# FlowState: Sampling Rate Invariant Time Series Forecasting

**Lars Graf**[1, 2], **Thomas Ortner**[1], **Stanisław Woźniak**[1], **Angeliki Pantazi**[1]

[1]IBM Research Europe – Zurich, Switzerland
[2]ETH Zurich / UZH Zurich, Switzerland



## Abstract

Foundation models (FMs) have transformed natural language processing, but their success has not yet translated to time series forecasting. Existing time series foundation models (TSFMs), often based on transformer variants, struggle with generalization across varying context and target lengths, lack adaptability to different sampling rates, and are computationally inefficient. We introduce FlowState[1], 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 and dynamic time-scale adjustment, allowing FlowState to inherently generalize across all possible temporal resolutions, and dynamically adjust the forecasting horizons. In contrast to other state-of-the-art TSFMs, which require training data across all possible sampling rates to memorize patterns at each scale, FlowState inherently adapts its internal dynamics to the input scale, enabling smaller models, reduced data requirements, and improved efficiency. We further propose an efficient pretraining strategy that improves robustness and accelerates training. Despite being the smallest model, FlowState outperforms all other models and is state-of-the-art for the GIFT-ZS and the Chronos-ZS benchmarks. Ablation studies confirm the effectiveness of its components, and we demonstrate its unique ability to adapt online to varying input sampling rates.

## Introduction

Machine learning (ML) models are ubiquitously found in many aspects of our daily lives. Especially foundation models (FMs), have received extensive research interest and are today employed in various natural language processing (NLP) tasks, such as text summarization, text generation or information retrieval from large, unstructured text databases (Hadi et al. 2023). The astounding capabilities of FMs are believed to arise in part from their specific training process, in which the FMs are trained on a vast collection of text-based data available on the public internet. Through this approach, the model can learn the underlying foundational principles of the data. Thus, we refer to this training procedure as the foundation model approach. Leveraging this extracted information, the model can then address various unseen downstream tasks, i.e., zero-shot generalization (Bommasani et al. 2021).

[1]Currently under review

Despite their astonishing performance in NLP, FMs struggle to be applied to other domains, such as time series processing. NLP and time series processing are both sequence processing tasks, but with major differences. In particular, a token in NLP carries substantially more information than an individual data point in time series. Furthermore, time series data can be multivariate and vary strongly within and especially across domains, e.g., the electrical power consumption of a city may look entirely differently than a stock price.

Therefore, other model capabilities compared to the NLP domain are required, which resulted in different model architectures to emerge as state-of-the-art (SOTA). For example, while FMs for NLP have so far undoubtedly been dominated by the transformer architecture, the same architecture is performing poorly in time series tasks (Zeng et al. 2022). Researchers have uncovered better architectures, based on linear layers that mix over time and features (Chen et al. 2023; Ekambaram et al. 2023) in an alternating manner. More recently, state space models (SSMs) (Gu et al. 2021) have emerged as viable alternatives that currently represent the SOTA in several time series tasks.

The above-mentioned challenges have for a long time hindered the emergence of time series foundation models (TSFMs). Only recently, researchers have found ways to utilize the foundation model training approach also for time series data and successfully trained TSFMs (Auer et al. 2025; Ansari et al. 2024; Ekambaram et al. 2024; Das et al. 2023; Liang et al. 2024). Still, these approaches are quite limited in their usability. For example, a TSFM should be able to process time series data of a particular length—the context—and produce a forecast of a different and potentially varying length—the target. However, most TSFMs currently employ a linear decoder of a fixed size, matched to the target, to produce their forecast. This results in the models being inherently specialized to a specific target length. On top of that, TSFM should generalize well to varying context and target lengths, where current TSFMs often struggle with. Finally, current TSFMs don't have a direct way to adjust to the specific characteristics of the time series encountered during evaluation, such as a change in sampling rate.

In this work, we propose a novel TSFM, called FlowState, that addresses these shortcomings. In particular, FlowState is based on an SSM, which naturally allows to process varying context lengths. Moreover, we introduce a novel functional

basis decoder (FBD) which leverages a set of basis functions to create a continuous forecast. The combination of the SSM encoder with the FBD allows to seamlessly adjust FlowState to the specific characteristics of the context time series, most importantly, its specific sampling rate. FBD also enables the model to produce varying target lengths, without retraining. Finally, we designed a massively parallel training scheme, that concurrently trains the model on a variety of context lengths for superior generalization capabilities.

In summary, we make the following key contributions:

- FlowState: We present an SSM-based time series foundation model that can be dynamically adjusted to the specific characteristics of the time series during evaluation
- Functional basis decoder (FBD): We propose a novel decoder, as a critical component of FlowState, that utilizes a set of continuous basis functions to allow seamless adjustment to specific input characteristics and to produce forecasts for varying target lengths
- Training approach with parallel predictions: We introduce a foundation model training scheme leveraging parallel predictions to enable efficient training and model robustness to varying context lengths

# Background

## Time Series Forecasting

Time series data is omnipresent in various domains and there are several tasks that can be performed with this data, such as anomaly detection, pattern search within time series, time series forecasting, etc. Although our model could be applied to several of these tasks, we specifically focus on time series forecasting. 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 $\mathbf{Y} \in \mathbb{R}^{T \times c}$ using various metrics, such as the mean absolute error (MAE) or the mean squared error (MSE).

## Models for time series data

Traditionally time series forecasting tasks have been addressed with classic machine learning models, such as the ARIMA model (Box et al. 2015), which to this day presents a strong baseline in some cases.

The successes of the transformer architecture in the NLP domain have inspired researchers to apply them to time series data, yet vanilla transformers were outperformed by simpler architectures (Zeng et al. 2022). Combining several timesteps to patches (Nie et al. 2023), or performing the self-attention alternatingly along the time and the feature dimension (Liu et al. 2024), improved their performance over classic baselines for some datasets. However, they still struggle on certain datasets and suffer from a high computational cost. This has inspired researchers to develop simpler approaches, solely based on multi-layer perceptrons (MLPs), that outperform transformers with a smaller computational footprint (Chen et al. 2023; Ekambaram et al. 2023).

Recently, State Space Models have emerged as another viable alternative. SSMs have a long history in control theory, are successfully applied to NLP tasks, and slowly make their way into other domains as well (Liu, Zhang, and Zhang 2024; Rahman et al. 2024; Wang et al. 2025). In contrast to the transformer- and MLP-based architectures, SSMs are stateful models more similar to the classic recurrent neural networks (RNNs), which in the past also served as strong baselines in time series tasks (Siami-Namini, Tavakoli, and Namin 2019; Che et al. 2018). Researchers have developed several SSM variants for NLP tasks, such as S4 (Gu, Goel, and Re 2022), S5 (Smith, Warrington, and Linderman 2023), S6 (Gu and Dao 2023) or Mamba2 (Dao and Gu 2024), which successively enhanced the model's capabilities and performance. A main advantage of SSMs over RNNs is that while the state update in conventional RNNs, such as LSTMs or GRUs, contains nonlinearities, the state update of SSMs block is linear, and only the output of the SSM contains a nonlinear activation. Thus, SSMs can be parallelized, which leads to a reduced time complexity compared to sequential RNNs.

Lastly, TiRex (Auer et al. 2025), a stateful model based on xLSTMs (Beck et al. 2024), has emerged and became state-of-the-art in several benchmarks. In addition, it introduced a capability to deal with various forecasting lengths by applying a novel Multi-Patch-Inference (MPI) process.

**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.

# 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, because of the parallel forecasts our model carries out during training, see Section "Parallel forecasts". Afterwards, the normalized inputs are embedded linearly and then provided to the SSM encoder directly without any patching, see Section "SSM Encoder" for more details. Importantly, while the time series before being processed by the SSM are considered to be in the feature space, where each element of the input represents features of the time series, the SSM encodes this information into a coefficient space, where it operates on coefficients of

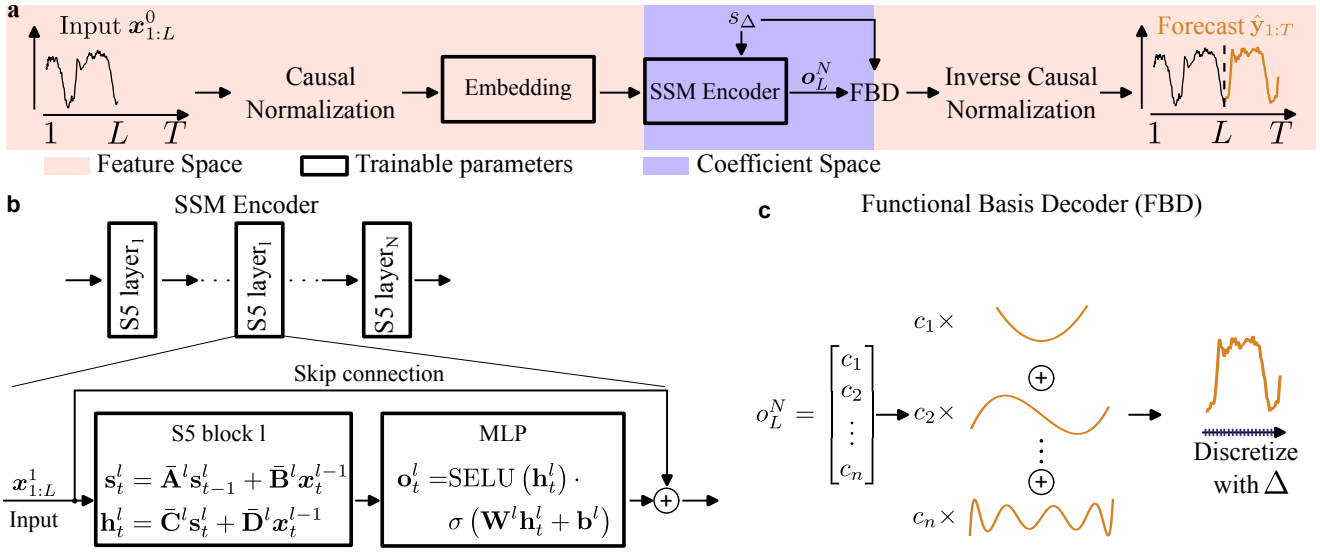


Figure 1: **Architecture overview. a** Overview of the FlowState architecture. The input context in the feature space (orange color) gets normalized, embedded and then processed by the SSM encoder. The SSM encoder transforms the input into the coefficient space (blue color) and provides the final encodings to the functional basis decoder, which then produces the final forecast. Modules with trainable parameters are highlighted in black rectangular blocks. **b** The SSM encoder consists of $N$ S5 layers, each composed of an S5 block extended with an MLP layer. A skip connection is used to allow inputs to propagate also to later encoder layers. **c** The functional basis decoder 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 $\Delta$ to produce the forecast.

continuous basis functions. The final output of the SSM encoder forms the basis for the FBD, see Section "Functional Basis Decoder" 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 $s_\Delta$, that allows to adjust these components to the sampling rate of the input data.

## SSM Encoder

FlowState utilizes a stack of S5 layers to form the SSM encoder, see Figure 1b. One S5 layer $l$ consists of an S5 block, followed by an MLP. The dynamics of the S5 block can be described through these governing equations:

$$\mathbf{s}^l_t = \bar{\mathbf{A}}^l \mathbf{s}^l_{t-1} + \bar{\mathbf{B}}^l \boldsymbol{x}^{l-1}_t \tag{1}$$

$$\mathbf{h}^l_t = \bar{\mathbf{C}}^l \mathbf{s}^l_t + \bar{\mathbf{D}}^l \boldsymbol{x}^{l-1}_t \tag{2}$$

where $\bar{\mathbf{A}}^l \in \mathbb{R}^{P \times P}$, $\bar{\mathbf{B}}^l \in \mathbb{R}^{c \times P}$, $\bar{\mathbf{C}}^l \in \mathbb{R}^{P \times H}$ and $\bar{\mathbf{D}}^l \in \mathbb{R}^{c \times H}$ 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 output size of the SSM block and $\mathbf{s}^l_t$ and $\mathbf{h}^l_t$ are the state and the output of the SSM block at timestep $t$. Note that the input is denoted as $\boldsymbol{x}^0_t$. As reported in (Smith, Warrington, and Linderman 2023), the matrices of the S5 block $l$ can be computed as

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

where $\mathbf{A}^l \in \mathbb{R}^{P \times P}$, $\mathbf{B}^l \in \mathbb{R}^{c \times P}$, $\mathbf{C}^l \in \mathbb{R}^{P \times H}$ and $\mathbf{D}^l \in \mathbb{R}^{c \times H}$ are the actual trainable parameters of the S5 and initialized using the HiPPO method (Gu et al. 2023).

In contrast to several other state-of-the-art works (Auer et al. 2025; Ekambaram et al. 2024; Ansari et al. 2024; Cohen et al. 2024), FlowState processes the time series as-is and does not perform any quantization or patching.

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, Warrington, and Linderman 2023). For example, an S5 model pretrained on data with a particular sampling rate can generalize without retraining to data with a different sampling rate. By supplying a different quantization parameter $\Delta$, the SSM can produce similar representations, irrespective of the sampling rate. While this effect can be beneficial for classification or regression tasks (Smith, Warrington, and Linderman 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.

## 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 $\Delta$, this continuous output is then sampled at an equally spaced interval, with the spacing $\Delta$. The FBD can be formalized as follows:

$$c_i = o_{L,i}^N \tag{3}$$

$$\tilde{\boldsymbol{y}} = \sum_{i=1}^{n} c_i p_i(a, b) \tag{4}$$

$$\hat{\boldsymbol{y}} = \text{sample}(\tilde{\boldsymbol{y}}, \Delta), \tag{5}$$

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 $\text{sample}(\cdot, \Delta)$ samples the argument equally spaced with $\Delta$.

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 the current parameter $\Delta$. Note that although we introduce the FBD as part of FlowState, it is a separate component and can be combined with other encoder architectures as well.

### Adjusting $\Delta$ for unseen sampling rates

As described above, the dynamics of the SSM encoder and the FBD can be adjusted to the input sampling rate by modifying $\Delta$. Typically, $\Delta$ 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 $\Delta$ with an additional scaling parameter $s_\Delta$, in particular:

$$\bar{\Delta} = f(\Delta, s_\Delta) = s_\Delta \cdot \Delta. \tag{6}$$

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

## Parallel forecasts

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 $L_{\text{min}}$ to $L$, see Figure 2.

These various forecasts can be formulated as

$$\hat{\boldsymbol{y}}_{t+1:t+T} = \text{FBD}\left(\text{SSM}\left(\boldsymbol{x}_{1:t}^0\right)\right), \tag{7}$$

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 $L_{\text{min}}$, 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.

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

### 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 $\boldsymbol{x}_{\leq t}^0$. 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 $\boldsymbol{x}_{1:L}^0$, see Eq. 2 of (Kim et al. 2022). However, this would result in information from $\boldsymbol{x}_{>t}^0$ to influence $\hat{\boldsymbol{y}}_{t+1:t+T}$. For example, if the average of the normalized sequence $\mu_t = \tilde{\boldsymbol{x}}_{1:t}^0$ is negative, the model will learn that positive values are to be expected for $\tilde{\boldsymbol{x}}_{>t}^0$, 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

$$\mu_{r,t} = \frac{\text{cumsum}\left(\boldsymbol{x}_{1:t}^0\right)}{t} \tag{8}$$

$$\sigma_{r,t}^2 = \frac{\text{cumsum}\left(\left(\mu_{r,t} - \boldsymbol{x}_{1:t}^0\right)^2\right)}{t} \tag{9}$$

$$\tilde{\boldsymbol{x}}_{1:t}^0 = \frac{\boldsymbol{x}_{1:t}^0 - \mu_{r,t}}{\sigma_{r,t}}, \tag{10}$$

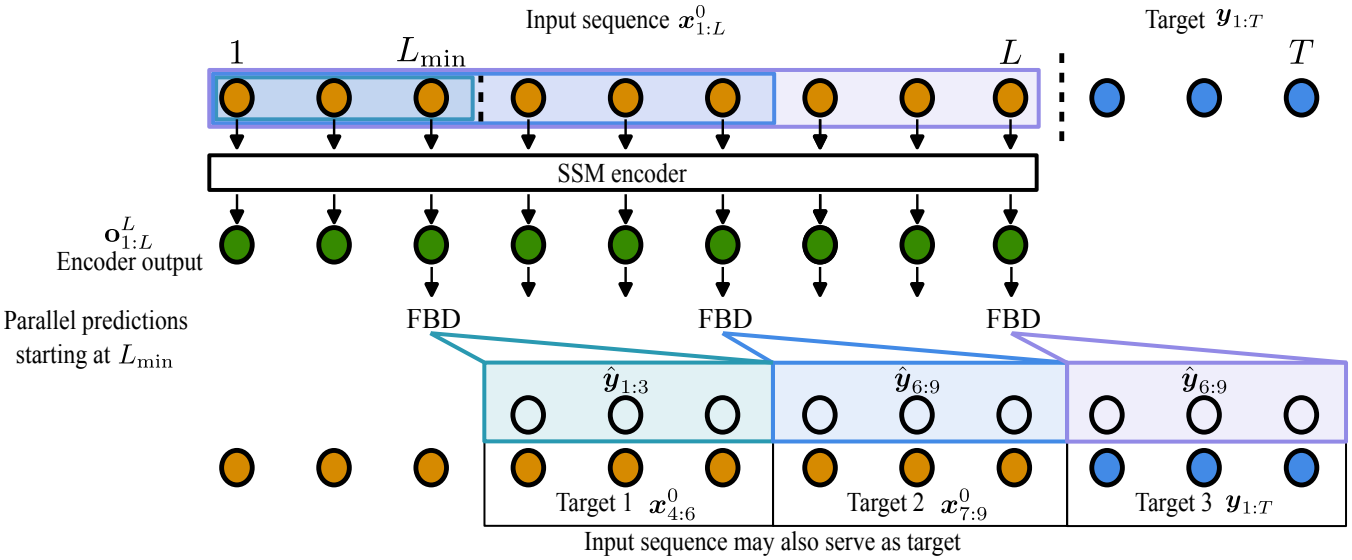


Figure 2: **Schematic illustration of our parallel prediction training scheme.** The input sequence $\mathbf{x}^0_{1:T}$ is encoded in parallel using the SSM encoder. Starting from $L_{\text{min}}$, multiple forecasts are produced in parallel, where each forecast has its own target and is based on an increasing context length. In particular, a prediction is made for every timestep after $L_{\text{min}}$, 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.

where 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 $\sigma_{r,t}$.

## Experiments

We pretrain the proposed FlowState model and then evaluate its forecasting capabilities on two commonly used benchmarks: **GIFT-Eval**[2] (Aksu et al. 2024), and Chronos Benchmark (II)[3] (Ansari et al. 2024) referred to as **Chronos-ZS**, as described in detail below.

For pretraining we consider two distinct corpora:

- **GIFT-Eval-Pretrain**: The official GIFT-Eval pretraining split, which ensures no data leakage into the GIFT-Eval benchmark. Models trained on this corpus are referred to as **FlowState(G)**.
- **TiRex Pretraining Data**: The same pretraining corpus used by TiRex, except for the synthetic data, which includes a subset of the GIFT-Eval-Pretrain data and the full Chronos pretraining dataset. This setup avoids data leakage into the Chronos-ZS benchmark and is used to train models referred to as **FlowState(T)**.

We additionally added synthetic time series generated via Gaussian Processes, following the methodology of KernelSynth (Ansari et al. 2024) to both corpora. All data—real and synthetic—is further enhanced using augmentation techniques introduced in Auer et al. (2025). For all models, except one ablation, CPM is used during pretraining, which allows for MPI, as introduced in the Background section.

### Evaluation Setup

We assess the forecasting performance of FlowState models in a zero-shot setting, following the standard evaluation protocols of both benchmarks.

**Avoiding data overlap** We observed that several state-of-the-art models, including TiRex—the current benchmark leader—use parts of the GIFT-Eval evaluation set during pretraining. The primary source of this overlap is the Chronos pretraining data corpus, which includes several GIFT-Eval tasks.

To ensure a fair comparison with all baseline models, we evaluate all models on a subset of GIFT-Eval consisting of all 79 out of 97 tasks that are not part of the Chronos pretraining corpus. We refer to this subset as **GIFT-ZS**, and all results reported in the Results section are based on this zero-shot evaluation protocol. Note that Auer et al. (2025) uses a different GIFT-ZS benchmark. In particular, we exclude three additional tasks—Solar/H short, medium, and long—which were present in the Chronos pretraining data but were not excluded in the GIFT-ZS from Auer et al. (2025).

Chronos-ZS has a lot of overlap with the GIFT-Eval pretraining data corpus, which is why only FlowState(T) models—trained without access to Chronos-ZS data—are evaluated on this benchmark.

**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

[2]https://huggingface.co/spaces/Salesforce/GIFT-Eval
[3]https://huggingface.co/spaces/autogluon/fev-leaderboard

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-ZS and WQL in Chronos-ZS. The normalized and averaged MASE are simply referred to as MASE.

**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.

**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 $s_\Delta$, corresponding to the pretraining context length of $2048$ steps, and to the base forecasting length $T = 24$ (=one season). Depending on $s_\Delta$ these effective lengths will change. In particular, the effective context length $L_{\text{eff}}$ and forecasting length $T_{\text{eff}}$ are computed as:

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

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, $s_\Delta$ decreases, resulting in larger $L_{\text{eff}}$ and $T_{\text{eff}}$. This allows FlowState to forecast far into the future for such datasets—precisely where long-range predictions are often most valuable.

## Results

We evaluate two variants of our model: a smaller FlowState-2.6M and a larger FlowState-9.1M, with 2.6M and 9.1M parameters, respectively.

**GIFT.** Table 1 presents the normalized MASE and CRPS metrics for GIFT-ZS, alongside the top seven models ranked by ascending MASE score on the full GIFT-Eval benchmark as of July 31, 2025. Despite its compact size, FlowState consistently outperforms much larger models. Notably, both FlowState(T)-9.1M and FlowState(G)-9.1M surpass all previously reported baselines, including TiRex, the current benchmark leader. Moreover, the FlowState-2.6M variants are by far the smallest models among the top performing ones, yet they perform on par with TiRex and much better than the other models. These results highlight the efficiency and scalability of our architecture.

Table 3 summarizes the full GIFT-Eval benchmark. Among all models without test data leakage, FlowState has the strongest forecasting performance. When comparing to models with test data leakage, FlowState still performs roughly on par with TiRex. However, we have not yet extensively analyzed the full GIFT-Eval and have already observed room to improve FlowState further.

**Chronos-ZS.** Table 2 presents the normalized MASE and WQL metrics, alongside the top seven models ranked by ascending MASE score on the Chronos-ZS benchmark as of July 31, 2025. Similarly to the GIFT-ZS results, FlowState(T)-9.1M outperforms current state-of-the-art models. In fact, our variant trained on the same data as TiRex, performs significantly better on the MASE and the WQL metric. Note, for Chronos-ZS we only evaluate our model variants without data leak, i.e., the models trained

| Model | #Par. | MASE↓ | CRPS↓ |
|---|---|---|---|
| **FlowState(T)-9.1M** | 9.1M | **0.719** | **0.489** |
| **FlowState(G)-9.1M** | 9.1M | 0.728 | 0.491 |
| **FlowState(T)-2.6M** | 2.6M | 0.731 | 0.498 |
| TiRex | 35M | 0.733 | 0.497 |
| **FlowState(G)-2.6M** | 2.6M | 0.733 | 0.502 |
| Toto-Open-Base-1.0 | 151M | 0.737 | 0.497 |
| Sundial-Base | 128M | 0.755 | 0.552 |
| TabPFN-TS | 11M | 0.762 | 0.531 |
| TimesFM-2.0 | 500M | 0.764 | 0.549 |
| YingLong | 300M | 0.796 | 0.537 |
| Chronos-bolt-b | 205M | 0.832 | 0.582 |

Table 1: GIFT-ZS results, sorted by ascending MASE.

| Model | #Par. | MASE↓ | WQL↓ |
|---|---|---|---|
| **FlowState(T)-9.1M** | 9.1M | **0.755** | **0.580** |
| **FlowState(T)-2.6M** | 2.6M | 0.777 | 0.614 |
| TiRex | 35M | 0.778 | 0.599 |
| TimesFM-2.0 | 500M | 0.789 | 0.700 |
| Moirai-l | 311M | 0.791 | 0.631 |
| Chronos-bolt-b | 205M | 0.795 | 0.624 |
| Chronos-b | 200M | 0.818 | 0.643 |
| Moirai-b | 91M | 0.819 | 0.637 |
| Chronos-bolt-s | 48M | 0.823 | 0.636 |

Table 2: Chronos-ZS results, sorted by ascending MASE.

| Model | #Par. | MASE↓ | CRPS↓ | TDL |
|---|---|---|---|---|
| **FlowState(G)-9.1M** | 9.1M | **0.738** | **0.508** | NO |
| Toto-Open-Base-1.0 | 151M | 0.750 | 0.517 | NO |
| Sundial-Base | 128M | 0.750 | 0.559 | NO |
| **FlowState(G)-2.6M** | 2.6M | 0.752 | 0.522 | NO |
| TabPFN-TS | 11M | 0.771 | 0.544 | NO |
| YingLong | 300M | 0.798 | 0.548 | NO |
| TiRex | 35M | **0.724** | **0.498** | YES |
| **FlowState(T)-9.1M** | 9.1M | 0.728 | 0.504 | YES |
| **FlowState(T)-2.6M** | 2.6M | 0.741 | 0.515 | YES |
| TimesFM-2.0 | 500M | 0.758 | 0.550 | YES |
| Chronos-bolt-b | 205M | 0.808 | 0.574 | YES |

Table 3: Full GIFT-Eval results, sorted by ascending MASE. TestData Leakage (TDL) indicates whether a model has seen parts of the evaluation data during pretraining.

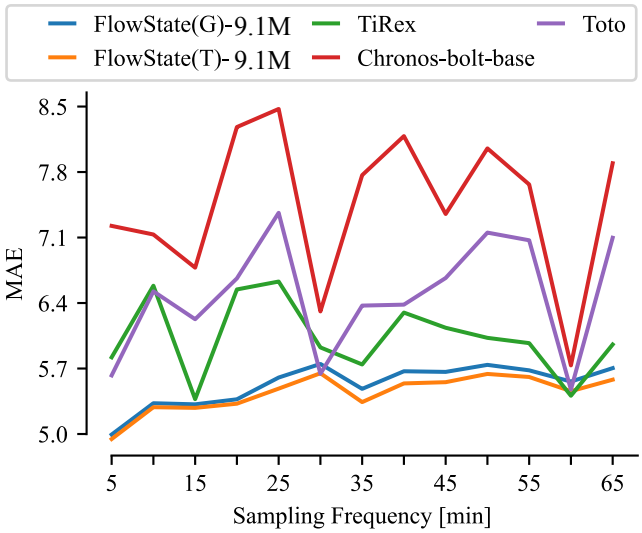


Figure 3: MAE performance across various sampling frequencies on the Loop Seattle dataset.

with the data from TiRex.

**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.

| Model Variant | MASE | CRPS |
|---|---|---|
| **FlowState(G)-2.6M (baseline)** | 0.733 | **0.502** |
| *Core Architectural Components* | | |
| w/o time-scale adjustment | 0.808 | 0.544 |
| w/o parallel predictions | 0.757 | 0.523 |
| *Decoder Variants* | | |
| Fourier basis | 0.736 | 0.505 |
| Half-Legendre basis | 0.737 | 0.504 |
| *CPM* | | |
| Autoregressive instead of MPI | 0.738 | 0.528 |
| No CPM training, longer target | **0.731** | 0.510 |

Table 4: Ablation results on FlowState(G)-2.6M.

**Ablation Study** To understand the contributions of individual components in FlowState, we conduct a series of ablations using the FlowState(G)-2.6M model. 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 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 similarly well to FlowState(G)-2.6M, only resulting in minor performance decrease.

**CPM Mechanism.** We assess the impact of the CPM / MPI mechanism by evaluating FlowState(G)-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, MASE performance recovers to a level even slightly above the original model. CRPS remains worse, highlighting the importance of MPI for accuracy of long probabilistic predictions.

## Conclusion

We introduce FlowState, a time series foundation model, that can dynamically adjust to the unique characteristics of the input time series, such as the 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 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-ZS and the Chronos-ZS 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.

## Acknowledgments

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

## References

Aksu, T.; Woo, G.; Liu, J.; Liu, X.; Liu, C.; Savarese, S.; Xiong, C.; and Sahoo, D. 2024. GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation. *arxiv preprint arxiv:2410.10393*.

Ansari, A. F.; Stella, L.; Turkmen, C.; Zhang, X.; Mercado, P.; Shen, H.; Shchur, O.; Rangapuram, S. S.; Arango, S. P.; Kapoor, S.; et al. 2024. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.

Auer, A.; Podest, P.; Klotz, D.; Böck, S.; Klambauer, G.; and Hochreiter, S. 2025. TiRex: Zero-Shot Forecasting Across Long and Short Horizons. In *1st ICML Workshop on Foundation Models for Structured Data*.

Beck, M.; Pöppel, K.; Spanring, M.; Auer, A.; Prudnikova, O.; Kopp, M.; Klambauer, G.; Brandstetter, J.; and Hochreiter, S. 2024. xLSTM: Extended Long Short-Term Memory. *arXiv*.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; Brynjolfsson, E.; Buch, S.; Card, D.; Castellon, R.; Chatterji, N.; Chen, A.; Creel, K.; Davis, J. Q.; Demszky, D.; Donahue, C.; Doumbouya, M.; Durmus, E.; Ermon, S.; Etchemendy, J.; Ethayarajh, K.; Fei-Fei, L.; Finn, C.; Gale, T.; Gillespie, L.; Goel, K.; Goodman, N.; Grossman, S.; Guha, N.; Hashimoto, T.; Henderson, P.; Hewitt, J.; Ho, D. E.; Hong, J.; Hsu, K.; Huang, J.; Icard, T.; Jain, S.; Jurafsky, D.; Kalluri, P.; Karamcheti, S.; Keeling, G.; Khani, F.; Khattab, O.; Koh, P. W.; Krass, M.; Krishna, R.; Kuditipudi, R.; Kumar, A.; Ladhak, F.; Lee, M.; Lee, T.; Leskovec, J.; Levent, I.; Li, X. L.; Li, X.; Ma, T.; Malik, A.; Manning, C. D.; Mirchandani, S.; Mitchell, E.; Munyikwa, Z.; Nair, S.; Narayan, A.; Narayanan, D.; Newman, B.; Nie, A.; Niebles, J. C.; Nilforoshan, H.; Nyarko, J.; Ogut, G.; Orr, L.; Papadimitriou, I.; Park, J. S.; Piech, C.; Portelance, E.; Potts, C.; Raghunathan, A.; Reich, R.; Ren, H.; Rong, F.; Roohani, Y.; Ruiz, C.; Ryan, J.; Ré, C.; Sadigh, D.; Sagawa, S.; Santhanam, K.; Shih, A.; Srinivasan, K.; Tamkin, A.; Taori, R.; Thomas, A. W.; Tramèr, F.; Wang, R. E.; Wang, W.; Wu, B.; Wu, J.; Wu, Y.; Xie, S. M.; Yasunaga, M.; You, J.; Zaharia, M.; Zhang, M.; Zhang, T.; Zhang, X.; Zhang, Y.; Zheng, L.; Zhou, K.; and Liang, P. 2021. On the Opportunities and Risks of Foundation Models. *arXiv*.

Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; and Ljung, G. M. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Sci. Rep.*, 8(6085): 1–12.

Chen, S.-A.; Li, C.-L.; Arik, S. O.; Yoder, N. C.; and Pfister, T. 2023. TSMixer: An All-MLP Architecture for Time Series Forecast-ing. *Transactions on Machine Learning Research*.

Cohen, B.; Khwaja, E.; Wang, K.; Masson, C.; Ramé, E.; Doubli, Y.; and Abou-Amal, O. 2024. Toto: Time Series Optimized Transformer for Observability. *arXiv*.

Dao, T.; and Gu, A. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 10041–10071. PMLR.

Das, A.; Kong, W.; Sen, R.; and Zhou, Y. 2023. A decoder-only foundation model for time-series forecasting. *arXiv*.

Ekambaram, V.; Jati, A.; Dayama, P.; Mukherjee, S.; Nguyen, N. H.; Gifford, W. M.; Reddy, C.; and Kalagnanam, J. 2024. Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series. *arXiv*.

Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. 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, 459–469. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701030.

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

Gu, A.; Dao, T.; Ermon, S.; Rudra, A.; and Ré, C. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487.

Gu, A.; Goel, K.; and Re, C. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*.

Gu, A.; Johnson, I.; Goel, K.; Saab, K. K.; Dao, T.; Rudra, A.; and Re, C. 2021. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Gu, A.; Johnson, I.; Timalsina, A.; Rudra, A.; and Re, C. 2023. How to Train your HIPPO: State Space Models with Generalized Orthogonal Basis Projections. In *International Conference on Learning Representations*.

Hadi, M. U.; Qureshi, R.; Shah, A.; Irfan, M.; Zafar, A.; Shaikh, M. B.; Akhtar, N.; Wu, J.; Mirjalili, S.; et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2022. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*.

Liang, Y.; Wen, H.; Nie, Y.; Jiang, Y.; Jin, M.; Song, D.; Pan, S.; and Wen, Q. 2024. 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, 6555–6565. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704901.

Liu, X.; Zhang, C.; and Zhang, L. 2024. Vision Mamba: A Comprehensive Survey and Taxonomy. *arXiv*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

Rahman, M. M.; Tutul, A. A.; Nath, A.; Laishram, L.; Jung, S. K.; and Hammond, T. 2024. Mamba in Vision: A Comprehensive Survey of Techniques and Applications. *arXiv*.

Siami-Namini, S.; Tavakoli, N.; and Namin, A. S. 2019. The Performance of LSTM and BiLSTM in Forecasting Time Series. In *2019 IEEE International Conference on Big Data (Big Data)*, 3285–3292.

Smith, J. T.; Warrington, A.; and Linderman, S. 2023. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations*.

Wang, Z.; Kong, F.; Feng, S.; Wang, M.; Yang, X.; Zhao, H.; Wang, D.; and Zhang, Y. 2025. Is Mamba effective for time series forecasting? *Neurocomputing*, 619: 129178.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2022. Are Transformers Effective for Time Series Forecasting? *arXiv*.