Moirai 2.0: When Less Is More for Time Series Forecasting

Chenghao Liu[*] **Taha Aksu**[*] **Juncheng Liu**[*] **Xu Liu**[*]
Hanshu Yan **Quang Pham** **Silvio Savarese** **Doyen Sahoo** **Caiming Xiong** **Junnan Li**[†]
Salesforce AI Research

 https://github.com/SalesforceAIResearch/uni2ts

 https://huggingface.co/Salesforce/moirai-2.0-R-small

Abstract

We introduce Moirai 2.0, a decoder-only time-series foundation model trained on a new corpus of 36M series. The model adopts quantile forecasting and multi-token prediction, improving both probabilistic accuracy and inference efficiency. On the GIFT-EVAL benchmark, it ranks among the top pretrained models while achieving a strong trade-off between accuracy, speed, and model size.

Compared to Moirai 1.0, Moirai 2.0 replaces masked-encoder training, multi-patch inputs, and mixture-distribution outputs with a simpler decoder-only architecture, single patch, and quantile loss. Ablation studies isolate these changes—showing that the decoder-only backbone along with recursive multi-quantile decoding contribute most to the gains. Additional experiments show that Moirai 2.0 outperforms larger models from the same family and exhibits robust domain-level results. In terms of efficiency and model size, Moirai 2.0 is twice as fast and thirty times smaller than its prior best version, Moirai 1.0-Large, while also performing better. Model performance plateaus with increasing parameter count and declines at longer horizons, motivating future work on data scaling and long-horizon modeling. We release code and evaluation details to support further research.

1 Introduction

Time series forecasting underpins capacity planning, anomaly response, and risk management across cloud infrastructure, observability, finance, energy, retail, and healthcare. Unlike text or images, time series exhibit nonstationarity, multi-scale temporal structure, irregular sampling, and incomplete observations, making generalization beyond a single domain both important and challenging. To tackle these, some early attempts on foundation models (FMs) for time series have been proposed and shown some promising results on cross-domain reuse, zero/few-shot adaptation, and unified deployment at scale.

Moreover, foundation models for time series have rapidly evolved from early exploratory attempts just a few years ago to becoming a central research direction with significantly growing adoption in industry. Today, the landscape includes dozens of pretrained time series foundation models from both academia and industry. Representative examples from industry include TimesFM [9] from Google, Chronos/Chronos-Bolt [2] and Chronos 2 [3] from Amazon, TOTO [8] from Datadog, and YingLong [32] from Alibaba. Community benchmarks from industry, such as GIFT-EVAL benchmark [1] and FEV-BENCH [29], also have accelerated progress by standarizing datasets, metrics and reporting performance of different models. A clear sign of this progress is that the GIFT-EVAL benchmark [1] has already received 25 foundation model submissions since its launch, showing the growing interest from the community.

Our previous work, Moirai 1.0 [34], was among the first attempts to scale the transformer architecture for time series. It demonstrated that large-scale time-series pretraining can yield strong generalization across domains, while also providing a blueprint for subsequent foundation models. While effective, Moirai 1.0 has also revealed several limitations. Its masked-encoder design led to inefficient data utilization during training, and the multi-patch setup constrained learning across different temporal frequencies. Moreover, while outputting a mixture of distributions was an intuitive way to enhance probabilistic forecasting, it proved empirically less effective in practice and added substantial complexity to both the model design and optimization process.

In this work, we introduce Moirai 2.0, a successor model that incorporates several architectural and training refinements. Key improvements include the adoption of quantile-based predictions with a refined loss function, a

---

[*]Equal Contribution.
[†]Corresponding author. Email: junnan.li@salesforce.com

decoder-only autoregressive architecture, and multi-token prediction strategy with a single patch size, along with additional optimization techniques applied during both pretraining and inference. We have also curated a new pretraining datasets to train this model which includes $36M$ time series and $\sim 295B$ observations. These changes substantially improve the accuracy and efficiency of our model, making it the best-performing model at the time of its release. As of today, it remains highly competitive, ranking 5th among 37 foundation models (including all size variants) on the GIFT-EVAL leaderboard, while offering one of the most favorable trade-offs between inference speed, model size, and accuracy.

The remainder of this paper is organized as follows. In Section 2, we provide the background and discuss related work on time series foundation models. Section 3 introduces the architectural design and training strategies behind Moirai 2.0. Section 4 lists the datasets we used to pretrain our new model. Section 5 presents evaluations on the GIFT-EVAL benchmark, including efficiency comparisons, scaling experiments, and ablation studies. Section 6 outlines the limitations of our current approach and highlights future research directions. Finally, Section 7 summarizes our findings and concludes the paper.

2 Background & Related Work

The time series point forecasting task is defined as predicting the future H values given a historical sequence with c values, *i.e.* let the historical context be $Y_{1:c} = \{y_1, y_2, ..., y_c\}$ the target sequence to predict would be $Y_{c+1:c+H} = \{y_{c+1}, y_{c+2}, ..., y_{c+H}\}$. The forecasting task can naturally extend to the multivariate setting where one is expected to predict multiple variates simultaneously *i.e.* $Y^1_{c+1:c+H}, Y^2_{c+1:c+H}, ...Y^N_{c+1:c+H}$. Moirai 2.0, however, does not support cross-variate forecasting. Instead, we treat multivariate forecasting as a collection of independent univariate tasks. Moirai 2.0 does not only predict point but quantile forecast, where the difference in definition comes at the prediction level. For quantile forecast given the same historical context $Y_{1:c}$, the target prediction is now the quantile levels for each future time step $Q_{c+1:c+H} = \{Q_{c+1}, Q_{c+2}, ..., Q_{c+H}\}$ where $Q_i = \{q^i_{l_1}, q^i_{l_2}, ..., q^i_{l_q}, \}$, l_1 through l_q being the quantile levels. We set the quantile levels as $\{0.1, 0.2, ..., 0.9\}$. Quantile forecasts offer a principled way to model uncertainty, as they are directly optimized for probabilistic forecasting accuracy.

The family of time series forecasting models can be categorized into 3 classes: statistical models, deep-learning models and foundation models. The statistical models work locally and rely solely on historical data statistics to predict future values. Some of the most widely used ones are ETS [18], Theta [13], and ARIMA [5].

Deep learning models, in contrast, are often dataset-specific, as each dataset typically requires training a dedicated model. Examples include N-BEATS [26], DLinear [37], and DeepAR [12], which are based on pre-transformer architectures. In contrast, transformer-based approaches such as Autoformer [35], Crossformer [39], and PatchTST [24] have also been proposed.

In the last few years, the success of foundation models in language and vision has also attracted time series researchers. What initially started as a handful of early attempts [14, 28, 34] has now grown into a rapidly expanding landscape, with more than 25 time series foundation models being publicized in the last 2 years.

The architecture and training strategies of time series foundation models vary considerably. Notably, two prominent approaches do not directly rely on transformer architectures: TabPFN-TS [17], which employs a Prior-Data Fitted Network (PFN), and TiRex [4], which is built on an xLSTM. The majority of other models, however, rely on transformer variants. Within the transformer family, some models adopt an encoder-only design, such as Chronos-2 [3], Yinglong [32], and the first version of our Moirai [34]. In this setup, a forecasting head is placed on top of the encoder backbone. This approach can mitigate error accumulation and accelerate inference, particularly when autoregression is avoided. In contrast, another group follows the decoder-only paradigm inspired by large language models (LLMs). Examples include Moirai-MoE [21], TimesFM family of models [9], and Sundial [22]. Others adopt a hybrid encoder–decoder architecture, such as Kairos [11], Chronos, Chronos-Bolt [2], and FlowState [15]. A further axis of differentiation is the choice of output representation and loss function. Some models predict point forecasts directly [9, 11]. Others, like Moirai, predict full distributions [34], or employ flow-based losses to learn continuous distributions from which forecasts can be sampled as applied by Graf et al. [15], Liu et al. [22], or directly output quantiles [2–4, 32].

Moirai 2.0 adopts a decoder-only architecture, with newly incorporated training and inference strategies to limit error accumulation and inference slowdown. Unlike point or distribution-based approaches, it outputs quantile forecasts, directly aligned with the CRPS metric through optimization with the quantile (pinball) loss.
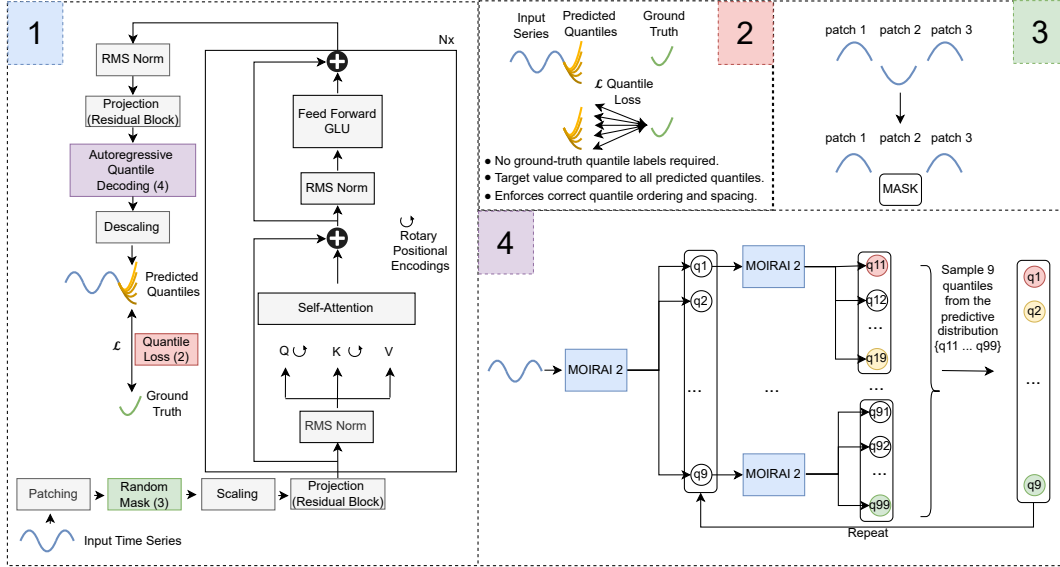
3   Model Architecture and Training

Figure 1: Overview of the Moirai 2.0 architecture. Panel 1 illustrates the end-to-end pipeline from patched input time series through the transformer backbone to predicted quantiles. Panel 2 highlights the quantile loss, which compares each ground-truth value against all predicted quantiles without requiring quantile labels, enforcing correct ordering and spacing. Panel 3 depicts patch-level random masking used to improve robustness. Panel 4 shows a simplified view of the autoregressive multi-step quantile decoding strategy, where quantile forecasts are recursively rolled out to construct predictive distributions across the horizon.

In this section, we first describe the model architecture of Moirai 2.0 in Section 3.1 and present the details of training and inference in Section 3.2. Finally, in Section 3.3, we discuss and summarize the differences between Moirai 2.0 and Moirai 1.0.

3.1   Model Architecture

For efficient training, Moirai 2.0 is structured as a decoder-only model built upon the Transformer architecture. We provide an overview of our architecture in Figure 1.

Input Projection

The input layers are responsible for preprocessing the raw time-series into input tokens for the transformer architecture. Currently, Moirai 2.0 is designed for univariate time-series input. Multivariate data is accommodated by treating each variable as an independent univariate series, an approach often found to perform well [24]. Instance normalization [20] is applied to each time series individually to ensure the model is robust to diverse and non-stationary input scales. Adhering to established practices [34], the input time-series is first partitioned into contiguous, non-overlapping T patches. Missing values are explicitly handled by concatenating a corresponding binary indicator $\mathbf{m}_i$ (representing presence or absence) with the value at each time step within the patch. Each resulting patch, augmented with this missing value information as $\hat{\mathbf{x}}_i = \mathbf{x}_i || \mathbf{m}_i$, is then processed into an input token with a residual block [16]. Formally, given an input patch size of p_{in} and token embedding dimension d, this input projection with a residual block defines a mapping $\mathbb{R}^{2p_{in}} \rightarrow \mathbb{R}^d$ as shown below:

$$\mathbf{z}_i = \text{PatchEmbed}(\hat{\mathbf{x}}_i) = \text{SiLU}(\mathbf{W}(\hat{\mathbf{x}}_i) + \mathbf{b}) + \hat{\mathbf{x}}_i \in \mathbb{R}^d, i = 1, ...T, \tag{1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are weights and bias for the input projection.

Additionally, we also apply normalization before input projection. Notably, as mentioned in Liu et al. [21], applying instance normalization globally risks future information leakage in decoder-only models. We aim to avoids this by

computing normalization statistics solely from the first 30% of the time series, reserving the subsequent 70% segment for the causal pre-training task.

Stacked Transformer

Following the input projection, the sequence of tokens is processed by multiple stacked Transformer layers, *c.f.* Panel 1 in Figure 1. Each layer within this stack adheres to the standard Transformer architecture [30], primarily comprising two sub-layers: a causal multi-head self-attention mechanism followed by a feed-forward network. The causal nature of the self-attention ensures that the prediction for a given token depends only on valid preceding tokens and the current token itself, maintaining the autoregressive property required by the decoder-only framework. Residual connections are employed around each of the two sub-layers, followed by layer normalization, promoting stable training and effective gradient flow through the deep network. The output sequence from the final Transformer layer represents the learned contextual embeddings of the input patches.

Output Projection

The final stage involves projecting the sequence of processed token embeddings, derived from the stacked transformer layers, onto the target prediction space. An output residual block [16] performs this transformation, converting each d-dimensional token embedding into forecasts. Two key features are incorporated into this projection: First, to facilitate probabilistic predictions, the output dimension is scaled to accommodate n_q quantile estimates per time step [33]. Second, the model employs **multi-token prediction**, generating forecasts for n_{token} future patches from each output token representation to improve effectiveness and efficiency for long term forecast. Thus, for an output patch size p, the output projection layer effectively realizes a mapping $\mathbb{R}^d \to \mathbb{R}^{n_{token} \times n_q \times p}$, transforming the learned representations into structured quantile forecasts for multiple future patches.

We choose multi-token prediction instead of single-token prediction to improve efficiency, especially for long-term forecasting with relatively large prediction lengths. Additionally, multi-token prediction also helps to reduce the error accumulation in long-term forecasting.

Loss Function

To support probabilistic forecasting capabilities, Moirai 2.0 is trained by optimizing the quantile loss (also known as the pinball loss). This objective function enables the model to learn the conditional distribution of future time series values. Specifically, the model is configured to predict $n_q = 9$ distinct quantile levels, equidistantly spaced from 0.1 to 0.9 ($Q = \{0.1, 0.2, \ldots, 0.9\}$). The quantile loss for a single prediction at time step t, for a given quantile level $q \in Q$, is defined as:

$$\ell_q(y_t, \hat{y}_t^{(q)}) = \begin{cases} q(y_t - \hat{y}_t^{(q)}) & \text{if } y_t \geq \hat{y}_t^{(q)}, \\ (1-q)(\hat{y}_t^{(q)} - y_t) & \text{if } y_t < \hat{y}_t^{(q)}, \end{cases}$$

where y_t is the ground truth value at time t, and $\hat{y}_t^q$ is the model's predicted value for the q-th quantile at that time step. During training, the total loss is calculated by averaging the quantile loss across all predicted quantile levels ($q \in Q$) and all H time steps in K predicted patches, while ignoring contributions from any masked or missing target values. We formalize the total loss as follows:

$$\mathcal{L}_Q = \frac{1}{H|Q|} \sum_{t=1}^{H} \sum_{q \in Q} \ell_q(y_t, \hat{y}_t^{(q)}) \tag{2}$$

$$= \frac{1}{H|Q|} \sum_{t=1}^{H} \sum_{q \in Q} \left[q \max\left(y_t - \hat{y}_t^{(q)}, 0\right) + (1-q) \max\left(\hat{y}_t^{(q)} - y_t, 0\right) \right], \quad H = Kp, \tag{3}$$

Note that, by default, we treat all quantile levels equally, which if often preferable in practice and encourages accurate and well-calibrated quantile forecasts. However, when specific regions of the distribution are of greater interest, we can assign different weights (e.g., w_q) on selected quantiles to emphasize them. We discuss more on this in the last paragraph of Section 3.3

3.2 Training and Inference

Training

To enhance model robustness, **patch-level random masking** is utilized: 50% of input patches in each training sample are randomly masked before being processed by the model. This encourages robust representation learning and

improves handling of missing data segments. Since normalization statistics are computed using only the initial 30% segment of each sample, distribution shifts between this segment and the remaining 70% utilized for training may arise, potentially causing significant instability during the training process. To mitigate this, we employ z-score based anomaly detection, samples are filtered out if the latter 70% segment exhibits significant statistical deviation from the initial segment used for normalization. Following a similar training procedure as Moirai 1.0 [34], Moirai 2.0 is trained for $100,000$ steps with a batch size of 256. We utilize the AdamW optimizer configured with a learning rate of 1×10^{-3}, weight decay of 1×10^{-1}, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. The learning rate schedule includes a linear warmup over the first 10,000 steps, followed by cosine annealing. Training is conducted using bf16 mixed-precision arithmetic to enhance efficiency.

Inference

In standard autoregressive decoding, a model generates forecasts step by step by feeding its output back as input. This works naturally when the model predicts a single value per step, since the input and output dimensions match.

For *quantile forecasting*, however, the model outputs multiple values (quantiles) at each step. Feeding all quantiles directly back into the model creates a dimensional mismatch, and collapsing them to a single value (e.g., the median or mean [9]) discards important information about uncertainty.

To address this, we propose to use **autoregressive multi-quantile decoding**. Conceptually, this procedure resembles a *beam search with depth 2*. Given the forecast with multiple quantiles at the previous step $t-1$, instead of committing to a single quantile path with the median value, we temporarily expand the search tree at the current step t, then collapse it back to a fixed set of quantiles. At each decoding step t, we first *expand* the forecast with several quantiles from step $t-1$ into a larger number of candidate quantiles (e.g., $9 \times 9 = 81$). Then, we *collapse* these samples back into the desired quantile set (e.g., 9 levels) by sampling from the expanded set and use the collapsed quantiles as the forecast for step t. This procedure preserves forecast uncertainty while keeping the inference tractable.

Algorithm 1 Autoregressive Multi-Quantile Decoding with Initialization (depth-2 expand $\rightarrow$ collapse)

Require: Context $Y_{1:c}$; target quantile set $\mathcal{Q}$ with $m = |\mathcal{Q}|$; horizon H
    Initialization (first prediction, no expansion).
1: Predict the first-step quantiles directly from the context: $\{\hat{y}_{c+1}^{(q)}\}_{q \in \mathcal{Q}} \leftarrow \text{MOIRAI2}(Y_{1:c})$.
2: Append $\{\hat{y}_{c+1}^{(q)} : q \in \mathcal{Q}\}$ to the context.
    Autoregressive steps (beam-search depth 2).
3: **for** $t = c+1$ to $c + H - 1$ **do**
4:     **Expand:** form m histories by appending each $\hat{y}_t^{(q_1)}$ to the current context.
5:     From each expanded history, decode one step ahead to obtain $\{\hat{y}_{t+1}^{(q_1, q_2)}\}_{q_2 \in \mathcal{Q}}$; pool all candidates

$$\mathcal{S}_{t+1} \leftarrow \{\hat{y}_{t+1}^{(q_1, q_2)} : q_1, q_2 \in \mathcal{Q}\}, \quad |\mathcal{S}_{t+1}| = m^2.$$

6:     **Collapse:** for each $q \in \mathcal{Q}$, set

$$\hat{y}_{t+1}^{(q)} \leftarrow \text{Quantile}_q(\mathcal{S}_{t+1}).$$

7:     Append $\{\hat{y}_{t+1}^{(q)} : q \in \mathcal{Q}\}$ to the context.
8: **end for**
    Note (patch-level decoding). For clarity the pseudocode shows single-value prediction; in practice, the model outputs multiple patch tokens per iteration.

In practice, this approach allows us to maintain the benefits of quantile forecasting while avoiding dimensional mismatch. It can be viewed as sampling multiple plausible futures (expansion) and then summarizing them into a coherent set of quantiles (collapse).

3.3 Discussion and Comparison on Moirai 2.0 and Moirai 1.0

In this section, we provide some discussions and comparisons on architecture changes from Moirai 1.0 to Moirai 2.0. From the architecture perspective, we mainly have 3 changes: 1) use decoder-only for Moirai 2.0 instead of masked encoder in Moirai 1.0, 2) change to single-path size in Moirai 2.0 from multi-patch size in Moirai 1.0, 3) formulate the training loss as quantile loss in Moirai 2.0 instead of distribution loss Moirai 1.0.

Decoder-only in Moirai 2.0 vs Masked encoder in Moirai 1.0

Moirai 1.0 [34] adopts a masked-encoder architecture, where both context and prediction lengths are randomly sampled during training to support flexible downstream usage. However, this design yields only a single loss per sampled input configuration, leading to suboptimal data utilization and slower training efficiency. In contrast, Moirai 2.0 replaces the masked encoder with a decoder-only architecture, enabling more effective use of training data and simplifying the overall modeling pipeline.

In terms of data efficiency during training, Moirai 2.0, built upon causal decoder-only transformers, directly computes $T-1$ losses across all tokens or patches for a time series with T tokens. In contrast, the masked-encoder architecture used in Moirai-1.0, assuming a masking rate of 15%, engages only 15% of tokens in the loss computation, producing a single loss value for each specific pair of context and prediction lengths. This difference in training strategy makes Moirai 2.0 substantially more data-efficient during training.

Another advantage of using the decoder-only autoregressive architecture is the potential inference speedup with KV caches [27]. A KV cache stores the intermediate key–value attention representations computed from the input tokens during the first forward pass; at later decoding steps, the model can reuse these cached representations instead of recomputing them, reducing inference cost. Thus, when the model is asked to provide predictions repeatedly (e.g., n times), autoregressive models such as Moirai 2.0 can pre-fill the context once and reuse cached KV states, avoiding redundant computation. In contrast, masked-encoder models such as Moirai 1.0 must recompute key–value features n times from scratch, making them less efficient for repeated queries. This scenario is not uncommon in real-world forecasting, where end users often do not know the exact prediction horizon in advance—they may start by forecasting a few steps and then gradually extend the prediction as needed. To quantify the potential benefit, we conducted a case study by implementing a KV-cache–enabled version of Moirai 2.0. The results show that inference speedup scales with both context and prediction length: for a context of approximately 10K and a prediction length of 1K, KV caching provides up to a $4\times$ speedup, while extending the prediction length to 10K increases the gain to $17\times$.

Multi patch size vs Single patch size

In Moirai 2.0, we standardize to a single patch size, in contrast to the multi-patch design of Moirai 1.0. Empirically, this simplification improves both computational efficiency and forecasting accuracy, while also simplifying the implementation of training and inference stages.

Quantile forecast vs Distribution forecast

In Moirai 1.0, the model outputs the distribution parameters for a mixture of distributions,and generates forecasts by sampling from this mixture. During inference, 100 samples per step are typically drawn, and during training the distribution negative log likelihood (NLL) loss is used. In contrast, we use quantile loss for training Moirai 2.0, which generates $|Q|$ quantiles as the forecast. The quantile loss can directly optimize operational quantiles (e.g., using $q = 0.9$ for capacity planning) and it naturally handles asymmetric penalties (over- vs under-prediction). Compared to the distribution NLL loss, which may suffer from variance collapse, explosion, or unstable gradients under outliers, the quantile loss is more robust to extreme tails.

4 Pretraining Datasets

Our pretraining corpus is constructed from five complementary sources: (1) the non-leaking GIFT-EVAL PRETRAIN dataset, (2) train split of GIFT-EVAL TRAINTEST dataset [1] (3) additional series generated via Chronos-Mixup, (4) KernelSynth data [2], and (5) anonymized internal Salesforce CloudOps telemetry data. In combination these five components add up to $36M$ time series, with $\sim 295B$ observations, providing a diverse mixture of real-world and synthetic time series, covering a wide range of domains, frequencies, and temporal characteristics. We describe each component in more detail below.

Gift-Eval Pretrain and Gift-Eval TrainTest

The original Moirai model, reported in the GIFT-Eval leaderboard, was trained using a non-leaking version of the pretraining corpus GIFT-EVAL PRETRAIN [1]. Our new pretraining corpus also includes this same non-leaking dataset as its foundation. It has $3.25M$ time series with a total of $230B$ observations[1]. GIFT-EVAL PRETRAIN provides a

---

[1]Note that GIFT-Eval Pretrain is a subset of LOTSA introduced by Woo et al. [34]. While the original paper reports the dataset size as 27B observations, this discrepancy arises solely from differences in counting: they treat each multivariate series as a single observation, whereas in our accounting each variable is counted independently.

large and diverse set of time series carefully curated to avoid overlap with benchmark evaluation tasks. Additionally we add the train split of GIFT-EVAL TRAINTEST dataset to train Moirai 2.0, which includes $144K$ time series. For details about both datasets composition and construction methodology, we refer the reader to Aksu et al. [1].

Chronos-Mixup

Time Series Mixup (TSMixup) is proposed originally by Ansari et al. [2] which takes inspiration from original Mixup used in image classification [38]. It randomly samples $k \sim \mathcal{U}\{1, K\}$ time series of length $l \sim \mathcal{U}\{l_{min}, l_{max}\}$ from a distribution of training data then takes their convex combination. We follow Auer et al. [4], and generate significantly more and longer series and following their configuration we set the max value of $K = 4$, and set $L_{min} = 128$, $L_{max} = 4096$. We only use non leaking subset of datasets from Chronos data to generate the mixup to keep our training free of leakage. We generate 30M time series that add up to 63B observations in total.

KernelSynth

We also include the KernelSynth data provided by Ansari et al. [2] which generate synthetic time series using Gaussian processes. Their method constructs a kernel bank including kernels for trend, local change and seasonalities. The final kernel time series is constructed by sampling kernel from the bank and combining them with random binary operations $+$ or $\times$. This set contains 1M time series, with $1.02B$ observation in total.

Internal Salesforce data

A core component of our pre-training corpus is derived from internal salesforce telemetry data. This dataset comprises approximately 2.15M univariate time series, totaling roughly 1.48B observations at a daily granularity. The data covers a period of approximately one year, starting from January 2024. To ensure data quality, time series containing substantial amounts of missing values were filtered out prior to training.
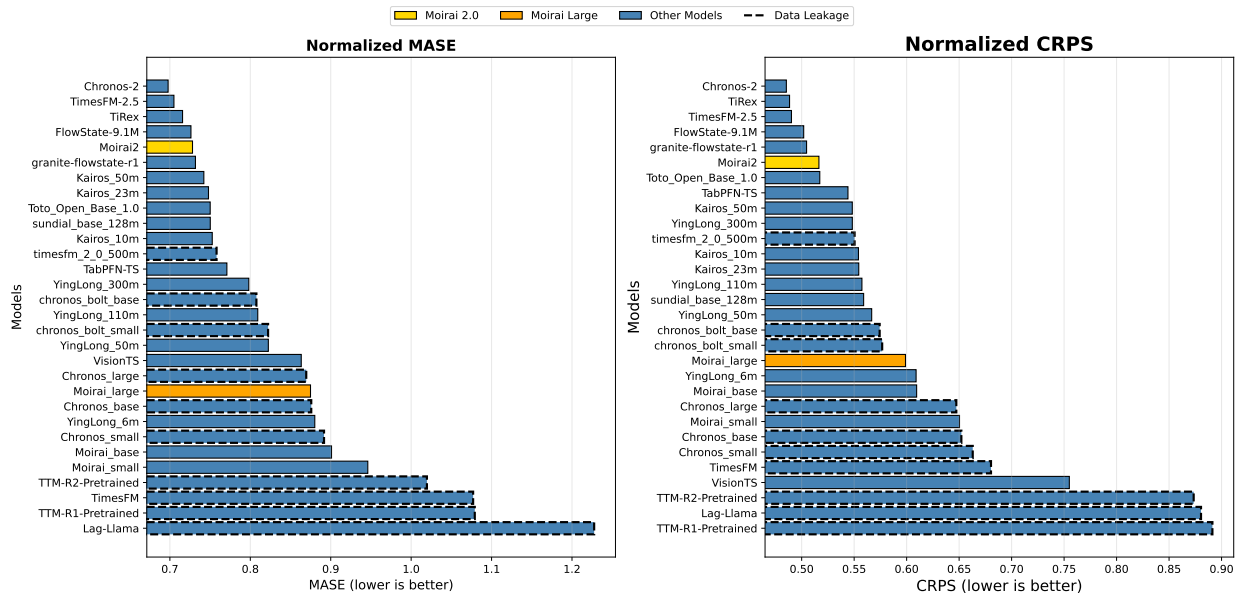

Figure 2: GIFTEVAL benchmark results for pretrained, zero-shot foundation models. Ensemble methods and models lacking reproducible code are excluded. Bars show normalized MASE (left) and normalized CRPS (right), where lower is better. Moirai 2.0 and its large variant rank among the top models under both metrics.

5   Evaluation

In this section, we present experimental results that demonstrate the capabilities of our model compared to other state-of-the-art foundation models. We primarily evaluate on the GIFT-Eval benchmark [1], comparing against all available pretrained foundation models with reproducible code and no test-data leakage. In addition to overall benchmark
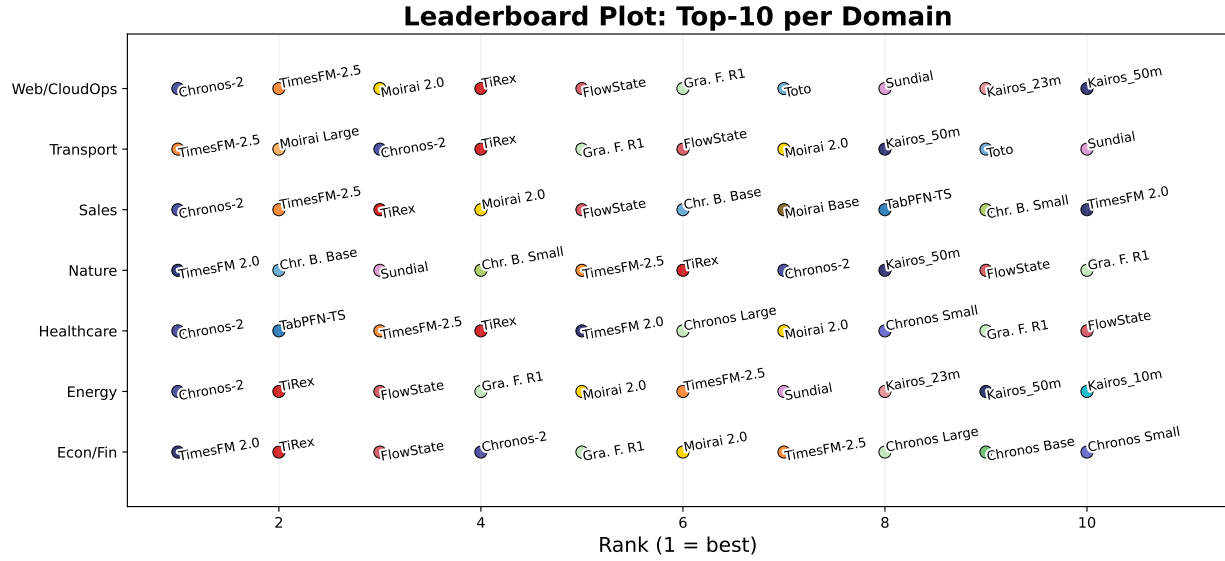
Leaderboard Plot: Top-10 per Domain

Figure 3: GIFTEVAL leaderboard results broken down by domain. For each domain, we display the top-10 foundation models ordered by their MASE rank (lower is better).
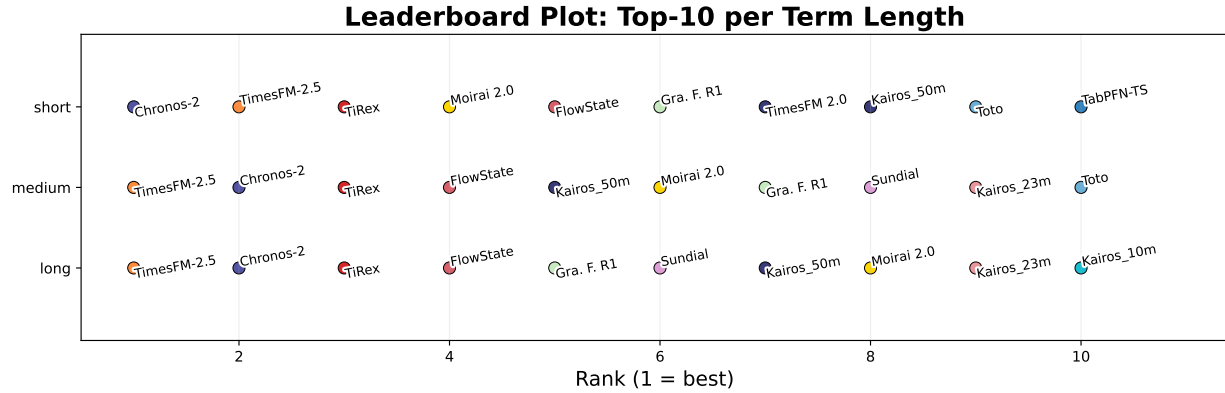
Leaderboard Plot: Top-10 per Term Length

Figure 4: GIFTEVAL leaderboard results broken down by prediction length. For each domain, we display the top-10 foundation models ordered by their MASE rank (lower is better).

performance, we analyze Moirai 2.0 from several perspectives: inference speed and models size and their relationship with accuracy compared to top leaderboard models, and the effect of scaling Moirai 2.0 to larger parameter counts. We also report domain-specific results across models and conclude with an ablation study to quantify the contribution of individual components.

5.1 Results on GIFT-Eval Benchmark

We evaluate Moirai 2.0 on the comprehensive GIFT-Eval benchmark, comparing it against state-of-the-art foundation models listed on the leaderboard. GIFT-Eval covers 97 task configurations across 55 datasets, spanning diverse domains, frequencies, and prediction lengths. For this study, we exclude agentic solutions, fine-tuned models, and foundation models without publicly available replication code. This filtering leaves 30 foundation models for comparison, including `Chronos-2` [3], `TimesFM-2.5`, `TimesFM-2.0`, `TimesFM` [9], `TiRex` [4], `FlowState` and `Granite-FlowState-R1` [15], `Kairos (10/23/50M)` [11], `Toto` [8], `Sundial` [22], `TabPFN-TS` [17], `YingLong (6/50/110/300M)` [32], `Chronos (small/base/large)` and `Chronos-Bolt (small/base)` [2], `Moirai (small/base/large)` [34], `TTM-R1-Pretrained` [10], and `Lag-Llama` [28].

Following leaderboard protocols, we report normalized MASE and CRPS, where each model's score is divided by the seasonal-naive baseline and aggregated using the geometric mean across tasks. As shown in Figure 2, Moirai 2.0 attains the 5th and 6th positions on MASE and CRPS, respectively—substantially outperforming its predecessor Moirai-Large despite having fewer parameters.

Fine Grained Results

Beyond overall benchmark scores, we analyze Moirai 2.0's performance across domains in GIFT-Eval. As shown in Figure 3, Moirai 2.0 consistently appears in the top-10 across most domains, with the notable exception of Nature tasks. This gap may suggest that the pretraining corpus underrepresents natural and environmental time series, motivating future work to balance domain coverage. Compared to its predecessor Moirai-Large, Moirai 2.0 delivers stronger results in nearly all domains, with Transport being the only case where Moirai-Large remains highly competitive. An additional perspective emerges when grouping tasks by forecast horizon Figure 4: Moirai 2.0 ranks 4th, 6th, and 8th on short, medium, and long prediction lengths, respectively. This trend highlights that while the model is highly competitive for short-range forecasting, its relative advantage diminishes as the prediction horizon grows.
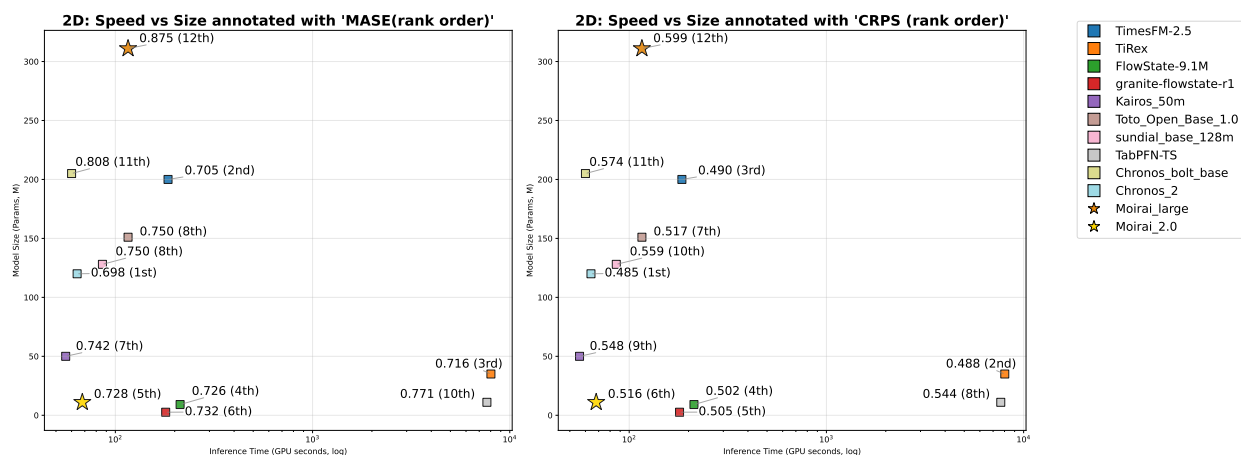

Figure 5: Speed–parameter count comparison across foundation models. Each point shows inference time (x-axis) vs. model size (y-axis), annotated with performance rank. The left plot annotates model rank by MASE, while the right plot by CRPS (lower is better). Moirai 2.0 variants achieve competitive accuracy while offering favorable size and inference efficiency.

5.2 Efficiency Comparison

To assess inference efficiency, we conducted experiments on 12 representative tasks from GIFT-Eval, selected to cover diverse dataset–frequency combinations and forecast horizons. Specifically, we included short-horizon tasks such as `M4--yearly`, `M4--hourly`, `Hospital--monthly`, `US Births--weekly`, `Electricity--daily`, `Saugeenday--daily`, `Bizitobs_l2c--hourly`, `SZ_Taxi--15 minutely`, and `Solar--hourly`, as well as medium/long-horizon tasks including `SZ_Taxi--15 minutely`, `Solar--hourly`, and `Bizitobs_l2c--hourly`. These tasks were chosen solely to maximize diversity in frequency and prediction length; accuracy on this subset is not emphasized, as we continue to use the full set of 97 tasks for performance reporting. For efficiency, we timed each model's total inference time to generate forecasts on all selected tasks, running on a single H200 GPU and using the official replication code provided in the GIFT-Eval repository.[2]

Figure 5 summarizes the trade-offs between model size, inference time, and predictive performance. Among baseline models, `Kairos-50M` emerges as the fastest, although it lags behind Moirai 2.0 in benchmark accuracy and is nearly five times larger in parameter count. `Granite-FlowState-R1`, on the other hand, achieves stronger accuracy than Moirai 2.0 while being smaller in size, but it is roughly three times slower in inference speed. Most notably, Moirai 2.0 demonstrates a substantial leap in efficiency compared to its predecessor: it is approximately 30× smaller and 2× faster than Moirai-Large, while simultaneously achieving significantly better accuracy.

---

[2]Replication code for all leaderboard models is available at: https://github.com/SalesforceAIResearch/gift-eval/tree/main/notebooks.

5.3 Scaling Experiments

We further investigate the effect of scaling model size on performance by training two larger variants of Moirai 2.0, namely base and large, which increase the parameter count by approximately $8\times$ and $30\times$ compared to the small version. As shown in Table 1, we do not observe scaling benefits: both larger variants underperform the small model on GIFT-Eval in terms of MASE and CRPS. In fact, the small variant already appears sufficient to fully exploit the available pretraining data, and simply increasing the parameter count does not translate into better results. This finding suggests that improvements may require further scaling of pretraining data, architectural innovations, or targeted regularization strategies, rather than model size alone.

This trend is consistent with the broader observations reported in the GIFT-Eval benchmark paper, where scaling across model families does not systematically lead to improved forecasting performance [1]. At the same time, other recent studies suggest that scaling parameter counts can yield benefits [11, 32]. We hypothesize that the interaction between model size and pretraining data availability is the critical factor here, without aligned increases in data scale and diversity, parameter growth alone does not guarantee gains [19].

Table 1: Performance of different Moirai 2.0 sizes.

Model	Params (M)	MASE	CRPS
Moirai 2.0 small	11.4	**0.728**	**0.516**
Moirai 2.0 base	87.1	0.732	0.525
Moirai 2.0 large	305	0.743	0.530

5.4 Ablation Study

We conduct an ablation study to trace the evolution from Moirai 1.0 to Moirai 2.0, with results summarized in Table 2. Starting from the original Moirai 1.0 small, which uses an encoder-only architecture and a distributional loss.

To disentangle the effect of new pretraining data, we first trained a v0 of decoder-only architecture with GIFT-Eval Pretrain, which applies the new decoder-only architecture with the original distributional loss. This resulted in improvements for MASE and minimal change for CRPS. Next in v1, we train the same architecture with the newly curated corpus and observe a substantial improvement on both metrics. The v2 version replaces the distributional loss with quantile loss, yielding the largest single improvement across the ablations. Building on this, v3 augments the decoder-only design with recursive decoding, further reducing error. In v4, we additionally apply random masking during pretraining. While this modification alone slightly degrades performance, we retained it in combination with multi-token prediction, where its benefits became more apparent in v5. Multi-token prediction enabled alongside recursive decoding and random masking yields the best CRPS while retaining similar MASE. The final version of Moirai 2.0 strengthens the projection layer by replacing the linear head with a residual block, while keeping quantile loss, decoder-only architecture, and all three pretraining/inference strategies. This final variant achieves the best overall performance, completing the incremental path of design decisions.

Table 2: Ablation study of Moirai 2.0 design choices. Starting from the Moirai 1.0 followed by a decoder-only variant using the original GIFT-Eval pretraining and distributional loss (v0), new pretraining corpus (v1), quantile loss (v2), autoregressive quantile decoding (v3), random masking (v4), and multi-token prediction (v5). The final Moirai 2.0 additionally replaces the linear projection with a residual block, achieving the best overall results.

Variant	Pretrain Data	Loss	Architecture	Projection	Multitoken pred.	Autoreg. quantiale dec.	Random mask	MASE	CRPS
Moirai 1.0 small	GIFT-Eval Pretrain	distribution	enc-only	linear	—	—	—	0.946	0.65
v0	GIFT-Eval Pretrain	distribution	dec-only	linear	—	—	—	0.9288	0.6469
v1	New Corpus	distribution	dec-only	linear	—	—	—	0.85	0.58
v2	New Corpus	quantile	dec-only	linear	—	—	—	0.744	0.553
v3	New Corpus	quantile	dec-only	linear	—	✓	—	0.736	0.533
v4	New Corpus	quantile	dec-only	linear	—	✓	✓	0.772	0.56
v5	New Corpus	quantile	dec-only	linear	✓	✓	✓	0.739	0.527
Moirai 2.0	New Corpus	quantile	dec-only	residual block	✓	✓	✓	0.728	0.516

6 Limitations and Future Work

Seeing minimal benefit in doing so, we have dropped support for multivariate forecasting and the use of covariates in Moirai 2.0. This observation likely stems from the limited availability of high-quality datasets with such properties, which may be revisited in future versions once this limitation is resolved — potentially through synthetic data generation, as demonstrated by Ansari et al. [3].

While Moirai 2.0 benefits substantially from its architectural changes—achieving better performance with a smaller parameter count—it also introduces new challenges for scaling. Counterintuitively, increasing the number of parameters degrades performance, suggesting a mismatch between the model architecture and the available data. Furthermore, the model exhibits weaker performance as the forecast horizons gets longer. In future work we would like to explore bridging this gap between architecture and data, enabling scalable model growth and potentially yielding further performance improvements.

For future work there are several exciting directions we believe are worth to explore. First, agentic solutions that combine time-series analysis capabilities with the reasoning power of large language models (LLMs) represent a highly promising avenue. The recent agent-based submissions to the GIFT-Eval benchmark further demonstrate the potential of this paradigm. We anticipate that such approaches will become increasingly practical and relevant as new benchmarks emerge that demand deeper reasoning and contextual understanding.

Another important direction lies in developing foundation models capable of multimodal reasoning, integrating text, image, and time-series modalities. While some early works have begun exploring this space [6, 7, 23, 25, 31, 36], much remains to be discovered. We believe that forecasting and time-series analysis will become far more valuable for both businesses and everyday users when enriched with additional contextual information which makes multimodal integration an exciting path for future research.

7 Conclusion

In this technical report we introduced our new model, Moirai 2.0 a decoder-only time-series foundation model that pairs quantile forecasting with multi-token prediction and trained on a newly curated diverse pretraining data with 36 million time series. Evaluated on the comprehensive GIFT-EVAL benchmark, Moirai 2.0 is highly competitive—achieving top 5 performance among pretrained models while offering a favorable trade-off between accuracy, inference speed, and parameter count. Our ablations isolate the effect of each design choice and show that switching from encoder-only to decoder-only architecture, together with recursive multi-quantile decoding accounts for most of the observed gains.

Along these advances, we also discussed limitations of Moirai 2.0. We observed diminishing returns from parameter scaling, and degraded performance as forecast horizons lengthen. Addressing these challenges may require: data scaling aligned with model capacity; and architectural improvements aimed at long-horizon performance. By releasing code, documenting and evaluating design choices, and reporting limitations, we hope to support and accelerate progress in this direction.

References

[1] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

[2] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

[3] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

[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. *arXiv preprint arXiv:2505.23719*, 2025.

[5] George E. P. Box and David A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970.

[6] Yifu Cai, Arvind Srinivasan, Mononito Goswami, Arjun Choudhry, and Artur Dubrawski. JoLT: Jointly Learned Representations of Language and Time-Series for Clinical Time-Series Interpretation (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21):23447–23448, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i21.30423. URL https://ojs.aaai.org/index.php/AAAI/article/view/30423.

[7] Winnie Chow, Lauren Gardiner, Haraldur T. Hallgrímsson, Maxwell A. Xu, and Shirley You Ren. Towards Time Series Reasoning with LLMs, December 2024. URL http://arxiv.org/abs/2409.11376. arXiv:2409.11376 [cs].

[8] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability. *arXiv preprint arXiv:2407.07874*, 2024.

[9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[10] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam Nguyen, Wesley M Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *Advances in Neural Information Processing Systems*, 37:74147–74181, 2024.

[11] Kun Feng, Shaocheng Lan, Yuchen Fang, Wenchao He, Lintao Ma, Xingyu Lu, and Kan Ren. Kairos: Towards adaptive and generalizable time series foundation models. *arXiv preprint arXiv:2509.25826*, 2025.

[12] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *ArXiv*, abs/1704.04110, 2017. URL https://api.semanticscholar.org/CorpusID:12199225.

[13] F. Garza, M. M. Canseco, C. Challu, and K. G. Olivares. Statsforecast: Lightning fast forecasting with statistical and econometric models. Presented at PyCon Salt Lake City, Utah, US, 2022. URL: https://github.com/Nixtla/statsforecast.

[14] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *ArXiv*, abs/2402.03885, 2024. URL https://api.semanticscholar.org/CorpusID:267500205.

[15] Lars Graf, Thomas Ortner, Stanisĺ Woĺṣniak, Angeliki Pantazi, et al. Flowstate: Sampling rate invariant time series forecasting. *arXiv preprint arXiv:2508.05287*, 2025.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: How tabpfn-v2 outperforms specialized time series forecasting models. *arXiv preprint arXiv:2501.02945*, 2025.

[18] Rob Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008.

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[20] 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.

[21] 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. *arXiv preprint arXiv:2410.10469*, 2024.

[22] Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *arXiv preprint arXiv:2502.00816*, 2025.

[23] Mike A. Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. Language Models Still Struggle to Zero-shot Reason about Time Series, April 2024. URL http://arxiv.org/abs/2404.11757. arXiv:2404.11757 [cs].

[24] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*.

[25] Wenzhe Niu, Zongxia Xie, Yanru Sun, Wei He, Man Xu, and Chao Hao. LangTime: A Language-Guided Unified Model for Time Series Forecasting with Proximal Policy Optimization, March 2025. URL http://arxiv.org/abs/2503.08271. arXiv:2503.08271 [cs].

[26] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *ArXiv*, abs/1905.10437, 2019. URL https://api.semanticscholar.org/CorpusID:166228758.

[27] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

[28] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

[29] Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting. *arXiv preprint arXiv:2509.26468*, 2025.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[31] Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. ChatTime: A Unified Multimodal Time Series Foundation Model Bridging Numerical and Textual Data, December 2024. URL http://arxiv.org/abs/2412.11376. arXiv:2412.11376.

[32] 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. *arXiv preprint arXiv:2506.11029*, 2025.

[33] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[34] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

[35] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235623791.

[36] Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. ChatTS: Aligning Time Series with LLMs via Synthetic Data for Enhanced Understanding and Reasoning, December 2024. URL http://arxiv.org/abs/2412.03104. arXiv:2412.03104 [cs].

[37] Ailing Zeng, Mu-Hwa Chen, L. Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2022. URL https://api.semanticscholar.org/CorpusID:249097444.

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[39] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.
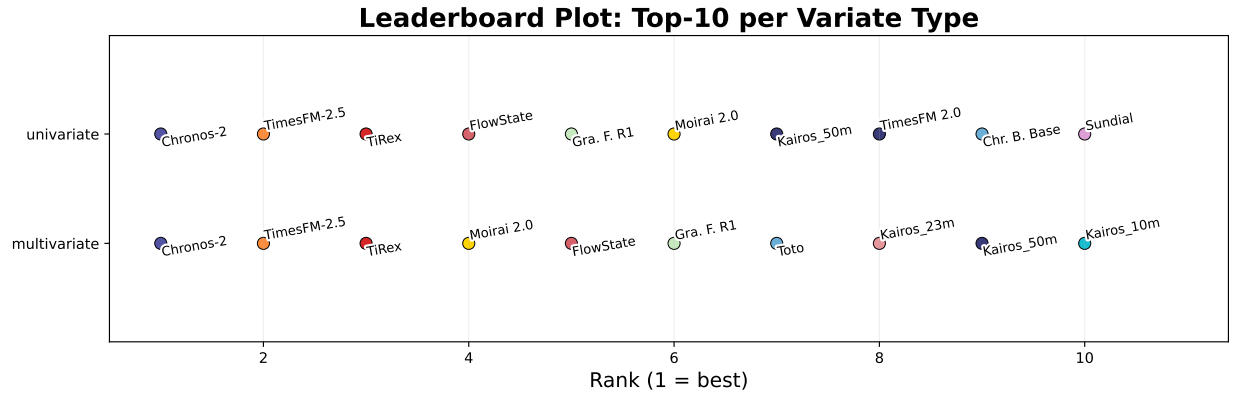
A   Appendix

Figure 6: GIFTEVAL leaderboard results broken down by variate type. For both univariate and multivariate, we display the top-10 foundation models ordered by their MASE rank (lower is better).
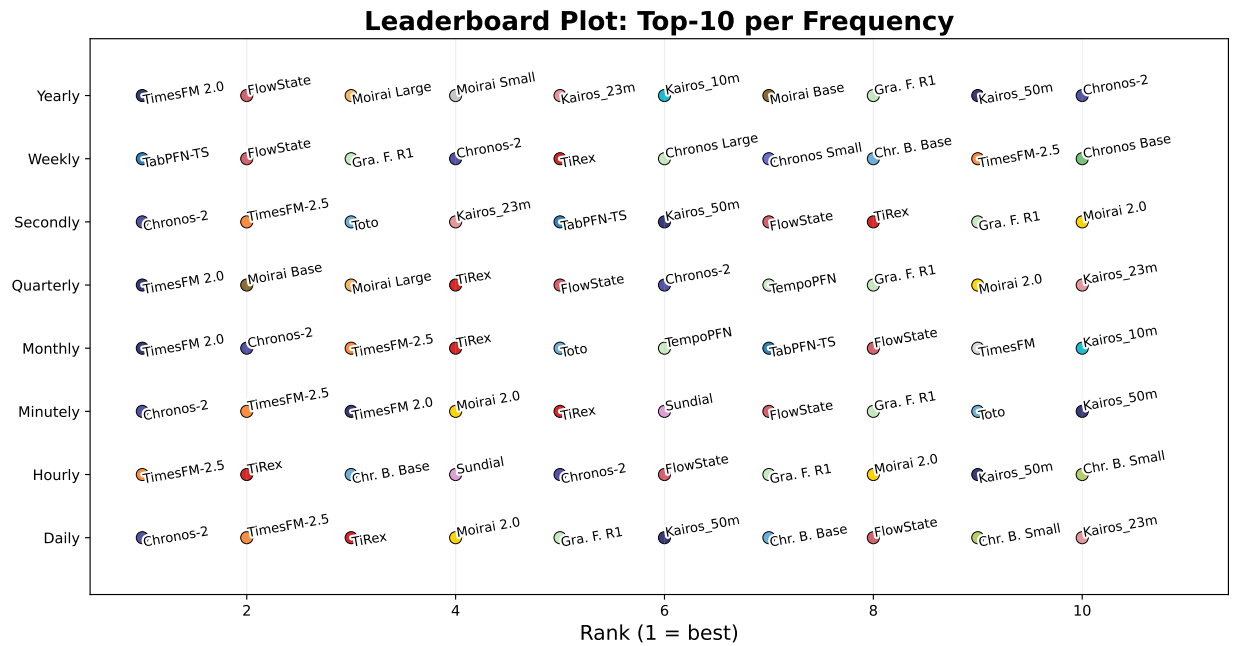

Figure 7: GIFTEVAL leaderboard results broken down by frequency. For each frequency type, we display the top-10 foundation models ordered by their MASE rank (lower is better).
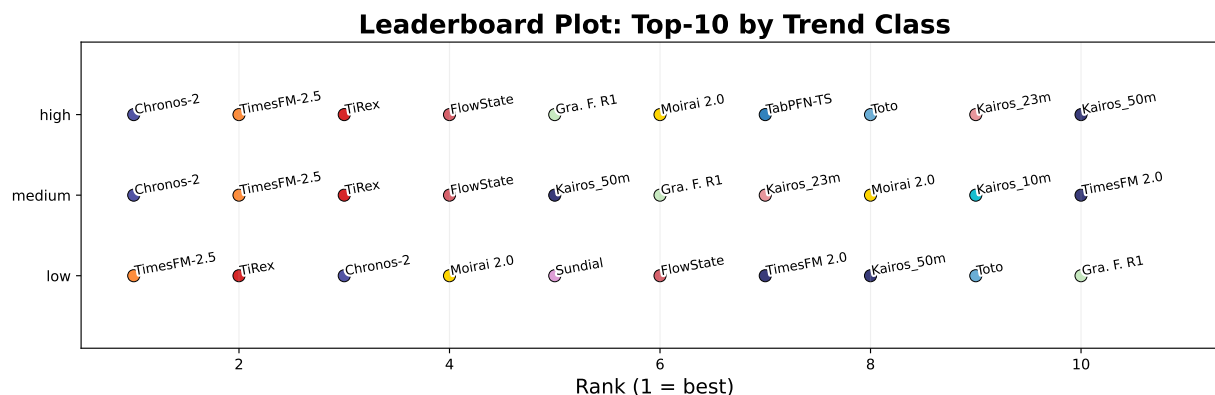
Figure 8: GIFTEVAL leaderboard results broken down by trendiness of time series. High value indicates strong trend and vice versa. We display the top-10 foundation models ordered by their MASE rank (lower is better).
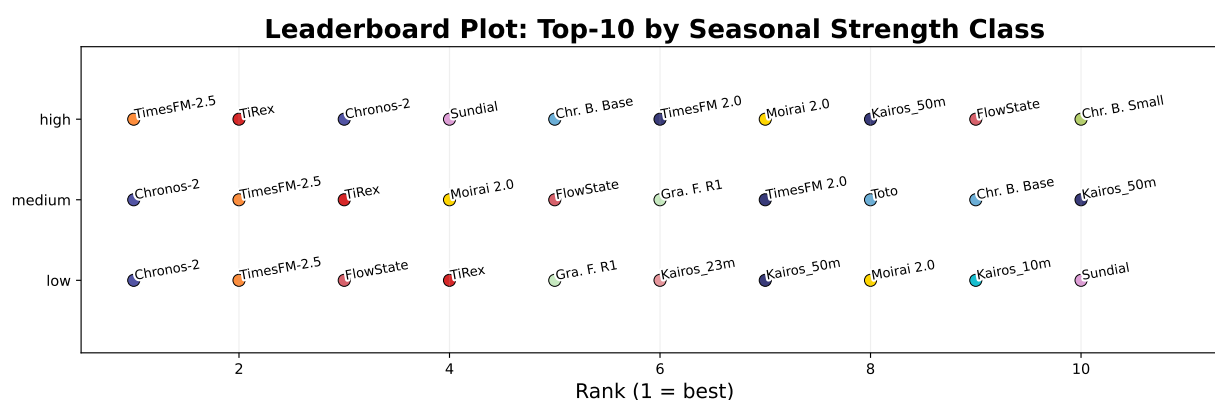

Figure 9: GIFTEVAL leaderboard results broken down by seasonal strength of the time series. High value indicates strong seasonal patterns and vice versa. We display the top-10 foundation models ordered by their MASE rank (lower is better).
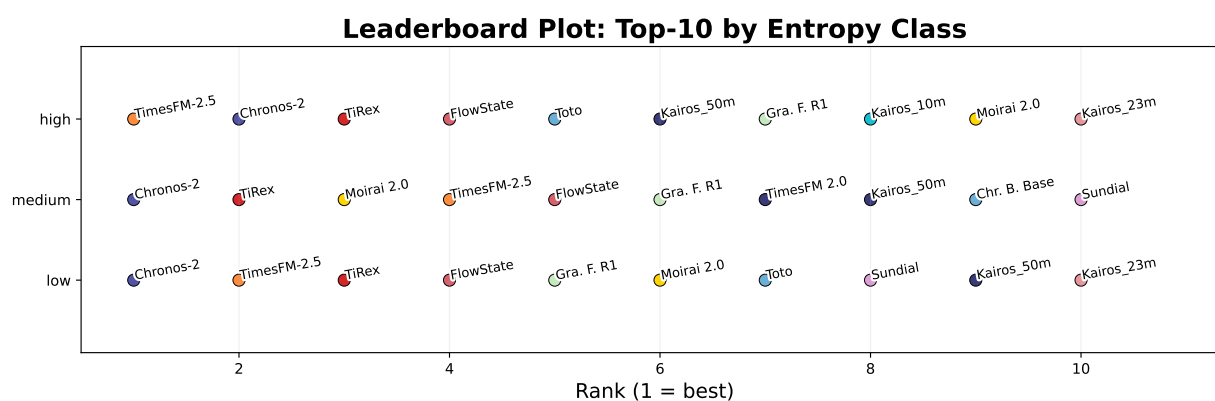

Figure 10: GIFTEVAL leaderboard results broken down by entropy of the time series. Entropy measures the "forecastability" of a time series, where low values indicate a high signal-to-noise ratio and high values occur when a series is difficult to forecast. We display the top-10 foundation models ordered by their MASE rank (lower is better).
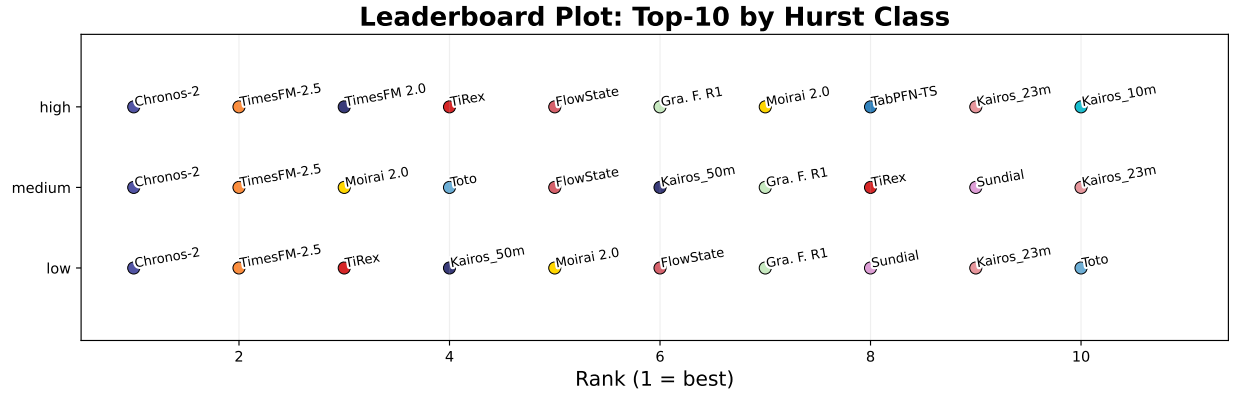
Leaderboard Plot: Top-10 by Hurst Class

Figure 11: GIFTEVAL leaderboard results broken down by hurts value of the time series. Hurst indicates the long-term memory or persistence of a time series and whether future values are likely to be influenced by past trends, revert to the mean, or behave randomly. Higher values indicate more persistence. We display the top-10 foundation models ordered by their MASE rank (lower is better).
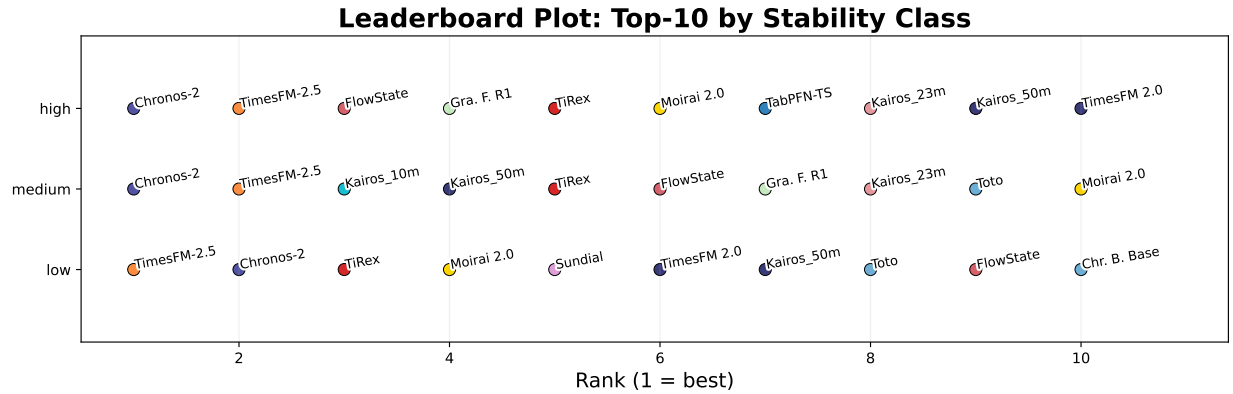
Leaderboard Plot: Top-10 by Stability Class

Figure 12: GIFTEVAL leaderboard results broken down by stability of the time series. Stability is the variance of the means. Lower values indicate more stable data. We display the top-10 foundation models ordered by their MASE rank (lower is better).
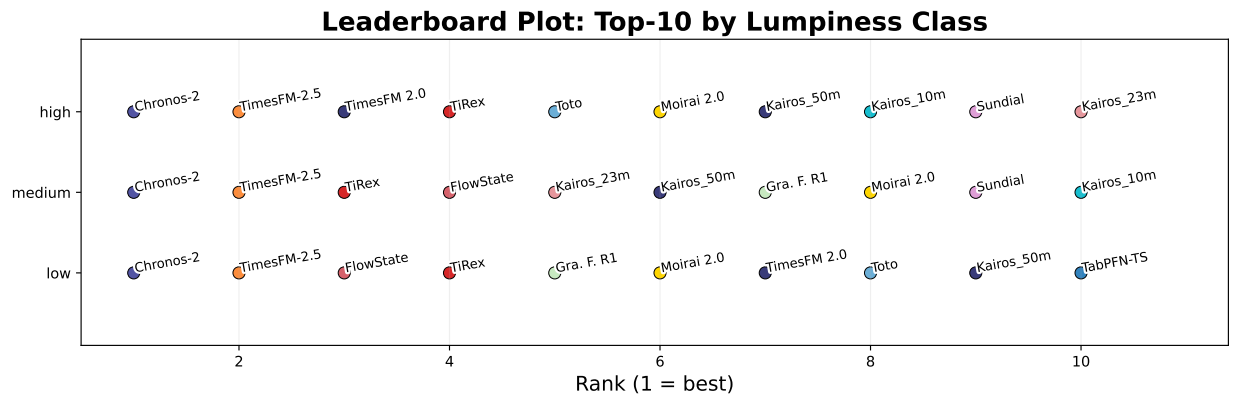
Leaderboard Plot: Top-10 by Lumpiness Class

Figure 13: GIFTEVAL leaderboard results broken down by lumpiness of the time series. Lumpiness is the variability of the variance across different segments of the time series. A high value of lumpiness indicates significant fluctuations in variability. We display the top-10 foundation models ordered by their MASE rank (lower is better).