Lag-Llama: Towards Foundation Models for Time Series Forecasting

Kashif Rasul[1] **Arjun Ashok**[2,3,4]
[†]**Andrew Robert Williams**[3,4] [†]**Arian Khorasani**[3,4] [†]**George Adamopoulos**[5]
[†]**Rishika Bhagwatkar**[3,4] [†]**Marin Biloš**[1] [†]**Hena Ghonia**[3,4] [†]**Nadhir Hassen**[3,4]
[◇]**Anderson Schneider**[1] [◇]**Sahil Garg**[1] [◇]**Alexandre Drouin**[2,4] [◇]**Nicolas Chapados**[2,4]
[♠]**Yuriy Nevmyvaka**[1] [♠]**Irina Rish**[3,4]
[1]Morgan Stanley, New York, USA
[2]ServiceNow Research, Montréal, Canada
[3]Université de Montréal, Montréal, Canada
[4]Mila, Montréal, Canada
[5]McGill University, Montréal, Canada

Abstract

Aiming to build foundation models for time-series forecasting and study their scaling behavior, we present here our work-in-progress on `Lag-Llama`, a general-purpose univariate probabilistic time-series forecasting model trained on a large collection of time-series data. The model shows good zero-shot prediction capabilities on unseen "out-of-distribution" time-series datasets, outperforming supervised baselines. We use *smoothly broken power-laws* [7] to fit and predict model scaling behavior. The open source code is made available at `https://github.com/kashif/pytorch-transformer-ts`.

1 Introduction

Probabilistic time-series forecasting is an important practical problem arising in a wide range of applications, from finance and weather forecasting to brain imaging and computer systems performance management [32]. Various methods have been proposed for this task, ranging from classical autoregressive models [19] to the more recent neural forecasting methods based on deep learning architectures [44]. The majority of these previous approaches focus on training the model on the data from the same domain wherein the prediction task is performed.

However, in the past several years, machine-learning is witnessing a paradigm shift due to the rise of *foundation models* [5]—large-scale, general-purpose neural networks pretrained in an unsupervised manner on large amounts of diverse data; such models demonstrate remarkable few-shot generalization capabilities on a wide range of downstream tasks [6], often outperforming task-specific models. Following the successes of foundation models in language and image processing domains[30, 33], we aim to develop foundation models for time-series, investigate their behavior at scale, and push the limits of transfer achievable across diverse time-series domains.

In this paper, we present preliminary results of our ongoing work along those lines. We train a transformer model on a large collection of time-series datasets and evaluate its performance on an unseen "out-of-distribution" dataset. Specifically, we investigate the use of pre-trained time series models for the *univariate* probabilistic time series forecasting use case and introduce the `Lag-Llama` model[1] trained on a large collection of time series from the Monash Time Series Repository [16].

[†]Equal Contribution , [◇]Equal Contribution , [♠]Equal Contribution
Corresponding authors: `kashif.rasul@gmail.com`, `arjun.ashok@servicenow.com`
[1]Note that the name of our model `Lag-Llama`, although inspired by Llama [45], is used to signify architectural similarities only, and *not* the use of any pretrained language model weights released by [45] or other works.

R0-FoMo: Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models at NeurIPS 2023.

We report the test-set performance of this model on unseen time-series datasets and present a neural scaling laws study on the number of parameters and training data.

Our contributions:

- We propose `Lag-Llama`, a model for univariate probabilistic time-series forecasting suitable for scaling law analyses of time series foundation models.
- We train `Lag-Llama` on a corpus of time series datasets and show that `Lag-Llama` outperforms or compares favorably to supervised baselines when tested *zero-shot* on unseen time series datasets, and we identify a "stable" regime where the model constantly outperforms the baselines beyond a certain model size.
- We fit *empirical scaling laws* of the zero-shot test performance of the model as a function of the model size, allowing us to potentially extrapolate and predict generalization beyond the models used in this paper.

2 Related Work

Neural forecasting is a rapidly developing research area [4]; its primary focus so far has been mainly on training and forecasting within individual datasets. We instead focus on building generic foundation models on a wide range of diverse time-series data. Recent work has investigated the use of pre-trained language models as frozen encoders, e.g., in `Time-LLM` [20], `LLM4TS` [8] and `GPT2(6)` [51], while simultaneously fine-tuning/adapting the input and distribution heads for forecasting. Whereas these models use patches, the `Lag-Llama` model proposed here differs primarily through its use of lag-features as explained in Sec. 4.1.

In parallel, *self-supervised learning* techniques have been proposed for time series [24, 48, 49]. Most related to our work is Yeh et al. [49] who train on a corpus of time series datasets. The key difference is that they validate their model only on classification tasks downstream, and do not validate on forecasting tasks. Another related work is `TimeGPT-1` [13] a proprietary foundation model for time series forecasting. One key difference between [13] and ours is that they utilize the machinery of conformal prediction for uncertainty quantification after the point-forecasting emission head, while our model is built directly for probabilistic forecasting. [23] demonstrate data scaling for transformer-based forecasting models trained on proprietary data.

Foundation models are an emerging paradigm of self-supervised deep learning on extensive datasets [5]. Many such models [11, 30, 10, 33, 47] have demonstrated adaptability across domains, extending beyond web data to scientific domains such as protein design [35]. Scaling the model and dataset size was shown to result in remarkable transfer capabilities and excellent few-shot learning on novel tasks [43]. Scale-driven transfer learning [43] unifies tasks, domains, and modalities. The main goal of our work is to apply the foundation model approach to the time-series data and to investigate the extent of the transfer achievable across a wide range of time-series domains.

When it comes to quantifying the resources needed to train foundation models, *neural scaling laws* seek to predict a neural network's performance based on quantities of interest such as model size, training dataset size, and computational resources, among others [27, 3, 7]. We adopt the methodology introduced in [7] to fit scaling laws to our model's performance. This approach employs a broken power law functional form to model and extrapolate nonlinear scaling behaviors of deep neural networks.

3 Probabilistic Time Series Forecasting

We assume a training dataset of $D \geq 1$ time series sampled in discrete time, $\mathcal{D}_{\text{train}} = \{x^i_{1:T^i}\}_{i=1}^D$, where at each time point $t \in \{1, \ldots, T^i\}$, $x^i_t \in \mathbb{R}$ (or a subset thereof, such as $\mathbb{N}$). We wish to predict $P \geq 1$ steps into the future; as such, we require a back-testing test set of these D time series denoted by $\mathcal{D}_{\text{test}} = \{x^i_{T^i+1:T^i+P}\}_{i=1}^D$ or some held-out time series dataset. Even though we use t to refer to the order of elements in a given time series, each x^i_t also has a distinct date-time associated with it, which increments regularly based on the frequency of the dataset. We also include covariates $\mathbf{c}^i_t$ as *vector-valued* information associated with each x^i_t, as described in Sec. 4.1. The covariates are assumed to be non-stochastic and available in advance for the P future time points.

In the *univariate* probabilistic time series forecasting problem, we wish to model the potentially complex unknown joint distribution of the P future values of a one-dimensional sequence given its observed past and covariate features:

$$p_{\mathcal{X}}(x^i_{t+1:t+P} \mid x^i_{1:t}, \mathbf{c}^i_{1:t+P}). \tag{1}$$

Rather than considering the whole history of each time series i, which can vary considerably, we can instead sub-sample fixed context windows of size $C \geq 1$ of our choosing from the full time series (which we assume, without loss of generality, to be sampled starting from $t = 1$) and learn an approximation of the unknown distribution of the next P future values given the covariates:

$$p_{\mathcal{X}}(x^i_{C+1:C+P} \mid x^i_{1:C}, \mathbf{c}^i_{1:C-1+P}). \tag{2}$$

Thus, if we denote the parameters of our deep learning model by θ, we can approximate Eqn. 2 by an autoregressive model which we can write via the chain rule of probability as

$$p_{\mathcal{X}}(x^i_{C+1:C+P} \mid x^i_{1:C}, \mathbf{c}^i_{1:C-1+P}; \theta) = \prod_{t=C+1}^{C+P} p_{\mathcal{X}}(x^i_t \mid x^i_{1:t-1}, \mathbf{c}^i_{1:t-1}; \theta),$$

where we assume that the covariates are available for all future time points.

4 Lag-Llama

Our strategy is to train a single model over a large corpus of time series, the details of which are in App. Table 4. A number of challenges arise in this setting. For one, we are limited to a univariate model as the multivariate dimension obtained by grouping the time series varies for each dataset. This work considers univariate probabilistic forecasting, much simpler than the multivariate case. Another reason is that we need to sample from each dataset in such a way as to learn representations from all available datasets without disproportionately sampling from any one dataset.

To use Transformer-based architectures [46], we need vector-valued inputs, but the frequency of the time series in our corpus varies. We therefore need to vectorize univariate data in a way that accounts for the frequency of the specific datasets that make up our corpus. We now present a general method to vectorize series from such a dataset given the frequency of the specific datasets.

4.1 Lag Features

The *only covariates* we employ in this model are from the target values, in particular lag features, where the lags are constructed from a set of appropriate lag indices for quarterly, monthly, weekly, daily, hourly, and second-level frequencies that correspond to the frequencies in our corpus of time series data (see App. Table 4). Given a sorted set of lag indices $\mathcal{L} = \{1, \ldots, L\}$, we define the lag operation on a particular time value as $x_t \mapsto \mathbf{c}_t \in \mathbb{R}^{|\mathcal{L}|}$ where each entry j of $\mathbf{c}_t$ is given by $\mathbf{c}_t[j] = x_{t-\mathcal{L}[j]}$. Thus to create lag features for some context-length window $x_{1:C}$ we need to sample a larger window with L more historical points denoted by $x_{-L+1:C}$. The resulting vectors capture the underlying temporal patterns of the signal.

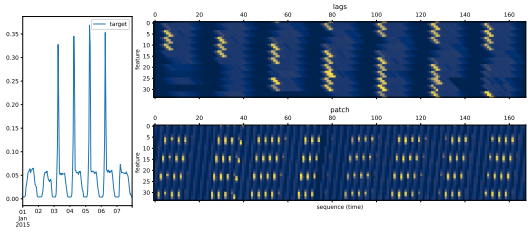

Figure 1: *For a time series (left), we depict the sequence of lag feature vector values (our method, top-right) and same-sized patch feature vector values (bottom-right). The sequence of lag vectors captures the periodicity of the time series, whereas the patch vectors are causally mixed. Each lag vector contains data from only the previous vectors, while each patch vector contains values from the previous and next vectors.*

An alternative approach to vectorizing a univariate series is to use potentially overlapped patches [29] or segments of a certain size and stride, resulting in a sequence of vectors whose dimension can be specified. However, this approach can lead to vectors whose entries are causally mixed. See Fig. 1 for an example of both approaches.

While both approaches essentially serve the same purpose, the indices of the lags correspond directly to the various possible seasonalities of the data, and have the advantage of preserving the date-time index causal structure. By relying on lag features, we can therefore use masked decoders during training and autoregressive sampling at inference time, which is not trivial with patches. Nonetheless, we note that a downside to using lags is that it requires an L-sized or larger context window at inference time.

4.2 Lag-Llama Architecture

Lag-Llama's architecture is based on the recent LlaMA [45] architecture which incorporates pre-normalization via the RMSNorm [50] and adds Rotary Positional Encoding (RoPE) [41] to each attention layer's query and key representations.

Fig. 2 shows a general schematic of this model with M decoder layers. A univariate sequence of length $x^i_{-L+1:C}$ along with its covariates is vectorized to a sequence of C vectors $\mathbf{x}^i_{1:C}$ via the lag operation. These are passed through a shared linear projection layer that maps the features to the hidden dimension of the attention module. After passing through the masked Transformer layers, the model predicts the parameters of some chosen distribution with a distribution head, as described in Sec. 4.3. The negative log-likelihood of the predicted distribution over the prediction window is minimized with the actual values as the ground truth.

At inference time, given a time series of size at least L, we can construct a feature vector that is passed to the model to obtain the distribution of the next time point. We can obtain many samples from the predicted distribution and concatenate them to the initial sequence to obtain further lag vectors. In this fashion, via *greedy* autoregressive decoding, we are able to obtain many simulated trajectories of the future up to our chosen prediction horizon $P \geq 1$. From these empirical samples, we can calculate the uncertainty intervals for downstream decision-making tasks and metrics with respect to held-out data.

4.3 Choice of Distribution Head

The last layer of `Lag-Llama` is a distinct layer known as the *distribution head*, which projects the model's features to the parameters of a probability distribution. We can combine different distribution heads with the representational capacity of the model to output the parameters of any parametric probability distribution. For our initial experiments, we use a Student's t-distribution [40] and output the three parameters corresponding to this distribution, namely its degrees of freedom, mean, and scale. More expressive choices of distributions, such as normalizing flows [34] and copulas [36, 12, 2] are left as future work.

4.4 Value Scaling

One particular challenge of time series data is that the time series in a dataset can have any numerical magnitude, which is perhaps not the case in image, audio, or NLP data. Since we train a single shared model with distribution head over such data, we utilize the scaling heuristic from [37]. For each univariate window, we calculate its mean value $\mu^i = \sum_{t=-L}^{C} x^i_t/(C+L)$ and variance σ^i. We can then replace the time series $\{x^i_t\}_{t=-L}^{C}$ in the window by $\{(x^i_t - \mu^i)/\sigma^i\}_{t=-L}^{C}$. We also incorporate μ^i via $\text{sign}(\mu^i)\log(1+|\mu^i|)$ as well as $\log(\sigma^i)$ as time independent real-valued covariates.

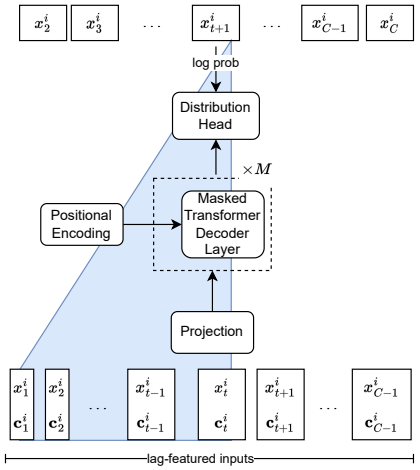

Figure 2: *The `Lag-Llama` architecture. `Lag-Llama` learns to output a distribution over the values of the next time step based on lagged input features. The input at each timestep is the value of a univariate time series at a given timestep x^i_t concatenated with its covariate vector $\mathbf{c}^i_t$ constructed via the lag operation described in Sec. 4.1. The inputs are projected through M masked decoder layers. The features are then mapped to the parameters of a probability distribution through an additional layer called the distribution head, on which we elaborate in Sec. 4.3.*

During training and obtaining likelihood, the values are transformed using the mean and variance, while sampling, every timestep of data that is sampled is de-standardized using the same mean and variance. This technique predates the recent related `RevIN` [22] method and Non-stationary Transformers [26].

4.5 Augmentation

To prevent overfitting, we apply time series augmentation techniques, namely `Freq-Mix` and `Freq-Mask` [9], to each batch with a probability of 0.5 and a randomized augmentation rate of 0.1. In order to not over/under sample we employ stratified sampling where datasets in our collection are weighed by the amount of *total* temporal time points when sampling random training windows.

5 Experiments

5.1 Dataset

We train `Lag-Llama` on all public time series in the Monash Time Series Repository [16], as well as from other sources typically used in time series research [1]. These datasets have diverse frequencies

Table 1: *Zero-Shot performance of* `Lag-Llama` *on two datasets, compared to baselines trained on the datasets. The best value of each dataset is in* ***bold****. Confidence intervals are reported wherever available. The second best value is in brown. Results are reported on the CRPS metric. Lower is better.*

Model	Dataset	
	`m4-weekly`	`traffic`
Supervised		
AutoARIMA	0.050	N/A
AutoETS	0.052	0.52
AutoGluon Best [39]	**0.041**	0.166
AutoTheta	0.053	1.054
DeepAR [38]	0.046	0.166
TFT [25]	0.049	0.167
Naive	N/A	0.79
SeasonalNaive	0.073	0.332
Zero shot		
`Lag-Llama` (ours)	0.0691 ± 0.0061	$\mathbf{0.149 \pm 0.0058}$
`Lag-Transformer` (ours)	0.0607 ± 0.0066	0.268 ± 0.0048

and come from different application domains; the properties of these datasets are outlined in App. Table 4. In total, our training set comprises a total of 305,443 individual time series.

We reserve the `M4 Weekly` and `Traffic` datasets as our test sets. These test sets are completely *unseen* during training and are used to test the out-of-distribution generalization performance of the model. We pick these test sets arbitrarily, but other choices of training-test splits will be explored in future work.

5.2 Model Training and Specification

We train the models with a batch size B of 100 and a learning rate α of 10^{-3}. Training stops if the validation loss does not improve for 50 validation epochs, where each epoch consists of 100 windows, sampled as described in Sec. 4.5.

We train models with a random grid across the parameters in Table 2. Early stopping is performed using the average loss on the *held-out splits* of the respective datasets used during training. Each model is trained on 6 different seeds that constitute different initializations and data orders. The other hyperparameters are held constant; their values can be found in App. Sec. A.1.

5.3 Evaluation

The models are evaluated on the *test split* of the unseen test set(s). Evaluation is performed by *sampling* from the model autoregressively starting with conditioning on the context, and until a prediction length as defined for the respective test dataset. We sample 100 samples for each timestep.

We then use the Continuous Ranked Probability Score (CRPS) metric [15, 28] score as our evaluation metric. CRPS is a *proper* scoring rule that measures the compatibility of a predicted cumulative distribution function (CDF) F with the ground-truth sample x as

$$\mathrm{CRPS}(F, x) = \int_{\mathbb{R}} (F(y) - \mathbb{I}\{x \leq y\})^2 \, \mathrm{d}y,$$

where $\mathbb{I}\{x \leq y\}$ is 1 if $x \leq y$ and 0 otherwise. We approximate the CDF via empirical samples at each time point, and the final metric is averaged over the prediction horizon of the time series and across the time series of a dataset.

6 Results

6.1 Zero-Shot Performance

Preliminary results on the zero-shot performance of `Lag-Llama` and `Lag-Transformer` are reported in Table 1, in comparison with baselines from [39]. All baselines are trained solely on the *train splits* of the respective test datasets with supervised learning. `Lag-Llama` and `Lag-Transformer` are only

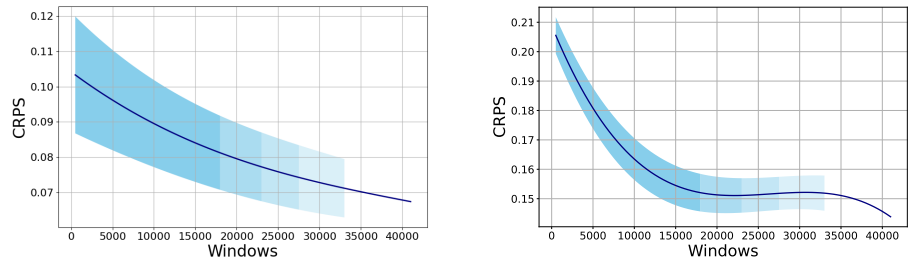

Figure 3: *Zero-shot CRPS of the best* `Lag-Llama` *with respect to the number of data windows seen during training from the training corpus of datasets (average and standard error of 6 seeds) for* `M4 Weekly` *(left) and* `Traffic` *(right).*

pretrained on a corpus of time series and are the only models that never see either of the two test datasets, as detailed in Table 4. `Lag-Llama` achieves better performance on `Traffic` *zero-shot*, surpassing all supervised baselines, while `Lag-Transformer` does not perform well on `Traffic`. While on the `m4-weekly` dataset, both `Lag-Llama` and `Lag-Transformer` do not outperform the baselines. This shows that there is great scope for detailed analysis of both architectures with different training and test splits that can lead to improved *data sampling* or *model selection* choices.

6.2 Performance of the model with increasing number of training windows

For the best model of `Lag-Llama` obtained above on both unseen test datasets, we zoom in further into how many windows the models need to see during training to reach the performance. Fig. 3 plots the test performance of `Lag-Llama` as a function of the number of training windows seen from the corpus of datasets. `Lag-Llama` achieves better zero-shot performance on both the unseen test sets as the number of windows seen from the training corpus of datasets increases. Further, in this case, we do not fix the size of the training dataset and instead let early stopping decide when to stop training. Instead, as in the literature of scaling laws [21, 7, 14], we further plan to fix the dataset size and understand the scaling properties of the model as a function of the dataset size when the entire dataset is exhausted during training. This will require careful reconsideration of our choices made for early stopping, model selection, and data sampling.

6.3 Performance of the model with respect to model size

We train a series of models of various sizes, with hyperparameters sampled from the grid given in Table 2. The total number of parameters of the trained models ranges from 10^3 to 3×10^7. Training such models allows us to test the performance of the model at various scales. Fig. 4 shows such a plot of `Lag-Llama` on the `traffic` dataset. `Lag-Llama` outperforms the baseline with no training on the `Traffic` dataset, after around 10^6 parameters, and achieves stable zero-shot performance with multiple possible configurations of hyperparameters between 10^6 and 10^7. We perform the same analysis on the `M4-weekly` dataset and for the `Lag-Transformer` model in App. Sec. A.3.

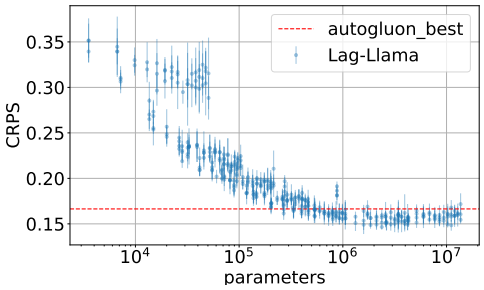

Figure 4: *Zero-shot CRPS of* `Lag-Llama` *as a function of parameter count on the* `Traffic` *dataset. The figure plots the average and standard error of 6 seeds. The red line represents the performance of the best supervised baseline trained solely on the* `Traffic` *dataset.*

6.4 Scaling Laws

We fit scaling laws to the zero-shot CRPS of `Lag-Llama` and `Lag-Transformer` as a function of model size. Fitting such scaling laws potentially allows us to extrapolate further with the obtained mathematical power law. The details of the scaling laws obtained and the extrapolations predicted can be found in App. Sec. A.4.

7 Discussion

In this work, we propose Lag-Llama, a step towards foundation models for probabilistic time-series forecasting. Lag-Llama's zero-shot performance beats or compares favorably to supervised baselines, and as the model size increases, its performance improves and stabilizes across hyperparameter specifications. We further fit smoothly broken power laws [7] that model's test performance with respect to its size.

Going beyond these preliminary experiments, we plan to first ablate various architectural design and model selection choices. We will assess the model's zero-shot performance across other datasets in a leave-one-out fashion, to obtain zero-shot performance results across a spectrum of datasets that would allow for interesting analyses. We also plan to fine-tune the pretrained models on the training splits of the downstream datasets and assess the few-shot and many-shot finetuning performance of our model across datasets. Finally, we plan to scale both the model size and the amounts of diverse time-series training data, while comparing scaling laws of this and other candidate architectures for time-series foundation models.

Acknowledgements

We are grateful to Viatcheslav Gurev, Mohammad Javad Darvishi Bayazi, Valentina Zantedeschi, Étienne Marcotte and Roland Riachi for useful discussions during the course of the project. We acknowledge and thank the authors and contributors of all the open-source libraries that were used in this work, especially: GluonTS [1], NumPy [17], Pandas [42], Matplotlib [18] and PyTorch [31].

We acknowledge the support from the Canada CIFAR AI Chair Program and from the Canada Excellence Research Chairs (CERC) Program. This research was made possible thanks to the computing resources on the Summit supercomputer, provided as a part of the INCITE program award "Scalable Foundation Models for Transferable Generalist AI". These resources were provided by the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Alexander Alexandrov et al. "GluonTS: Probabilistic and Neural Time Series Modeling in Python". In: *Journal of Machine Learning Research* 21.116 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/19-820.html.

[2] Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Nicolas Chapados, and Alexandre Drouin. *TACTiS-2: Better, Faster, Simpler Attentional Copulas for Multivariate Time Series*. 2023. arXiv: 2310.01327 [cs.LG].

[3] Yasaman Bafhri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. "Explaining Neural Scaling Laws". In: *ArXiv* abs/2102.06701 (2021). URL: https://www.arxiv.org/abs/2102.06701.

[4] Konstantinos Benidis et al. "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey". In: *ACM Computing Surveys* 55.6 (Dec. 2022), pp. 1–36. DOI: 10.1145/3533382. URL: https://doi.org/10.1145%2F3533382.

[5] Rishi Bommasani et al. *On the Opportunities and Risks of Foundation Models*. 2022. arXiv: 2108.07258 [cs.LG].

[6] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. "Broken Neural Scaling Laws". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://arxiv.org/abs/2210.14891.

[8] Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. *LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs*. 2023. arXiv: 2308.08469 [cs.LG].

[9] Muxi Chen, Zhijian Xu, Ailing Zeng, and Qiang Xu. *FrAug: Frequency Domain Augmentation for Time Series Forecasting*. 2023. URL: https://openreview.net/forum?id=j83rZLZgYBv.

[10] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL].

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[12] Alexandre Drouin, Étienne Marcotte, and Nicolas Chapados. "TACTiS: Transformer-Attentional Copulas for Time Series". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 5447–5493. URL: https://proceedings.mlr.press/v162/drouin22a.html.

[13] Azul Garza and Max Mergenthaler-Canseco. *TimeGPT-1*. 2023. arXiv: 2310.03589 [cs.LG].

[14] Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. "Scaling Laws for Neural Machine Translation". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=hR_SMu8cxCV.

[15] Tilmann Gneiting and Adrian E Raftery. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (2007), pp. 359–378. DOI: 10.1198/016214506000001437. URL: https://doi.org/10.1198/016214506000001437.

[16] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. "Monash Time Series Forecasting Archive". In: *Neural Information Processing Systems Track on Datasets and Benchmarks*. 2021.

[17] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[18] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[19] R.J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and practice*. OTexts, 2021. ISBN: 978-0987507136.

[20] Ming Jin et al. *Time-LLM: Time Series Forecasting by Reprogramming Large Language Models*. 2023. arXiv: 2310.01728 [cs.LG].

[21] Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. "Scaling Laws for Neural Language Models". In: *ArXiv* abs/2001.08361 (2020). URL: https://www.arxiv.org/abs/2001.08361.

[22] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. "Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=cGDAkQo1C0p.

[23] Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, and Tim Januschowski. "Deep Learning Based Forecasting: A Case Study from the Online Fashion Industry". In: *Forecasting with Artificial Intelligence: Theory and Applications*. Ed. by Mohsen Hamoudia, Spyros Makridakis, and Evangelos Spiliotis. Cham: Springer Nature Switzerland, 2023, pp. 279–311. ISBN: 978-3-031-35879-1. DOI: 10.1007/978-3-031-35879-1_11. URL: https://doi.org/10.1007/978-3-031-35879-1_11.

[24] Zhe Li, Pengyun Wang, Zhongwen Rao, Lujia Pan, and Zenglin Xu. *Ti-MAE: Self-Supervised Masked Time Series Autoencoders*. 2023. URL: https://openreview.net/forum?id=9AuIMiZhkL2.

[25] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[26] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. "Non-Stationary Transformers: Exploring the Stationarity in Time Series Forecasting". In: *Advances in Neural Information*

Processing Systems. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 9881–9893. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/4054556fcaa934b0bf76da52cf4f92cb-Paper-Conference.pdf.

[27] Alexander Maloney, Daniel A. Roberts, and James Sully. "A Solvable Model of Neural Scaling Laws". In: *ArXiv* abs/2210.16859 (2022). URL: https://www.arxiv.org/abs/2210.16859.

[28] James E. Matheson and Robert L. Winkler. "Scoring Rules for Continuous Probability Distributions". In: *Management Science* 22.10 (1976), pp. 1087–1096.

[29] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *International Conference on Learning Representations*. 2023.

[30] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[31] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8026–8037.

[32] Martin Peterson. *An Introduction to Decision Theory*. Second. Cambridge Introductions to Philosophy. Cambridge University Press, 2017. DOI: 10.1017/9781316585061.

[33] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[34] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M Bergmann, and Roland Vollgraf. "Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=WiGQBFuVRv.

[35] Yilun Du et al Robert Verkuil Ori Kabeli. *Language models generalize beyond natural proteins*. 2022. https://doi.org/10.1101/2022.12.21.521521: https://doi.org/10.1101/2022.12.21.521521 (cs.CV).

[36] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. "High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 6827–6837.

[37] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks". In: *International Journal of Forecasting* (2019). ISSN: 0169-2070. URL: http://www.sciencedirect.com/science/article/pii/S0169207019301888.

[38] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191.

[39] Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. "AutoGluon–TimeSeries: AutoML for Probabilistic Time Series Forecasting". In: *AutoML Conference 2023 (ABCD Track)*. 2023. URL: https://openreview.net/forum?id=XHIY3cQ8Tew.

[40] Student. "The probable error of a mean". In: *Biometrika* (1908), pp. 1–25.

[41] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2021. arXiv: 2104.09864 [cs.CL].

[42] The Pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[43] Sebastian Thrun and Lorien Pratt. "Learning to Learn: Introduction and Overview". In: *Learning to Learn*. USA: Kluwer Academic Publishers, 1998, pp. 3–17. ISBN: 0792380479.

[44] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. "Deep learning for time series forecasting: a survey". In: *Big Data* 9.1 (2021), pp. 3–21.

[45] Hugo Touvron et al. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971* (2023).

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R.

Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[47] Wenhui Wang et al. *Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks*. 2022. arXiv: 2208.10442 [cs.CV].

[48] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. "CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=PilZY3omXV2.

[49] Chin-Chia Michael Yeh et al. *Toward a Foundation Model for Time Series Data*. 2023. arXiv: 2310.03916 [cs.LG].

[50] Biao Zhang and Rico Sennrich. "Root Mean Square Layer Normalization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

[51] Tian Zhou, PeiSong Niu, Xue Wang, Liang Sun, and Rong Jin. *One Fits All:Power General Time Series Analysis by Pretrained LM*. 2023. arXiv: 2302.11939 [cs.LG].

A Appendix

A.1 Hyperparameter Choices

Table 2: *Hyperparameter choices for a random grid search. The dimension of the feedforward layers* `n_hidden` *is derived from* `n_embed` *as described in (App. Sec. A.2).*

Hyperparameter	**Symbol**	`Lag-Llama`	`Lag-Transformer`
Layers	M	[1-16]	[2-16]
Decoder layers	M_{dec}	[1-16]	[1-8]
Encoder layers	M_{enc}	0	[1-8]
Heads	`n_head`	[1, 2, 4, 8]	[1, 2, 4, 8]
Latent dimension	`n_embed`	[256, 512, 768]	[256, 512, 768]
Feedforward dimension	`n_hidden`	[256, 512, 768]	[256, 512, 768]

Table 3: *Fixed hyperparameter values used for experiments in Sec. 5.*

Hyperparameter	**Symbol**	**Values**
Context length	C	256
Augmentation probability	`aug_prob`	0.5
Augmentation rate	`aug_rate`	0.1
Batch size	B	100
Batches per epoch	B_e	100
Early stopping patience		50
Limit of validation batches		10
Max training epochs		1000
Lag sequence indices	$\mathcal{L}$	[Q, M, W, D, T, M, S]
Size of $\mathbf{c}_t$	$\lvert\mathcal{L}\rvert, \mu, \sigma$	$84 + 2$
Learning rate	α	10^{-3}
Weight decay	λ	10^{-8}
Dropout		0

Table 2 describes the hyperparameter choices of `Lag-Llama` and `Lag-Transformer`.

Table 3 describes the model hyperparameters and other parameters that are used during training, for all experiments described in Sec. 5. The lag-sequence indices are those indices obtained using the lags at several granularity levels - quarterly, monthly, weekly, daily, minute-level, and second-level. For reference, we give below the `GluonTS` [1] code used to obtain these lags:

Listing 1: *Lags used as in the code based on* `GluonTS`

```
from gluonts.time_feature import get_lags_for_frequency


lags = sorted(
    list(
        set(
            get_lags_for_frequency(freq_str="Q", num_default_lags=1)
            + get_lags_for_frequency(freq_str="M", num_default_lags=1)
            + get_lags_for_frequency(freq_str="W", num_default_lags=1)
            + get_lags_for_frequency(freq_str="D", num_default_lags=1)
            + get_lags_for_frequency(freq_str="H", num_default_lags=1)
            + get_lags_for_frequency(freq_str="T", num_default_lags=1)
            + get_lags_for_frequency(freq_str="S", num_default_lags=1)
        )
    )
)
```

A.2 Dimension of the Feedforward Layer

To ensure comparability, the dimension of the feedforward layers in both architectures is derived as the largest multiple of 256 less than or equal to $\frac{8}{3}$`n_embed` as in Llama [45]:

$$\texttt{n\_hidden} = 256 \times \left\lfloor \frac{8 \times \texttt{n\_embed}}{3 \times 256} \right\rfloor \tag{3}$$

Practically, the possible dimensions of the feedforward layers are $[256, 512, 768]$

A.3 Parameter Scaling Plots on all Architectures and Datasets

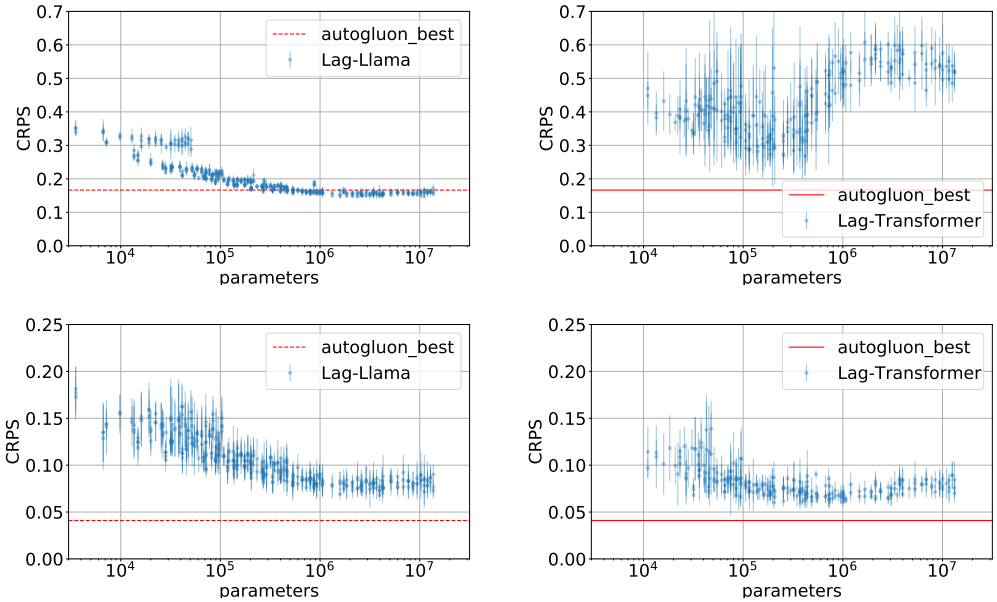

Figure 5: *`Lag-Llama` (left) and `Lag-Transformer` (right) performance as a function of parameter count on the unseen datasets `Traffic` (top) and `M4 Weekly` (bottom). Error bars are from the 6 different runs.*

Additional plots on the performance of the model as the model size is scaled can be found in Fig. 5.

A.4 Fitting Scaling Laws

As detailed in [7], the functional form of a broken neural scaling law is

$$y = a + \left(bx^{-c_0}\right) \prod_{i=1}^{n} \left(1 + \left(\frac{x}{d_i}\right)^{1/f_i}\right)^{-c_i * f_i}$$

To avoid choosing the number of breaks n manually, we fit the functional form to the leftmost 85% of the data (black points) using nonlinear least squares for multiple values of n. We then select the value of n that minimizes RMSE on the subsequent 5% of the data, meaning that the rightmost 10% of the data, i.e. the largest model sizes (green points), are never used to estimate nor validate the fit of the scaling law (red line). The scaling laws are plotted for each architecture and dataset in Fig. 6.

For example, the functional form of `Lag-Llama` on `Traffic`, which has two breaks, is:

$$y = 0.351 + \left((3 \times 10^{-16} x^{4.75}\right) \left(1 + \left(\frac{x}{1389.77}\right)^{1/1.62}\right)^{-2.36*1.62} \left(1 + \left(\frac{x}{1418.26}\right)^{1/1.61}\right)^{-2.44*1.61}$$

A.5 Dataset details

The details of all datasets used during training are given in Table 4.

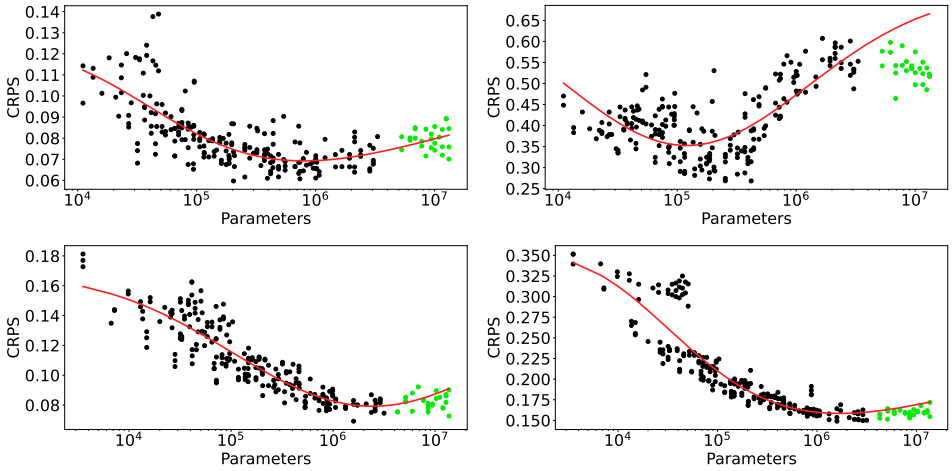

Figure 6: *Two breaks Neural Scaling laws fit for* `Lag-Llama` *(left) and* `Lag-Transformer` *(right), on* `traffic` *(top) and* `m4-weekly` *(bottom) respectively. Green points are the held-out data for the fit.*

Table 4: *Properties of the datasets used in the experiments, in particular the domain of time series, the total number of time series, the temporal frequency of the time series, the minimal length of the time series, and the maximum length. We use all the datasets apart from* `M4 Weekly` *and* `Traffic` *for training.*

Name	Domain	No. of series	Freq.	Min. len.	Max. len.
Air Passengers	Transport	1	1M	108	108
Aus. Elec. Demand	Energy	5	30T	230,736	232,272
Car Parts	Sales	2674	1M	51	51
CIF	Finance	72	1M	34	120
Covid	Health	266	1D	212	212
Electricity	Energy	321	1H	26,304	26,304
Exchange Rate	Finance	8	1B	6,071	6,071
Fred-MD	Finance	107	1M	728	728
Hospital	Health	767	1M	84	84
Kaggle Web Traffic	Web	145,063	1D	803	803
KDD Cup	Nature	270	1H	9,504	10,920
London Smart Meters	Energy	5,560	30T	288	39,648
NN5	Finance	111	1D	791	791
Pedestrian Count	Transport	66	1H	576	96,424
Ride Share	Transport	2,304	1H	541	541
Saugeen River	Nature	1	1D	23,741	23,741
Solar	Energy	137	10T	52,560	52,560
Taxi	Transpose	1,214	30T	1,488	1,488
Temperature	Nature	32,072	1D	725	725
Tourism	Transpose	1,311	1M	11	333
Uber	Transpose	262	1H	2,602	4,320
Vehicle Trips	Transpose	329	1D	70	243
Weather	Nature	3,010	1D	1,332	65,981
Wikipedia	Web	9,535	1D	762	762
M4 Hourly	Diverse	414	1H	700	960
M4 Daily	Diverse	4,227	1D	93	9,919
M4 Monthly	Diverse	48,000	1M	42	2,794
M4 Quarterly	Diverse	24,000	1Q	16	866
M4 Yearly	Diverse	23,000	1Y	13	278
Wind Farms	Energy	339	1T	6,345	527,040
M4 Weekly	Diverse	359	1W	80	2,597
Traffic	Transport	862	1H	17,544	17,544