Linear Regression as a Litmus Test for Time Series Forecasting Benchmarks

Walid Bouainouche
Huawei Noah's Ark Lab
Paris, France

Gabriel Singer
Huawei Noah's Ark Lab
Paris, France

Vasilii Feofanov
Huawei Noah's Ark Lab
Paris, France

Ievgen Redko
Huawei Noah's Ark Lab
Paris, France

Abstract

Recent work on foundation models for time series forecasting has been accompanied by benchmarks such as GIFT-Eval, which aim to standardize comparison and establish leaderboards. These studies typically include simple baselines such as Seasonal Naïve or DLinear, establishing a low bar that new foundation models are expected to surpass. However, we show that this bar can be substantially raised: with careful tuning, a vanilla linear regression model achieves surprisingly strong performance, outperforming many deep learning methods (e.g., iTransformer) and even popular foundation models such as Chronos Base. This finding highlights the need to recalibrate evaluation practices in time series forecasting, both by adopting stronger baselines that meaningfully challenge foundation models and by incorporating more diverse, non-linear datasets. We argue that linear regression can serve as a litmus test for benchmark design, revealing that current evaluation practices may obscure progress in foundation model forecasting.

1 Introduction

Time series forecasting has long been recognized as a challenging problem, owing to the fundamental differences between time series data and other modalities such as text or images. Time series are highly heterogeneous across domains, including energy, retail, finance, mobility, and weather, while also exhibiting nonstationarity, variable sampling rates, multiple seasonalities, and diverse forecasting horizons. These characteristics make the design and evaluation of general-purpose forecasting models particularly complex.

In recent years, the field has witnessed a surge of interest in large-scale deep learning architectures, particularly foundation models (FMs) that promise strong performance across domains without task-specific tuning. While these models have quickly gained traction, their empirical improvements over prior methods have often been modest, and their evaluation practices have not always been rigorous. In particular, comparisons are frequently made against very simple baselines such as Seasonal Naïve or DLinear [17] that may not adequately capture the difficulty of the task. As a result, it remains unclear whether foundation models truly provide consistent and meaningful advances in time series forecasting.

In this paper, we revisit an overlooked yet surprisingly competitive baseline (further called Linear++): a single linear layer, coupled with RevIN normalization [9] and a context length selected by validation. Despite its simplicity, this model achieves strong results in both deterministic and probabilistic forecasting settings. Our study builds on the spirit of Zeng et al. [17], who introduced DLinear and demonstrated that linear models could outperform transformer architectures at the time.

We take this line of inquiry further by (a) slightly refining the linear regression formulation, (b) comparing it against modern architectures and recent foundation models on two benchmarks, including GIFT-Eval [1], and (c) exploring its capabilities in the probabilistic and deterministic setting.

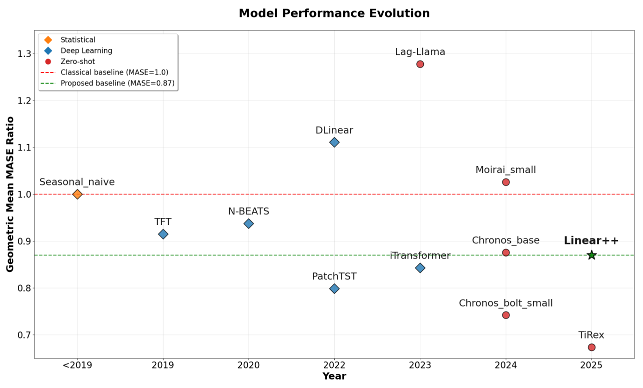

Figure 1: Performance of forecasting models sorted by year.

Our results reveal that a carefully tuned linear regression model can rival, and in some cases outperform, state-of-the-art deep learning methods such as iTransformer [12] as well as foundation models with millions or even hundreds of millions of parameters, such as Chronos Base [3]. This finding suggests that evaluation practices in time series forecasting need to be recalibrated, considering stronger baselines to ensure that new methods surpass more than trivial models, and incorporating more complex datasets where sophisticated architectures truly add value. We argue that linear regression can serve as a litmus test for benchmark design, helping to clarify whether progress in foundation model forecasting is genuine or simply a byproduct of insufficient evaluation.

2 Methodology

In this section, we introduce the framework and present our proposed baseline.

2.1 Problem Setup

Given a D-dimensional time series of length L (*context size*) $\mathbf{X} \in \mathbb{R}^{D \times L}$, the goal of time series forecasting is to predict next H values (*prediction horizon*), denoted by $\mathbf{Y} \in \mathbb{R}^{D \times H}$. We assume that we have access to a training set that consists of N observations $(\mathcal{X}, \mathcal{Y}) = (\{\mathbf{X}^{(i)}\}_{i=0}^{N}, \{\mathbf{Y}^{(i)}\}_{i=0}^{N})$.

We aim to train a predictor $f_{\boldsymbol{\omega}} : \mathbb{R}^{D \times L} \to \mathbb{R}^{D \times H}$ (deterministic) or $f_{\boldsymbol{\omega}} : \mathbb{R}^{D \times L} \to \mathcal{P}(\mathbb{R}^{D \times H})$ (probabilistic) parameterized by $\boldsymbol{\omega}$.

Deterministic setting: The model outputs point predictions and is trained by minimizing the Mean Squared Error (MSE):

$$\mathcal{L}_{\text{det}}(\boldsymbol{\omega}) = \frac{1}{ND} \sum_{i=0}^{N} \|\mathbf{Y}^{(i)} - f_{\boldsymbol{\omega}}(\mathbf{X}^{(i)})\|_{\text{F}}^{2}. \tag{1}$$

The evaluation metric is the standard mean squared error (MSE).

Probabilistic setting: The model outputs a predictive distribution and is trained by minimizing negative log-likelihood:

$$\mathcal{L}_{\text{prob}}(\boldsymbol{\omega}) = -\frac{1}{N} \sum_{i=1}^{N} \log P(\mathbf{Y}^{(i)} \mid \mathbf{X}^{(i)}; \boldsymbol{\omega}). \tag{2}$$

We forecast the entire horizon $y_{t+1:t+H}$ under the assumption that each future value follows an independent Student-*t* distribution. For every step $h \in \{1, \ldots, H\}$ the network therefore outputs the three parameters that characterise this density μ_{t+h}, scale $\sigma_{t+h} > 0$, and degrees-of-freedom $\nu_{t+h} > 2$ and is trained by minimising the sum of Student-*t* negative log-likelihoods over the horizon. The Student-*t* likelihood and its parameterisation are taken from GLUONTS [2].

In practice, to evaluate the accuracy of predicted cumulative distribution functions, the Continuous Ranked Probability Score (CRPS) [6] is typically employed.

2.2 Proposed Method

Our proposed baseline, Linear++, is frustratingly easy. We employ a linear layer $\mathbf{W} : \mathbb{R}^L \to \mathbb{R}^H$ that is shared between all the channels. Before and after the linear layer we apply instance normalization and de-normalization (RevIN), respectively, which improve stability and align all the channels to same units. Inspired by Xu et al. [16] and recent practices on GIFT-Eval, we tune the context size L (using a grid search with values proportional to the horizon) based on a hold-out validation set, which helps the model to find the right periodicity (if any) for each dataset. We also tune the learning rate and the weight decay, though the linear model is not very sensitive to these hyperparameters. We provide more details in Appendix.

3 Experiments

In this section, we present our experimental results when comparing Linear++ with various supervised and foundation models on two benchmarks.

3.1 Experimental Setup

We evaluate Linear++ on two benchmarks used in literature. First, Gift-Eval [1] consists of 24 datasets tested under various conditions. These datasets span short-, medium-, and long-term horizons, as well as multiple frequencies, resulting in a total of 97 evaluation settings. The evaluation adheres to the benchmark protocols, using the mean absolute scaled error (MASE) and Continuous Ranked Probability Score (CRPS). In practice, CRPS is approximated by the mean weighted quantile loss (WQL) over nine quantiles ranging from 0.1 to 0.9 in increments of 0.1. To compute aggregated performance, each evaluation score is first normalized against a seasonal naive baseline, and then combined using the geometric mean across all settings. Our second benchmark is the one used in long-term mutlivariate forecasting literature with 6 datasets [7]: ETTh1, ETTh2, ETTm1 ETTm2, Exchange Rate, Weather. In this case, the evaluation is deterministic only, and MSE is used for evaluation.

3.2 GIFT-Eval Benchmark

The results show that Linear++ achieves strong performance across diverse time series datasets, outperforming many supervised models (iTransformer [12], TFT [10], N-BEATS [14]) and even foundation models (Moirai Small/Large [11], Chronos Small/Base [3], YingLong 6M [15]). A relative comparison with all methods on all datasets from GIFT-Eval can be found in Appendix B.1. Linear++ also significantly surpasses DLinear, which is probably connected to the hyperparameter tuning and integration of RevIN. In their paper, they showed that the proposed trend/seasonality decomposition improves the performance, so tuning properly DLinear will probably boost its performance. It is important to mention that Linear++ ignores the multivariate nature of data as it learns one layer for all channels. From the multi-task learning perspective [8], it means that we assume that all channels collaborate without needing to introduce individual channel biases. A quite high performance of Linear++ can be explained by RevIN, but may be also due to the lack of diversity of multivariate problems.

In addition, our comparison of methods reveal that there are datasets where almost all models are worse than Linear++ in terms of performance. For example, this is the case for bizitobs l2c/5T (Figure 8), a highly periodic dataset close to linear forecasting. On the other hands, ett2/W (Figure [9]) is the dataset which contains different subpopulations, so it is more suitable for foundation models, while a linear model is less appropriate in this context.

3.3 Deterministic Long-term Forecasting Benchmark

In this benchmark, we compare Linear++ with such methods as SAMformer, [7], PatchTST [13], iTransformer [12], TSMixer [5] and DLinear [17]. The performance results are displayed in Table 1.

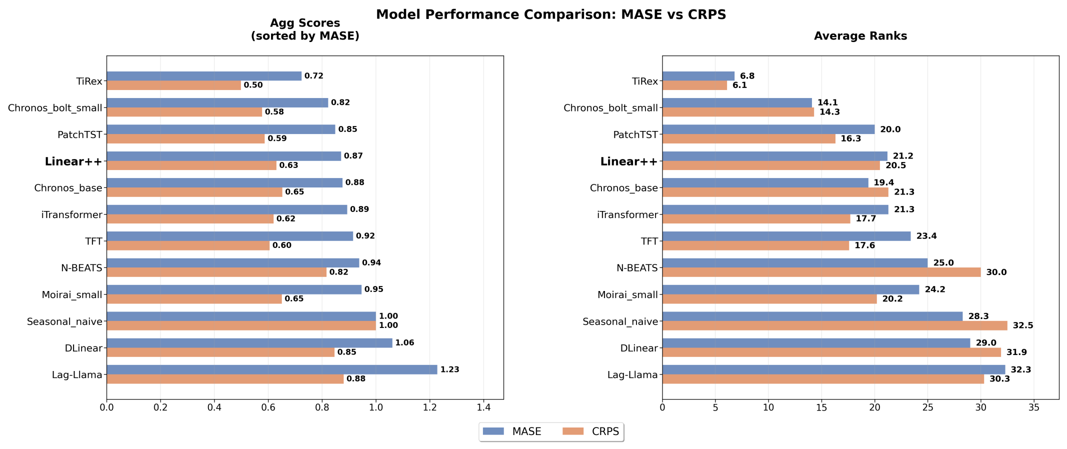

Figure 2: Performance results on GIFT-Eval benchmark.

Table 1: The test MSE averaged over different prediction horizons $H \in \{96, 192, 336, 720\}$. We extract results from Ilbert et al. [7] and the full table can be found in Appendix B.2.

Dataset	Linear++	SAMformer	TSMixer	DLinear	iTransformer	PatchTST
ETTh1	<u>0.413</u>	**0.410**	0.439	0.423	0.454	0.469
ETTh2	0.379	**0.344**	<u>0.357</u>	0.431	0.383	0.387
ETTm1	0.387	<u>0.373</u>	0.385	**0.357**	0.407	0.387
ETTm2	**0.185**	0.269	0.289	<u>0.267</u>	0.288	0.281
Exchange	0.385	0.445	0.593	**0.296**	<u>0.360</u>	0.366
Weather	**0.239**	0.261	0.267	<u>0.246</u>	0.258	0.259

We can see that Linear++ is again a strong baseline while being the most lightweight model. Although both PatchTST and SAMformer made attempts to improve the transformer architecture, the advantage of transformers remains unclear for this benchmark.

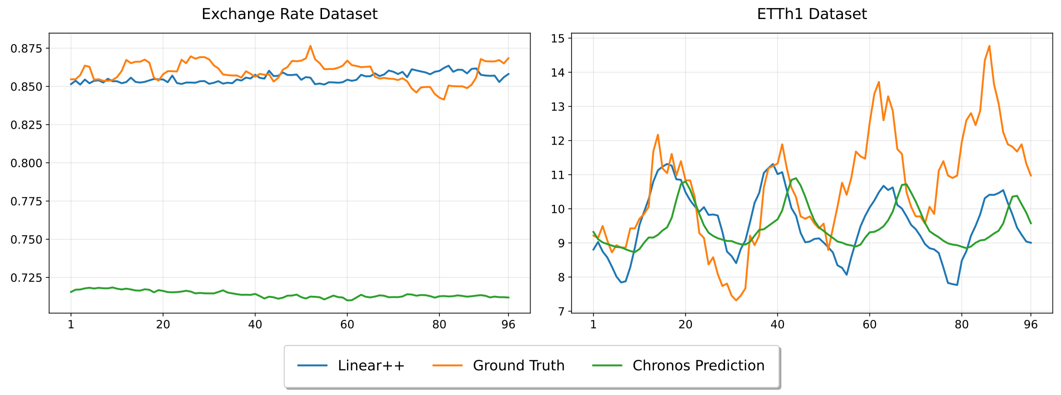

Figure 3: Forecasting examples on ETTh1 and Exchange Rate for horizon $H = 96$.

In Figure 3, we visualize some examples of predictions and true responses from ETTh1 and Exchange Rate datasets to compare forecasting quality of Linear++ and Chronos Base. We can see that for clearly periodic data (ETTh1), Linear++ estimated the periodicity very well, though failing to anticipate the change of the trend; meanwhile, Chronos has a phase shift problem. When the data is complex and non-periodic as Exchange Rate, a linear model is not able to capture any intrinsic patterns, predicting roughly the mean value, which interestingly is a very strong baseline, and we can see that Chronos fails to do so.

4 Conclusion and Future Work

We raised concerns about the current evaluation setup in time series forecasting. While foundation models enable zero-shot forecasting, we argue they should at least surpass strong linear baselines. Our results show that a carefully tuned linear regression model remains highly competitive, suggesting that existing benchmarks set the bar too low. Although recent models such as TiRex [4] report stronger performance, issues like data leakage and benchmark overfitting highlight the need for caution. We call for fairer evaluation practices, including stronger baselines and more complex datasets, to ensure that progress in foundation model forecasting is both genuine and meaningful.

References

[1] T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

[2] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024. URL https://arxiv.org/abs/2403.07815.

[4] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning, 2025. URL https://arxiv.org/abs/2505.23719.

[5] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting, 2023. URL https://arxiv.org/abs/2303.06053.

[6] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

[7] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. Samformer: Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention, 2024. URL https://arxiv.org/abs/2402.10198.

[8] Romain Ilbert, Malik Tiomoko, Cosme Louart, Ambroise Odonnat, Vasilii Feofanov, Themis Palpanas, and Ievgen Redko. Analysing multi-task regression via random matrix theory with application to time series forecasting. *Advances in Neural Information Processing Systems*, 37: 115021–115057, 2024.

[9] 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*, 2021. URL https://openreview.net/forum?id=cGDAkQo1C0p.

[10] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020. URL https://arxiv.org/abs/1912.09363.

[11] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts, 2024. URL https://arxiv.org/abs/2410.10469.

[12] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[13] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023. URL `https://arxiv.org/abs/2211.14730`.

[14] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2020. URL `https://arxiv.org/abs/1905.10437`.

[15] Xue Wang, Tian Zhou, Jinyang Gao, Bolin Ding, and Jingren Zhou. Output scaling: Yinglong-delayed chain of thought in a large pretrained time series forecasting model, 2025. URL `https://arxiv.org/abs/2506.11029`.

[16] Zongzhe Xu, Ritvik Gupta, Wenduo Cheng, Alexander Shen, Junhong Shen, Ameet Talwalkar, and Mikhail Khodak. Specialized foundation models struggle to beat supervised baselines. *arXiv preprint arXiv:2411.02796*, 2024.

[17] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

A Appendix

A.1 Linear++

Given an input time series $\mathbf{X} \in \mathbb{R}^{D\times L}$, the predictor $f_{\boldsymbol{\omega}}$ is defined as:

$$f_{\boldsymbol{\omega}}(\mathbf{X}) = g_1\left(\mathbf{W} \cdot g_0(\mathbf{X})\right),$$

where:

- $g_0 : \mathbb{R}^{D\times L} \to \mathbb{R}^{D\times L}$ is the RevIN normalization transformation,
- $g_1 : \mathbb{R}^{D\times H} \to \mathbb{R}^{D\times H}$ is the inverse RevIN denormalization,
- $\mathbf{W} \in \mathbb{R}^{H\times L}$ is the weight matrix of the linear layer (shared across all channels).

A.2 Probabilistic Linear++

Let $\mathbf{X} \in \mathbb{R}^{D\times L}$ be the context window and $\mathbf{Y} \in \mathbb{R}^{D\times H}$ the forecast horizon. The model directly predicts the three natural Student-*t* parameters (μ, σ, ν) simultaneously over the horizon.

Linear layer predicting parameters. The model uses a single linear layer with weights $\mathbf{W} \in \mathbb{R}^{3H\times L}$, shared by all channels, to output simultaneously all 3 parameters for the horizon:

$$\hat{Z}_k = \mathbf{W}\tilde{X}_k \in \mathbb{R}^{3H},$$

where $\hat{Z}_k$ concatenates the raw predictions for $\big(\mu_{k,1:H}, \log\sigma_{k,1:H}, \log\nu_{k,1:H}\big)$.

Predictive distribution.

$$Y_{k,t+h}|\mathbf{X} \sim \text{Student-t}(\mu_{k,h}, \sigma_{k,h}, \nu_{k,h}), \quad h = 1, \dots, H.$$

Loss function. Training minimises the negative log-likelihood of the Student-*t*, summed across all steps h and all channels k:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{D}\sum_{h=1}^{H} \log p_{\text{Student-t}}(y_{i,k,t+h}|\mu_{k,h}, \sigma_{k,h}, \nu_{k,h}).$$

De-normalization. The network outputs are de-normalized with inverse RevIN to align location and scale to the original time series units.

A.3 Probabilistic Metrics

For a predicted distribution with CDF F and ground truth value y, the CRPS is defined as:

$$\text{CRPS}(F, y) = \int_0^1 2\Lambda_\alpha(F^{-1}(\alpha), y)d\alpha,$$

where the quantile loss $\Lambda_\alpha(q, y)$ is given by: $\Lambda_\alpha(q, y) = (\alpha - \mathbb{I}\{y < q\})(y - q)$.

In practice, we approximate the CRPS using a discrete sum over quantile levels:

$$\text{CRPS} \approx \frac{1}{K}\sum_{k=1}^{K} \text{wQL}[\alpha_k],$$

where K is the number of quantile levels, and $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ are selected quantile levels (e.g., $\alpha_k = 0.1k$ for $k = 1, 2, \dots, 9$ when $K = 9$). The weighted quantile loss $\text{wQL}[\alpha]$ for quantile level α is calculated as:

$$\text{wQL}[\alpha] = 2\frac{\sum_t \Lambda_\alpha(\hat{q}_t(\alpha), y_t)}{\sum_t |y_t|},$$

where: $\hat{q}_t(\alpha)$ is the predicted α-quantile at time step t, y_t is the actual observed value at time t, $\Lambda_\alpha(\hat{q}_t(\alpha), y_t)$ is the quantile loss at time t for quantile level α.

A.4 RevIN

We adopt the formalism presentend in [7]. RevIN uses trainable affine parameters $\beta, \gamma \in \mathbb{R}^K$.

For a sample $\mathbf{X}^{(i)} \in \mathbb{R}^{K \times L}$ and feature $k \in \{1, \dots, K\}$, let's define the empirical mean and standard deviation:

$$\hat{\mu}_k^{(i)} = \frac{1}{L} \sum_{t=1}^{L} \mathbf{X}_{kt}^{(i)}, \qquad \hat{\sigma}_k^{2\,(i)} = \frac{1}{L} \sum_{t=1}^{L} \left(\mathbf{X}_{kt}^{(i)} - \hat{\mu}_k^{(i)}\right)^2. \tag{3}$$

The input sequence is normalized feature-wise to $\tilde{\mathbf{X}}^{(i)} \in \mathbb{R}^{K \times L}$:

$$\tilde{\mathbf{X}}_{kt}^{(i)} = \gamma_k, \frac{\mathbf{X}_{kt}^{(i)} - \hat{\mu}_k^{(i)}}{\sqrt{\hat{\sigma}_k^{2,(i)} + \varepsilon}} \beta_k, \tag{4}$$

with a small $\varepsilon > 0$ to avoid division by zero. The network takes $\tilde{\mathbf{X}}^{(i)}$ as input and produces $\tilde{\mathbf{Y}}^{(i)} \in \mathbb{R}^{K \times H}$.

The forecast used for evaluation is the denormalized $\hat{\mathbf{Y}}^{(i)} \in \mathbb{R}^{K \times H}$:

$$\hat{\mathbf{Y}}_{kt}^{(i)} = \sqrt{\hat{\sigma}_k^{2,(i)} + \varepsilon}; \frac{\tilde{\mathbf{Y}}_{kt}^{(i)} - \beta_k}{\gamma_k} \hat{\mu}_k^{(i)}. \tag{5}$$

As noted by [9], the tuple $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2, \boldsymbol{\beta}, \boldsymbol{\gamma})$ carries the non-stationary information of each input sequence $\mathbf{X}^{(i)}$.

B Experimental Details

B.1 GIFT-Eval Benchmark

In this Section we display the ranking of the models with geometric mean computed over datasets for MASE and CRPS. In addition to that we add 4 heatmaps that show a performance gap of forecasting methods with respect to 2 baselines, Seasonal Naive and Linear++, for two metrics, MASE and CRPS. The ratio is calculated as $\frac{\text{MASE(model)}}{\text{MASE(baseline)}} - 1$, where values below 0 (blue regions) indicate the model outperforms the baseline, and values above 0 (red regions) indicate the baseline is superior.

From Figure 4, 5, 6, 7 we can see that Linear++ significantly increases a low bar both for deterministic and probabilistic forecasting. In addition, Figure 8 and 9 illustrate some examples from 2 datasets, which are discussed in Section 3.2.

Table 2: Performance Ranking vs Seasonal Naive Baseline (sorted by MASE)

Rank	Model	MASE	CRPS
1	TiRex	0.724	0.498
2	Moirai2	0.728	0.516
3	TimeCopilot	0.741	0.508
4	Toto Open Base 1.0	0.750	0.517
5	sundial base 128m	0.750	0.559
6	TTM-R2-Finetuned	0.756	0.583
7	timesfm 2.0 500m	0.758	0.550
8	TabPFN-TS	0.771	0.544
9	TEMP-ENSEMBLE	0.788	0.462
10	YingLong 300m	0.798	0.548
11	Chronos bolt base	0.808	0.574
12	YingLong 110m	0.809	0.557
13	Chronos bolt small	0.822	0.577
14	YingLong 50m	0.822	0.567
15	PatchTST	0.849	0.587
16	VisionTS	0.863	0.755
17	Chronos large	0.870	0.647
18	**Linear++**	**0.870**	**0.630**
19	Moirai large	0.875	0.599
20	Chronos base	0.876	0.652
21	YingLong 6m	0.880	0.609
22	Chronos small	0.892	0.663
23	iTransformer	0.893	0.620
24	Moirai base	0.901	0.610
25	TFT	0.915	0.605
26	N-BEATS	0.938	0.816
27	Moirai small	0.946	0.650
28	Seasonal naive	1.000	1.000
29	TTM-R2-Pretrained	1.020	0.873
30	DLinear	1.061	0.846
31	Auto Arima	1.074	0.912
32	timesfm	1.077	0.680
33	TTM-R1-Pretrained	1.079	0.891
34	Auto Theta	1.090	1.244
35	TIDE	1.091	0.772
36	Timer	1.136	0.970
37	Auto ETS	1.212	7.489
38	Lag-Llama	1.228	0.880
39	Naive	1.270	1.591
40	DeepAR	1.343	0.853
41	Crossformer	2.574	1.637

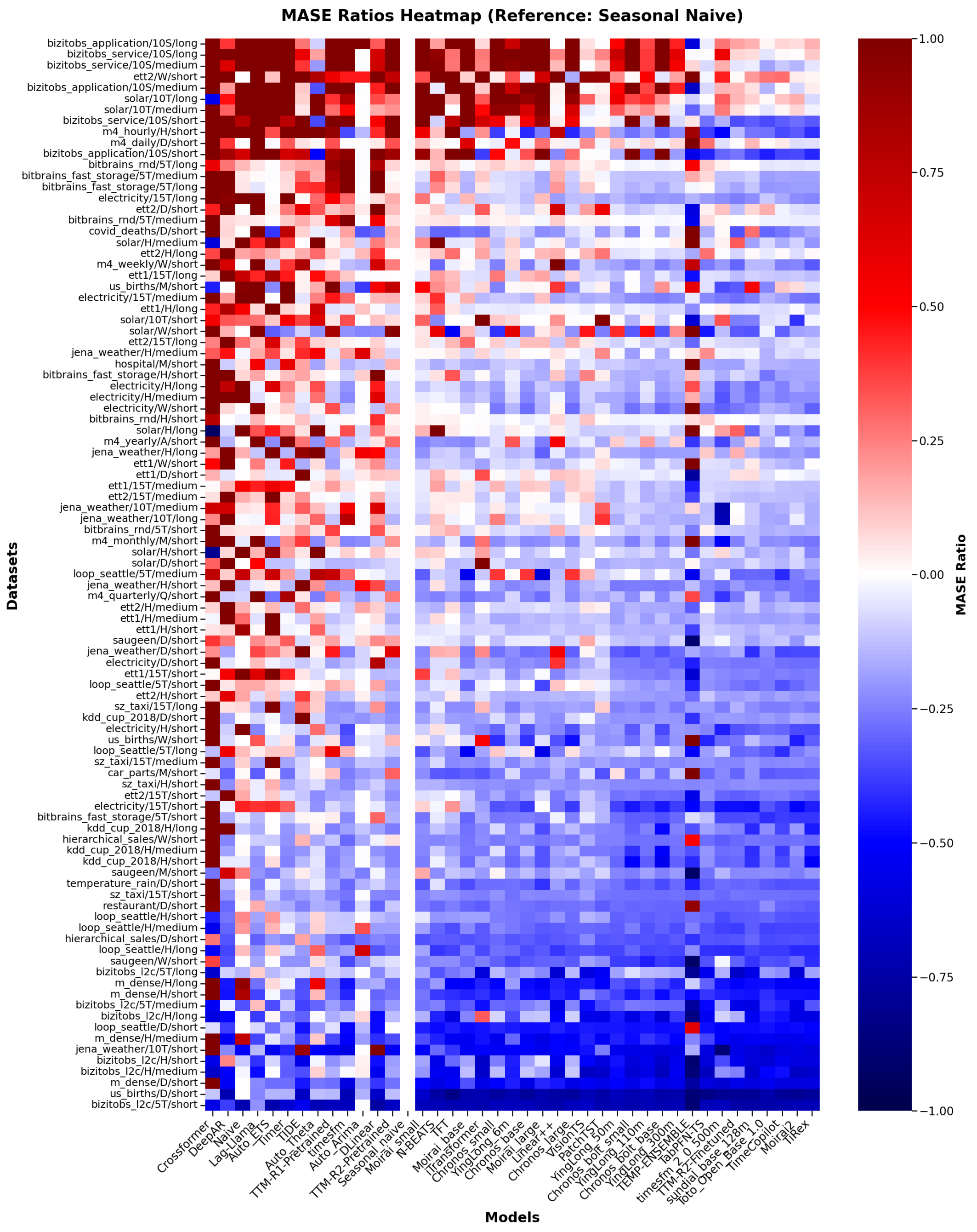

Figure 4: MASE Ratios Heatmap (Reference: Seasonal Naive)

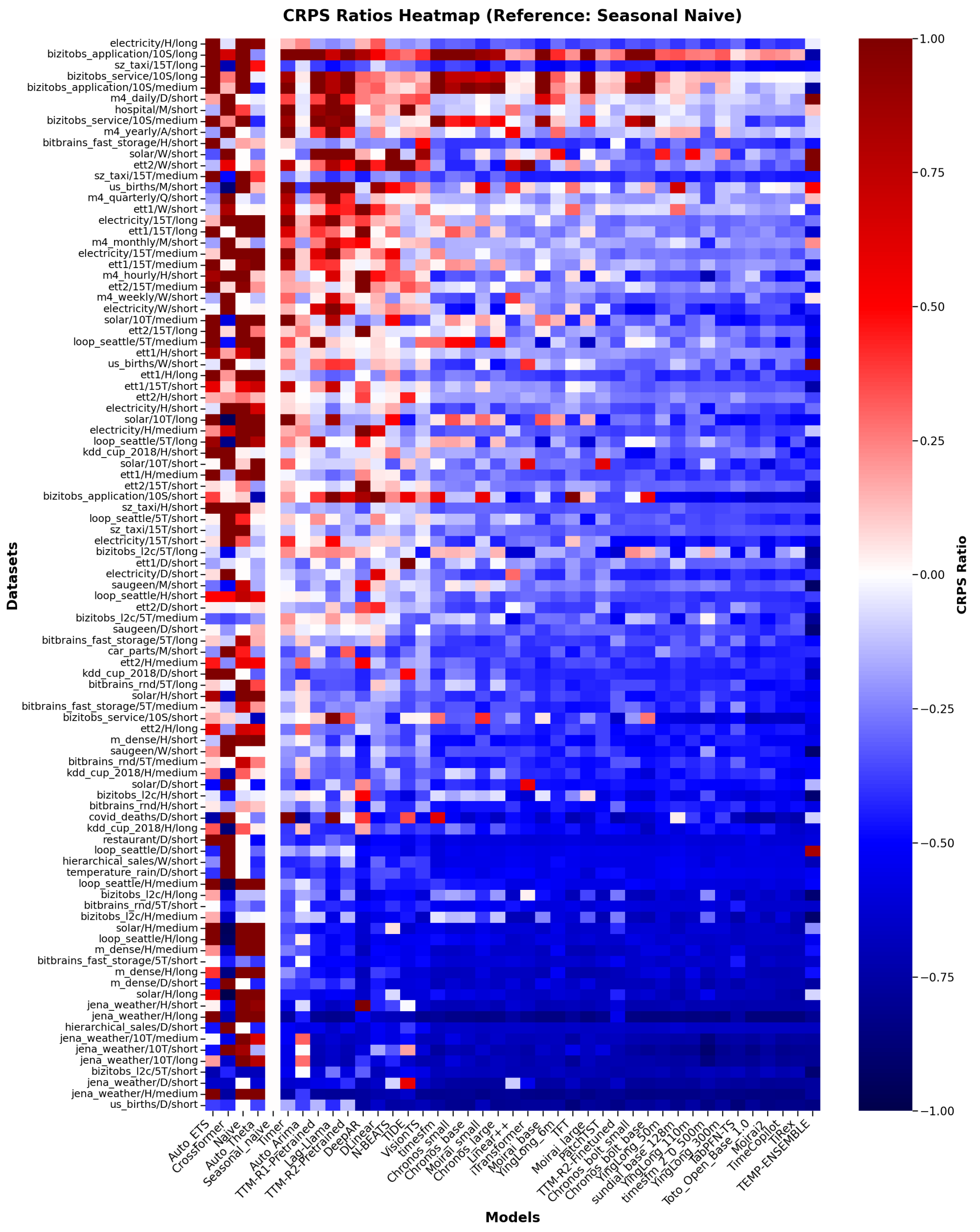

Figure 5: CRPS Ratios Heatmap (Reference: Seasonal Naive).

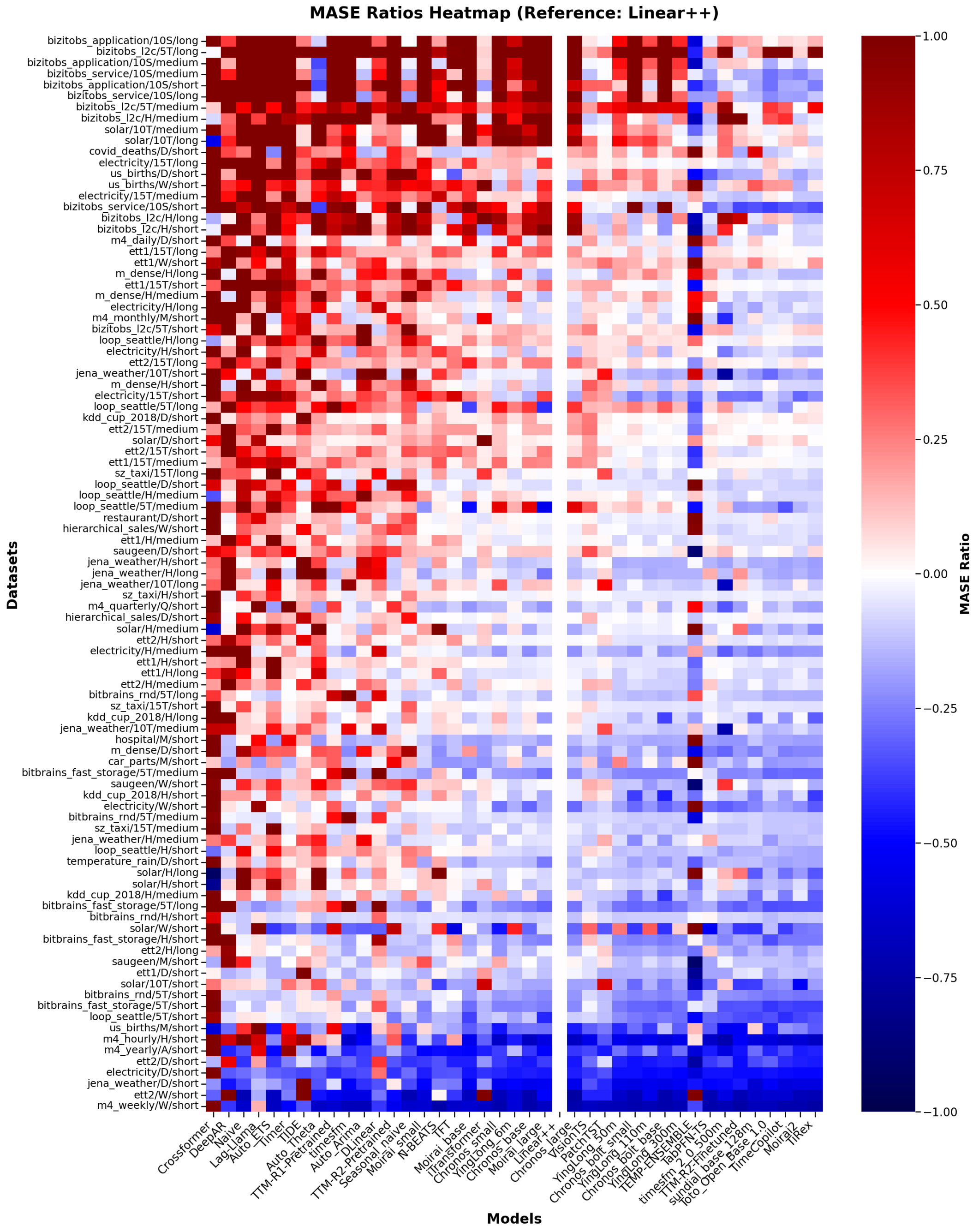

Figure 6: MASE Ratios Heatmap (Reference: Linear++).

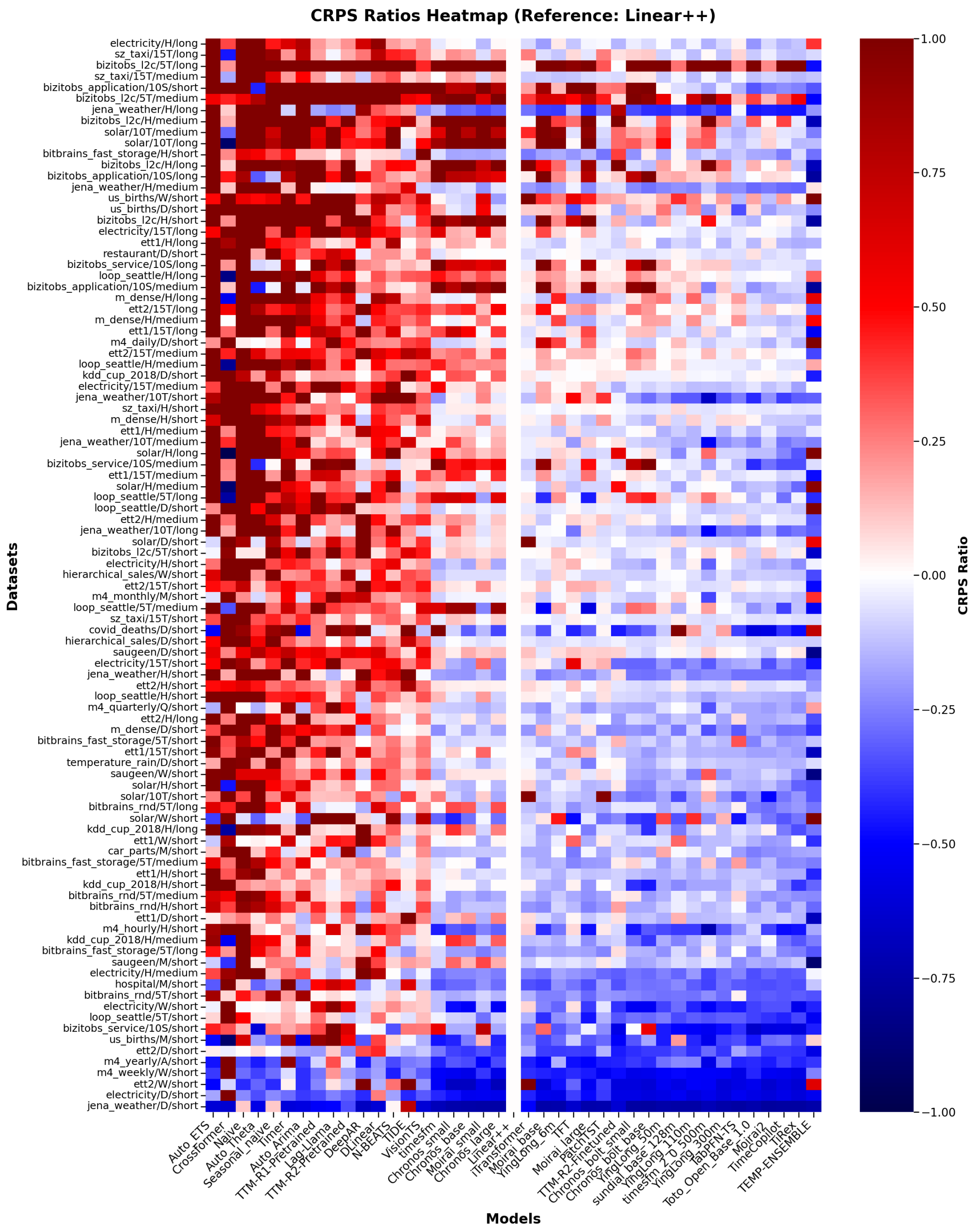

Figure 7: CRPS Ratios Heatmap (Reference: Linear++).

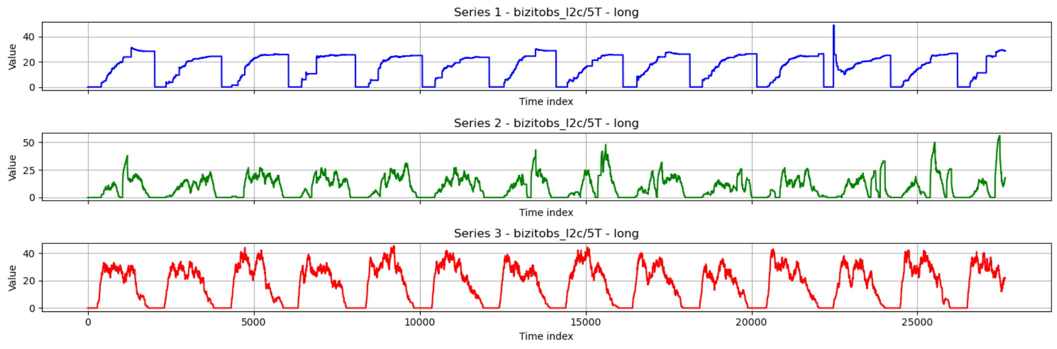

Figure 8: Example of instances for Bizitobs-l2c long term dataset.

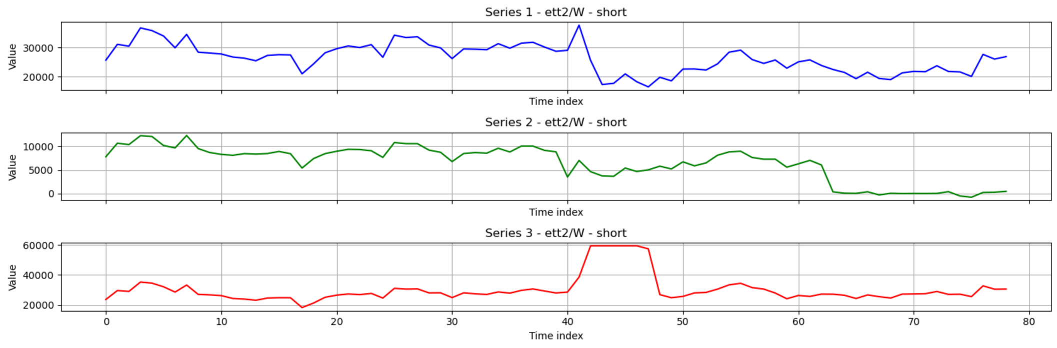

Figure 9: Example of instances for ETT2 weekly short term dataset.

B.2 Long-term Forecasting Benchmark

Table 3: Performance comparison between our model (Linear++) and baselines for multivariate long-term forecasting with different horizons H.

Dataset	H	Linear++	SAMformer	TSMixer	DLinear	iTransformer	PatchTST
ETTh1	96	**0.366**$_{\pm 0.005}$	0.381$_{\pm 0.003}$	0.398$_{\pm 0.001}$	$\underline{0.375}$	0.386	0.414
	192	**0.401**$_{\pm 0.008}$	0.409$_{\pm 0.002}$	0.426$_{\pm 0.003}$	$\underline{0.405}$	0.441	0.460
	336	$\underline{0.427}$$_{\pm 0.009}$	**0.423**$_{\pm 0.001}$	0.435$_{\pm 0.003}$	0.439	0.487	0.501
	720	$\underline{0.459}$$_{\pm 0.002}$	**0.427**$_{\pm 0.002}$	0.498$_{\pm 0.076}$	0.472	0.503	0.500
ETTh2	96	**0.279**$_{\pm 0.000}$	0.295$_{\pm 0.002}$	0.308$_{\pm 0.003}$	$\underline{0.289}$	0.297	0.302
	192	0.372$_{\pm 0.001}$	**0.340**$_{\pm 0.002}$	$\underline{0.352}$$_{\pm 0.004}$	0.383	0.380	0.388
	336	0.416$_{\pm 0.002}$	**0.350**$_{\pm 0.000}$	$\underline{0.360}$$_{\pm 0.002}$	0.448	0.428	0.426
	720	0.449$_{\pm 0.006}$	**0.391**$_{\pm 0.001}$	$\underline{0.409}$$_{\pm 0.006}$	0.605	0.427	0.431
ETTm1	96	$\underline{0.315}$$_{\pm 0.012}$	0.329$_{\pm 0.001}$	0.336$_{\pm 0.004}$	**0.299**	0.334	0.329
	192	0.380$_{\pm 0.034}$	$\underline{0.353}$$_{\pm 0.006}$	0.362$_{\pm 0.006}$	**0.335**	0.377	0.367
	336	0.405$_{\pm 0.030}$	$\underline{0.382}$$_{\pm 0.001}$	0.391$_{\pm 0.003}$	**0.369**	0.426	0.399
	720	0.449$_{\pm 0.028}$	$\underline{0.429}$$_{\pm 0.000}$	0.450$_{\pm 0.006}$	**0.425**	0.491	0.454
ETTm2	96	**0.126**$_{\pm 0.029}$	0.181$_{\pm 0.005}$	0.211$_{\pm 0.014}$	$\underline{0.167}$	0.180	0.175
	192	**0.162**$_{\pm 0.047}$	0.233$_{\pm 0.002}$	0.252$_{\pm 0.005}$	$\underline{0.224}$	0.250	0.241
	336	**0.198**$_{\pm 0.061}$	0.285$_{\pm 0.001}$	0.303$_{\pm 0.004}$	$\underline{0.281}$	0.311	0.305
	720	**0.253**$_{\pm 0.085}$	$\underline{0.375}$$_{\pm 0.001}$	0.390$_{\pm 0.003}$	0.397	0.412	0.402
Exchange	96	0.089$_{\pm 0.001}$	0.161$_{\pm 0.007}$	0.343$_{\pm 0.082}$	**0.081**	$\underline{0.086}$	0.088
	192	$\underline{0.181}$$_{\pm 0.001}$	0.246$_{\pm 0.009}$	0.342$_{\pm 0.031}$	**0.157**	0.177	0.176
	336	0.372$_{\pm 0.004}$	0.368$_{\pm 0.006}$	0.484$_{\pm 0.062}$	$\underline{0.305}$	0.331	**0.301**
	720	0.899$_{\pm 0.005}$	1.003$_{\pm 0.018}$	1.204$_{\pm 0.028}$	**0.643**	$\underline{0.847}$	0.901
Weather	96	**0.168**$_{\pm 0.000}$	0.197$_{\pm 0.001}$	0.214$_{\pm 0.004}$	0.176	$\underline{0.174}$	0.177
	192	**0.211**$_{\pm 0.001}$	0.235$_{\pm 0.000}$	0.231$_{\pm 0.003}$	$\underline{0.220}$	0.221	0.225
	336	**0.256**$_{\pm 0.000}$	0.276$_{\pm 0.001}$	0.279$_{\pm 0.007}$	$\underline{0.265}$	0.278	0.278
	720	**0.320**$_{\pm 0.001}$	0.334$_{\pm 0.000}$	0.343$_{\pm 0.024}$	$\underline{0.323}$	0.358	0.354

Table 4: Relative difference (% reduction in MSE) of Linear++ vs. baselines: $\Delta\% = 100 \times \frac{\text{Baseline}-\text{Linear++}}{\text{Baseline}}$. Positive = Linear++ better (lower MSE).

Dataset	H	vs SAMformer	vs TSMixer	vs DLinear	vs ITransformer	vs PatchTST
ETTh1	96	3.9%	8.0%	2.4%	5.2%	11.6%
	192	2.0%	5.9%	1.0%	9.1%	12.8%
	336	-0.9%	1.8%	2.7%	12.3%	14.8%
	720	-7.5%	7.8%	2.8%	8.7%	8.2%
ETTh2	96	5.4%	9.4%	3.5%	6.1%	7.6%
	192	-9.4%	-5.7%	2.9%	2.1%	4.1%
	336	-18.9%	-15.6%	7.1%	2.8%	2.3%
	720	-14.8%	-9.8%	25.8%	-5.2%	-4.2%
ETTm1	96	4.3%	6.3%	-5.4%	5.7%	4.3%
	192	-7.6%	-5.0%	-13.4%	-0.8%	-3.5%
	336	-6.0%	-3.6%	-9.8%	4.9%	-1.5%
	720	-4.7%	0.2%	-5.6%	8.6%	1.1%
ETTm2	96	30.4%	40.3%	24.6%	30.0%	28.0%
	192	30.5%	35.7%	27.7%	35.2%	32.8%
	336	30.5%	34.7%	29.5%	36.3%	35.1%
	720	32.5%	35.1%	36.3%	38.6%	37.1%
Exchange	96	44.7%	74.1%	-9.9%	-3.5%	-1.1%
	192	26.4%	47.1%	-15.3%	-2.3%	-2.8%
	336	-1.1%	23.1%	-22.0%	-12.4%	-23.6%
	720	10.4%	25.3%	-39.8%	-6.1%	0.2%
Weather	96	14.7%	21.5%	4.5%	3.4%	5.1%
	192	10.2%	8.7%	4.1%	4.5%	6.2%
	336	7.2%	8.2%	3.4%	7.9%	7.9%
	720	4.2%	6.7%	0.9%	10.6%	9.6%

C Experimental Details

C.1 Gift-Eval benchmark

Datasets. We run our experiments on the GIFT-EVAL benchmark, which spans 45 public data sets covering a wide range of domains, frequencies and target dimensions.

Each data set is provided in three regime splits *short*, *medium* and *long*exactly as defined by the benchmark.

Hyper-parameter optimisation. For every *(data set, regime)* pair we launch an OPTUNA study with 20 trials, minimising the validation loss. The search space is:

Table 5: Search space used in the OPTUNA study.

Hyper-parameter	Interval / Set	Distribution
learning-rate	$[10^{-5}, 10^{-2}]$	log-uniform
weight-decay	$[10^{-6}, 10^{-2}]$	log-uniform
context-multiplier	$\{2, 5, 10, 15, 20\}$	categorical

where the effective context length equals $L =$ context-multiplier $\times$ prediction length.

Training protocol. Each trial trains for at most 15 epochs during optimisation and up to 150 epochs for the final model, both with early stopping (patience $= 5$). The optimiser is Adam with the sampled learning rate and weight decay.

Evaluation. We report the official Gift-Eval metrics: MSE, MAE, RMSE, MAPE, SMAPE, ND, MSIS and MEAN_WEIGHTED_SUM_QUANTILE_LOSScomputed with the open-source evaluation module of GLUONTS.