs].

Eli Schwartz, Leshem Choshen, Joseph Shtok, Sivan Doveh, Leonid Karlinsky, and Assaf Arbelle. NumeroLogic: Number Encoding for Enhanced LLMs' Numerical Reasoning, March 2024. URL http://arxiv.org/abs/2404.00459. arXiv:2404.00459 [cs] version: 1.

Aliaksandra Shysheya, John Bronskill, James Requeima, Shoaib Ahmed Siddiqui, Javier Gonzalez, David Duvenaud, and Richard E. Turner. JoLT: Joint Probabilistic Predictions on Tabular Data Using LLMs, February 2025. URL http://arxiv.org/abs/2502.11877. arXiv:2502.11877 [stat].

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

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen

Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288. arXiv:2307.09288 [cs].

Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From Words to Numbers: Your Large Language Model Is Secretly A Capable Regressor When Given In-Context Examples, September 2024. URL http://arxiv.org/abs/2404.07544. arXiv:2404.07544 [cs].

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP Models Know Numbers? Probing Numeracy in Embeddings, September 2019. URL http://arxiv.org/abs/1909.07940. arXiv:1909.07940 [cs].

Jiaxin Wen, Pei Ke, Hao Sun, Zhexin Zhang, Chengfei Li, Jinfeng Bai, and Minlie Huang. Unveiling the Implicit Toxicity in Large Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1322–1338, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.84. URL https://aclanthology.org/2023.emnlp-main.84/.

Hao Xue and Flora D. Salim. PromptCast: A New Prompt-based Learning Paradigm for Time Series Forecasting, December 2023. URL http://arxiv.org/abs/2210.08964. arXiv:2210.08964 [stat].

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

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers Can Achieve Length Generalization But Not Robustly, February 2024. URL http://arxiv.org/abs/2402.09371. arXiv:2402.09371 [cs].

Angela Yaqian Zhu, Nandita Mitra, and Jason Roy. 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.

APPENDIX CONTENTS

A RELATED WORKS

LLMs for Structured Data. With the advances in the performance of LLMs, several methods have been proposed which utilise LLM's pre-training and prior knowledge to make predictions on structured data, namely *tabular data* (e.g. Requeima et al., 2024; Hegselmann et al., 2023; Shysheya et al., 2025; Vacareanu et al., 2024) and *time series data* (e.g. Gruver et al., 2024; Xue & Salim, 2023). These works show that LLMs can serve as competitive predictive models even *without task-specific fine-tuning*. This is particularly evident in the small sample regime, where large-scale pre-training, access to prior knowledge and ability to condition on textual information allows LLMs to match or outperform the performance of purpose-built regressors.

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 and LLMs. Probe classifiers, or simply probes, are models trained to uncover specific properties directly from the intermediate representations of neural models Alain & Bengio (2018). In the context of LLMs, a growing body of works from the area of mechanistic interpretability studied what kind of properties can be directly recovered from the internal representations of the LLMs (without the need for decoding), and where within the internal representations they are located (e.g. which layer). Amongst others, prior work has probed LLMs for properties such as factuality of the generated responses Obeso et al. (2025); Orgad et al. (2025); Azaria & Mitchell (2023), semantic entropy Kossen et al. (2024), toxicity Roy et al. (2023); Wen et al. (2023), text sentiment Palma et al. (2025) and even training order recency Krasheninnikov et al. (2025). While the previous works focus primarily on training probe classifiers, in this work we focus on *probe regressors*, proposing our magnitude-factorised probe as a method for handling targets with large range of continuous values (spanning several orders of magnitude). By focusing on continuous values, we uncover that the hidden states of the LLMs uncover much more fine-grained information than suggested by previous works (which probe primarily for binary–or binarised–properties).

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 several tokens ahead (Pal et al., 2023; Lindsey et al., 2025; Belrose et al., 2025). In contrast to these works, we show that similar results hold also when using LLMs for numerical predictions, allowing to uncover not only the marginal distributions of tokens at each of the steps ahead, but also the properties of the entire numerical predictive distribution of the LLM (mean, median, quantiles).

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.

Taken together, these results support the hypothesis that it should be possible to train probes that efficiently approximate the numerical *predictive distribution* of the LLM, motivating our work.

B Details of the Experimental Setup

B.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)
- **Llama-3-8B model** Grattafiori et al. (2024)
- **Llama-3-3B model** Grattafiori et al. (2024)
- **Phi-3.5-mini model** Abdin et al. (2024)
- **DeepSeek-R1-Distill-Llama-8B model** DeepSeek-AI (2025)

B.2 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. These instances are a part of Azure's GPU-optimised virtual machine series, with their hardware specifications summarised in Table 5.

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

Table 5: 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

B.3 Details of the datasets

B.3.1 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 $x \in [0, 60]$, discretised into 120 equidistant points. The functions are summarised in Table 6. 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 $b \sim \mathcal{U}(0, D_{\text{scale}})$
- Vertical translation by $d \sim \mathcal{U}(-D_{\text{scale}}, D_{\text{scale}})$

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 $D_{\text{scale}} = 1.0$, $p = 3$ for $D_{\text{scale}} = 10.0$, $p = 2$ for $D_{\text{scale}} = 1000.0$ and $p = 1$ for $D_{\text{scale}} = 10000.0$). This results in 33600 generated time series for each value of D_{scale}.

Concatenated dataset. Having constructed the individual dataset for each scaling factor $D_{\text{scale}} \in \{1, 10, 1000, 10000\}$, we also construct one concatenated dataset. In doing that, we limit the number of datapoints to 70000 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 $[-D_{\text{scale}}, D_{\text{scale}}]$.

B.3.2 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 complete 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.

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

- **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.

B.3.3 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.

B.3.4 LLM GENERATION SETTINGS

We generate the LLM hidden states from LLMs available through the `huggingface` library. For each of the input time series, we obtain 100 samples from the LLM, generated auto-regressively, as well as the greedy generation. During generation for Llama-2-7b, as its tokenizer encodes each digit separately, we narrow down the generated tokens to just the 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.

B.3.5 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.

B.4 DETAILS OF THE PROBING MODELS

Our magnitude-factorised regression models, used both for the purpose of point prediction and for the purpose of quantile regression have the hyperparameters as reported in Table 7 and Table 8. We train the model using the ADAM optimiser.

We also report any deviations from these values for specific experiments:

- **Figure 2.** `max_mag` $= 4$.
- **Table 1.** `lr` $= 10^{-4}$, `max_epochs` $= 2000$.

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	$\log_{10}(D_{\text{scale}})$
`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
`activation_func`	Activation function used in the MLP	GELU
`hidden_states_list`	A list of the hidden states $\mathcal{H}$ to use as input	$[25, \ldots, 32]$

Table 7: Model-specific hyperparameters for the magnitude-factorised models.

Hyperparameter	Description	Default Value
`learning_rate`	Learning rate for the optimizer	10^{-5}
`weight_decay`	L2 regularization weight	1.0
`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 8: Optimizer and training-related hyperparameters.

- **Figure 6.** `min_mag` $= 10$ `max_mag` $= 10$.

C Additional Experimental Results

C.1 Comparison against vanilla MLP probe

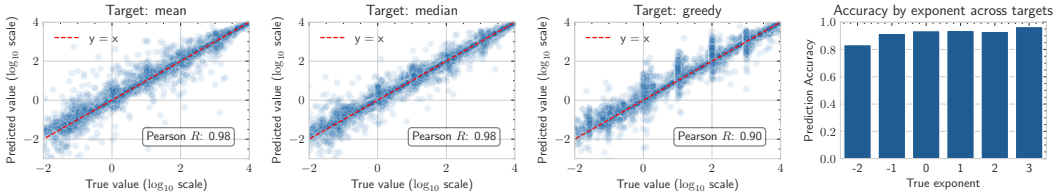
In Table 9, we compare our magnitude-factorised probe with a vanilla MLP trained on the same hidden representations. Both models use a single hidden layer of size 512 and a learning rate of 10^{-5}. Evaluation is conducted on Llama-2 using the synthetic time series dataset with scale $D_{\text{scale}} = 1.0$, which provides the narrowest range of magnitudes among those we study. Even in this relatively simple setting, the magnitude-factorised probe achieves substantial improvements over the baseline, reducing MSE by 41%, 33%, and 42% for greedy, mean, and median predictions respectively. These results highlight the importance of probe design: a negative finding with one architecture does not necessarily imply that information is absent from the LLM's hidden states—it may simply reflect the limitations of the probing model.

Table 9: MSE of the values predicted with the magnitude-factorised (MFP) model vs. a vanilla MLP.

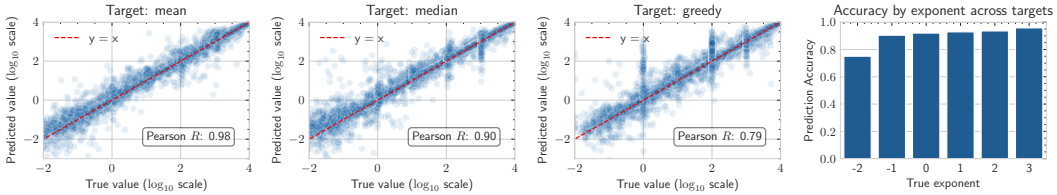
predicted value	MFP probe (ours)	MLP probe (log-scaling)	MLP probe (no log-scaling)
greedy	0.0150	0.0400	0.0255
mean	0.0061	0.0091	0.0092
median	0.0058	0.0101	0.0100

C.2 RESULTS WITH OTHER LLMS

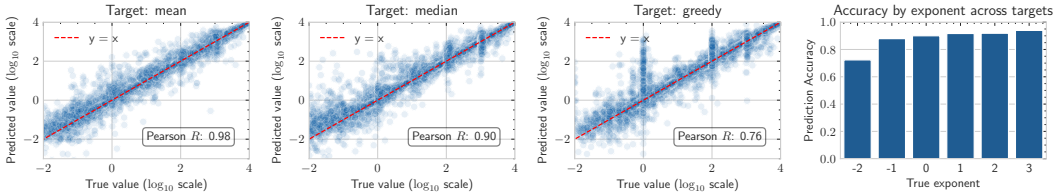
We provide results for the key experiments in the main paper with more LLMs. As the tokenizers of models outside of the Llama-2 family do not encode digits separately, we *do not* narrow down the generated tokens during decoding. For obtaining the random samples, we use `temperature=1.0` and `top_p=0.95`. We perform repeated sampling until for each time series, we obtain 100 LLM samples $y_i^j \sim p_{\text{LLM}}(\cdot|\mathbf{x}_i)$.



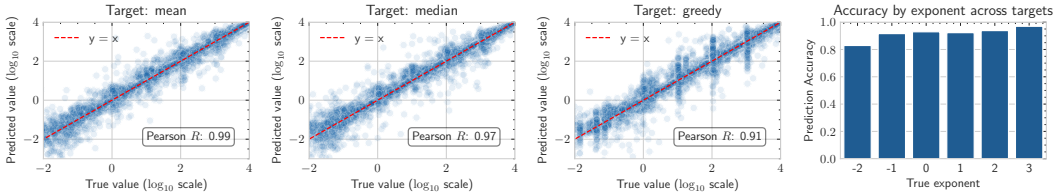
(a) Llama-2-7B



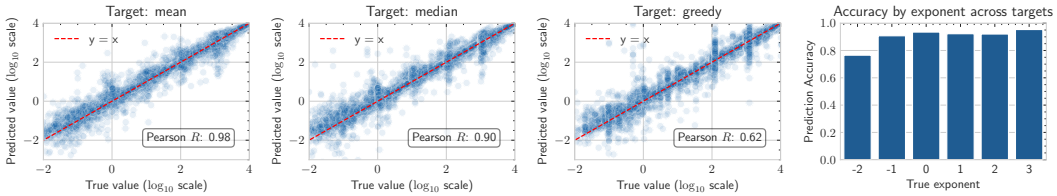
(b) Llama-3-8B



(c) Llama-3.2-3B

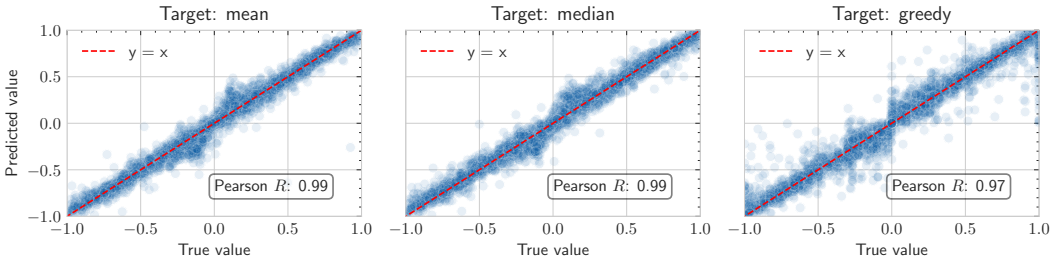


(d) Phi-3.5-instruct

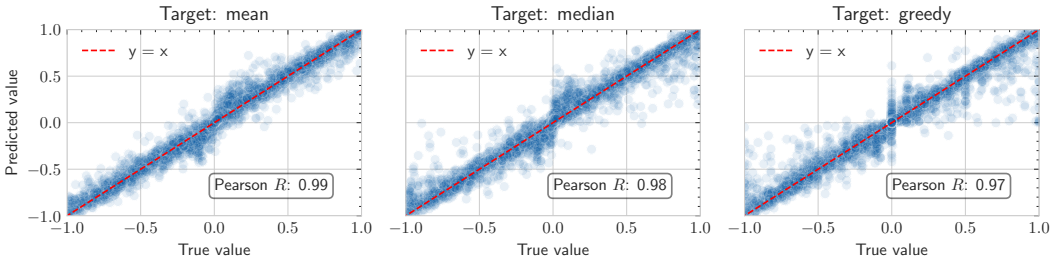


(e) DeepSeek-R1-Distill-Llama-8B

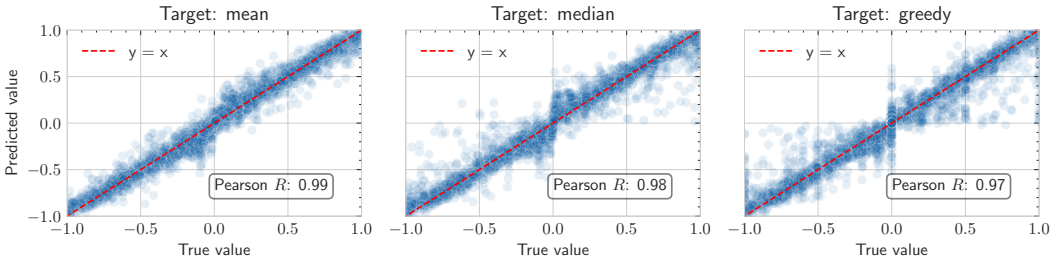
Figure 7: Predicted vs. sample mean, median and greedy prediction (on $\log_{10}$ scale). Results analogous to Figure 2.



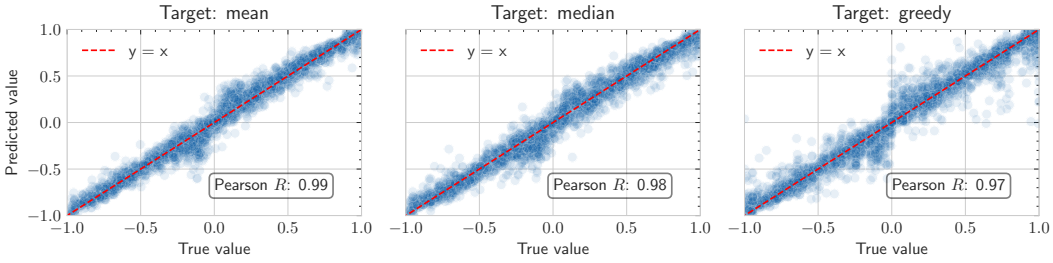
(a) Llama-2-7B



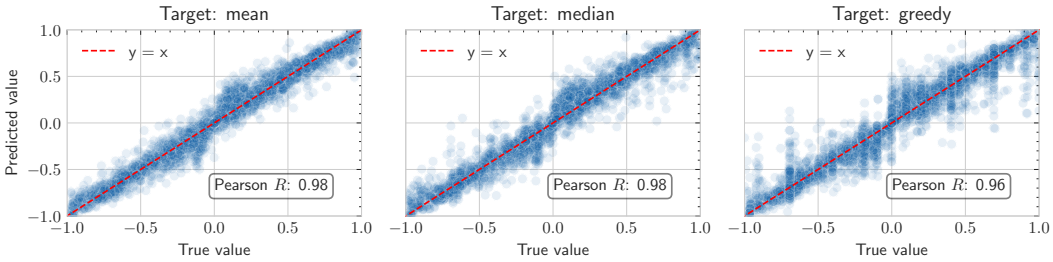
(b) Llama-3-8B



(c) Llama-3.2-3B

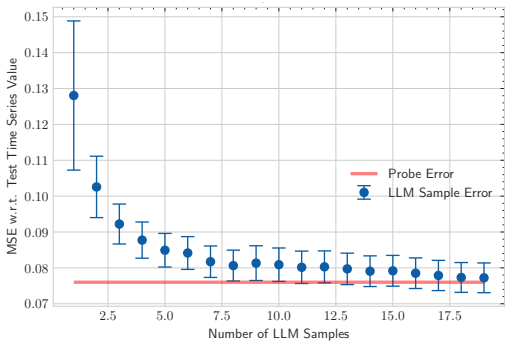


(d) Phi-3.5-instruct

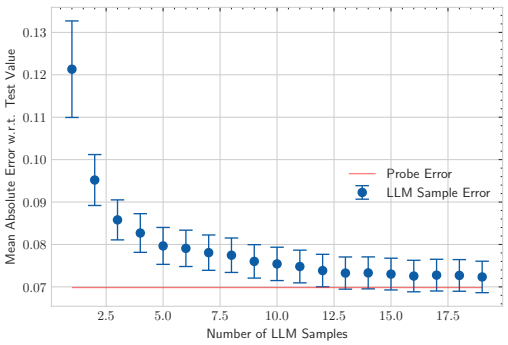


(e) DeepSeek-R1-Distill-Llama-8B

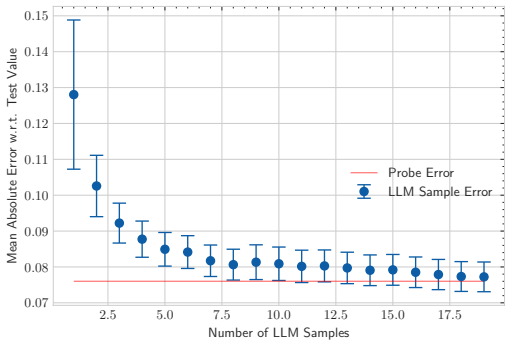
Figure 8: Predicted vs. sample mean, median and greedy prediction on the dataset with scale $D_{\text{scale}} = 1.0$.



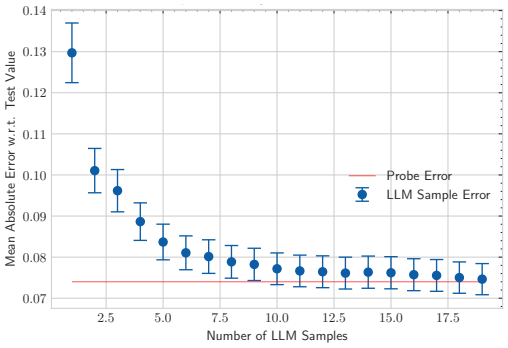
(a) Llama-2-7B



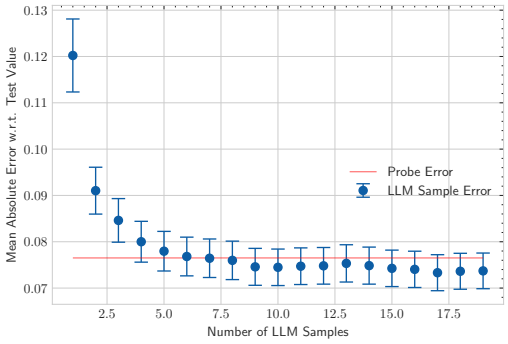
(b) Llama-3-8B



(c) Llama-3.2-3B



(d) Phi-3.5-instruct



(e) DeepSeek-R1-Distill-Llama-8B

Figure 9: Sample efficiency on the task of one step ahead predictions. Results analogous to Figure 4.

Table 10: MSE for the predictions on the dataset with scale $D_{\text{scale}} = 1.0$, reported for all models. Results analogous to Table 1.

(a) Llama-2-7B

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$* =$ mean	0.006	0.256	0.035	0.085
$* =$ median	0.006	0.260	0.041	0.087
$* =$ greedy	0.015	0.273	0.065	0.109

(b) Llama-3-8B

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$* =$ mean	0.007	0.253	0.047	0.093
$* =$ median	0.012	0.264	0.061	0.106
$* =$ greedy	0.016	0.255	0.072	0.122

(c) Llama-3.2-3B

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$* =$ mean	0.007	0.260	0.0484	0.092
$* =$ median	0.013	0.271	0.062	0.104
$* =$ greedy	0.018	0.267	0.075	0.119

(d) Phi-3.5-mini-instruct

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$* =$ mean	0.008	0.248	0.042	0.100
$* =$ median	0.012	0.252	0.047	0.104
$* =$ greedy	0.020	0.270	0.060	0.113

(e) DeepSeek-R1-Distill-Llama-8B

y_i^*, target	$\hat{y}_i^*$ (ours)	$\bar{\mathbf{x}}$	$\bar{\mathbf{x}}_i$	$x_{i,n}$
$* =$ mean	0.009	0.247	0.045	0.110
$* =$ median	0.013	0.253	0.056	0.120
$* =$ greedy	0.020	0.264	0.069	0.135

Table 11: Coverage of the CI for all models. Results analogous to Table 2.

(a) Llama-2-7B

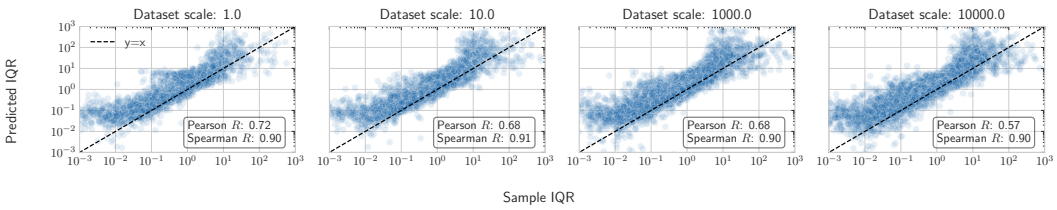
α dataset	50%	90%	95%
1.0	52.0 ± 0.4	90.9 ± 0.3	95.5 ± 0.2
10.0	52.7 ± 0.5	91.3 ± 0.3	96.1 ± 0.2
1000.0	51.4 ± 0.3	90.7 ± 0.3	95.7 ± 0.2
10000.0	48.2 ± 0.3	90.5 ± 0.2	95.4 ± 0.2

(b) Llama-3-8B

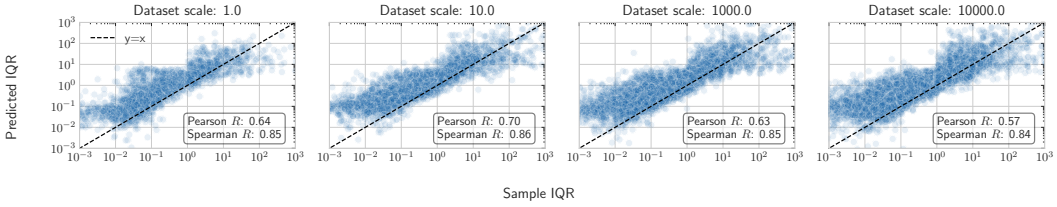
α dataset	50%	90%	95%
1.0	54.4 ± 0.6	90.4 ± 0.4	94.7 ± 0.3
10.0	53.8 ± 0.6	91.8 ± 0.4	96.3 ± 0.2
1000.0	51.7 ± 0.4	91.2 ± 0.3	96.0 ± 0.2
10000.0	50.8 ± 0.4	91.1 ± 0.3	95.3 ± 0.2

(c) Phi-3.5-mini-instruct

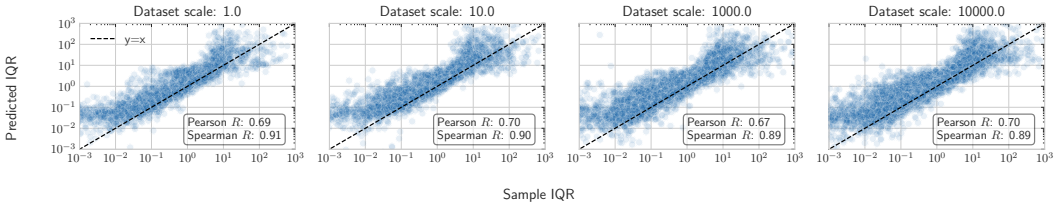
α dataset	50%	90%	95%
1.0	49.9 ± 0.5	89.3 ± 0.4	94.3 ± 0.3
10.0	50.9 ± 0.5	89.3 ± 0.4	95.1 ± 0.3
1000.0	47.6 ± 0.4	88.0 ± 0.3	93.3 ± 0.3
10000.0	47.3 ± 0.3	87.3 ± 0.3	93.8 ± 0.2



(a) Llama-2-7B



(b) Llama-3-8B



(c) Phi-3.5-instruct

Figure 10: Predicted IQR vs. Sample IQR (median adjusted). Results analogous to Figure 3.

C.3 Computational Costs and Inference Time

Crucial for understanding the validity of our approach is the comparison of the inference cost of the proposed probe vs. autoregressive sampling. Ignoring the cost of encoding the input time series with the LLM (as this cost is shared by both methods) we note that decoding each token with the 7B-parameter LLMs (such as Llama-2) requires $14B$ FLOPS, which means that, for a model with digit-by-digit tokenization obtaining n samples of a 5-digit number requires $n \times 70B$ FLOPS. In comparison, using our magnitude-factorised probe (which we assume uses 8 concatenated layers of the LLM and thus has input of size $8 \times 4,096 = 32,768$, 1 hidden layer of dimension 512, and a maximum of $k = 9$ separate heads corresponding to each of the magnitude bins) incurs the constant cost of $32,768 \times 512 + 512 \times 9 \approx 17M$ FLOPS for the magnitude classification model and $32,768 \times 512 + 512 \times 1 \approx 17M$ FLOPS for the regression model, so $34M$ parameters in total per one statistic. If we wanted to predict, say, 7 different quantile values, we would need 7 models of the same size, requiring $234M$ FLOPS. Thus, the required number of computations for using the probe vs. generating even 1 LLM sample is far lower.

We further validate these estimates by comparing the times needed to generate n samples from the LLM vs. the time needed to obtain a single median estimate from our probe. We run the experiments using the Llama-2-7B model, averaging the results over 100 random time series samples consisting of 20 data points each. We run all the timing experiments on one instance of a H100 GPU. In this setting, inference with a trained probe model (including the process of obtaining the hidden representation of the LLM) takes 0.034 ± 0.006s. We include the times required for obtaining n LLM samples in Table 12. These results confirm that even generating a single sample via autoregression is roughly 47× slower than running the full inference pipeline with our probe.

Table 12: Average time to generate n LLM samples.

n samples	1	5	10	20	50	100
time (s)	1.59 ± 0.08	1.62 ± 0.07	1.71 ± 0.07	1.83 ± 0.07	2.35 ± 0.08	3.28 ± 0.08

C.4 Layer Ablation Study

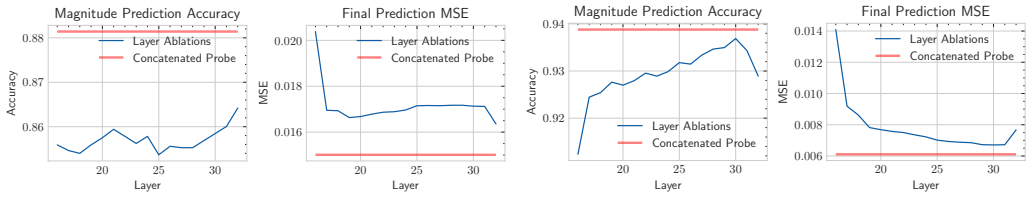
In this section, we ablate the choice of the layers $\mathcal{H}$ used by our probing models by training on individual layers of the LLM, letting $\mathcal{H} = \{\ell\}$ for each $\ell \in [16, \ldots, 32]$. Our results provide further insights into how information about the LLM's numerical predictive distribution is spread across the layers.

Figure 11 shows the results for the probing models from section 2. For the target of the greedy LLM prediction, we see that the most relevant information is contained in the last layers of the LLM, although generally the variations between the layers are not that significant. We further observe that concatenating information from across different layers leads to significant performance improvements. For the mean target, which is predicted with a higher accuracy, we observe that information relevant for predicting the magnitude and the scaled value of the target is distributed less uniformly, concentrating in the 30th layer.

Table 12 shows analogous results for the quantile regression model from section 3.2. The observed pattern for median prediction is similar to the one of mean prediction from Figure 11. In terms of the uncertainty information, we find it to be more uniformly encoded across the layers and again that concatenating information across several layers leads to an improved probing performance.

C.5 Mean Absolute Loss per quantile

In addition to the results presented in Section 3, we report the mean absolute error between the the empirical quantile $\tilde{q}^s$ and the predicted quantile $\hat{q}^s$ for all quantile levels. Results are shown in Table 13. We observe that the absolute errors are higher for quantile levels further from the median.



(a) Prediction of y_i^{greedy} (b) Prediction of y_i^{mean}

Figure 11: *Layer ablation for Llama-2-7B on the probing model from section 2.*

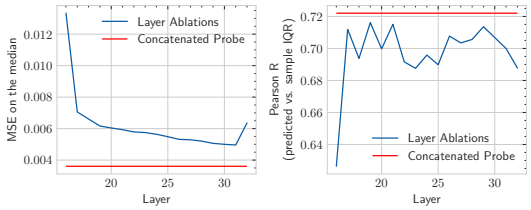


Figure 12: *Layer ablation for Llama-2-7B on the quantile probing model from section 3.* MSE on the median (left, lower is better) and the Pearson R between predicted IQR and sample IQR (right, higher is better).

Table 13: *Quantile regression – mean absolute error per quantile.* Values represent the mean absolute error between the empirical quantile $\tilde{q}^s$ and the predicted quantile $\hat{q}^s$ for all quantile levels.

quantile level τ^s dataset	0.025	0.05	0.25	0.5	0.75	0.95	0.97
1.0	0.58	0.40	0.16	0.05	0.15	0.40	0.60
10.0	3.12	2.16	0.45	0.28	0.49	1.91	3.26
1000.0	111.16	61.54	14.83	12.10	14.30	46.04	115.71
10000.0	843.11	380.06	119.80	100.41	118.90	483.33	1172.27