FlowState: Sampling-Rate Invariant Time Series Foundation Model with Dynamic Forecasting Horizons

Lars Graf[*]
IBM Research Europe – Zurich
ETH Zurich / UZH Zurich

Thomas Ortner
IBM Research Europe – Zurich

Stanisław Woźniak
IBM Research Europe – Zurich

Angeliki Pantazi
IBM Research Europe – Zurich

Abstract

Foundation models (FMs) have transformed natural language processing (NLP), but their successes have not yet translated to the time series domain. Existing time series foundation models (TSFMs) struggle with generalization across varying context and target lengths, lack adaptability to different sampling rates, and are computationally inefficient. We introduce FlowState, a novel TSFM architecture that addresses these challenges through two key innovations: a state space model (SSM) based encoder and a functional basis decoder. This design enables continuous-time modeling, adjustment to various sampling rates, and flexible forecasting horizons without retraining, paving the way for a "BERT moment" for TSFM. We further propose a parallel training strategy that enhances robustness and accelerates training. Despite being the smallest model, FlowState achieves state-of-the-art results on the GIFT and the Chronos benchmarks, while demonstrating superior adaptability to unseen sampling rates.

1 Introduction

Foundation models (FMs) have revolutionized natural language processing (NLP) through pretraining on large-scale text corpora, enabling strong zero-shot generalization Bommasani et al. [2021], Hadi et al. [2023]. This "BERT moment" initiated a paradigm shift from task-specific models towards universal models, that can generalize in a zero-shot manner to numerous tasks. However, their successes have not yet translated to the time series domain. Unlike NLP, time series data is often multivariate, domain-specific, and sampled at varying rates. These differences limit the applicability of FMs from NLP Zeng et al. [2022]. Recent advances in time series modeling have favored architectures like MLP mixers Chen et al. [2023], Ekambaram et al. [2023] and state space models (SSMs) Gu et al. [2021], which better capture temporal dynamics. Based on these architectures, time series foundation models (TSFMs) have emerged Auer et al. [2025], Ansari et al. [2024], Ekambaram et al. [2024], Das et al. [2023], Liang et al. [2024], but they still struggle with generalization across varying context and target lengths, and lack adaptability to different sampling rates.

We propose **FlowState**, a novel TSFM that addresses these limitations through:

- **SSM-based encoder**: Enables efficient processing of variable-length contexts and dynamic adjustment to input sampling rates.
- **Functional Basis Decoder (FBD)**: Produces continuous forecasts that can be sampled at arbitrary resolutions, supporting variable target lengths.
- **Parallel forecasting pretraining scheme**: Speeds up training and improves generalization to varying context lengths.

NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models (BERT^{2}S).
[*] Correspondance to Lars.Graf1@ibm.com

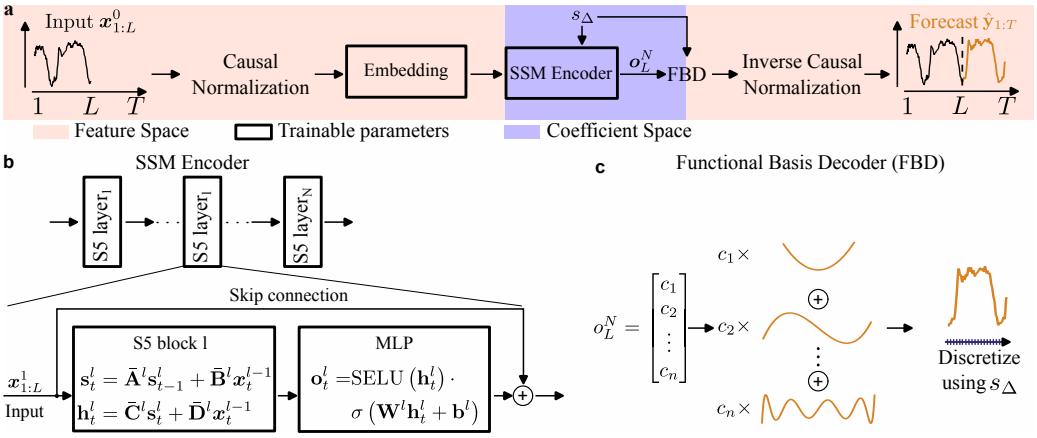

Figure 1: **Architecture overview. a** The input gets normalized, embedded and processed by the SSM encoder. The encoder transforms the input into the coefficient space and provides its outputs to the FBD, to produce the forecast. **b** The SSM encoder consists of N S5 layers, each composed of an S5 block extended with an MLP layer, and a skip to allow inputs to propagate directly to later layers. **c** The FBD interprets the outputs $\mathbf{o}^l_t$ of the SSM encoder as coefficients of a functional basis and creates a continuous output, which can be sampled at regular intervals to produce the forecast.

2 Time Series Foundation Models

Traditionally, data for time series forecasting has been processed with classic machine learning models, such as the ARIMA model Box et al. [2015], which to this day still presents a strong baseline. In such a task, the model receives an input time series $\mathbf{X} \in \mathbb{R}^{L \times c} = \{\mathbf{x}_1, ..., \mathbf{x}_L\} = \mathbf{x}_{1:L}$, where $\mathbf{x}_t \in \mathbb{R}^c$ is the c-channel multivariate time series at timestep t and L is the context length. Given this input data, the task of the model is to produce a forecast for the proceeding T timesteps, i.e., to produce $\hat{\mathbf{Y}} \in \mathbb{R}^{T \times c} = \{\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_T\} = \hat{\mathbf{y}}_{1:T} = \mathbf{x}_{L+1:L+T}$, where T is the forecasting length. The quality of the forecast can be measured by comparing it against the ground truth $\mathbf{Y} \in \mathbb{R}^{T \times c}$.

The time series domain has also experienced a shift towards FMs, which are typically trained on a large pretrain corpus of univariate ($c = 1$) time series, extract essential foundational knowledge from it and generalize this knowledge to various downstream tasks in a zero-shot manner. These TSFMs typically require deep learning architectures and transformers Nie et al. [2023], Liu et al. [2024], MLP-mixer Chen et al. [2023], Ekambaram et al. [2023] or recently also SSM Wang et al. [2025], Gu et al. [2022], Smith et al. [2023], Gu and Dao [2023], Dao and Gu [2024] and other stateful architectures such as TiRex Auer et al. [2025], representing the current state-of-the-art, have emerged.

In particular, TiRex introduced a series of novel concepts, such as a Multi-Patch-Inference (MPI) process and contiguous patch masking (CPM) for training, which significantly improved performance. The authors also showed that the ability to keep track of states and dynamics of the underlying time series is critical, making stateful models especially well-suited for time series tasks.

3 FlowState

Our proposed model, FlowState, is an encoder-decoder architecture, employing an S5-based encoder and a functional basis decoder. Figure 1a shows an overview of its architecture. The input time series with length L is first normalized in a causal manner.

This causality is critical, to enable a novel parallel prediction technique we employ during pretraining. To make optimal use of all intermediate results during pretraining, we predict a unique target from each hidden state of the SSM Encoder in parallel. This also speeds up training, and increases robustness to varying context lengths. This parallel prediction process is further detailed in Appendix A.5.

After normalization, the inputs are embedded linearly and then provided to the SSM encoder directly without any patching, see Section 3.1 for more details. Importantly, while the time series, before being processed by the SSM, is considered to be in the feature space, where each element of it

represents features of the time series, the SSM encodes the time series into a coefficient space, where it operates on coefficients of continuous basis functions. The final output of the SSM encoder forms the basis for the FBD, see Section 3.2 for details, whose outputs are then inverse normalized, using the inverse method of the input normalization, and form the forecasts of our model. Importantly, the FBD maps from the coefficient space back to the feature space to provide the forecasts. Furthermore, the SSM encoder, as well as the FBD are controlled by an additional scaling factor s_Δ, that allows to adjust these components to the sampling rate of the input data. Additional model details can be found in the Appendix A.

3.1 SSM Encoder

FlowState utilizes a stack of S5 layers to form the SSM encoder, see Figure 1b. $\bar{\mathbf{A}}^l \in \mathbb{R}^{n\times n}$, $\bar{\mathbf{B}}^l \in \mathbb{R}^{1\times n}$, $\bar{\mathbf{C}}^l \in \mathbb{R}^{n\times m}$ and $\bar{\mathbf{D}}^l \in \mathbb{R}^{m\times m}$ are the state transition, the input, the output and the skip connections matrices of layer l, m and n are the hidden state size and the in-/output size of the SSM block and $\mathbf{s}_t^l$ and $\mathbf{h}_t^l$ are the state and the output of the SSM block at timestep t. Note that the input is denoted as $\boldsymbol{x}_t^0$. As reported in Smith et al. [2023], the matrices of the S5 block l can be computed as

$$\bar{\mathbf{A}}^l = e^{\text{diag}(\mathbf{A}^l\cdot\Delta)}, \bar{\mathbf{B}}^l = \mathbf{A}^{l^{-1}}\left(\bar{\mathbf{A}}^l - 1\right)\mathbf{B}^l, \bar{\mathbf{C}}^l = \mathbf{C}^l, \bar{\mathbf{D}}^l = \mathbf{D}^l,$$

where $\mathbf{A}^l \in \mathbb{R}^n$, $\mathbf{B}^l \in \mathbb{R}^{1\times n}$, $\mathbf{C}^l \in \mathbb{R}^{n\times m}$, $\mathbf{D}^l \in \mathbb{R}^{1\times m}$, and $\Delta \in \mathbb{R}^n$ are the actual trainable parameters of the continuous S5 and initialized using the HiPPO method Gu et al. [2023]. Subsequently, the output of each SSM block is further processed by an MLP layer, see Figure 1b.

The S5 architecture can naturally be adjusted to a change of the input sampling rate Smith et al. [2023] without the need for retraining. By adapting the quantization parameter Δ, the SSM can produce similar representations, irrespective of the sampling rate. While this effect can be beneficial for classification or regression tasks Smith et al. [2023], it is insufficient for time series forecasting tasks. In particular, current decoders cannot distinguish those similar representations, hence they cannot adjust the forecasting sampling rate properly. To remedy this issue, we propose a novel decoder.

3.2 Functional Basis Decoder

For the functional basis decoder, we take inspiration from how SSMs are initialized from an input sequence. The HiPPO approach ensures that their hidden state expresses coefficients of a polynomial basis, which optimally approximates the input sequence. In particular, Gu et al. [2020] demonstrated a possibility to use the hidden state of their SSM at timestep t to reconstruct the input sequence until t with a functional basis. We adopt this approach for our decoder, but instead of extracting coefficients that can be used to reconstruct the input, we use a continuous functional basis to construct the forecast from the final outputs of the SSM encoder $\mathbf{o}_L^N$, see Figure 1c. In particular, our proposed FBD interprets the final outputs of the SSM encoder, $\mathbf{o}_L^N$, as coefficients of a functional basis, which can in turn be used to produce a continuous output function. To obtain the forecast with a desired quantization, this continuous output is then sampled at an equally spaced interval, with the spacing adjusted with s_Δ. The FBD can be formalized as follows:

$$c_i = o_{L,i}^N, \quad \tilde{\boldsymbol{y}} = \sum_{i=1}^{n} c_i p_i(a,b), \quad \hat{\boldsymbol{y}} = \text{sample}(\tilde{\boldsymbol{y}}, s_\Delta), \tag{1}$$

where $p_i(\cdot,\cdot)$ is the i-th basis function evaluated at an interval $[a,b]$, $\tilde{\boldsymbol{y}}$ is the continuous forecast and $\text{sample}(\cdot, s_\Delta)$ samples the argument equally.

Our functional basis decoder offers several key advantages. Firstly, it produces a continuous forecast, which can then be sampled with any desired sampling rate. Secondly, it draws inspiration from a well-established procedure to map from coefficient to feature space, and thus can leverage various functional basis functions, depending on the task. For our main experiments, we use the Legendre polynomials to be consistent with the SSM input encoding used by the HiPPO initialization. Another viable option is to use the Fourier basis functions to better match periodic signals. Finally, and most importantly, it enables the decoder to produce forecasts at the correct sampling rate, based on s_Δ. Note that although we introduce the FBD as part of FlowState, it can be combined with other encoder architectures as well.

Model	Param.[M]	MASE	CRPS
FlowState-9.1M	9.1	**0.726**	**0.502**
FlowState-2.6M	2.6	0.735	0.508
Toto-Open-Base-1.0	151	0.750	0.517
Sundial-Base	128	0.750	0.559
TabPFN-TS	11	0.771	0.544
YingLong	300	0.798	0.548
YingLong	110	0.809	0.557

Table 1: GIFT results for zero-shot models, sorted by ascending MASE.

Model	Param.[M]	MASE	WQL
FlowState(T)-9.1M	9.1	**0.755**	**0.580**
FlowState(T)-2.6M	2.6	0.777	0.614
TiRex	35	0.778	0.599
TimesFM-2.0	500	0.789	0.700
Moirai-l	311	0.791	0.631
Chronos-bolt-b	205	0.795	0.624
Chronos-b	200	0.818	0.643

Table 2: Chronos results, sorted by ascending MASE.

4 Results

We evaluate FlowState's forecasting capabilities on two commonly used benchmarks: GIFT-Eval[1] Aksu et al. [2024], referred to as **GIFT**, and the Chronos benchmark (II)[2] Ansari et al. [2024], referred to as **Chronos**.

As discussed before, a critical component of TSFMs is their zero-shot capability. Therefore, we carefully ensured that FlowState does not have any testdata leakage and performs zero-shot forecasting on the two benchmarks. Therefore, we pretrain FlowState as a TSFM in two sizes, 9.1M and 2.6M parameters. In particular, we train FlowState on the GIFT-Eval-Pretrain dataset, augmented with some data from Chronos Ansari et al. [2024] that do not have any overlap with the GIFT benchmark. Moreover, we train another variant, FlowState(T), with the exact same data as TiRex in order to avoid testdata leakage with the Chronos benchmark. More details about pretraining can be found in Appendix A.1. In the following we describe the results for the two benchmarks as of 28^{th} August 2025 and additional results, including ablations and experiments evaluating performance on unseen sampling rates, can be found in Appendix B.

GIFT benchmark. Table 1 shows the normalized MASE and CRPS metrics for the GIFT-Eval benchmark, with ascending MASE score. FlowState-9.1M and FlowState-2.6M surpass all previously reported baselines. Notably, the FlowState-2.6M variant is by far the smallest model, yet outperforms all other baselines.

Chronos benchmark. Table 2 shows the normalized MASE and WQL metrics for the Chronos benchmark, with ascending MASE score. Similarly to the GIFT results, the FlowState(T) variants outperform the current state-of-the-art models. Importantly, although FlowState(T) has been trained on the same data as TiRex, it performs significantly better.

5 Conclusion

We introduce FlowState, a time series foundation model, that can dynamically adjust to the unique characteristics of the input time series, such as its specific sampling rate. To do so, we developed a functional basis decoder (FBD), a novel component that leverages a set of basis functions to create continuous forecasts. FlowState's combination of an SSM-based encoder and the FBD enables the seamless adjustment and the production of forecasts of varying lengths. To further enhance FlowState's efficiency and robustness, we propose a training scheme that through multiple parallel predictions exposes the model to diverse context lengths during training. FlowState establishes the new state-of-the-art on the GIFT and the Chronos benchmarks, outperforming strong baselines that are up to 192× larger. Finally, FlowState demonstrates superior robustness and adaptability to unseen sampling rates and our ablation studies confirm the individual and collective benefits of our proposed components. Thus, FlowState demonstrates progress toward a "BERT moment" for TSFMs.

[1] `https://huggingface.co/spaces/Salesforce/GIFT-Eval`

[2] `https://huggingface.co/spaces/autogluon/fev-leaderboard`

Acknowledgments and Disclosure of Funding

This work has been partly funded by the EU under the Horizon Europe (HE) programme through the CloudSkin project (Grant No. 101092646).

References

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.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the Opportunities and Risks of Foundation Models. *arXiv*, August 2021. doi: 10.48550/arXiv.2108.07258.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv*, October 2023. doi: 10.48550/arXiv.2310.10688.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 459–469, New York, NY, USA, 2023. Association for Computing Machinery. ISBN

9798400701030. doi: 10.1145/3580305.3599533. URL https://doi.org/10.1145/3580305.3599533.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. 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. *arXiv*, January 2024. doi: 10.48550/arXiv.2401.03955.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Albert Gu, Isys Johnson, Karan Goel, Khaled Kamal Saab, Tri Dao, Atri Rudra, and Christopher Re. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=yWd42CWN3c.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Re. How to train your HIPPO: State space models with generalized orthogonal basis projections. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=klK17OQ3KB.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

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.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6555–6565, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671451. URL https://doi.org/10.1145/3637528.3671451.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

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*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is mamba effective for time series forecasting? *Neurocomputing*, 619:129178, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2024.129178. URL https://www.sciencedirect.com/science/article/pii/S0925231224019490.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? *arXiv*, May 2022. doi: 10.48550/arXiv.2205.13504.

A Additional details

A.1 Pretraining

We pretrain two distinct variants of FlowState:

- **FlowState**: The pretraining data for this variant is carefully designed to have no overlap with the GIFT-Eval benchmark. It consists of the official GIFT-Eval pretraining dataset, as well as of some training datasets from Chronos, that are not overlapping with the GIFT-Eval benchmark. In addition, we added synthetic time series data generated via Gaussian Processes, following the methodology of KernelSynth Ansari et al. [2024]. Moreover, we enhance the diversity of the entire pretraining dataset with augmentation techniques introduced in Auer et al. [2025].
- **FlowState(T)**: The pretraining data for this variant is carefully designed to have no overlap with the Chronos benchmark. We use the same pretraining corpus as TiRex, except for the synthetic data, which we generate in the same way as for FlowState.

To enable probabilistic forecasting, we optimize the model using the quantile loss. For all models, except one ablation, CPM is used during pretraining, which allows for MPI.

A.2 Metrics

Both benchmarks use a probabilistic evaluation metric based on the Weighted Quantile Loss (**WQL**). While GIFT-Eval refers to this metric as **CRPS**, the underlying formulation is equivalent, as WQL can be interpreted as a quantile-based approximation of the CRPS.

For each time series, we generate probabilistic forecasts and compute two normalized metrics: Mean Absolute Scaled Error (**MASE**) measures point forecast accuracy, while CRPS and WQL measure the probabilistic forecast accuracy. Both metrics are normalized per task using a Seasonal Naive baseline. Final scores are reported as the geometric mean across tasks. For consistency with the leaderboards, we refer to the normalized probabilistic metric as CRPS in GIFT and WQL in Chronos. The normalized and averaged MASE are simply referred to as MASE.

A.3 Temporal Scaling

FlowState's continuous-time formulation introduces two key considerations during evaluation. First, we determine a suitable *scale factor* for each dataset. Since datasets vary in both sampling rates and temporal dynamics, we base this factor on *seasonality* rather than raw sampling frequency. For example, hourly temperature data typically exhibits a 24-step daily cycle, while weekly peak temperatures follow a seasonal pattern of $365/7 \approx 52$ steps. Even though a week contains 168 hours, a more appropriate scale factor between these two examples is determined by the ratio of their relative seasonality. We define a base seasonality of 24 and compute the scale factor as:

$$s_\Delta = \frac{\text{Base Seasonality}}{\text{Seasonality}}$$

This ensures that all datasets are mapped to a common temporal scale in the model's continuous space. A scale factor of 1 corresponds to seasonality 24.

A.4 Context and Forecasting Length

FlowState's architecture enables flexible adaptation of both context and forecasting lengths across datasets with diverse temporal resolutions. Unlike discrete models, where these lengths are fixed, FlowState operates in a scale-adjusted latent space, representing continuous signals. To maintain consistency with pretraining, we define effective lengths which the model actually uses, relative to the scale factor s_Δ, corresponding to the pretraining context length of 2048 steps, and to the base forecasting length $T = 24$ (one season). Depending on s_Δ these effective lengths will change. In particular, the effective context length L_{eff} and forecasting length T_{eff} are computed as:

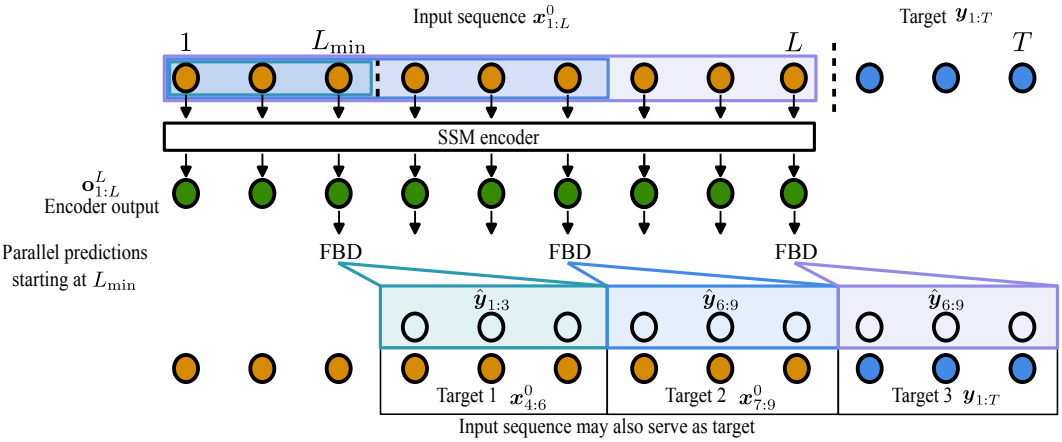

Figure 2: **Schematic illustration of our parallel prediction training scheme.** The input sequence $\mathbf{x}^0_{1:T}$ is encoded in parallel using the SSM encoder. Starting from L_{min}, multiple forecasts are produced in parallel, where each forecast has its own target and is based on an increasing context length. In particular, a prediction is made for every timestep after L_{min}, but for clarity only three are shown. For example, for the first forecast (green color) only the first three timesteps $\mathbf{x}^0_{1,2,3}$ are used as the context, while for the last prediction (purple color) the full context $\mathbf{x}^0_{1:T}$, is used. Note that for some of the forecasts the input sequence itself serves as the target and thus a causal processing of the input is essential to avoid information leakage.

$$L_{\text{eff}} = \frac{L}{s_\Delta}, \quad T_{\text{eff}} = \frac{T}{s_\Delta}$$

This formulation is particularly beneficial for datasets with large seasonality, which typically require longer historical context and benefit from extended forecasting horizons. As seasonality increases, s_Δ decreases, resulting in larger L_{eff} and T_{eff}. This allows FlowState to forecast far into the future for such datasets—precisely where long-range predictions are often most valuable.

A.5 Parallel prediction details

To enable efficient training of FlowState and to enhance its robustness to varying context lengths, we introduce an advanced foundation training scheme. In particular, it utilizes multiple parallel forecasts with increasingly longer contexts, ranging from L_{min} to L, see Figure 2.

These various forecasts can be formulated as

$$\hat{\boldsymbol{y}}_{t+1:t+T} = \text{FBD}\left(\text{SSM}\left(\boldsymbol{x}^0_{1:t}\right)\right), \tag{2}$$

where $t \in [L_{\text{min}}, L]$. Importantly, because FlowState is an SSM-based architecture, the inputs can be processed in parallel and in turn also the multiple forecasts can be produced in parallel. Thus, this approach allows to produce $(L - L_{\text{min}})$ forecasts from any given context-target pair, while other models traditionally only produce a single forecast per context-target pair. Depending on the choice of L and L_{min}, this amount can be very large, for example for $L = 2048$ and $L_{\text{min}} = 20$, as was used for our main results, FlowState produces 2028 forecasts per sample in parallel.

The benefits of this training scheme can either materialize in significantly reduced training times, because one can iterate through the dataset faster using a larger stride to the next context-target pair, or in an increased number of training examples, when the original stride to the next context-target pair is kept constant. Another advantage of this training procedure is that the model inherently learns to produce forecasts from varying context lengths. This naturally increases the models' generalization capabilities and robustness. Note that our novel training scheme is not limited to the FlowState architecture but can be applied to any causal architecture that can produce multiple forecasts in parallel. A non-causal model is not compatible with this parallel prediction approach, because it could use information from future inputs. This is further detailed for the example of global normalization in Section A.7.

Note, parallel forecasting can also speed-up inference if combined with MPI.

A.6 Adjusting Δ for unseen sampling rates

As described in the main text, the dynamics of the SSM encoder and the FBD can be adjusted to the input sampling rate by modifying Δ. Typically, Δ is only adjusted, as a trainable parameter, during training, while during inference, the parameter remains fixed. In order to enable adjustments to the sampling rates during inference, we modify the parameter Δ with an additional scaling parameter s_Δ, in particular:

$$\bar{\Delta} = f(\Delta, s_\Delta) = s_\Delta \cdot \Delta. \tag{3}$$

The adaptation of the parameter Δ by multiplication with the correct scale factor s_Δ is crucial, because it is used to discretize the continuous SSM for a given sampling rate.

A.7 Causal normalization

For parallel forecasting to work properly an architecture has to be strictly causal. Otherwise, the model may learn to exploit this information leakage during training. In particular, the prediction $\hat{\boldsymbol{y}}_{t+1:t+T}$ should only use the data from $\boldsymbol{x}^0_{\leq t}$. The SSM and FBD of FlowState naturally satisfy this requirement, but the commonly applied normalization and inverse normalization technique, RevIN Kim et al. [2022], would violate it.

RevIN normalizes every context-target pair, based on the mean and the standard deviation of the entire context $\boldsymbol{x}^0_{1:L}$, see Eq. 2 of Kim et al. [2022]. However, this would result in information from $\boldsymbol{x}^0_{>t}$ to influence $\hat{\boldsymbol{y}}_{t+1:t+T}$. For example, if the average of the normalized sequence $\mu_t = \tilde{\boldsymbol{x}}^0_{1:t}$ is negative, the model will learn that positive values are to be expected for $\tilde{\boldsymbol{x}}^0_{>t}$, because, per definition, RevIN produces a zero mean for the whole time series.

To address this problem, we use a causal form of RevIN. Specifically, instead of using the average and standard deviation of the entire context to normalize, we leverage a running mean and a running standard deviation. Each element of the input at time t is then normalized using these quantities at time t. This can be formulated as

$$\mu_{r,t} = \frac{\text{cumsum}\left(\boldsymbol{x}^0_{1:t}\right)}{t} \tag{4}$$

$$\sigma^2_{r,t} = \frac{\text{cumsum}\left(\left(\mu_{r,t} - \boldsymbol{x}^0_{1:t}\right)^2\right)}{t} \tag{5}$$

$$\tilde{\boldsymbol{x}}^0_{1:t} = \frac{\boldsymbol{x}^0_{1:t} - \mu_{r,t}}{\sigma_{r,t}}, \tag{6}$$

where $\text{cumsum}(\cdot)$ is the cumulative sum function.

Similarly, each forecast of the FBD is de-normalized by the statistics of the last timestep of the context. For example, $\hat{\boldsymbol{y}}_{t:t+T}$ is de-normalized with $\mu_{r,t}$ and $\sigma_{r,t}$.

A.8 Autoregressive Forecasting and Multi-Patch-Inference

Typically TSFMs use autoregressive techniques to extend their forecasting horizon. Namely, they produce several shorter forecasts of size $p < T$ sequentially, always appending their forecast to the original context. Auer et al. [2025] have improved this autoregressive technique with MPI. In particular, they adopt contiguous patch masking (CPM) during training, which accustoms the model to make a prediction after a certain number of unknown timesteps. This setup enables MPI to forecast future patches by treating intermediate ones as missing. The main advantage of MPI / CPM over autoregressive forecasting is the increased model's robustness to noise and uncertain data, as well as the ability to propagate uncertainty over multiple forecasting patches.

A.9 Hyperparameters

Model hyperparameters can be found in Table 3. Note that no extensive hyperparameter search has been performed yet.

Component	FlowState-2.6M	FlowState-9.1M
# Layers	6	6
Hidden Size	256	512
State Dimension	256	512
# Basis Functions	256	256

Table 3: Model hyperparameters for FlowState variants.

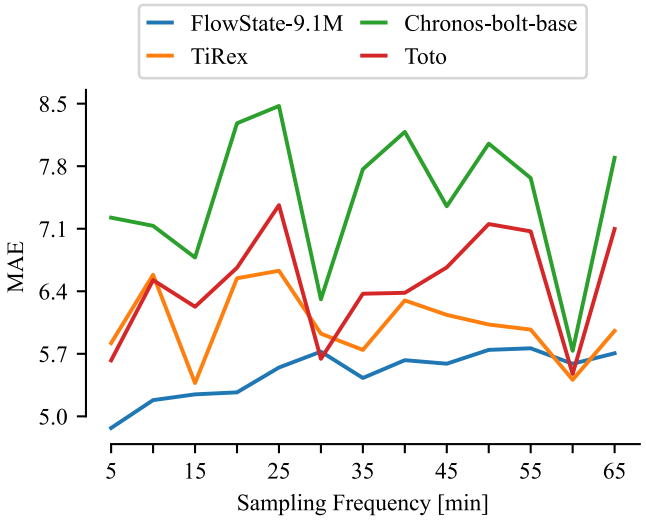

Figure 3: MAE performance across various sampling frequencies on the Loop Seattle dataset.

B Additional results

B.1 Robustness to Unseen Sampling Rates

To assess the robustness of FlowState to varying temporal resolutions, we conduct a controlled experiment on the Loop Seattle dataset. Originally sampled at 5-minute intervals, we subsample the data to create versions with sampling intervals ranging from 5 to 65 minutes in 5-minute increments. We then evaluate FlowState-9.1M, TiRex, Chronos-bolt-b, and Toto on each version using the standard GIFT-Eval evaluation framework and a target length of 480 timesteps.

Figure 3 shows the MAE of all models for each sampling frequency. FlowState-9.1M consistently outperforms all baselines across most frequencies, with a particularly large margin at uncommon sampling intervals. The only exceptions are at 15T, 30T, and 60T—common frequencies likely seen during pretraining—where baseline models briefly close the gap. These results highlight FlowState's ability to generalize to unseen sampling rates without requiring exposure to every possible frequency during training.

Loop Seattle was selected because none of the baselines use it during pretraining, and due to its small original sampling rate, and long time series, making it well-suited for controlled subsampling experiments.

B.2 Ablation Study

To understand the contributions of individual components in FlowState, we conduct a series of ablations by retraining variations of the FlowState-2.6M model and evaluating on GIFT. The results are summarized in Table 4, and organized into the following categories:

Core Architectural Components. This group isolates the impact of FlowState's key design choices. Removing the time-scale adjustment mechanism leads to a significant drop in performance, confirming its importance for generalization across sampling rates. Disabling parallel predictions, by always only

Model Variant	MASE	CRPS
FlowState-2.6M (baseline)	**0.735**	**0.508**
Core Architectural Components		
w/o time-scale adjustment	0.796	0.548
w/o parallel predictions	0.761	0.531
Decoder Variants		
Fourier basis	0.737	0.511
Half-Legendre basis	0.737	0.511
CPM		
Autoregressive instead of MPI	0.749	0.541
No CPM training, longer target	0.739	0.526

Table 4: Ablation results for FlowState-2.6M evaluated on GIFT.

predicting from the last context point, also significantly degrades performance, though to a lesser extent.

Decoder Variants. We evaluate two alternative sets of basis functions to the default Legendre basis: a Fourier basis, and a Legendre basis defined over the interval $[0, 1]$ instead of $[-1, 1]$ (referred to as "Half-Legendre"). Both sets of basis functions have performed slightly worse compared to FlowState-2.6M.

CPM Mechanism. We assess the impact of the CPM / MPI mechanism by evaluating FlowState-2.6M, which was trained with CPM, autoregressively, instead of using MPI. This leads to a substantial performance drop. However, when we retrain the model without CPM, and using a longer base target length of 60 instead of 24, performance recovers. Both MASE and CRPS remain worse, but CRPS to a larger degree, highlighting the importance of MPI for accuracy of long probabilistic predictions.