TEST-TIME STATE-AUGMENTED ADAPTIVE FORECASTING FOR TIME SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) have demonstrated zero-shot forecasting capability on unseen time series data from various domains. When forecasting with a TSFM, a fixed-length input window is fed into the TSFM. Since typical TSFMs are transformer-based, they incur high computational costs when the input window length is large, due to the transformers' quadratic computational complexity with respect to the input length. Although shortening the input window length can save computational costs, there is a trade-off between the computational costs and forecast performance because transformers are stateless and cannot capture temporal dependencies beyond an input window, and short input windows make a distributional gap relative to the TSFM's pre-training. In this paper, we address the trade-off between the computational cost and performance when forecasting with a TSFM. To mitigate accuracy drops with short input windows, we propose *State-augmented Adaptive Forecasting (SAAF)*, which enables off-the-shelf pre-trained TSFMs to capture temporal dependencies beyond the input window under restricted window lengths. During inference, SAAF incorporates a stateful module that captures temporal dependencies with short input windows. In addition, SAAF transforms a TSFM's outputs to mitigate the TSFM's performance degradation caused by short input windows. Experimental results show that SAAF had lower forecast errors than baselines, especially with short input windows, across various forecasting benchmarks, at lower computational costs.

1 INTRODUCTION

Time series foundation models (TSFMs) have attracted attention (Brinkmeyer et al., 2022; Garza et al., 2023; Rasul et al., 2023b; Jin et al., 2024; Das et al., 2024; Ansari et al., 2024a; Woo et al., 2024b; He et al., 2025) in time series forecasting. Since time series forecasting has a wide range of applications and domains (Sezer et al., 2020; Wen et al., 2022; Kanai et al., 2025) including edge environments, such as healthcare on smartwatches (Narayanswamy et al., 2025), manufacturing, etc. (A et al., 2025), independently building a forecasting model for each task is expensive. Hence, TSFMs have been developed as domain-agnostic base models. By learning forecasting on large-scale time series datasets, TSFMs show zero-shot forecasting capability on unseen time series data from various domains.

In forecasting with TSFMs, *rolling window forecasting* (Zivot & Wang, 2003; Inoue et al., 2017) is typically adopted because of its simplicity and applicability. Rolling window forecasting feeds a fixed-length interval of recent input sequence (window) into a model, predicts the future values, and then slides the window repeatedly (Fig. 1). The past sequence behind the fixed interval is not fed into the TSFM. The window length directly affects forecast accuracy because time series data often have long-term dependencies (Zhou et al., 2021; Cirstea et al., 2022; Su et al., 2025).

In typical TSFMs, however, the computational cost grows in a quadratic order with respect to input lengths because the majority of open-source TSFMs are based on transformers (Vaswani et al., 2017; Keles et al., 2023). Although shortening the input window can reduce computational cost, there is a trade-off between computational cost and forecast performance. This is because short input windows degrade forecasting accuracy due to the stateless nature of transformers. That is, transformers cannot access the past information beyond the current input window, whereas stateful models, such as recurrent neural networks (RNNs), can accumulate the history of the past input windows into hidden

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

states and incorporate them for the current prediction. Thus, with short input windows, capturing temporal dependencies with transformer-based TSFMs becomes difficult. Moreover, short input windows make a distributional gap relative to the TSFM's pre-training input windows. In fact, we empirically observed that the performance of TSFMs degrades when input windows are short and fail to capture the temporal dependencies (Sec. 5.5).

To capture temporal dependencies with transformer-based TSFMs, all related contexts, i.e., sufficiently long input windows, must be fed into the models. However, with limited computational resources, such as edge environments (Narayanswamy et al., 2025; A et al., 2025), handling long input windows with TSFMs can be infeasible due to the high computational cost. One practical application of TSFMs is to continue forecasting for a long time in resource-limited environments. To forecast with low computational costs, only a minimal amount of observed data should be retained. Hence, we address the trade-off between the computational cost and accuracy of TSFM, which has not been considered in previous TSFM studies.

To mitigate accuracy drops with short input windows, we propose *State-augmented Adaptive Forecasting (SAAF)* to enable off-the-shelf pre-trained TSFMs to capture temporal dependencies with short input windows during inference. The key idea of SAAF is to incorporate an auxiliary, lightweight, stateful forecasting module to capture temporal dependencies that stateless transformers cannot. SAAF consists of three components: (i) a *context accumulation forecaster module (CAFM)* is the lightweight stateful module that accumulates input sequences into a fixed-dimensional state vector. By using HiPPO (Gu et al., 2020), a recurrent memory component, as the state vector, it can efficiently encode time series data while retaining as much past information as possible. The state vector enables CAFM to capture long-term dependencies beyond an input window. CAFM learns auxiliary forecasting using the state vector online from past time series observations during inference and complements TSFM's outputs. (ii) A *forecast transformation module (FTM)* learns to transform a TSFM's outputs for adapting to short input windows to mitigate the performance degradation of the backbone TSFM by learning the current dynamics online. (iii) A *dynamic weighting module (DWM)* consolidates the forecasts from CAFM and FTM by adjusting the weights of the two modules based on observed past forecast errors. SAAF can be applied to arbitrary pre-trained TSFMs post-hoc because it does not access TSFM parameters or intermediate features. Moreover, SAAF does not require backpropagation; SAAF has negligible computational overhead compared to TSFMs' forward pass. Our experiments show that SAAF improved forecast performance, especially on short input windows, and the trade-off between computational cost and forecast performance.

2 Related Work

Time Series Foundation Models. Inspired by the success of large language models (LLMs), various TSFMs have been developed (Brinkmeyer et al., 2022; Garza et al., 2023; Rasul et al., 2023b; Jin et al., 2024; Das et al., 2024; Ansari et al., 2024a; Woo et al., 2024b; He et al., 2025). By learning to predict future values of time series data from various domains, TSFMs can dorecast unseed time series data in a zero-shot manner. For inference, a TSFM receives a sequence of the recent timesteps and outputs forecasts of future timesteps. Lag-LLaMA (Rasul et al., 2023b) and Chronos (Ansari et al., 2024a) were especially inspired by LLMs and adopt model architectures used in LLMs (LLaMA (Touvron et al., 2023) and T5 (Raffel et al., 2020), respectively). TimesFM (Das et al., 2024) and Moirai (Woo et al., 2024b) incorporate methodologies of time series models (e.g., patching (Nie et al., 2023)). They split the input time series data into patches and make a token for each patch rather than each timestep to reduce computation. To further reduce computational cost, they output a forecast patch at once rather than auto-regressively generating it one-by-one timesteps. Most TSFMs are transformer-based and incur quadratic computational cost with respect to input sequence length. Although reducing the input length can reduce computational cost, forecast accuracy degrades with short inputs, as discussed in Sec. 1.

Recurrent Models. Recurrent models, such as SSM-based models and recurrent transformers, which can inherently handle long-term dependencies, have emerged. Recurrent memory transformers (RMTs) (Bulatov et al., 2022; 2023) introduce a special token called a memory token to transformers. By passing the embedded memory token to the input of the next inference, RMTs can handle long-term dependencies. However, memory tokens cannot be added to pre-trained models post-hoc because memory token handling must be learned with backpropagation through time, as in RNNs. SSM-based models, such as S4 (Gu et al., 2022) and Mamba (Gu & Dao, 2024), have

also attracted attention as next-generation architectures. More recently, SSM-based TSFMs (Auer et al., 2025; Graf et al., 2025) have been developed. These models are efficient because they incur linear computational costs with respect to input length. However, due to well-explored training techniques and scaling laws in transformers (Jia et al., 2021; Kaplan et al., 2020), transformers are the mainstream of TSFMs, and only a small number of SSM-based TSFMs are publicly available. In this work, we incorporate HiPPO (Gu et al., 2020), a fundamental memory component in S4 and Mamba, into pre-trained transformer-based TSFMs post-hoc to handle temporal dependencies beyond an input window without additional memorization training.

Test-time Adaptation. Test-time adaptation (TTA) has been studied in time series forecasting (Zhang et al., 2023; Kim et al., 2025; Lee et al., 2025). TTA is a setting for on-the-fly adaptation of models with only unlabeled test data and was originally proposed for non-time-series models (Wang et al., 2021; Zhou & Levine, 2021; Niu et al., 2022; Zhang et al., 2022; Enomoto et al., 2024; Adachi et al., 2023; 2025). In the time series setting, TTA aims to adapt forecasting models to temporal changes online during inference, using observed ground truth, unlike the original TTA setting. OneNet (Zhang et al., 2023) dynamically ensembles two different types of forecasters. TAFAS (Kim et al., 2025) introduces a transformation module called the gated calibration module (CGM) attached to the input and output of a target model. The GCM is trained on observed data during testing, which requires backpropagation through the model, whereas not all TSFMs support backpropagation. ELF (Lee et al., 2025) trains an additional linear forecaster that directly projects an input sequence into a forecast and takes an ensemble of forecasts from the TSFM and additional forecaster. However, these methods assume that input windows are long enough to capture context, leaving unexplored cases with short input windows. We address an adaptive forecasting with short input windows to improve TSFMs' trade-off between the computational cost and accuracy, which affects practical use but has not been explored.

3 Preliminaries

Problem setting. A pre-trained TSFM $f : \mathbb{R}^{C\times L} \to \mathbb{R}^{C\times H}$ takes a recent L timestep sequence (input window) and outputs H future values (horizon) as a forecast, where C is the number of channels (variates). We denote values of a time series data at the t-th timestep as $\mathbf{x}_t \in \mathbb{R}^C$ and values of a timespan from t to t' $(> t)$ as $\mathbf{x}_{t:t'} = [\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t'}] \in \mathbb{R}^{C\times(t'-t+1)}$. We consider the rolling window forecasting, where input windows $\mathcal{T} = \{\mathbf{x}_{t_i-L+1:t_i} \in \mathbb{R}^{C\times L}\}_{i=0}^{T-1}$ are sequentially given in chronological order. At each timestep t_i, the TSFM f forecasts the H-horizon from the past L timesteps:

$$\tilde{\mathbf{x}}_{t_i+1:t_i+H} = f\left(\mathbf{x}_{t_i-L+1:t_i}\right). \tag{1}$$

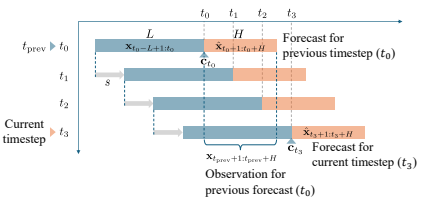


Figure 1: Rolling window forecasting.

The window slides $s \leq \min(L, H)$ timesteps at once. That is, the i-th window $\mathbf{x}_{t_i-L+1:t_i}$ is given at timestep $t_i = si$. Fig. 1 illustrates the problem setting.

Motivation. As the length of an input window L becomes shorter, we can reduce the computational cost of the transformer-based TSFM f. However, f has more difficulty capturing the temporal dependencies of time series data due to the stateless nature of transformers, unlike stateful models that can accumulate information of the past input windows. When forecasting on the current input window $\mathbf{x}_{t_i-L+1:t_i}$ in Eq. (1), f cannot incorporate information before timestep $t_i - L$ since past input windows are discarded under the rolling window forecasting setting. We seek to refine the TSFM's raw forecasts $\{\tilde{\mathbf{x}}_{t_i+1:t_i+H}\}_{i=0}^{T-1}$ sequentially during inference for enabling the TSFM to incorporate temporal dependencies beyond an input window.

4 State-augmented Adaptive Forecasting

We introduce our proposed method, *State-augmented Adaptive Forecasting (SAAF)*. The key idea of SAAF is to incorporate an auxiliary, lightweight, stateful forecasting module to capture temporal dependencies that stateless transformers cannot. Fig. 2 illustrates its overview. SAAF consists of three components: (i) a *context accumulation forecaster module (CAFM)* for accumulating the input

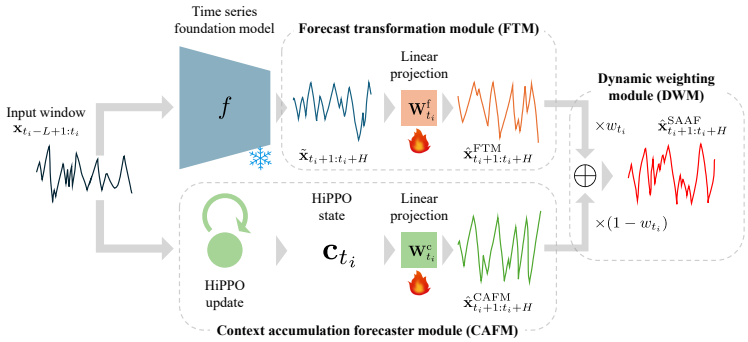


Figure 2: Overview of State-augmented Adaptive Forecasting (SAAF).

contexts beyond the input window length L into a fixed-dimensional state vector and learning auxiliary forecasting (Sec. 4.1), (ii) a *forecast transformation module (FTM)* for learning to transform the TSFM's forecasts to mitigate the performance degradation by short input windows, and (iii) a *dynamic weighting module (DWM)* for adaptively adjusting the weights of the two forecasts from CAFM and FTM on the basis of observed forecast errors (Sec. 4.3).

4.1 Context Accumulation Forecaster Module

Hereinafter, we assume the number of channels $C = 1$ for simplicity. However, SAAF can be easily extended to multivariate time series data ($C \geq 2$) by applying it to each channel.

To capture the temporal dependencies beyond L, CAFM accumulates past input windows into a fixed-dimensional state vector $\mathbf{c}_{t_i} \in \mathbb{R}^D$. Then, an auxiliary linear regressor forecasts $\mathbf{x}_{t_i+1:t_i+H}$ from $\mathbf{c}_{t_i}$ for complementing forecasts from f. To achieve this, the state vector should retain information of the past input sequence before the current input window as much as possible. Specifically, we adopt high-order polynomial projection operators (HiPPO) (Gu et al., 2020), which can efficiently encode sequential data with minimum information loss without any training.

HiPPO is based on the approximation of an input sequence with an orthogonal polynomial function basis $\{p_1(t), \ldots, p_D(t)\}$, where $p_d : \mathbb{R} \to \mathbb{R}$ is d-th polynomial function basis. In other words, HiPPO approximates all history of the input sequence with the polynomial basis: $\mathbf{x}_{0:t_i} \approx g(t) := \sum_d c_{t_i,d} p_d(t)$, where $\mathbf{c}_{t_i} = [c_{t_i,1}, \ldots, c_{t_i,D}]$ are coefficients that minimize the error $\sum_{t=0}^{t_i} |x_t - g(t)|^2$ and $x_t \in \mathbb{R}$ is the value of the input sequence $\mathbf{x}_{0:t_i}$ at timestep t. Interestingly, Gu et al. (2020) reveal that the time evolution of $\mathbf{c}_{t_i}$ can be expressed as the following:

$$\mathbf{c}_{t+1} = \bar{\mathbf{A}}\mathbf{c}_t + \bar{\mathbf{b}}x_t, \tag{2}$$

where $\bar{\mathbf{A}} \in \mathbb{R}^{D \times D}$ and $\bar{\mathbf{b}} \in \mathbb{R}^D$ are determined by the choice of polynomial bases without training. Given that $\mathbf{c}_{t_i}$ contains the optimal polynomial approximation coefficients of the input sequence, $\mathbf{c}_{t_i}$ efficiently encodes the context information from 0 to t_i. Since we receive the input window $\mathbf{x}_{t_i-L+1:t_i}$ at every s timesteps, as illustrated in Fig. 1, we update $\mathbf{c}_{t_{i-1}}$ to $\mathbf{c}_{t_i}$ by applying Eq. (2) s times for $x_{t_{i-1}+1}, \ldots, x_{t_i}$. While Gu et al. (2020) proposed several variants of HiPPO, we adopt HiPPO-LagT (Translated Laguerre), which prioritizes information of recent timesteps and exponentially decays the importance of older timesteps Sec. C in the appendix for more details including $\bar{\mathbf{A}}$ and $\bar{\mathbf{b}}$.

We employ a linear regressor from $\mathbf{c}_{t_i}$ for auxiliary forecasting along with f during inference: $\hat{\mathbf{x}}^{\mathrm{CAFM}}_{t_i+1:t_i+H} := \mathbf{W}^{\mathrm{c}}_{t_i}\mathbf{c}'_{t_i} \in \mathbb{R}^H$. Here, $\mathbf{W}^{\mathrm{c}}_{t_i} = [\mathbf{w}^{\mathrm{c}}_{t_i,1}, \ldots, \mathbf{w}^{\mathrm{c}}_{t_i,H}]^\top \in \mathbb{R}^{H \times (D+1)}$ and $\mathbf{c}'_{t_i} := [\mathbf{c}_{t_i}; 1] \in \mathbb{R}^{D+1}$, for which we concatenate a constant dimension 1 to $\mathbf{c}_{t_i}$ for learning the bias term. As illustrated in Fig. 1, by collecting the last s timesteps from the most recent $\lceil H/s \rceil$ input windows, we can observe the ground truth $\mathbf{x}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H}$ corresponding to a state vector $\mathbf{c}_{t_{\mathrm{prev}}}$, which contains information from 0 to a previous step $t_{\mathrm{prev}} := t_{i-\lceil H/s \rceil}$. Hence, at each step t_i, we update $\mathbf{W}^{\mathrm{c}}_{t_i}$ by using the last state-observation pair with Bayesian linear regression because it works in online settings and is stable by using prior distributions (Yahya et al., 2014). Specifically, we update the H regressors $\mathbf{W}^{\mathrm{c}}_{t_i} = [\mathbf{w}^{\mathrm{c}}_{t_i,1}, \ldots, \mathbf{w}^{\mathrm{c}}_{t_i,H}]^\top$ for predicting $\mathbf{x}_{t_i+1:t_i+H} = [x_{t_i+1}, \ldots, x_{t_i+H}]^\top$,

where $\mathbf{w}^{\mathrm{c}}_{t_i,h} \in \mathbb{R}^{D+1}$ and $h \in \{1,\dots,H\}$. We assume each $\mathbf{w}^{\mathrm{c}}_{t_i,h}$ follows a Gaussian prior $\mathcal{N}^{\mathrm{c}}_{t_i,h} := \mathcal{N}(\mathbf{m}^{\mathrm{c}}_{t_i,h}, (\mathbf{P}^{\mathrm{c}}_{t_i})^{-1})$, where $\mathbf{m}^{\mathrm{c}}_{t_i,h} \in \mathbb{R}^{D+1}$ and $\mathbf{P}^{\mathrm{c}}_{t_i} \in \mathbb{R}^{(D+1)\times(D+1)}$ are the mean and precision. Note that $\mathbf{P}^{\mathrm{c}}_{t_i}$ is shared across the horizon $h \in \{1,\dots,H\}$ because it is determined by only $\mathbf{c}_{t_{\mathrm{prev}}}$ and does not depend on h, as in Eq. (3). Using the last previous state vector $\mathbf{c}_{t_{\mathrm{prev}}}$ and corresponding observation $\mathbf{x}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H} = [x_{t_{\mathrm{prev}}+1},\dots,x_{t_{\mathrm{prev}}+H}]$, we update the prior distribution $\mathcal{N}^{\mathrm{c}}_{t_{i-1},h}$ to $\mathcal{N}^{\mathrm{c}}_{t_i,h}$ by the following equations (Bishop & Nasrabadi, 2006):

$$\mathbf{P}^{\mathrm{c}}_{t_i} = \mathbf{P}^{\mathrm{c}}_{t_{i-1}} + \beta \mathbf{c}'_{t_{\mathrm{prev}}} \mathbf{c}'^{\top}_{t_{\mathrm{prev}}}, \quad \mathbf{m}^{\mathrm{c}}_{t_i,h} = (\mathbf{P}^{\mathrm{c}}_{t_i})^{-1} \left(\mathbf{P}^{\mathrm{c}}_{t_{i-1}} \mathbf{m}^{\mathrm{c}}_{t_{i-1},h} + \beta \mathbf{c}'_{t_{\mathrm{prev}}} x_{t_{\mathrm{prev}}+h} \right), \tag{3}$$

where β is a hyperparameter corresponding to the observation noise. Eq. (3) can be efficiently computed by using the Woodbury matrix identity (Woodbury, 1950) (see Sec. B in the appendix).

We initialize $\mathbf{m}^{\mathrm{c}}_{t_0,h}$ and $\mathbf{P}^{\mathrm{c}}_{t_0}$ as follows:

$$\mathbf{m}^{\mathrm{c}}_{t_0,h} = \mathbf{0}, \quad \mathbf{P}^{\mathrm{c}}_{t_0} = \alpha \mathbf{I}, \tag{4}$$

where α is a hyperparameter corresponding to precision. With the updated prior, CAFM outputs the expectation

$$\hat{\mathbf{x}}^{\mathrm{CAFM}}_{t_i+1:t_i+H} = \mathbb{E}_{\mathbf{w}^{\mathrm{c}}_{t_i,h} \sim \mathcal{N}^{\mathrm{c}}_{t_i,h}} \left[\mathbf{W}^{\mathrm{c}}_{t_i} \mathbf{c}'_{t_i} \right] = \left[\mathbf{m}^{\mathrm{c}}_{t_i,1}, \dots, \mathbf{m}^{\mathrm{c}}_{t_i,H} \right]^{\top} \mathbf{c}'_{t_i}. \tag{5}$$

Since the update is in a closed form (Eq. (3)), SAAF does not rely on backpropagation.

4.2 Forecast Transformation Module

FTM projects the TSFM's raw forecast $\tilde{\mathbf{x}}_{t_i+1:t_i+H} = f(\mathbf{x}_{t_i-L+1:t_i}) \in \mathbb{R}^H$ into an adapted forecast $\hat{\mathbf{x}}^{\mathrm{FTM}}_{t_i+1:t_i+H} = \mathbf{W}^{\mathrm{f}}_{t_i} \tilde{\mathbf{x}}'_{t_i+1:t_i+H} \in \mathbb{R}^H$ to mitigate the TSFM's performance degradation by short input windows. Here, $\tilde{\mathbf{x}}'_{t_i+1:t_i+H} := [\tilde{\mathbf{x}}_{t_i+1:t_i+H}; 1] \in \mathbb{R}^{H+1}$ (a constant dimension 1 is concatenated, similar to CAFM) and $\mathbf{W}^{\mathrm{f}}_{t_i} \in \mathbb{R}^{H\times(H+1)}$ is a linear transformation. Note that we freeze the TSFM f. Similar to CAFM, we consider H regressors $\mathbf{W}^{\mathrm{f}}_{t_i} = [\mathbf{w}^{\mathrm{f}}_{t_i,1},\dots,\mathbf{w}^{\mathrm{f}}_{t_i,H}]^{\top}$ for predicting $\mathbf{x}_{t_i+1:t_i+H} = [x_{t_i+1},\dots,x_{t_i+H}]^{\top}$ and assume each $\mathbf{w}^{\mathrm{f}}_{t_i,h}$ follows a Gaussian prior $\mathcal{N}^{\mathrm{f}}_{t_i,h} := \mathcal{N}(\mathbf{m}^{\mathrm{f}}_{t_i,h}, (\mathbf{P}^{\mathrm{f}}_{t_i})^{-1})$, where $\mathbf{m}^{\mathrm{f}}_{t_i,h} \in \mathbb{R}^{H+1}$ and $\mathbf{P}^{\mathrm{f}}_{t_i} \in \mathbb{R}^{(H+1)\times(H+1)}$. By using observed $\mathbf{x}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H}$ corresponding to a previous TSFM's raw forecast $\tilde{\mathbf{x}}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H}$, as depicted in Fig. 1, we update the prior distribution as follows, like CAFM:

$$\mathbf{P}^{\mathrm{f}}_{t_i} = \mathbf{P}^{\mathrm{f}}_{t_{i-1}} + \beta \tilde{\mathbf{x}}'_{t^H_{\mathrm{prev}}} \tilde{\mathbf{x}}'^{\top}_{t^H_{\mathrm{prev}}}, \quad \mathbf{m}^{\mathrm{f}}_{t_i,h} = (\mathbf{P}^{\mathrm{f}}_{t_i})^{-1} \left(\mathbf{P}^{\mathrm{f}}_{t_{i-1}} \mathbf{m}^{\mathrm{f}}_{t_{i-1},h} + \beta \tilde{\mathbf{x}}'_{t^H_{\mathrm{prev}}} x_{t_{\mathrm{prev}}+h} \right). \tag{6}$$

Here, we denote $\tilde{\mathbf{x}}'_{t^H_{\mathrm{prev}}} := \tilde{\mathbf{x}}'_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H}$ for simplicity. With the updated prior, the output of FTM is

$$\hat{\mathbf{x}}^{\mathrm{FTM}}_{t_i+1:t_i+H} = \mathbb{E}_{\mathbf{w}^{\mathrm{f}}_{t_i,h} \sim \mathcal{N}^{\mathrm{f}}_{t_i,h}} \left[\mathbf{W}^{\mathrm{f}}_{t_i} \tilde{\mathbf{x}}'_{t_i+1:t_i+H} \right] = \left[\mathbf{m}^{\mathrm{f}}_{t_i,1}, \dots, \mathbf{m}^{\mathrm{f}}_{t_i,H} \right]^{\top} \tilde{\mathbf{x}}'_{t_i+1:t_i+H}. \tag{7}$$

Unlike CAFM, we set the initial prior mean to identity so that the TSFM's output is directly passed to the output in Eq. (7):

$$\left[\mathbf{m}^{\mathrm{f}}_{t_0,1}, \dots, \mathbf{m}^{\mathrm{f}}_{t_0,H} \right]^{\top} = \mathbf{I}, \quad \mathbf{P}^{\mathrm{f}}_{t_0} = \alpha \mathbf{I}. \tag{8}$$

4.3 Dynamic Weighting Module

The final forecast of SAAF is the ensemble of the two forecasts from CAFM and FTM:

$$\hat{\mathbf{x}}^{\mathrm{SAAF}}_{t_i+1:t_i+H} = w_{t_i} \hat{\mathbf{x}}^{\mathrm{CAFM}}_{t_i+1:t_i+H} + (1 - w_{t_i}) \hat{\mathbf{x}}^{\mathrm{FTM}}_{t_i+1:t_i+H}, \tag{9}$$

where $w_{t_i} \in [0,1]$ is the ensemble weight. We dynamically adjust w_{t_i} on the basis of which forecast is more reliable. For estimating the reliability, we monitor recent errors of the two modules and increase the weight of the one with smaller error. Specifically, at each step t_i, we track the exponential moving average (EMA) of the two errors:

$$\bar{e}^{\mathrm{c}}_{t_i} = m \bar{e}^{\mathrm{c}}_{t_{i-1}} + (1-m) \| \hat{\mathbf{x}}^{\mathrm{CAFM}}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H} - \mathbf{x}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H} \|_2^2 / H, \tag{10}$$

$$\bar{e}^{\mathrm{f}}_{t_i} = m \bar{e}^{\mathrm{f}}_{t_{i-1}} + (1-m) \| \hat{\mathbf{x}}^{\mathrm{FTM}}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H} - \mathbf{x}_{t_{\mathrm{prev}}+1:t_{\mathrm{prev}}+H} \|_2^2 / H, \tag{11}$$

where $m \in (0, 1)$ is the momentum. Then, we compute the weight as follows:

$$w_{t_i} = \text{sigmoid}(\gamma(\bar{e}^{\text{f}}_{t_i} - \bar{e}^{\text{c}}_{t_i})), \tag{12}$$

where $\gamma > 0$ is a hyperparameter determining the sharpness of switching the two forecasts. By Eq. (12), larger weights are assigned to forecasts with smaller recent errors. We set $w_{t_i} = 0$ before the observations are ready and the errors can be computed because FTM directly outputs the TSFM's raw forecast at the initial state, as in Eq. (8), whereas CAFM requires warmup for producing meaningful forecasts.

Sec. B of the appendix provides the procedure of SAAF in detail.

5 Experiments

We compared forecasting accuracy of our SAAF and baselines on various time series data.

5.1 Datasets

We selected datasets commonly used in studies on time series data (Kong et al., 2025; Kim et al., 2025; Lee et al., 2025).

ETT (Zhou et al., 2021): This dataset contains oil temperature and electricity load data of electricity transformers. ETT contains four domains (h1, h2, m1, and m2) and seven variates. The total numbers of timesteps are 4,416 (h1 and h2) and 17,664 (m1 and m2), respectively. We used all variates (not only the oil temperature) for forecasting.
Monash (Godahewa et al., 2021): This archive contains various datasets from various domains (e.g., energy, transport, and nature). We selected 12 datasets that contains sufficient records and timesteps. We further sub-sampled the datasets so that each dataset contains about 100,000 timesteps in total (e.g., 5 records $\times$ 20,000 timesteps/record) due to the computational budget. We also standardized the datasets, similar to ETT.

More details of the datasets can be found in Sec. E.1 of the appendix.

5.2 Time Series Foundation Models

We selected recently developed off-the-shelf TSFMs. We mainly used TimesFM (Das et al., 2024) in the main experiments and also report the results of Moirai (Woo et al., 2024b), Chronos (Ansari et al., 2024a), and Lag-LLaMA (Rasul et al., 2023b) in the appendix (Sec. F). We used these TSFMs with their default configurations except for the context length L. More details can be found in Sec. E.2 of the appendix.

5.3 Implementation

We set the forecast horizon $H = 64$ and the step size $s = 8$ throughout our experiments. Note that the effect of s to the forecasting results is limited, since the output of a TSFM is determined solely by the current input window, and our CAFM processes the input sequence sequentially, one timestep at a time, regardless of s.

For our SAAF, we set the hyperparameters of Bayesian linear regression $(\alpha, \beta) = (70, 0.01)$, momentum of EMA $m = 0.6$, the switching parameter $\gamma = 10$, and the dimensionality of HiPPO $D = 128$. These hyperparameters were selected by running SAAF on ETT-h1 with several configurations and fixed throughout the experiments.

5.4 Baselines

We compared our SAAF with the following baselines. **No adapt**: performs no adaptation (inference only). **ELF** (Lee et al., 2025): trains an additional linear forecaster and adaptively ensembles forecasts from a TSFM and the linear forecaster. We set the hyperparameters to the default values. **TAFAS** (Kim et al., 2025): trains an adapter module called gated calibration module (GCM) attached to the output of a TSFM. While the original TAFAS attaches another GCM to the input,

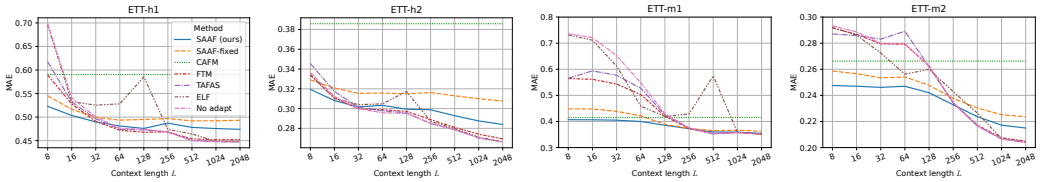


Figure 3: Forecast performance of TimesFM on the ETT dataset. Results for the other TSMFs can be found in Sec. F.1.

Table 1: Test MAEs and average ranking scores and average ranking scores of TimesFM on the Monash datasets. Each score is averaged over context lengths. The best and second scores are in **bold** and underlined.

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	0.82 (3.8)	0.77 (6.0)	0.21 (2.4)	0.77 (5.0)	0.74 (4.3)	0.66 (3.8)	0.87 (4.2)	0.59 (4.1)	0.56 (5.3)	0.60 (5.2)	0.14 (1.8)	0.14 (1.8)	0.57 (4.0)
ELF (Lee et al., 2025)	2506.85 (3.6)	0.79 (4.3)	1708.70 (4.9)	0.82 (4.5)	0.82 (5.6)	0.67 (4.0)	1.00 (4.4)	0.63 (4.0)	0.61 (4.8)	31.77 (5.5)	416.77 (4.8)	14.40 (4.4)	390.32 (4.6)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.61 (4.9)	0.62 (4.7)	0.22 (2.9)	0.38 (3.4)	0.73 (3.6)	0.66 (5.2)	0.78 (4.2)	0.35 (4.3)	0.56 (5.5)	0.59 (4.0)	0.14 (**2.8**)	0.14 (**2.5**)	0.48 (4.0)
FTM	0.53 (3.8)	0.56 (3.7)	**0.18** (**2.3**)	0.35 (**2.8**)	0.57 (**2.5**)	0.46 (4.8)	0.51 (4.2)	0.34 (**2.8**)	0.48 (4.8)	0.50 (4.5)	0.10 (2.9)	0.10 (3.2)	0.39 (3.5)
CAFM	0.51 (5.0)	0.38 (3.2)	0.35 (6.6)	0.33 (5.1)	0.58 (6.1)	0.44 (4.9)	0.34 (4.0)	0.34 (5.3)	**0.42** (2.7)	0.48 (3.7)	0.14 (6.8)	0.14 (6.9)	0.37 (5.0)
SAAF (fixed)	0.50 (3.4)	0.43 (3.2)	0.25 (4.9)	0.31 (3.9)	0.55 (2.8)	0.43 (2.9)	0.38 (3.3)	0.31 (4.2)	0.44 (2.2)	**0.46** (**2.1**)	0.11 (4.9)	0.11 (5.2)	0.36 (3.6)
SAAF (ours)	**0.49** (**3.0**)	**0.36** (**2.0**)	0.18 (3.8)	**0.26** (2.8)	**0.54** (2.9)	**0.42** (**1.9**)	**0.33** (**3.0**)	**0.26** (2.8)	0.43 (**1.7**)	0.46 (2.2)	**0.08** (3.8)	**0.09** (3.9)	**0.32** (**2.8**)

we omitted the input GCM because several TSFMs do not support backpropagation due to non-differentiable sampling, API design, and computational cost. We set the hyperparameters to the default values. **FTM**: uses only FTM of SAAF, described in Sec. 4.2. **CAFM**: uses only CAFM of SAAF, described in Sec. 4.1. Note that this module does not rely on the target TSFM. **SAAF (fixed)**: omits DWM in Eq. (12) from SAAF and fixes the weight to 0.5.

5.5 Forecast Performance

We evaluated the mean absolute error (MAE) between forecasts and corresponding observations over timesteps.

ETT: Fig. 3 shows the MAE of TimesFM with changing the context length L. Our SAAF significantly improved MAE especially with short context length L. Although individual FTM and CAFM can improve performance to some extent, their ensemble had more gains. On the other hand, ELF and TAFAS outperformed in a certain range of L. However, they degraded the performance with shorter L because they cannot consider the context beyond L. When L is large, SAAF underperformed the baselines on ETT-h1 and h2. This is because TimesFM can capture the context and performed well around the default context length (2,048), and CAFM could not be sufficiently trained before reaching the end of the input time series ($T = 4416$ in h1 and h2). In contrast, SAAF outperformed the baselines also with larger L on m1 and m2, where they have longer total timesteps T than h1 and h2, and CAFM could be sufficiently trained. For instance, SAAF achieved the same level of error with only $L = 8$, as No adapt did with $L > 128$ on m1 and m2.

To summarize, SAAF works well especially when L is short and the total number of timesteps T is long, which is practical where forecasting is continued for a long time, while holding a minimum amount of observation data, which is computationally efficient.

Monash: Tabs. 1 and 2 show the MAEs and average ranking scores comparing the baselines computed across domains and context lengths on the Monash datasets with TimesFM. Our SAAF had

Table 2: Test MAEs and average ranking scores of TimesFM on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	0.53 (5.4)	0.48 (5.2)	0.45 (4.9)	0.41 (4.6)	0.34 (3.9)	0.30 (2.8)	0.28 (2.5)	0.27 (2.0)	0.27 (2.1)	0.37 (3.7)
ELF (Lee et al., 2025)	0.53 (4.5)	0.46 (4.5)	0.42 (4.2)	0.40 (4.1)	0.35 (3.9)	0.37 (4.3)	0.55 (4.0)	0.93 (4.0)	6.10 (3.8)	1.12 (4.1)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.45 (4.7)	0.43 (4.8)	0.43 (5.2)	0.41 (4.7)	0.35 (4.4)	**0.31** (3.8)	**0.29** (3.6)	**0.28** (**2.9**)	**0.27** (**2.8**)	0.36 (4.1)
FTM	0.44 (4.2)	0.42 (4.1)	0.41 (4.2)	0.39 (3.8)	0.34 (3.6)	**0.31** (**3.5**)	**0.29** (**3.3**)	**0.28** (3.2)	**0.27** (3.1)	0.35 (3.7)
CAFM	0.38 (3.0)	0.38 (3.5)	0.38 (3.8)	0.38 (4.6)	0.38 (5.2)	0.38 (6.0)	0.38 (6.2)	0.38 (6.5)	0.38 (6.4)	0.38 (5.0)
SAAF (fixed)	0.39 (3.4)	0.38 (3.5)	0.38 (3.3)	0.36 (3.7)	0.34 (3.8)	0.32 (3.9)	0.32 (4.4)	0.31 (5.0)	0.31 (5.2)	0.35 (4.0)
SAAF (ours)	**0.35** (**2.8**)	**0.35** (**2.4**)	**0.35** (**2.3**)	**0.34** (**2.7**)	**0.32** (**3.2**)	**0.31** (3.8)	0.30 (3.9)	0.29 (4.3)	0.29 (4.7)	**0.32** (**3.3**)

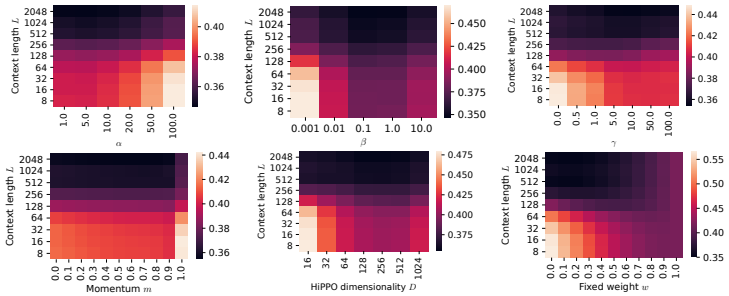


Figure 4: Sensitivity analysis on the hyperparameters of SAAF with TimesFM.

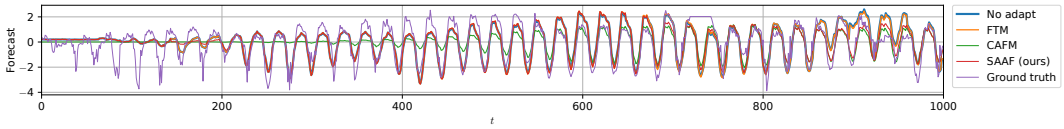


Figure 5: Forecast visualization of TimesFM on the first 1,000 timesteps of ETT-h1 (1st channel, $L = 256$). We plot the TSFM's forecast by connecting the last s timesteps of each output horizon since the output horizons have overlaps due to $s < H$.

best or competitive errors on most L and datasets. Moreover, the average ranking scores are also lowest or competitive in most cases, which indicates that SAAF robustly yields better results than the baselines.

5.6 Sensitivity Analysis

We investigated the effect of each hyperparameter of SAAF. We tested SAAF varying each hyperparameter while the others were fixed to the default values. Fig. 4 shows the results of TimesFM on ETT-m1.

For α and β of Bayesian linear regression, smaller α and $0.1 \leq \beta \leq 1.0$ produced better results, but the effects were not significant. For γ in Eq. (12), $5.0 \leq \gamma \leq 10.0$ produced better results. When γ is small, the performance becomes close to that of SAAF (fixed) because the weights are close to $\mathrm{sigmoid}(0) = 0.5$ unless the error difference is large. In contrast, when γ is large, the weight takes almost binary values, which sharply selects one of CAFM or FTM and the ensemble effect becomes small. For the momentum m, larger m produces slightly better results. However, $m = 1.0$ had the largest MAE. This is because $m = 1.0$ fixes the weights to values on the basis of the initial errors $\bar{e}^{\mathrm{f}}_{t_0}$ and $\bar{e}^{\mathrm{c}}_{t_0}$, which deviate from actual errors during adaptation. For the HiPPO dimensionality D, $128 \leq D \leq 512$ hits better performance. With smaller D, the ability of $\mathbf{c}_{t_i}$ to capture context information is limited and the performance of CAFM decreases. On the other hand, with larger D, $\mathbf{c}_{t_i}$ can capture rich context information, but the regression weights $\mathbf{W}^{\mathrm{c}}_{t_i}$ become difficult to learn. Lastly, we investigated the effect of the ensemble weight w_{t_i} while they were fixed for validating the efficacy of our dynamic weighting. When the weights are fixed, weights around 0.7 and 0.8 produced the best MAEs. However, our dynamic weighting further outperformed the best MAE with fixed weights.

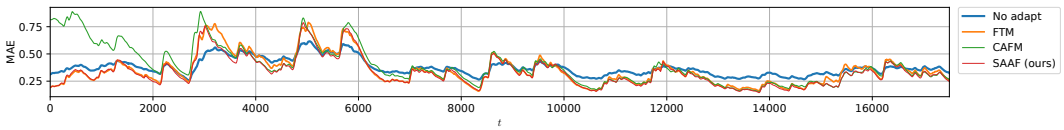
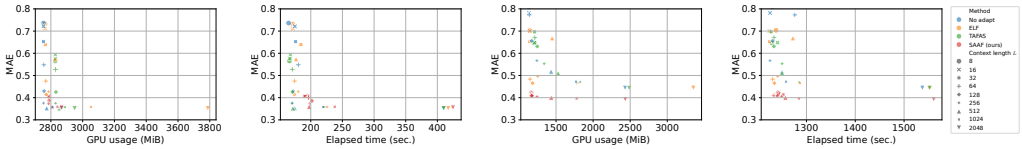


Figure 6: Error visualization of TimesFM on ETT-m1 ($L = 256$). Here, we plot the exponential moving average of MAE for readability.



(a) TimesFM (b) Moirai

Figure 7: Computational costs of (a) TimesFM and (b) Moirai on ETT-m1. Each score is the average over three runs.

5.7 Visualization

Fig. 5 shows the visualizations of the first 1,000 timestep forecasts with SAAF and its components on ETT-h1. First, FTM quickly adapts to the dataset while CAFM's adaptation is slow since the regression weight $\mathbf{W}^{\mathrm{c}}_{t_0}$ is initialized to zero, as in Eq. (4). As the timestep t increases, CAFM also accurately follows the ground truth time series. By adaptively ensembling CAFM and FTM, SAAF produces accurate forecasts throughout timesteps.

Fig. 6 shows the MAE of the forecasts on ETT-m1 (exponential-moving-averaged for readability). Although CAFM produced larger errors than the others on the first 7,500 timesteps, the error decreased along with timesteps. We can also see that SAAF produced smaller errors in most timesteps by adaptively selecting CAFM and FTM. This trend aligns with the results in Fig. 3. On ETT-h1 and h2 ($T = 4416$), SAAF underperformed the baselines with larger L since CAFM could not be sufficiently trained before reaching the end of the time series, as discussed in Sec. 5.5. In contrast, on ETT-m1 and m2 ($T = 17664$), SAAF had competitive MAEs with larger L since CAFM forecasted more accurately.

5.8 Computational Efficiency

We evaluated the computational cost for adaptation in terms of GPU usage and elapsed time. Here, we evaluated TimesFM and Moirai on ETT-m1. The evaluation is performed on a single NVIDIA V100 GPU (32GB) and Intel Xeon Silver 4214R CPU. Fig. 7 shows the results. While the GPU memory usage and computation time grow as the context length L increases, the overhead of SAAF is negligible compared to No adapt. With (a) TimesFM, the increase in computational costs with L is slow, where we could not observe improvements in the trade-offs. This is because TimesFM is a patch-based model, i.e., it splits an input time series into patches of size P, yielding about L/P tokens per input window. In contrast, with (b) Moirai, SAAF had lower MAE with smaller GPU usage and computation time compared to the baselines. It achieved the lowest MAEs with the lowest GPU memory and time. While Moirai is also a patch-based model, its computational costs grow faster because it incorporates multivariate dependencies and runs the forward pass multiple times with different patch sizes to generate a single forecast.

Although patching reduces computational costs, it does not compromise the practicality of our method, because the speedup from patching is only a constant factor, while the computational cost of transformers is $\mathcal{O}(L^2)$.

6 Conclusion

We proposed State-augmented Adaptive Forecasting (SAAF) for efficient and accurate forecasting with time series foundation models (TSFMs) by enabling to capture temporal dependencies with short input window lengths. By incorporating a context accumulation forecaster module (CAFM), SAAF can efficiently capture contexts beyond input window lengths. Further, a forecast transformation module (FTM) and dynamic weighting module (DWM) complementarily work with CAFM to enable TSFMs to quickly adapt to current temporal dynamics and ensemble. In the experiment, we demonstrated that SAAF outperforms baselines especially with short input windows by capturing temporal dependencies beyond input windows, which improved the trade-off between the computational cost and forecast performance.

AI USE STATEMENT

In this work, we used generative AI tools for editing code and grammatical correction of the paper to improve readability. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and interpreting results. Generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs are not applicable to this work. We have reviewed all AI-assisted work. We carefully checked the correction on the code and paper by AI. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We have carefully considered the ethical implications of this work in accordance with the ICLR Code of Ethics. This work does not involve human subjects, personally identifiable or sensitive data, or other activities requiring ethical approval. Although our method enables TSFMs to adapt to changes in the temporal dynamics of time series data, it may also adapt to unintentionally biased data, which can cause fairness or reliability issues. The model's behavior should be carefully monitored in sensitive or critical applications.

REPRODUCIBILITY STATEMENT

The details of the datasets and models used in the experiments are explained in Secs. 5.1 and E.1 and Secs. 5.2 and E.2, respectively. The procedure of our method and the hyperparameter setting are explained in Secs. B and 5.3. Also, the code is provided in the supplementary material.

REFERENCES

Sherly A, Mary Subaja Christo, and Jesi V Elizabeth. A hybrid approach to time series forecasting: Integrating ARIMA and prophet for improved accuracy. *Results in Engineering*, 27:105703, 2025. ISSN 2590-1230.

Kazuki Adachi, Shin'ya Yamaguchi, and Atsutoshi Kumagai. Covariance-aware feature alignment with pre-computed source statistics for test-time adaptation to multiple image corruptions. In *Proceedings of the 2023 IEEE International Conference on Image Processing (ICIP)*, pp. 800–804, 2023.

Kazuki Adachi, Shin'ya Yamaguchi, Atsutoshi Kumagai, and Tomoki Hamagami. Test-time adaptation for regression by subspace alignment. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

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 (TMLR)*, 2024a. ISSN 2835-8856.

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: Pretrained models for time series forecasting (v1.5.2). GitHub Repository, 2024b. `https://github.com/amazon-science/chronos-forecasting/tree/v1.5.2`.

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. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:57529–57580, 2025.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Lukas Brinkmeyer, Rafael Rego Drumond, Johannes Burchert, and Lars Schmidt-Thieme. Few-shot forecasting of time-series with heterogeneous channels. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pp. 3–18. Springer, 2022.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:11079–11091, 2022.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S Burtsev. Scaling transformer to 1M tokens and beyond with RMT. *arXiv preprint arXiv:2304.11062*, 2023.

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1994–2001, 2022.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the Forty-First International Conference on Machine Learning (ICML)*, pp. 10148–10167, 2024.

Shohei Enomoto, Naoya Hasegawa, Kazuki Adachi, Taku Sasaki, Shin'ya Yamaguchi, Satoshi Suzuki, and Takeharu Eda. Test-time adaptation meets image enhancement: Improving accuracy via uncertainty-aware logit switching. In *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2024.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. TimeGPT-1. *arXiv preprint arXiv:2310.03589*, 2023.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Lars Graf, Thomas Ortner, Stanisław Woźniak, and Angeliki Pantazi. FlowState: Sampling-rate invariant time series foundation model with dynamic forecasting horizons. In *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'? (BERT2S) at NeurIPS 2025.*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the First Conference on Language Modeling (COLM)*, 2024.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*, 2022.

Hui He, Kun Yi, Yuanchi Ma, Qi Zhang, Zhendong Niu, and Guansong Pang. SEMPO: Lightweight foundation models for time series forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:161846–161879, 2025.

Atsushi Inoue, Lu Jin, and Barbara Rossi. Rolling window selection for out-of-sample forecasting with time-varying parameters. *Journal of Econometrics*, 196(1):55–67, 2017.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the Thirty-Eighth International Conference on Machine Learning (ICML)*, pp. 4904–4916, 2021.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Sekitoshi Kanai, Yasutoshi Ida, Kazuki Adachi, Mihiro Uchida, Tsukasa Yoshida, and Shin'ya Yamaguchi. Evaluation of time-series training dataset through lens of spectrum of deep state space models. In *Proceedings of the 2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, 2025.

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.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *Proceedings of the Thirty-Fourth International Conference on Algorithmic Learning Theory (ALT)*, pp. 597–619, 2023.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 17868–17876, 2025.

Taekyun Kim, Dmitry V Dolgy, Dae San Kim, Hye Kyung Kim, and Seong Ho Park. A note on degenerate generalized Laguerre polynomials and Lah numbers. *Advances in Difference Equations*, 2021(1):421, 2021.

Xiangjie Kong, Zhenghao Chen, Weiyao Liu, Kaili Ning, Lechao Zhang, Syauqie Muhammad Marier, Yichen Liu, Yuhao Chen, and Feng Xia. Deep learning for time series forecasting: a survey. *International Journal of Machine Learning and Cybernetics*, pp. 1–34, 2025.

Zdzislaw Kowalczuk. Discrete approximation of continuous-time systems: a survey. *IEE Proceedings G (Circuits, Devices and Systems)*, 140:264–278, 1993.

Thomas L Lee, William Toner, Rajkarn Singh, Artjom Joosen, and Martin Asenov. Lightweight online adaption for time series foundation model forecasts. In *Proceedings of the Forty-Second International Conference on Machine Learning (ICML)*, pp. 33736–33764, 2025.

Girish Narayanswamy, Xin Liu, Kumar Ayush, Yuzhe Yang, Xuhai Xu, shun liao, Jake Garrison, Shyam A. Tailor, Jacob Sunshine, Yun Liu, Tim Althoff, Shrikanth Narayanan, Pushmeet Kohli, Jiening Zhan, Mark Malhotra, Shwetak Patel, Samy Abdel-Ghaffar, and Daniel McDuff. Scaling wearable foundation models. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the Thirty-Ninth International Conference on Machine Learning (ICML)*, pp. 16888–16905, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67, 2020.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. GitHub Repository, 2023a. `https://github.com/time-series-foundation-models/lag-llama/tree/df7531a`.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. In *Proceedings of the Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models (R0-FoMo) in the Thirty-Seventh Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023b.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

Liyilei Su, Xumin Zuo, Rui Li, Xin Wang, Heng Zhao, and Bingding Huang. A systematic review for transformer-based long-term series forecasting. *Artificial Intelligence Review*, 58(3):80, 2025.

Chenxi Sun, Moxian Song, Derun Cai, Baofeng Zhang, Shenda Hong, and Hongyan Li. A systematic review of echo state networks from design to application. *IEEE Transactions on Artificial Intelligence*, 5(1):23–37, 2024. doi: 10.1109/TAI.2022.3225780.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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 (NIPS)*, 30, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*, 2021.

Qingsong Wen, Linxiao Yang, Tian Zhou, and Liang Sun. Robust time series analysis and applications: An industrial perspective. In *Proceedings of the Twenty-Eighth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4836–4837, 2022.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers (v1.2.0). GitHub Repository, 2024a. `https://github.com/SalesforceAIResearch/uni2ts/tree/1.2.0`.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the Forty-First International Conference on Machine Learning (ICML)*, pp. 53140–53164, 2024b.

M.A. Woodbury. *Inverting Modified Matrices*. Memorandum Report / Statistical Research Group, Princeton. Department of Statistics, Princeton University, 1950.

Waheed Babatunde Yahya, OR Olaniran, and Samuel Oluwasuyi Ige. On bayesian conjugate normal linear regression and ordinary least square regression methods: A monte carlo study. *Ilorin Journal of science*, 1(1):216–227, 2014.

Guofeng Zhang and Tongwen Chen. Comparing digital implementation via the bilinear and step-invariant transformations. *Automatica*, 40(2):327–330, 2004. ISSN 0005-1098.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:38629–38642, 2022.

Yifan Zhang, Qingsong Wen, Xue Wang, Weiqiu Chen, Liang Sun, Z. Zhang, Liang Wang, Rong Jin, and Tien-Ping Tan. OneNet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:69949–69980, 2023.

Algorithm 1 State-augmented Adaptive Forecasting (SAAF).

Input: Pre-trained TSFM f, rolling windows of input time series data $\mathcal{T} = \{\mathbf{x}_{t_i-L+1:t_i}\}_{i=0}^{T-1}$
Output: Forecast for each window $\{\hat{\mathbf{x}}^{\text{SAAF}}_{t_i+1:t_i+H}\}_{i=0}^{T-1}$

```
Initialize weight mean vectors and precision matrices {m^c_{t_0,1}, ..., m^c_{t_0,H}}, P^c_{t_0}, {m^f_{t_0,1},...,m^f_{t_0,H}}, and P^f_{t_0} in accordance with Eqs. (4) and (8).
Initialize HiPPO state: c_{t_0} ← 0.
Initialize EMA errors for DWM: (ē^c_{t_0}, ē^f_{t_0}) ← (null, null).
for i ∈ {0, ..., T − 1} do
  if Previous TSFM prediction x̃_{t_prev+1:t_prev+H}, HiPPO state c_{t_prev}, and corresponding observation x_{t_prev+1:t_prev+H} are available then
    Update {m^c_{t_i,1}, ..., m^c_{t_i,H}}, P^c_{t_i}, {m^f_{t_i,1}, ..., m^f_{t_i,H}}, and P^f_{t_i} in accordance with Eqs. (3) and (6).
    # Update EMA errors
    if ē^c_{t_i} is null or ē^f_{t_i} is null then
      ē^c_{t_i} ← ‖x̂^CAFM_{t_prev+1:t_prev+H} − x_{t_prev+1:t_prev+H}‖_2^2 / H.
      ē^f_{t_i} ← ‖x̂^FTM_{t_prev+1:t_prev+H} − x_{t_prev+1:t_prev+H}‖_2^2 / H.
    else
      Update ē^c_{t_i} and ē^f_{t_i} in accordance with Eqs. (10) and (11).
    end if
  end if
  Update HiPPO state c_{t_i} in accordance with Eq. (2).
  Forecast with CAFM: x̂^CAFM_{t_i+1:t_i+H} ← [m^c_{t_i,1}, ..., m^c_{t_i,H}]^⊤ c_{t_i}.
  Forecast with FTM: x̂^FTM_{t_i+1:t_i+H} ← [m^f_{t_i,1}, ..., m^f_{t_i,H}]^⊤ f(x_{t_i−L+1:t_i}).
  # Compute dynamic weight w_{t_i}
  if ē^c_{t_i} is null or ē^f_{t_i} is null then
    w_{t_i} ← 0
  else
    Compute w_{t_i} in accordance with Eq. (12).
  end if
  Compute final forecast x̂^SAAF_{t_i+1:t_i+H} in accordance with Eq. (9).
  yield x̂^SAAF_{t_i+1:t_i+H}
end for
```

Aurick Zhou and Sergey Levine. Bayesian adaptation for covariate shift. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:914–927, 2021.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11106–11115, 2021.

Eric Zivot and Jiahui Wang. *Rolling Analysis of Time Series*, pp. 299–346. Springer New York, 2003.

A Limitations

As described in Secs. 5.5 and 5.7, CAFM requires warmup for producing meaningful forecasts since the weight $\mathbf{W}^{\text{c}}_{t_0}$ is initialized to zero. While our method is suitable in case of the total timesteps T is large, which is a realistic setting, fast adaptation when T is small is one direction of our future work.

B Detailed Procedure of SAAF

Alg. 1 lists the detailed procedure of SAAF.

Algorithm 2 PyTorch-like pseudocode for CAFM.

```
def cafm_update(c_prev, x_prev_observed, P, m, P_inv):
    """
    c_prev: (C, D)
    x_prev_observed: (C, H)
    P: (C, 1, D+1, D+1)
    m: (C, H, D+1, 1)
    P_inv: (C, 1, D+1, D+1)
    """

    # concat constant dimension
    c_prev = torch.cat([c_prev, torch.ones(D, 1)], dim=1) \  # (C, D+1)
                  .unsqueeze(2).unsqueeze(1)  # (C, 1, D+1, 1)

    # (C, 1, D+1, D+1)
    P_new = P + beta * c_prev @ c_prev.mT

    P_new_inv = P_inv - ((P_inv @ c_prev) @ (c_prev.mT @ P_inv)) / \
                        (1 / beta + c_prev.mT @ P_inv @ c_prev)

    # (C, H, D+1, 1)
    m_new = P_new_inv @ (
        P @ m + beta * c_prev @ x_prev_observed.reshape(C, H, 1, 1)
    )

    return P_new, m_new, P_new_inv
```

For the updates of CAFM and FTM in Eqs. (3) and (6), naively computing these equations incur $\mathcal{O}(D^3)$ and $\mathcal{O}(H^3)$ computational costs for the inverse of $\mathbf{P}^{\mathrm{c}}_{t_i}$ and $\mathbf{P}^{\mathrm{f}}_{t_i}$, respectively. However, we can further reduce these computational costs to $\mathcal{O}(D^2)$ and $\mathcal{O}(H^2)$ by reusing the inverse matrices in the previous step $(\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1}$ and $(\mathbf{P}^{\mathrm{f}}_{t_{i-1}})^{-1}$ and using the Woodbury matrix identity (Woodbury, 1950):

$$(\mathbf{P}^{\mathrm{c}}_{t_i})^{-1} = \left(\mathbf{P}^{\mathrm{c}}_{t_{i-1}} + \beta \mathbf{c}'_{t_{\mathrm{prev}}} \mathbf{c}'^{\top}_{t_{\mathrm{prev}}}\right)^{-1} = (\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1} - \frac{(\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1} \mathbf{c}'_{t_{\mathrm{prev}}} \mathbf{c}'^{\top}_{t_{\mathrm{prev}}} (\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1}}{\beta^{-1} + \mathbf{c}'^{\top}_{t_{\mathrm{prev}}} (\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1} \mathbf{c}'_{t_{\mathrm{prev}}}}. \tag{13}$$

Since $(\mathbf{P}^{\mathrm{c}}_{t_{i-1}})^{-1}$ is computed in the previous step, the right hand side of Eq. (13) can be computed in $\mathcal{O}(D^2)$. $(\mathbf{P}^{\mathrm{f}}_{t_i})^{-1}$ can also be computed in the same way.

We provide a PyTorch-like pseudocode in Alg. 2 for the CAFM update in Eq. (3). While we explained our method with assuming the number of channels $C = 1$ in Sec. 4, Alg. 2 handles multi-channels all at once. The FTM update in Eq. (6) can be implemented in the same way.

C Details of HiPPO

Here, we briefly describe HiPPO (high-order polynomial projection operators) (Gu et al., 2020), which is incorporated with our method in Sec. 4.1. HiPPO can efficiently encode an input time series data into a fixed-dimensional vector while retaining temporal dependencies. HiPPO is formalized as online function approximation: given an input function $f_t \in \mathbb{R}$, we seek $g^{(t)}$ that approximates $f_{\leq t}$ at every timestep t. In other words, we seek $g^{(t)}$ that minimizes L2 norm $\|g^{(t)} - f_{\leq t}\|^2_{\mu^{(t)}} = \int_0^t |g^{(t)} - f_t|^2 \, \mathrm{d}\mu^{(t)}$, where $\mu^{(t)}$ is a measure that controls the importance weight of approximation over time. Specifically, HiPPO employs a D-dimensional orthogonal polynomial basis for $g^{(t)}$, i.e., $g^{(t)} = \sum_{d=1}^{D} c_n^{(t)} P_d(t)$, and $\{P_d(t)\}_{d=1}^{D}$ is the orthogonal polynomial basis, e.g., Legendre or Laguerre polynomials. The optimal coefficients $\mathbf{c}^{(t)} = [c_1^{(t)}, \ldots, c_D^{(t)}] \in \mathbb{R}^D$ are called HiPPO, which encode the input function $f_{\leq t}$. Intriguingly, such $\mathbf{c}^{(t)}$ can be written in a form of ODE:

$$\frac{\mathrm{d}\mathbf{c}^{(t)}}{\mathrm{d}t} = \mathbf{A}(t)\mathbf{c}^{(t)} + \mathbf{b}(t) f_t, \tag{14}$$

where $\mathbf{A}(t) \in \mathbb{R}^{D\times D}$ and $\mathbf{b}(t) \in \mathbb{R}^D$ are determined by the selection of the polynomial basis and measure; we do not have to train $\mathbf{A}$ and $\mathbf{b}$.

While Gu et al. (2020) proposed several variants of HiPPO, we adopt HiPPO-LagT (Translated Laguerre), which employs an exponentially decaying measure $\mu^{(t)}(x)$, which assigns larger weights to recent approximation errors, and the translated Laguerre polynomials $L_n(t-x)$ (Gu et al., 2020; Kim et al., 2021):

$$\mu^{(t)}(x) = \exp(-(t-x))\mathbb{I}_{(-\infty,t]}(x), \tag{15}$$

$$L_n(t-x) = \frac{\exp(t-x)}{n!}\frac{\mathrm{d}^n}{\mathrm{d}(t-x)^n}\left[(t-x)^n\exp(-(t-x))\right], \tag{16}$$

where $\mathbb{I}_R(x)$ is indicator function that returns 1 when $x \in R$ and otherwise 0.

In HiPPO-LagT, $\mathbf{A}$ and $\mathbf{b}$ for Eq. (14) are as follows (Theorem 1 in (Gu et al., 2020)):

$$A_{nk} = \begin{cases} -1 & \text{if } n \geq k \\ 0 & \text{if } n < k \end{cases}, \quad b_n = 1. \tag{17}$$

Since the above discussion is in continuous-time, we have to discretize the ODE in Eq. (14) to obtain the update rule in Eq. (2). For discretization, we adopted bilinear transformation in accordance with the official implementation of HiPPO[1]. With bilinear transformation, $\bar{\mathbf{A}}$ and $\bar{\mathbf{b}}$ for Eq. (2) can be obtained by the following (Kowalczuk, 1993; Zhang & Chen, 2004):

$$\bar{\mathbf{A}} = \left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} + \frac{h}{2}\mathbf{A}\right), \quad \bar{\mathbf{b}} = \frac{h}{2}\left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1}\mathbf{b}, \tag{18}$$

where $h > 0$ is the discretization step size.

More details can be found in (Gu et al., 2020).

D Computational Cost

While the target TSFM f incurs $\mathcal{O}(L^2)$ computation in each step, our SAAF's update does not depend on L. Our method requires $\mathcal{O}(D^2)$ and $\mathcal{O}(H^2)$ for updating priors $\mathbf{P}^{\mathrm{c}}_{t_i}, \mathbf{m}^{\mathrm{c}}_{t_i,h}, \mathbf{P}^{\mathrm{f}}_{t_i}$, and $\mathbf{m}^{\mathrm{f}}_{t_i,h}$ in Eqs. (3) and (6), and $\mathcal{O}(sD^2)$ for Eq. (2). Since H, D, and s do not take values that are too large (see Sec. 5) and are fixed in practice, the computational overhead of SAAF is negligible compared to the inference with f, which is also empirically observed in Sec. 5.8.

E Detailed Experimental Settings

E.1 Datasets

Here, we provide additional information of the datasets we used in the experiments.

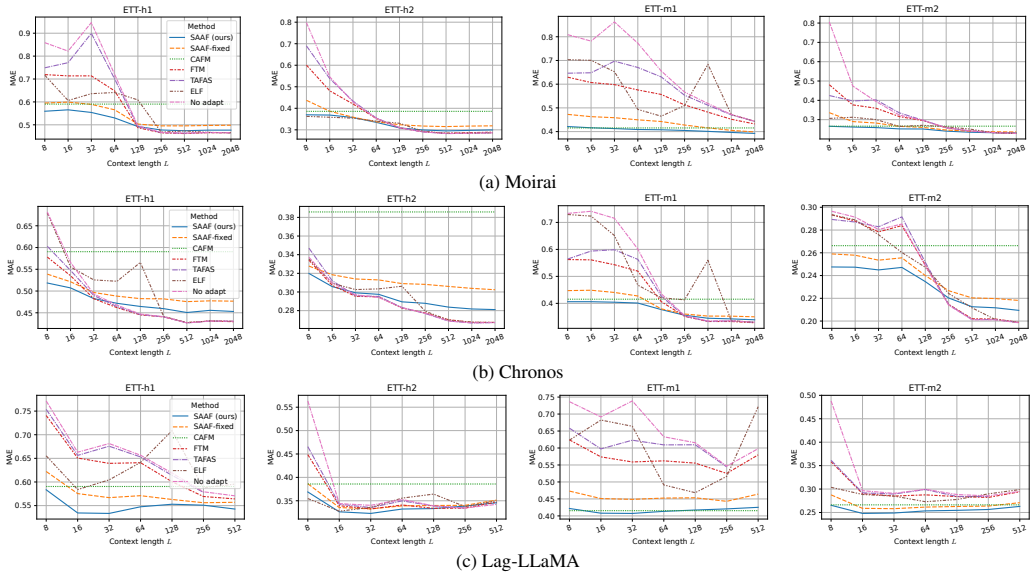
ETT (Zhou et al., 2021) We used timestamps from Jul. 1, 2017 to Dec. 31, 2017 for adaptation. We standardized the dataset by subtracting the mean and then dividing the standard deviation, which were computed using data from before Jul. 1, 2017. We downloaded the dataset from the official GitHub repository[2]. The dataset is released under the CC BY-ND 4.0 license.

Monash (Godahewa et al., 2021) We downloaded the datasets from the official homepage[3]. For license information of each dataset, please refer to its download link.

[1] https://github.com/state-spaces/s4/blob/main/notebooks/hippo_function_approximation.ipynb

[2] https://github.com/zhouhaoyi/ETDataset

[3] https://forecastingdata.org/



(a) Moirai

(b) Chronos

(c) Lag-LLaMA

Figure 8: Forecast performance (MAE) of (a) Moirai, (b) Chronos, and (c) Lag-LLaMA on the ETT dataset. The results for TimesFM are provided in Fig. 3.

E.2 Models

TimesFM (Das et al., 2024) We downloaded the model from the official GitHub repository (v1.2.0) (Woo et al., 2024a). Specifically, we used TimesFM 2.0 (`google/timesfm-2.0-500m-pytorch`). The model is released under the Apache-2.0 license.

Moirai (Woo et al., 2024b) We downloaded the model from the official GitHub repository (v1.2.0) (Woo et al., 2024a). Specifically, we used the Moirai-base model (`Salesforce/moirai-1.1-R-base`). The model is released under the Apache-2.0 license.

Chronos (Ansari et al., 2024a) We downloaded the model from the official GitHub repository (v1.5.2) (Ansari et al., 2024b). Specifically, we used Chronos-bolt (`amazon/chronos-bolt-small`). The model is released under the Apache-2.0 license.

Lag-LLaMA (Rasul et al., 2023b) We downloaded the model following zero-shot forecasting example in the official GitHub repository (Rasul et al., 2023a). The model is released under the Apache-2.0 license.

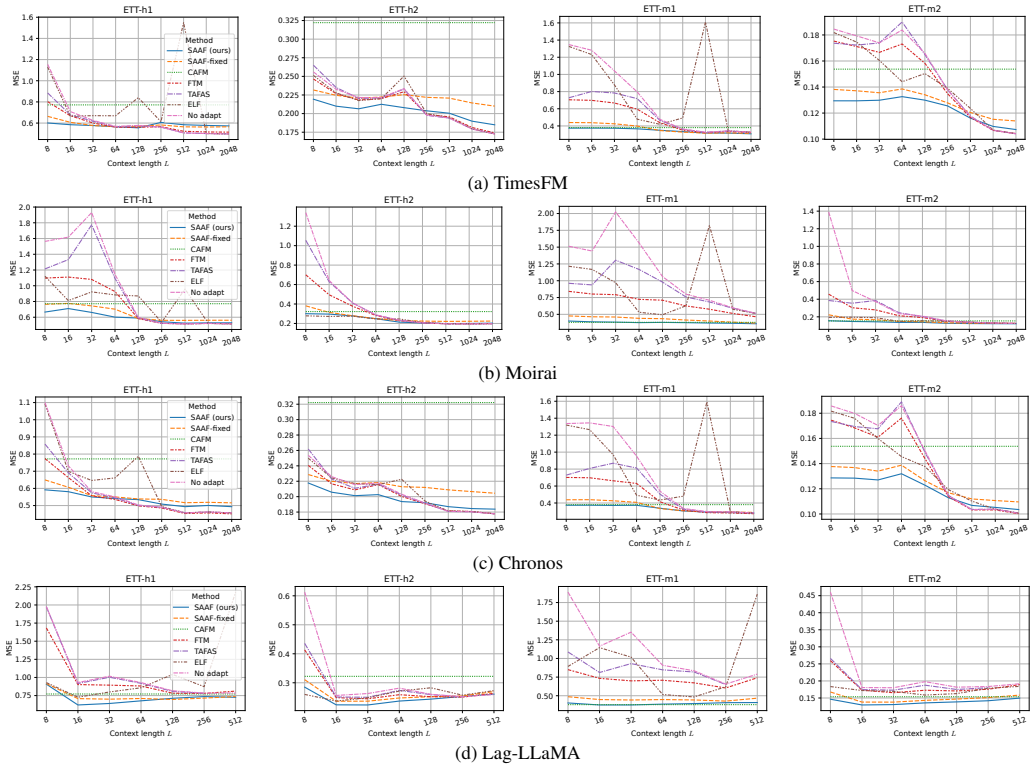
The ETT dataset is not used to pre-train any TSFM used in the experiment. For the Monash datasets, Chronos and TimesFM do not use them for pre-training. In Moirai and Lag-LLaMA, there are overlaps between the pre-training set and the datasets used in the experiments. However, these overlaps do not matter because the main purpose of the experiments is to compare adaptation methods using the same TSFM backbone, not to compare TSFMs' performance.

F Additional Experimental Results

F.1 Forecast Performance

Here, we provide additional results of forecasting benchmarks presented in Sec. 5.5.

Fig. 8 shows MAEs of Moirai, Chronos, and Lag-LLaMA on the ETT datasets. Similar to TimesFM in Fig. 3, SAAF had lower errors especially with smaller input window length L and larger total timesteps T (ETT-m1), which is a realistic situation, e.g., continuous adaptation and forecasting in



(a) TimesFM

(b) Moirai

(c) Chronos

(d) Lag-LLaMA

Figure 9: Forecast performance (MSE) of (a) TimesFM (b) Moirai, (c) Chronos, and (d) Lag-LLaMA on the ETT dataset.

Table 3: Test MAEs and average ranking scores of Moirai on the Monash datasets. Each score is averaged over context lengths.

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	0.55 (3.4)	0.59 (4.8)	0.16 (1.8)	0.58 (4.9)	0.62 (4.3)	0.42 (3.4)	0.83 (5.8)	0.69 (3.8)	0.52 (5.3)	0.54 (4.7)	0.08 (2.1)	0.09 (2.1)	0.47 (3.9)
ELF (Lee et al., 2025)	0.51 (**2.9**)	0.46 (**3.2**)	214.93 (5.0)	0.39 (3.9)	0.57 (4.3)	**0.41** (**3.0**)	0.67 (4.7)	0.34 (**2.7**)	0.44 (3.8)	0.51 (4.6)	73.23 (4.6)	0.74 (3.8)	24.44 (3.9)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.53 (4.6)	0.49 (3.7)	**0.15** (**2.6**)	0.42 (3.2)	0.59 (4.3)	0.43 (4.6)	0.72 (5.5)	0.45 (4.6)	0.48 (5.9)	0.50 (3.6)	0.08 (**2.7**)	0.09 (**2.7**)	0.41 (4.0)
FTM	0.51 (3.6)	0.46 (4.1)	0.16 (2.8)	0.40 (**2.6**)	0.56 (3.3)	0.43 (4.3)	0.65 (5.2)	0.39 (3.1)	0.48 (5.7)	0.49 (4.5)	**0.07** (3.2)	**0.08** (3.4)	0.39 (3.8)
CAFM	0.51 (4.3)	0.38 (3.9)	0.35 (6.4)	0.33 (4.3)	0.58 (4.9)	0.44 (4.6)	**0.34** (**1.0**)	0.34 (4.3)	**0.42** (**1.9**)	0.48 (3.4)	0.14 (6.3)	0.14 (6.6)	0.37 (4.3)
SAAF (fixed)	0.50 (4.2)	0.39 (4.4)	0.24 (5.2)	0.33 (4.9)	**0.55** (**3.2**)	0.42 (3.9)	0.45 (3.0)	0.34 (5.1)	0.43 (2.2)	**0.46** (**3.0**)	0.09 (5.2)	0.10 (5.5)	0.36 (4.1)
SAAF (ours)	**0.49** (4.5)	**0.33** (3.4)	0.18 (4.0)	**0.26** (3.7)	0.55 (3.5)	0.42 (3.8)	0.36 (1.8)	**0.26** (3.9)	0.43 (2.3)	0.46 (3.5)	0.08 (3.7)	0.09 (3.9)	**0.33** (**3.5**)

edge environments. Here, we also provide MSEs in Fig. 9. We can observe the same tendency in MSE compared to MAE.

Tabs. 3 to 8 show detailed MAEs and average ranking scores comparing the baselines computed across dataset domains and context lengths on the Monash datasets. Also, Tabs. 9 to 16 show MSEs and average ranking scores of the four TSFMs.

Our SAAF had best or competitive errors on most L and datasets. Moreover, the average ranking scores are also lowest or competitive in most cases, which indicates that SAAF robustly yields better results than the baselines.

Table 4: Test MAEs and average ranking scores of Moirai on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	0.74 (7.0)	0.74 (6.6)	0.82 (5.7)	0.70 (5.4)	0.39 (4.0)	0.35 (3.4)	0.31 (2.8)	0.30 (2.0)	0.30 (1.9)	0.52 (4.3)
ELF (Lee et al., 2025)	0.57 (4.3)	0.50 (3.8)	0.50 (3.4)	0.51 (4.4)	0.35 (3.8)	0.38 (4.1)	0.58 (4.2)	0.95 (4.3)	6.10 (3.8)	1.16 (4.0)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.58 (5.2)	0.54 (5.1)	0.60 (5.2)	0.57 (5.0)	0.39 (4.5)	0.35 (3.8)	0.32 (**3.2**)	**0.30** (**2.9**)	**0.30** (**3.1**)	0.44 (4.2)
FTM	0.55 (4.5)	0.51 (4.2)	0.52 (4.4)	0.50 (4.1)	0.35 (3.4)	0.33 (**3.1**)	**0.31** (**3.2**)	0.31 (3.8)	0.31 (3.9)	0.41 (3.8)
CAFM	0.38 (**1.9**)	0.38 (2.7)	0.38 (**2.7**)	0.38 (2.8)	0.38 (5.1)	0.38 (5.8)	0.38 (6.1)	0.38 (6.0)	0.38 (6.0)	0.38 (4.3)
SAAF (fixed)	0.43 (3.2)	0.42 (3.4)	0.42 (3.9)	0.41 (3.8)	0.35 (4.0)	0.34 (4.2)	0.33 (4.3)	0.33 (4.6)	0.32 (4.8)	0.37 (4.0)
SAAF (ours)	**0.37** (**1.9**)	**0.36** (**2.2**)	**0.36** (**2.7**)	**0.35** (**2.5**)	**0.33** (**3.2**)	**0.32** (3.7)	**0.31** (4.2)	**0.30** (4.4)	**0.30** (4.6)	**0.33** (**3.3**)

Table 5: Test MAEs and average ranking scores of Chronos on the Monash datasets. Each score is averaged over context lengths.

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	0.47 (2.6)	0.48 (5.2)	0.14 (1.4)	0.27 (4.3)	0.47 (3.1)	0.39 (2.9)	0.60 (4.8)	0.33 (4.2)	0.45 (6.2)	0.51 (4.4)	0.05 (1.2)	0.06 (1.3)	0.35 (3.5)
ELF (Lee et al., 2025)	**0.47** (**2.7**)	0.42 (4.1)	197.03 (4.9)	0.26 (4.4)	0.50 (4.5)	**0.40** (**3.1**)	0.52 (4.1)	0.31 (3.8)	0.44 (4.6)	0.50 (4.5)	67.30 (4.8)	0.08 (4.0)	22.35 (4.1)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.48 (4.5)	0.42 (3.7)	**0.14** (**2.6**)	0.23 (2.8)	**0.47** (3.2)	0.42 (4.8)	0.53 (4.7)	0.29 (4.0)	0.45 (5.8)	0.49 (3.3)	**0.05** (**2.5**)	**0.06** (**2.0**)	0.34 (3.6)
FTM	0.47 (3.3)	0.41 (**2.7**)	0.15 (2.7)	0.24 (**2.0**)	0.47 (**2.2**)	0.41 (4.2)	0.49 (4.8)	0.29 (**2.1**)	0.45 (4.2)	0.48 (4.8)	0.05 (2.8)	0.07 (3.4)	0.33 (**3.3**)
CAFM	0.51 (5.3)	0.38 (3.9)	0.35 (6.6)	0.33 (5.2)	0.58 (6.3)	0.44 (4.9)	0.34 (**2.6**)	0.34 (4.7)	0.42 (1.8)	0.48 (3.2)	0.14 (6.8)	0.14 (7.0)	0.37 (4.9)
SAAF (fixed)	0.48 (4.8)	0.36 (4.2)	0.24 (5.4)	0.26 (4.9)	0.50 (4.6)	0.41 (3.8)	0.37 (3.3)	0.29 (4.9)	**0.42** (2.3)	**0.46** (**2.8**)	0.09 (5.5)	0.10 (5.8)	0.33 (4.4)
SAAF (ours)	0.47 (4.2)	**0.32** (3.4)	0.18 (4.3)	**0.22** (3.9)	0.48 (4.0)	0.41 (3.8)	**0.33** (2.7)	**0.26** (3.8)	0.42 (**1.8**)	0.46 (3.9)	0.07 (4.3)	0.08 (4.5)	**0.31** (3.7)

Table 6: Test MAEs and average ranking scores of Chronos on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	0.55 (5.2)	0.48 (5.2)	0.46 (5.0)	0.43 (4.7)	0.37 (3.9)	0.30 (3.2)	0.25 (2.3)	0.24 (2.1)	0.24 (2.0)	0.37 (3.7)
ELF (Lee et al., 2025)	0.54 (4.9)	0.47 (4.5)	0.42 (4.4)	0.41 (4.4)	0.34 (4.7)	0.35 (4.3)	0.53 (4.2)	0.90 (4.0)	6.05 (3.8)	1.11 (4.4)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.45 (4.7)	0.43 (4.7)	0.42 (5.0)	0.40 (4.8)	0.36 (3.9)	0.30 (3.6)	**0.26** (**3.0**)	**0.24** (**2.8**)	**0.25** (**2.9**)	0.35 (3.9)
FTM	0.45 (4.1)	0.42 (4.0)	0.40 (4.0)	0.37 (3.2)	0.33 (**3.0**)	**0.29** (**3.1**)	**0.26** (3.2)	0.25 (**2.8**)	**0.25** (**2.9**)	0.34 (**3.4**)
CAFM	0.38 (**2.8**)	0.38 (3.5)	0.38 (3.8)	0.38 (4.2)	0.38 (4.8)	0.38 (5.8)	0.38 (6.1)	0.38 (6.3)	0.38 (6.3)	0.38 (4.9)
SAAF (fixed)	0.39 (3.5)	0.38 (3.4)	0.37 (3.3)	0.35 (3.8)	0.34 (4.2)	0.32 (4.3)	0.30 (4.8)	0.30 (5.3)	0.30 (5.2)	0.34 (4.2)
SAAF (ours)	**0.35** (**2.8**)	**0.35** (**2.7**)	**0.34** (**2.4**)	**0.33** (**2.8**)	**0.31** (3.5)	**0.29** (3.8)	0.28 (4.3)	0.27 (4.6)	0.27 (4.8)	**0.31** (3.5)

F.2 More Visualizations

Fig. 10 shows the additional forecast visualizations for Moirai, Chronos, and Lag-LLaMA on ETT-h1, which is compatible with Fig. 5. As the timestep t increases, forecasts become more accurate by CAFM's capturing long-term dependencies. Especially, relatively low-accuracy models, like Lag-LLaMA, significantly mitigates errors, as in $800 \leq t \leq 1000$.

Fig. 11 shows the additional error visualizations for the TSFMs on ETT-m1, which is compatible with Fig. 6. Although FTM can mitigate errors in the early phase of the adaptation, like in (b) Chronos, it sometimes generates larger errors than No adapt. On the other hand, CAFM reduces errors in the latter phase of the adaptation while it had larger errors in the initial phase. SAAF had lower errors also with these models by dynamically adjusting the weights between FTM and CAFM, which corresponds to adjusting the incorporation of temporal dependencies.

Also, we visualized the transition of the CAFM's weight w_{t_i} defined in Eqs. (9) and (12). Fig. 12 shows the visualized w_{t_i}. While the weight fluctuates, a gradual upward trend toward 0.5 is observed. This observation aligns with the CAFM's warmup discussed in Secs. 5.5 and 5.7.

F.3 More Hyperparameter Sensitivity Analysis

We conducted additional hyperparameter sensitivity analysis of our SAAF with changing the model and dataset from Sec. 5.6.

Fig. 13 shows the result when the model is changed to Chronos from TimesFM. Compared to the TimesFM result Fig. 4, Chronos exhibits a similar tendency of sensitivity, which implies that the same hyperparameter setting may work well on other TSFMs.

On the other hand, sensitivity analysis with TimesFM on Monash (London Smart Meters) in Fig. 14 shows a different tendency, especially α and β. While our setting used in the main experiments worked well, we recommend to tune α and β on target datasets to further improve performance for practitioners.

Table 7: Test MAEs and average ranking scores of Lag-LLaMA on the Monash datasets. Each score is averaged over context lengths.

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	0.52 (3.9)	0.73 (7.0)	0.25 (2.9)	0.69 (7.0)	0.61 (6.1)	0.47 (5.0)	0.82 (5.7)	0.65 (7.0)	0.48 (6.4)	0.51 (5.0)	0.11 (3.0)	0.14 (3.0)	0.50 (5.2)
ELF (Lee et al., 2025)	0.51 (**2.4**)	0.60 (4.1)	0.71 (3.4)	0.48 (2.9)	0.61 (5.9)	0.44 (3.3)	0.71 (5.3)	0.47 (3.9)	0.44 (3.9)	0.54 (6.0)	**0.09** (**2.0**)	**0.10** (**2.4**)	0.48 (3.8)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.55 (6.7)	0.66 (5.7)	0.24 (3.3)	0.58 (4.9)	0.60 (5.1)	0.51 (7.0)	0.74 (5.4)	0.56 (4.9)	0.48 (6.4)	0.52 (5.9)	0.10 (3.4)	0.12 (3.1)	0.47 (5.2)
FTM	0.53 (5.1)	0.63 (4.7)	0.24 (3.6)	0.60 (5.9)	0.59 (3.6)	0.47 (5.6)	0.72 (5.3)	0.57 (5.4)	0.47 (5.1)	0.51 (5.1)	0.11 (4.7)	0.13 (4.9)	0.46 (4.9)
CAFM	0.51 (4.1)	**0.38** (**1.4**)	0.35 (6.4)	0.33 (**1.9**)	0.58 (4.1)	0.44 (3.1)	**0.34** (**1.0**)	**0.34** (**1.7**)	**0.42** (2.3)	0.48 (2.7)	0.14 (6.3)	0.14 (6.1)	0.37 (3.4)
SAAF (fixed)	0.51 (3.0)	0.47 (3.0)	0.28 (5.3)	0.43 (3.3)	**0.57** (**1.1**)	0.43 (3.0)	0.48 (3.3)	0.42 (3.3)	0.43 (2.4)	**0.47** (**1.1**)	0.11 (5.0)	0.13 (4.9)	0.39 (3.2)
SAAF (ours)	**0.50** (2.7)	0.39 (2.0)	**0.22** (**3.1**)	**0.33** (2.3)	0.57 (2.0)	**0.41** (**1.0**)	0.36 (2.0)	0.35 (1.9)	0.43 (**1.4**)	0.47 (2.1)	0.10 (3.6)	0.11 (3.6)	**0.35** (**2.3**)

Table 8: Test MAEs and average ranking scores of Lag-LLaMA on the Monash datasets. Each score is averaged over datasets.

	Context length L							
Method	8	16	32	64	128	256	512	Mean
No adapt	0.61 (6.4)	0.53 (5.8)	0.55 (4.9)	0.51 (4.9)	0.48 (4.7)	0.46 (4.8)	0.42 (4.8)	0.51 (5.2)
ELF (Lee et al., 2025)	0.53 (4.0)	0.47 (3.5)	0.47 (3.8)	0.46 (3.7)	0.39 (3.3)	0.43 (3.9)	0.64 (4.2)	0.48 (3.8)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.57 (5.7)	0.49 (5.2)	0.50 (5.5)	0.49 (5.0)	0.46 (5.2)	0.45 (4.9)	0.41 (4.7)	0.48 (5.2)
FTM	0.56 (5.1)	0.49 (4.7)	0.47 (4.4)	0.47 (4.6)	0.46 (5.1)	0.44 (5.2)	0.42 (5.3)	0.47 (4.9)
CAFM	0.38 (2.1)	0.38 (3.4)	0.38 (3.6)	0.38 (3.8)	0.38 (3.6)	0.38 (3.3)	0.38 (4.2)	0.38 (3.4)
SAAF (fixed)	0.43 (2.8)	0.41 (3.3)	0.40 (3.2)	0.40 (3.4)	0.39 (3.6)	0.39 (3.4)	0.38 (2.8)	0.40 (3.2)
SAAF (ours)	**0.37** (**2.0**)	**0.35** (**2.2**)	**0.35** (**2.5**)	**0.36** (**2.6**)	**0.36** (**2.6**)	**0.37** (**2.4**)	**0.36** (**1.9**)	**0.36** (**2.3**)

Table 9: Test MSEs and average ranking scores of TimesFM on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	1.32 (6.1)	1.24 (5.9)	1.18 (5.9)	1.18 (5.8)	0.99 (5.6)	0.85 (5.3)	0.81 (5.1)	0.77 (5.0)	0.77 (5.1)	1.01 (5.5)
ELF (Lee et al., 2025)	1.29 (5.1)	1.17 (5.2)	1.11 (5.2)	1.20 (5.6)	1.09 (5.1)	35.02 (5.4)	373.67 (5.4)	884.94 (5.7)	37562.27 (5.8)	4317.97 (5.4)
TAFAS (Kim et al., 2025) (w/o input GCM)	1.01 (4.5)	1.02 (4.7)	1.04 (4.6)	1.10 (4.5)	0.97 (4.2)	0.84 (3.9)	0.81 (3.9)	0.77 (3.8)	0.76 (3.5)	0.92 (4.2)
FTM	0.94 (3.8)	0.95 (3.8)	0.97 (3.8)	0.93 (3.6)	0.82 (3.1)	0.76 (3.0)	0.73 (**2.8**)	**0.71** (**2.3**)	**0.70** (**2.3**)	0.84 (3.2)
CAFM	1.17 (2.8)	1.17 (2.8)	1.17 (2.9)	1.17 (3.4)	1.17 (3.9)	1.17 (4.4)	1.17 (4.8)	1.17 (4.8)	1.17 (4.8)	1.17 (3.8)
SAAF (fixed)	0.83 (3.2)	0.83 (3.2)	0.85 (3.1)	**0.84** (2.8)	0.80 (3.3)	0.78 (3.2)	0.77 (3.0)	0.77 (3.5)	0.78 (3.5)	0.81 (3.2)
SAAF (ours)	**0.76** (**2.6**)	**0.79** (**2.5**)	**0.82** (**2.5**)	0.85 (**2.3**)	**0.78** (**2.8**)	**0.73** (**2.8**)	**0.72** (3.0)	**0.71** (2.9)	0.71 (3.0)	**0.76** (**2.7**)

F.4 Selection of HiPPO Types

As described in Sec. 4.1, we adopted HiPPO-LagT (Translated Laguerre) for CAFM, while Gu et al. (2020) recommend HiPPO-LegS (Scaled Legendre). In HiPPO-LegS, a uniform measure and the Legendre polynomials are employed:

$$\mu^{(t)}(x) = \frac{1}{t}\mathbb{I}_{[0,t]}(x), \tag{19}$$

$$P_n(x) = \frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left(x^2-1\right)^n. \tag{20}$$

$\mathbf{A}$ and $\mathbf{b}$ for Eq. (14) are as follows (Theorem 2 in (Gu et al., 2020)):

$$A_{nk} = -\frac{1}{t}\begin{cases}(2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k\\ n+1 & \text{if } n = k\\ 0 & \text{if } n < k\end{cases}, \quad b_n = \frac{1}{t}(2n+1)^{1/2}. \tag{21}$$

HiPPO-LegS assigns the importance weights for the polynomial approximation error of an input sequence uniformly from timestep 0 to t, as in Eq. (19). In other words, HiPPO-LegS can equally retain the information of the input sequence from 0 to t. However, the information of the input sequence encoded into $\mathbf{c}_{t_i}$ becomes uniformly inaccurate as the total timestep T grows since the dimensionality of $\mathbf{c}_{t_i}$, i.e., the memory capacity, is fixed. Although HiPPO-LegS is suitable if T is known and fixed, T can be very long or unknown in time series forecasting applications.

Here, we compared our SAAF while changing HiPPO. We tested SAAF and CAFM with TimesFM on the ETT dataset.

Fig. 15 shows the results. SAAF with HiPPO-LagT (ours) had significantly smaller errors than SAAF with HiPPO-LegS in most cases. The difference comes from CAFM, which CAFM (LegS) produced larger errors than CAFM (LagT). HiPPO-LagT can hit good balance between encoding long-term memory and forgetting old information within the fixed memory capacity (HiPPO dimensionality D) by the exponentially decaying measure in Eq. (15).

Table 10: Test MSEs and average ranking scores of TimesFM on the Monash datasets. Each score is averaged over context lengths. The best scores are in **bold**. Cases that produced MSEs larger than 10^5 are displayed as '-.'

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	3.44 (6.5)	1.39 (6.2)	5.61 (4.0)	2.49 (5.1)	2.21 (5.8)	1.98 (6.4)	3.16 (6.4)	1.51 (5.2)	0.74 (6.2)	1.72 (6.4)	3.71 (2.8)	0.30 (3.3)	2.35 (5.4)
ELF (Lee et al., 2025)	-	668.81 (4.7)	-	77.26 (4.8)	841.91 (6.1)	13.42 (6.1)	392.44 (5.8)	47.79 (4.8)	545.91 (5.5)	-	-	-	-
TAFAS (Kim et al., 2025) (w/o input GCM)	1.45 (4.5)	0.79 (4.8)	5.63 (4.8)	0.42 (3.1)	2.16 (4.2)	1.80 (4.4)	2.50 (5.0)	0.40 (3.1)	0.73 (5.2)	1.59 (4.8)	3.71 (2.7)	0.29 (2.8)	1.79 (4.1)
FTM	1.02 (3.8)	0.61 (3.7)	4.81 (**1.7**)	0.33 (3.1)	0.76 (2.8)	0.99 (4.1)	0.57 (3.8)	0.38 (2.8)	0.45 (4.1)	0.97 (3.8)	3.72 (**2.2**)	0.07 (**1.3**)	1.22 (3.1)
CAFM	**0.83** (2.1)	0.30 (3.4)	8.66 (6.6)	0.24 (4.6)	0.76 (5.6)	**0.84** (**1.0**)	**0.24** (1.9)	0.32 (5.1)	**0.34** (**1.6**)	**0.77** (**1.0**)	**0.06** (6.7)	0.08 (6.6)	1.12 (3.8)
SAAF (fixed)	0.87 (2.4)	0.36 (2.7)	4.91 (3.2)	0.22 (3.8)	**0.66** (**1.2**)	0.88 (2.7)	0.31 (2.0)	0.29 (3.9)	0.36 (2.2)	0.81 (1.9)	0.95 (4.8)	**0.05** (4.8)	0.89 (3.0)
SAAF (ours)	**0.83** (**1.6**)	**0.28** (**1.6**)	**4.46** (**1.7**)	**0.17** (**2.9**)	0.67 (2.1)	0.85 (1.8)	0.25 (**1.8**)	**0.24** (**2.7**)	**0.34** (1.8)	0.82 (2.6)	0.83 (3.8)	**0.05** (3.8)	**0.82** (**2.3**)

Table 11: Test MSEs and average ranking scores of Moirai on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	2.03 (6.8)	2.28 (6.7)	2.60 (6.3)	1.89 (6.2)	1.08 (5.9)	0.90 (5.4)	0.83 (5.1)	0.88 (4.9)	0.85 (4.8)	1.48 (5.8)
ELF (Lee et al., 2025)	1.35 (4.9)	1.27 (4.5)	1.30 (4.0)	1.40 (5.3)	1.11 (5.2)	35.05 (5.0)	373.78 (5.5)	885.01 (5.8)	37562.38 (5.6)	4318.07 (5.1)
TAFAS (Kim et al., 2025) (w/o input GCM)	1.20 (5.2)	1.24 (5.0)	1.48 (5.2)	1.33 (4.8)	1.01 (4.6)	0.87 (4.1)	0.82 (3.8)	0.87 (3.5)	0.84 (3.4)	1.07 (4.4)
FTM	1.08 (4.2)	1.06 (3.8)	1.13 (4.1)	1.06 (3.7)	0.81 (3.4)	0.75 (**2.9**)	0.77 (**2.7**)	0.86 (**2.9**)	0.84 (**2.9**)	0.93 (3.4)
CAFM	1.17 (**2.1**)	1.17 (2.6)	1.17 (**2.6**)	1.17 (2.6)	1.17 (3.8)	1.17 (4.2)	1.17 (4.2)	1.17 (4.5)	1.17 (4.6)	1.17 (3.5)
SAAF (fixed)	0.85 (2.8)	0.85 (3.0)	0.83 (3.2)	0.82 (3.1)	0.77 (2.7)	0.74 (3.2)	0.76 (3.7)	**0.78** (3.2)	0.79 (3.5)	0.80 (3.1)
SAAF (ours)	**0.76** (**2.1**)	**0.78** (**2.4**)	**0.74** (2.7)	**0.77** (**2.4**)	**0.73** (**2.4**)	**0.71** (3.1)	**0.71** (3.1)	0.79 (3.2)	**0.76** (3.2)	**0.75** (**2.7**)

Table 12: Test MSEs and average ranking scores of Moirai on the Monash datasets. Each score is averaged over context lengths. The best scores are in **bold**. Cases that produced MSEs larger than 10^5 are displayed as '-.'

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	1.21 (6.6)	0.94 (5.0)	4.95 (4.0)	1.11 (5.1)	1.06 (6.3)	1.26 (6.5)	1.85 (6.5)	1.96 (5.5)	0.60 (6.5)	1.64 (6.6)	0.04 (3.1)	0.06 (3.7)	1.39 (5.5)
ELF (Lee et al., 2025)	1.01 (5.6)	0.53 (3.3)	-	3.56 (4.0)	0.90 (5.7)	1.13 (6.2)	1.93 (5.8)	0.42 (4.2)	0.39 (4.6)	1.15 (5.9)	-	2160.97 (4.8)	-
TAFAS (Kim et al., 2025) (w/o input GCM)	0.98 (5.0)	0.57 (4.7)	4.91 (3.4)	0.50 (3.2)	0.90 (5.0)	1.00 (4.8)	1.21 (5.0)	0.70 (3.6)	0.46 (5.5)	1.08 (4.9)	0.04 (**2.4**)	0.06 (2.5)	1.03 (4.2)
FTM	0.96 (4.0)	0.49 (**3.1**)	4.96 (3.1)	0.42 (**3.0**)	0.76 (3.0)	0.97 (3.8)	0.77 (4.0)	0.50 (**2.5**)	0.43 (4.5)	0.95 (3.8)	**0.02** (2.5)	**0.04** (**1.6**)	0.94 (3.2)
CAFM	**0.83** (**1.6**)	0.30 (3.9)	8.66 (6.6)	0.24 (4.3)	0.76 (3.8)	**0.84** (**1.7**)	**0.24** (**1.0**)	0.32 (4.3)	**0.34** (**1.0**)	**0.77** (**1.0**)	0.06 (6.3)	0.08 (6.2)	1.12 (3.5)
SAAF (fixed)	0.85 (2.0)	0.31 (3.9)	**4.69** (2.7)	0.25 (4.4)	**0.67** (**1.0**)	0.87 (2.1)	0.37 (2.8)	0.33 (4.5)	0.35 (2.5)	0.80 (1.8)	0.03 (4.9)	0.05 (4.7)	0.80 (3.1)
SAAF (ours)	0.84 (2.2)	**0.26** (3.5)	4.72 (**2.0**)	**0.18** (3.5)	0.71 (2.6)	0.86 (2.0)	0.28 (1.8)	**0.25** (2.7)	0.35 (2.5)	0.83 (2.8)	**0.02** (3.7)	0.05 (4.3)	**0.78** (**2.8**)

F.5 Computational Efficiency

In Sec. 5.8, we reported the computational costs of TimesFM and Moirai. Here, we also report the results of Chronos-bolt and Lag-LLaMA.

Fig. 16 shows the result. In (a) Chronos-bolt, we could not observe the trade-off's improvement by SAAF due to the TSFM's low computational cost by patching and the model size, similar to TimesFM. On the other hand, compared within the same context length L, SAAF improves the error. Thus, as mentioned in Sec. 1, SAAF is also suitable to even such a TSFM when continuing forecasting for a long period with a minimal amount of observation data retained. In contrast, in (b) Lag-LLaMA, SAAF significantly improved the trade-off and achieved the lowest error with the lowest computational cost. This is because Lag-LLaMA treats each timestep as a token and auto-regressively generates forecasts one timestep at a time, where L largely affects the computational cost.

F.6 Effect of CAFM's Warmup

As described in Sec. 5.5, SAAF performs better on time series data that have longer total timesteps T because CAFM requires warmup to produce meaningful forecasts, as the weight is initialized to zero in Eq. (4). We evaluated the effect of T on the performance. Fig. 17 plots the accumulated MAE with reference to T on ETT-m1. CAFM produces large errors with small T, and the error decreases as T increases. While FTM's error also decreases with larger T, the effect of T on the error is smaller than in CAFM; the context length L affects more than T. Since SAAF combines CAFM and FTM, it is also affected by CAFM's warmup. Thus, SAAF performs well, especially when T is large, e.g., when continually forecasting for a long period.

Table 13: Test MSEs and average ranking scores of Chronos on the Monash datasets. Each score is averaged over datasets.

	Context length L									
Method	8	16	32	64	128	256	512	1,024	2,048	Mean
No adapt	1.35 (5.8)	1.22 (6.0)	1.17 (5.9)	2.25 (5.9)	1.58 (5.5)	0.80 (5.0)	0.72 (4.8)	0.71 (4.9)	0.68 (4.8)	1.16 (5.4)
ELF (Lee et al., 2025)	1.30 (5.5)	1.16 (5.3)	1.05 (5.3)	2.13 (5.8)	1.61 (5.3)	34.98 (5.2)	373.65 (5.5)	884.88 (5.5)	37562.15 (5.5)	4318.10 (5.4)
TAFAS (Kim et al., 2025) (w/o input GCM)	0.99 (4.5)	0.99 (4.4)	0.97 (4.4)	1.49 (4.6)	1.29 (4.1)	0.78 (3.7)	0.71 (3.3)	0.71 (3.4)	0.68 (3.1)	0.96 (3.9)
FTM	0.92 (3.6)	0.92 (3.6)	0.88 (3.7)	0.92 (3.6)	0.84 (3.2)	0.70 (**2.8**)	0.66 (**2.5**)	0.67 (**2.2**)	0.66 (**2.2**)	0.80 (3.0)
CAFM	1.17 (2.8)	1.17 (2.9)	1.17 (2.8)	1.17 (3.2)	1.17 (4.0)	1.17 (4.6)	1.17 (4.8)	1.17 (5.3)	1.17 (5.3)	1.17 (4.0)
SAAF (fixed)	0.81 (3.2)	0.82 (3.1)	0.80 (3.2)	0.76 (2.8)	0.73 (3.2)	0.73 (3.4)	0.70 (3.8)	0.70 (3.6)	0.71 (3.8)	0.75 (3.3)
SAAF (ours)	**0.75** (**2.6**)	**0.77** (**2.7**)	**0.74** (**2.6**)	**0.69** (**2.2**)	**0.68** (**2.7**)	**0.69** (3.3)	**0.65** (3.3)	**0.65** (3.1)	**0.65** (3.2)	**0.70** (**2.8**)

Table 14: Test MSEs and average ranking scores of Chronos on the Monash datasets. Each score is averaged over context lengths. The best scores are in **bold**. Cases that produced MSEs larger than 10^5 are displayed as '-.'

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	0.96 (6.4)	0.59 (5.6)	4.80 (4.4)	0.30 (4.7)	0.64 (4.7)	2.34 (6.2)	1.11 (6.5)	0.43 (4.9)	0.41 (6.6)	1.52 (6.5)	0.01 (2.0)	0.03 (3.0)	1.10 (5.1)
ELF (Lee et al., 2025)	0.95 (5.8)	0.42 (4.0)	-	0.27 (4.3)	0.70 (5.6)	2.34 (6.3)	1.37 (5.7)	0.37 (4.3)	0.39 (5.0)	1.15 (5.7)	-	0.56 (5.2)	-
TAFAS (Kim et al., 2025) (w/o input GCM)	0.88 (3.8)	0.42 (3.8)	4.80 (3.9)	0.19 (2.7)	0.61 (3.2)	1.55 (4.6)	0.73 (4.8)	0.30 (2.8)	0.40 (5.3)	1.19 (5.0)	**0.01** (2.5)	**0.03** (2.3)	0.93 (3.7)
FTM	0.88 (4.2)	0.38 (**2.8**)	4.46 (2.5)	0.18 (**2.2**)	**0.57** (**1.9**)	0.93 (3.9)	0.53 (3.8)	0.31 (**2.4**)	0.39 (4.1)	0.93 (3.8)	**0.01** (**1.7**)	**0.03** (**1.0**)	0.80 (**2.8**)
CAFM	0.83 (3.0)	0.30 (4.0)	8.66 (6.6)	0.24 (4.7)	0.76 (5.9)	**0.84** (**1.7**)	**0.24** (**1.9**)	0.32 (4.6)	**0.34** (**1.1**)	**0.77** (**1.2**)	0.06 (6.7)	0.08 (6.6)	1.12 (4.0)
SAAF (fixed)	**0.82** (**1.8**)	0.27 (4.1)	4.50 (2.9)	0.17 (4.8)	0.58 (3.2)	0.86 (2.1)	0.30 (2.2)	0.27 (4.8)	**0.34** (2.2)	0.80 (1.6)	0.03 (5.2)	0.04 (5.3)	0.75 (3.4)
SAAF (ours)	**0.82** (2.1)	**0.24** (3.0)	**4.09** (**1.5**)	**0.14** (4.0)	0.58 (3.2)	0.85 (1.9)	0.26 (2.0)	**0.24** (3.6)	**0.34** (2.2)	0.82 (2.8)	0.02 (4.2)	0.04 (4.5)	**0.70** (2.9)

Table 15: Test MSEs and average ranking scores of Lag-LLaMA on the Monash datasets. Each score is averaged over datasets.

	Context length L							
Method	8	16	32	64	128	256	512	Mean
No adapt	3.27 (6.9)	1.40 (6.6)	1.46 (5.9)	1.40 (6.0)	1.39 (6.5)	1.39 (6.8)	1.04 (6.4)	1.62 (6.4)
ELF (Lee et al., 2025)	1.72 (4.4)	1.21 (4.5)	1.23 (4.8)	1.25 (4.8)	1.14 (4.0)	35.11 (4.2)	373.84 (4.8)	59.36 (4.5)
TAFAS (Kim et al., 2025) (w/o input GCM)	4.37 (5.2)	1.26 (4.9)	1.20 (5.0)	1.25 (4.8)	1.30 (5.3)	1.32 (5.6)	1.02 (5.0)	1.67 (5.1)
FTM	2.65 (5.0)	1.10 (4.1)	1.08 (4.0)	1.21 (4.4)	1.24 (4.8)	1.35 (4.8)	0.97 (4.2)	1.37 (4.5)
CAFM	1.17 (**1.7**)	1.17 (2.7)	1.17 (2.7)	1.17 (**2.6**)	1.17 (2.5)	1.17 (**2.1**)	1.17 (2.8)	1.17 (2.4)
SAAF (fixed)	1.13 (2.8)	0.86 (3.0)	0.86 (3.2)	0.93 (2.8)	1.00 (2.7)	1.07 (2.5)	0.95 (2.8)	0.97 (2.8)
SAAF (ours)	**0.89** (2.1)	**0.78** (**2.2**)	**0.80** (**2.5**)	**0.92** (**2.6**)	**0.98** (**2.2**)	**1.05** (2.2)	**0.88** (**2.1**)	**0.90** (**2.3**)

F.7 Comparison with Recurrent Models

Recently, TSFMs that have recurrent architectures have been proposed, as discussed in Sec. 2. In this section, we compared the following recurrent baselines and demonstrate that our SAAF can be used beyond transformer-based TSFMs.

TiRex (Auer et al., 2025): An LSTM-based TSFM. Since SAAF is model-agnostic, it can be combined with non-transformer-based TSFMs. Here, using TiRex as the backbone TSFM, we compared No adapt, ELF (Lee et al., 2025), TAFAS (Kim et al., 2025) (w/o input GCM), FTM, and SAAF.

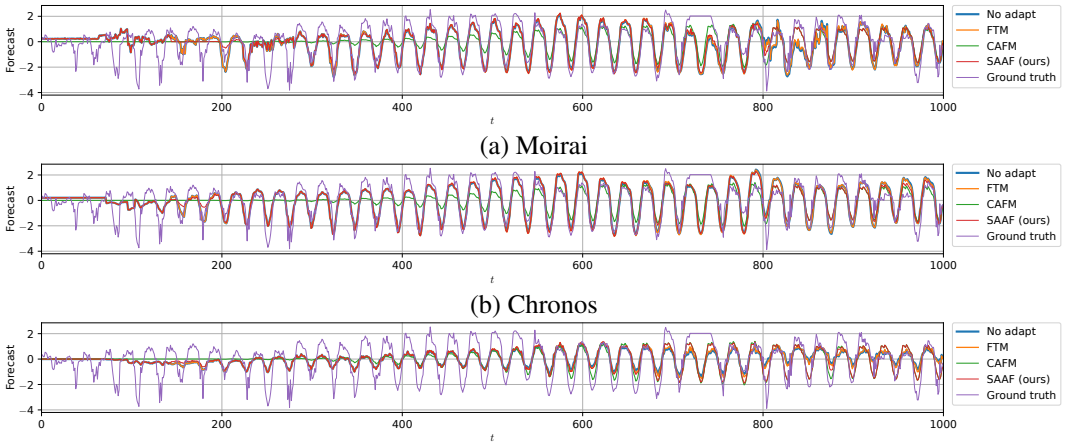
ESN (Sun et al., 2024): We adopted a leaky ESN (echo state network) as a simple stateful forecasting baseline. Following a standard setting in ESN, we randomly initialized and froze the reservoir layer and trained the readout layer with Bayesian linear regression, similar to CAFM. For fair comparison, we set the state dimensionality to $128 \times 7\ (\text{num_channels}) = 896$ to align with CAFM.

Fig. 18 shows the result. As with the other transformer-based TSFMs, we ran TiRex with varying the context length L because TiRex does not support online forecasting like CAFM. SAAF can also be used to TiRex and improved MAE in $L \leq 32$ on ETT-h1 and $L \leq 128$ on ETT-m1. Similar to the result in Sec. 5.5, SAAF works better when the total number of timesteps T is long (e.g., ETT-m1).

Comparing ESN and CAFM, CAFM achieved lower errors than ESN by using HiPPO, which is explicitly designed to memorize history, as its state. This implies that just using a stateful forecasting model does not improve performance. On the other hand, while FTM improves performance to some extent, its gains are limited when L is small. SAAF achieves lower or competitive error compared to each module by complementarily ensembling CAFM and FTM.

Table 16: Test MSEs and average ranking scores of Lag-LLaMA on the Monash datasets. Each score is averaged over context lengths. The best scores are in **bold**.

Method	London Smart Meters	Australian Electricity Demand	Wind Farms	Pedestrian Counts	KDD Cup 2018	Weather	Solar	San Francisco Traffic	Sunspot	Saugeen River Flow	Solar Power	Wind Power	Mean
No adapt	1.15 (7.0)	1.04 (7.0)	9.11 (4.9)	0.92 (7.0)	1.08 (6.6)	1.11 (7.0)	1.57 (6.6)	0.95 (7.0)	0.49 (7.0)	1.13 (6.6)	0.06 (5.1)	0.10 (5.6)	1.56 (6.4)
ELF (Lee et al., 2025)	1.06 (5.3)	0.69 (4.4)	669.07 (4.4)	0.51 (3.1)	1.00 (6.0)	1.03 (5.4)	1.29 (5.7)	0.55 (3.6)	0.39 (4.0)	1.13 (6.0)	**0.04** (**2.1**)	0.12 (3.9)	56.41 (4.5)
TAFAS (Kim et al., 2025) (w/o input GCM)	1.03 (5.4)	0.80 (5.7)	11.35 (4.4)	0.62 (4.7)	1.00 (5.4)	1.06 (5.0)	1.09 (5.3)	0.69 (5.0)	0.46 (5.9)	1.03 (5.1)	0.05 (4.6)	0.09 (4.7)	1.61 (5.1)
FTM	1.00 (4.3)	0.72 (4.6)	8.52 (4.3)	0.66 (5.7)	0.85 (4.0)	1.01 (4.6)	0.87 (4.3)	0.72 (5.6)	0.45 (5.0)	0.98 (4.3)	0.05 (3.9)	0.08 (3.0)	1.33 (4.5)
CAFM	**0.83** (**1.0**)	**0.30** (**1.3**)	8.66 (6.0)	**0.24** (**1.9**)	0.76 (2.4)	**0.84** (**1.0**)	**0.24** (**1.0**)	**0.32** (**1.7**)	**0.34** (**1.0**)	**0.77** (**1.0**)	0.06 (5.7)	0.08 (5.0)	1.12 (2.4)
SAAF (fixed)	0.88 (3.0)	0.41 (3.1)	5.89 (**2.0**)	0.36 (3.3)	**0.73** (**1.1**)	0.88 (3.0)	0.42 (3.1)	0.43 (3.3)	0.36 (3.1)	0.83 (2.0)	**0.04** (3.9)	**0.06** (**2.7**)	0.94 (2.8)
SAAF (ours)	0.84 (2.0)	0.32 (1.9)	**5.50** (**2.0**)	**0.24** (2.3)	0.77 (2.4)	0.85 (2.0)	0.28 (2.0)	0.33 (1.9)	0.35 (2.0)	0.85 (3.0)	**0.04** (2.7)	**0.06** (3.1)	**0.87** (**2.3**)

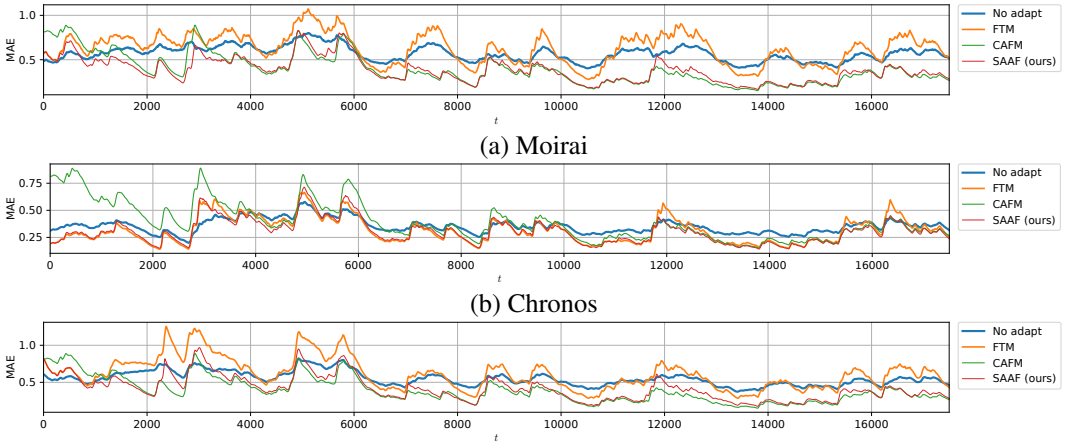


(a) Moirai

(b) Chronos

(c) Lag-LLaMA

Figure 10: Forecast visualizations of TSFMs on the first 1,000 steps of ETT-h1 (1st channel, $L = 256$). The plotting procedure is the same as Fig. 5.



(a) Moirai

(b) Chronos

(c) Lag-LLaMA

Figure 11: Error visualization of Moirai, Chronos, and Lag-LLaMA on ETT-m1 ($L = 256$). The plotting procedure is the same as Fig. 6.

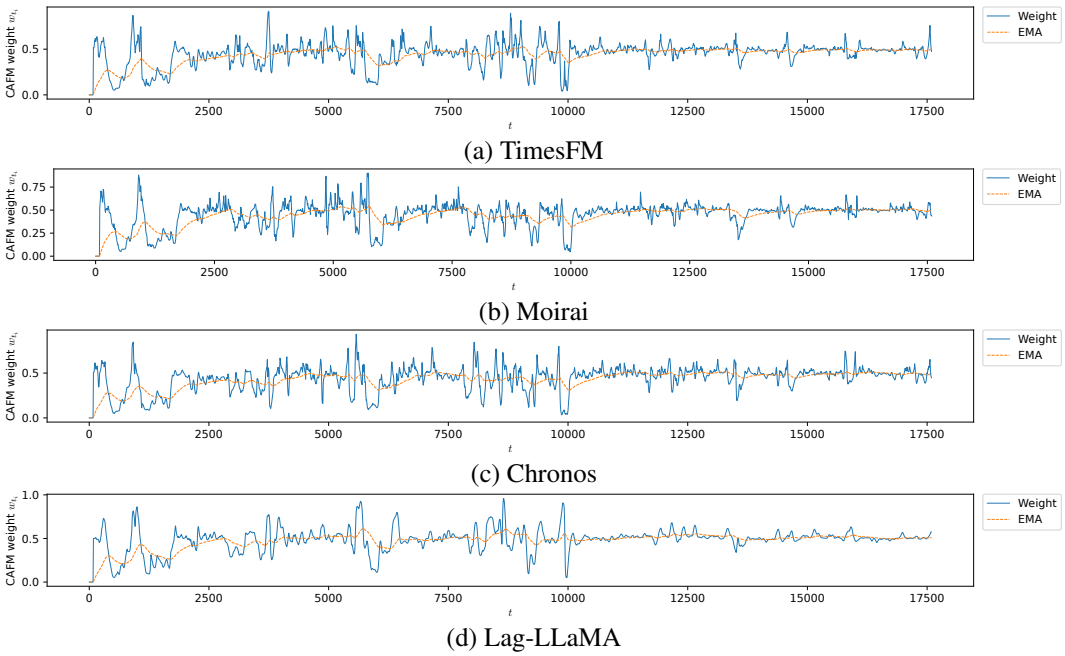


Figure 12: Transition of the CAFM weight w_{t_i} in Eqs. (9) and (12) on ETT-m1 (the oil temperature channel). The original w_{t_i} and its EMA are plotted.

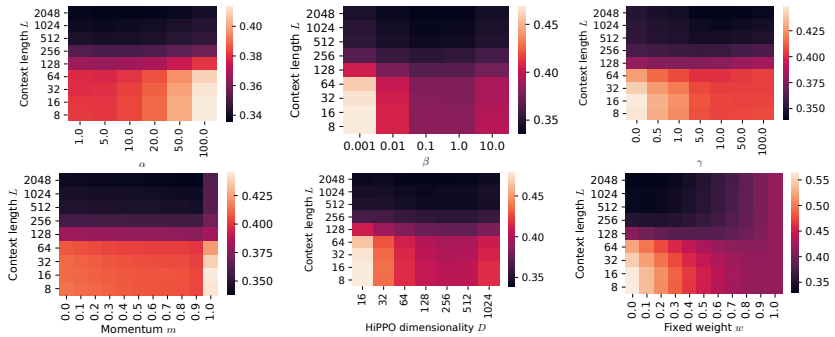


Figure 13: Sensitivity analysis on the hyperparameters of SAAF with Chronos on ETT-m1.

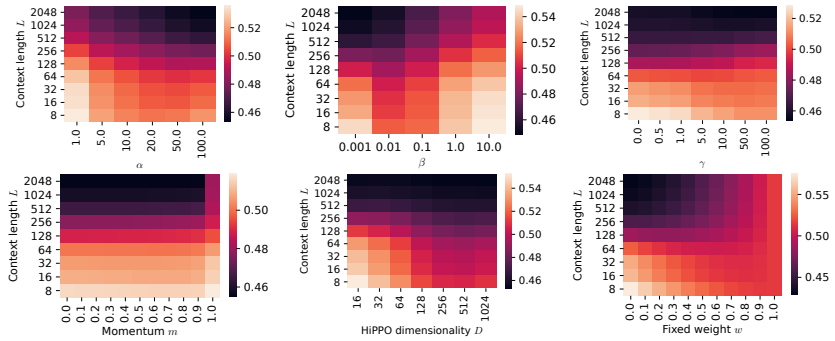


Figure 14: Sensitivity analysis on the hyperparameters of SAAF with TimesFM on Monash (London Smart Meters).

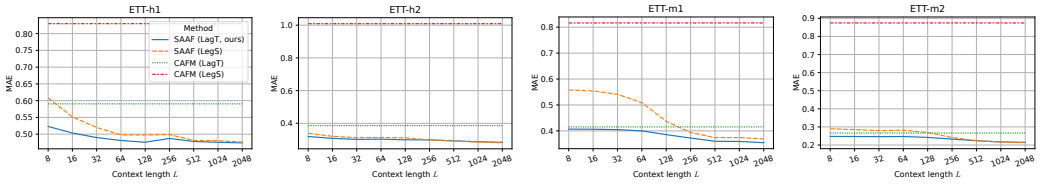
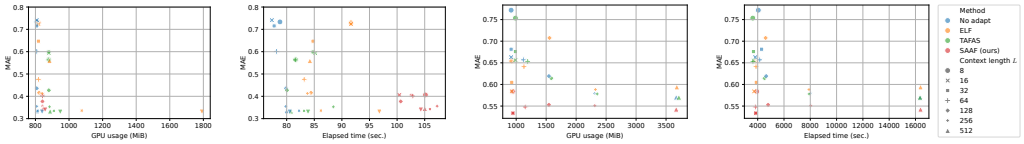
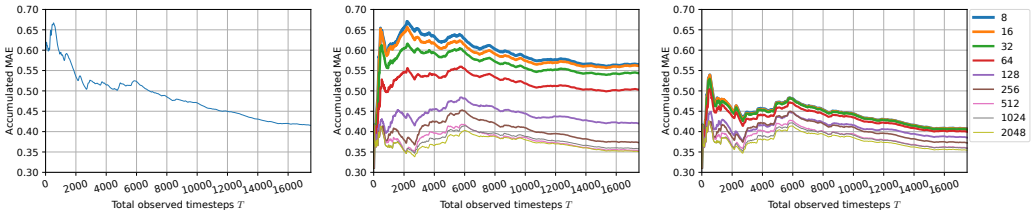


Figure 15: Forecast performance of TimesFM with SAAF and CAFM on the ETT dataset. The type of HiPPO is varied.



(a) Chronos-bolt (b) Lag-LLaMA

Figure 16: Computational costs of (a) Chronos-bolt on ETT-m1 and (b) Lag-LLaMA on ETT-h1. Each score is the average over three runs.



(a) CAFM (b) FTM (TimesFM) (c) SAAF (TimesFM)

Figure 17: Accumulated MAE vs. total timesteps T on ETT-m1.

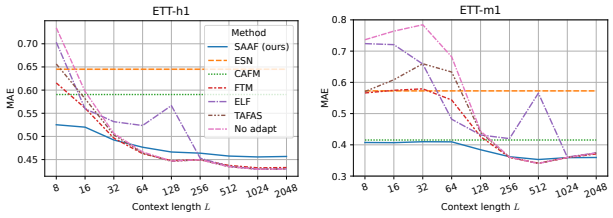


Figure 18: Forecast performance of TiRex and ESN on ETT-h1 and m1.