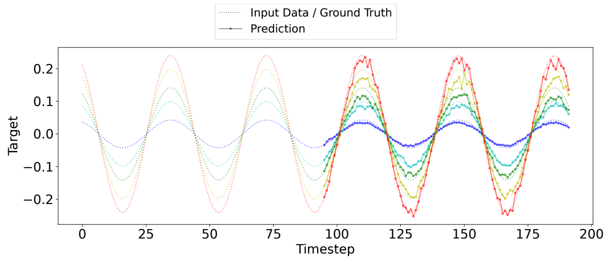
Data Adjustment Based on Model Characteristics for Few-Shot Time Series Forecasting

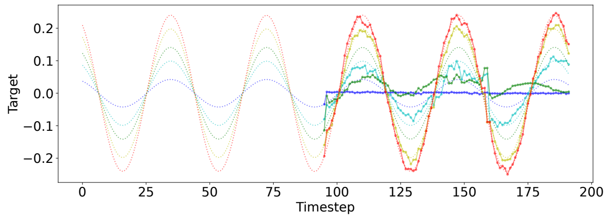
Yuna Saka, Tomoaki Yamazaki, Kouzou Ohara
Aoyama Gakuin University
5-10-1 Fuchinobe, Chuo-ku, Sagamihara-shi, Kanagawa, 252-5258 Japan
yuna.saka@dslabo.org, yamazaki@it.aoyama.ac.jp, ohara@it.aoyama.ac.jp

Abstract

Few-shot time series forecasting is crucial in real-world scenarios where training data is scarce. Among existing approaches, time series foundation models have achieved high accuracy on this task by leveraging their general knowledge of time series data. To further enhance these models, plug-and-play modules specialized for time series forecasting offer an effective way to improve performance with minimum computational costs. However, existing plug-and-play modules focus only on data characteristics and ignore the prediction characteristics unique to each model. To address this issue, we propose a plug-and-play module for few-shot time series forecasting that enhances the performance of time series foundation models by leveraging their prediction characteristics. The proposed module operates through two phases: the module optimization phase and the downstream forecasting phase. For the module optimization phase, the model performs forecasting using specialized datasets to obtain the trend and periodicity prediction characteristics as parameters. In the downstream forecasting phase, the parameters are used to adjust the trend and periodicity components of the input data to enhance the prediction ability of the model. Experimental results on real-world datasets show that incorporating the proposed module into existing time series foundation models yields comparable or better forecasting performance than using the models alone.



(a) Examples of predictions by GPT4TS (Zhou et al. 2023).



(b) Examples of predictions by Time-MoE (Shi et al. 2025).

Figure 1: Examples of predictions by two models on synthetic sinusoidal periodicity data with varying ranges in the 5% few-shot forecasting task. Although the input data is identical, the prediction characteristics are different between models.

Introduction

Time series forecasting is essential in various real-world scenarios, including sales forecasting for retail inventory management (Wellens, Boute, and Udenio 2024) and stock prediction for investment decision-making (Lu and Xu 2024). Most forecasting methods need to be trained on a large-scale time series dataset to capture complex temporal patterns and generate accurate predictions, but collecting a sufficiently large dataset requires significant time and effort (Jiang et al. 2025). To address this issue, few-shot time series forecasting, a variant of time series forecasting that uses only a small proportion of training data, is an effective solution.

One approach for handling few-shot time series forecasting is to leverage time series foundation models, which incorporate general knowledge of time series patterns by pretraining on a large dataset (Liang et al. 2024). Nevertheless, these models still have room for improvement, such as boosting their capability to capture properties unique to time series data (Liang et al. 2024). However, since time series foundation models require large-scale datasets for pretraining and contain a large number of parameters, modifying these models can become excessively costly.

To address the high computational costs of such models, plug-and-play modules tailored for time series forecasting can be utilized. These modules can be installed directly into an existing time series forecasting framework, allowing the reduction of the computational cost required for improving the performance of a model. Existing plug-and-play modules attempt to enhance the forecasting ability of a model by focusing on characteristics of the input data such as trend or periodicity. However, as indicated by the examples in Figure 1, different forecasting models possess unique forecasting characteristics, which is an essential property that existing modules overlook.

Regarding this issue, we propose a plug-and-play module for few-shot time series forecasting that considers the prediction characteristics inherent in different time series foundation models. The basic workflow of the proposed module consists of two phases, namely the module optimization phase and the downstream forecasting phase. During the module optimization phase, forecasting on synthetic datasets consisting of linear trend and sinusoidal periodicity is performed. Based on the differences in trend slopes and periodicity ranges between the predicted and true data, the proposed module learns the optimal degree of adjusting the input data for the model. In the downstream forecasting phase, trend and periodicity components are extracted from the input data and adjusted according to the learned degree to generate the model input. To evaluate the effectiveness of the proposed method, we conducted forecasting experiments on real-world datasets using existing time series foundation models. The results show that incorporating the proposed module can improve the performance of the models.

Related Work

Few-Shot Time Series Forecasting

Few-shot time series forecasting aims to predict time series data when the amount of training data is limited. One common approach is meta-learning. Frameworks such as BiLO-Auto-TSF/ML (Xu and Li 2022) and Feature-Adaptive Time Series Forecasting Framework (Ouyang et al. 2025) incorporate meta-learning to achieve accurate few-shot forecasting. Other methods adapt existing machine learning models, such as Bidirectional Long Short-Term Memory (Iwata and Kumagai 2020) and Siamese Network (Fan et al. 2025), for effective few-shot forecasting. In addition to these approaches, time series foundation models are also commonly used for this task.

Time Series Foundation Models

Time series foundation models, also known as large time series models, generally refer to deep time series models that conduct pre-training on a substantial dataset. By leveraging universal knowledge gained from pre-training, these models can effectively adapt to various downstream time series analysis tasks (Liang et al. 2024). Time series foundation models can be roughly classified into two categories based on how they acquire pre-trained knowledge. One approach is to conduct pre-training on time series datasets containing up to several billion data points, as adopted by models including Timer-XL (Liu et al. 2025) and Time-MoE (Shi et al. 2025). Another approach is to utilize Large Language Models (LLMs) as a backbone, as in ChatTime (Wang et al. 2025) and GPT4TS (Zhou et al. 2023). The latter approach leverages the cross-domain data processing knowledge of LLMs to enable the models to function without costly pre-training.

Plug-and-Play Modules for Time Series Forecasting

Plug-and-play modules for time series forecasting are often used with existing forecasting methods to improve their prediction performance. These modules can be directly installed into a forecasting framework without modifying the model architecture, thereby mitigating high computational costs. A popular approach is to decompose the input data into several components and predict them separately using an existing model and a different method, as in Dual-domain Dynamic Normalization (Dai et al. 2024) and Frequency Adaptive Normalization (Ye et al. 2024). Another common approach is to modify the data to emphasize its characteristics, as in Spectral Attention (Kang et al. 2024) and TimeBase (Huang et al. 2025).

Methodology

While utilizing plug-and-play modules is promising for improving the prediction accuracy of time series foundation models, existing modules fail to consider the intrinsic prediction characteristics of these models. To address this limitation, we propose a plug-and-play module that adjusts the input data by leveraging the prediction characteristics of the backbone time series foundation model.

Framework Overview

Figure 2 shows an overview of the few-shot time series forecasting framework implementing the proposed plug-and-play module. The goal of the proposed module is to adjust the input data before processing by the model to account for its prediction characteristics. This framework consists of two phases, namely the module optimization phase and the downstream forecasting phase. In the module optimization phase, a synthetic dataset containing linear trend and sinusoidal periodicity data is utilized to conduct forecasting with the backbone model. Based on the differences between the predicted and true data in terms of trend slopes and periodicity ranges, the optimal adjustment degrees are determined as parameters for the model. For the downstream forecasting phase, the proposed module first decomposes the input data into linear trend, sinusoidal periodicity, and residual components. Among these components, the trend and periodicity components are then adjusted using degrees calculated from the parameters optimized for the model. The adjusted components are integrated with the residual component to be used as the model input for forecasting. Note that if the model is determined to be incapable of predicting linear trend patterns during module optimization, a different adjustment procedure for the output trend component is conducted.

Acquirement of Model Prediction Characteristics

Before conducting forecasting with the proposed module, module optimization is performed to obtain the prediction characteristics of the backbone model. In this phase, specialized datasets consisting of synthetic linear trend and sinusoidal periodicity data are used. Through forecasting using these datasets and an additional test to determine the

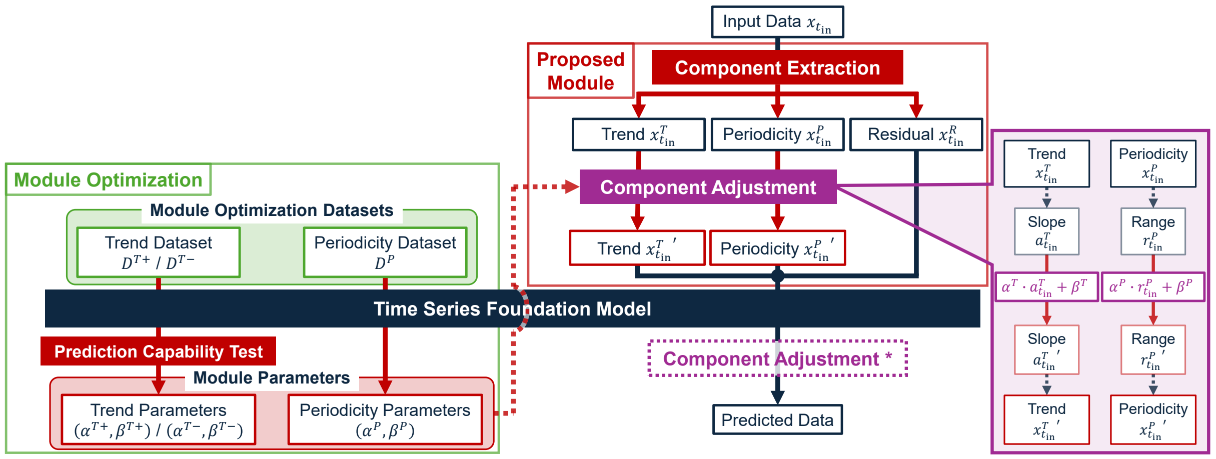

Input Data $x_{t_{\text{in}}}$
Proposed Module
Component Extraction
Trend $x^T_{t_{\text{in}}}$
Periodicity $x^P_{t_{\text{in}}}$
Residual $x^R_{t_{\text{in}}}$
Component Adjustment
Trend $x^T_{t_{\text{in}}}{}'$
Periodicity $x^P_{t_{\text{in}}}{}'$
Module Optimization
Module Optimization Datasets
Trend Dataset D^{T+} / D^{T-}
Periodicity Dataset D^P
Time Series Foundation Model
Prediction Capability Test
Module Parameters
Trend Parameters $(\alpha^{T+}, \beta^{T+})$ / $(\alpha^{T-}, \beta^{T-})$
Periodicity Parameters (α^P, β^P)
Component Adjustment *
Predicted Data
Trend $x^T_{t_{\text{in}}}$
Periodicity $x^P_{t_{\text{in}}}$
Slope $a^T_{t_{\text{in}}}$
Range $r^P_{t_{\text{in}}}$
$\alpha^T \cdot a^T_{t_{\text{in}}} + \beta^T$
$\alpha^P \cdot r^P_{t_{\text{in}}} + \beta^P$
Slope $a^T_{t_{\text{in}}}{}'$
Range $r^P_{t_{\text{in}}}{}'$
Trend $x^T_{t_{\text{in}}}{}'$
Periodicity $x^P_{t_{\text{in}}}{}'$

Figure 2: Overview of the few-shot forecasting framework incorporating the proposed module. During module optimization, prediction characteristics of trend and periodicity data are derived as parameters based on the predictions on specialized datasets by the backbone model. For downstream forecasting, the proposed module extracts components from the input data, adjusts them based on the module parameters, and merges them to create the model input. If the model fails the trend prediction capability test during module optimization, the output trend component adjustment with "*" is conducted.

trend prediction capability, the trend and periodicity prediction characteristics of the model are defined as parameters.

Module Optimization Datasets For module optimization, a specialized dataset consisting of synthetic trend and periodicity datasets is adopted. In detail, linear trends with positive and negative slopes given by $y = ax$ and $y = -ax$ are used as the trend datasets D^{T+} and D^{T-}, and a sinusoidal pattern represented by $y = \sin cx$ serves as the periodicity dataset D^P. The coefficients of the trend and periodicity equations are set to $a = 5$ and $c = 25$ in this study. These datasets each contain 15,000 data points, and the input sequence length l_{in} is assumed to be 96. As preprocessing, each dataset is first divided into the training, validation, and test data in the ratio of 7:1:2, and standardization is applied based on the training data. Then, each dataset is multiplied by five factors, $S^T = \{s^T_1, \ldots, s^T_5\}$ or $S^P = \{s^P_1, \ldots, s^P_5\}$, to adjust its range to approximately 0.1, 0.2, 0.3, 0.4, and 0.5. Specifically, $S^T = \{3.1, 6.3, 9.5, 12.7, 15.9\}$ is adopted for each trend dataset, and $S^P = \{0.03, 0.07, 0.10, 0.14, 0.17\}$ is adopted for the periodicity dataset. This results in five data variants for each dataset: $D^{T+} = \{D^{T+,s^T_n} \mid s^T_n \in S^T\}$ as the positive slope trend dataset, $D^{T-} = \{D^{T-,s^T_n} \mid s^T_n \in S^T\}$ as the negative slope trend dataset, and $D^P = \{D^{P,s^P_n} \mid s^P_n \in S^P\}$ as the periodicity dataset.

Characteristic Extraction from Forecasting Results Using the generated datasets, a few-shot forecasting task is conducted where the model optimized with a limited amount of training data makes predictions on the test data. From the predicted and true test data of each dataset, necessary characteristics are extracted to define the module parameters. For the linear trend datasets, the slope of each predicted and true sequence is derived by calculating their linear fit. The linear fit is derived using the Theil-Sen Estimator (Theil 1950; Sen 1968), a robust regression algorithm that can mitigate the effect of outliers in the data. Given an output sequence of length l_{out} with timesteps $t_{\text{out}} = (t_{l_{\text{in}}}, \ldots, t_{l_{\text{in}}+l_{\text{out}}-1})$ from a trend dataset D^{T,s^T_n} with the scale s^T_n, linear fitting using the Theil-Sen Estimator can be formalized as follows:

$$\text{TheilSen}(y^{T,s^T_n}_{t_{\text{out}}}) = a^{T,s^T_n}_{t_{\text{out}}} \cdot t_{\text{out}} + b^{T,s^T_n}_{t_{\text{out}}}, \tag{1}$$

where $y^{T,s^T_n}_{t_{\text{out}}}$ is either the predicted sequence $\hat{y}^{T,s^T_n}_{t_{\text{out}}}$ or the true sequence $\tilde{y}^{T,s^T_n}_{t_{\text{out}}}$, and $a^{T,s^T_n}_{t_{\text{out}}}$ and $b^{T,s^T_n}_{t_{\text{out}}}$ are the derived slope and intercept, respectively. The slopes $a^{T,s^T_n}_{t_{\text{out}}}$ calculated for all sequences in the test data are averaged for each dataset and scale variant:

$$a^{T,s^T_n} = \frac{1}{|\mathcal{T}_{\text{out}}|} \sum_{t_{\text{out}} \in \mathcal{T}_{\text{out}}} a^{T,s^T_n}_{t_{\text{out}}}, \tag{2}$$

where $\mathcal{T}_{\text{out}}$ is a set containing all output timestep sequences. The above procedures result in the predicted and true trend slopes for each positive slope dataset scale, $\hat{a}^{T+} = \{\hat{a}^{T+,s^T_n} \mid s^T_n \in S^T\}$ and $\tilde{a}^{T+} = \{\tilde{a}^{T+,s^T_n} \mid s^T_n \in S^T\}$, and negative slope dataset scale, $\hat{a}^{T-} = \{\hat{a}^{T-,s^T_n} \mid s^T_n \in S^T\}$ and $\tilde{a}^{T-} = \{\tilde{a}^{T-,s^T_n} \mid s^T_n \in S^T\}$.

For the sinusoidal periodicity dataset, the range of each predicted and true sequence is obtained by calculating the average local maxima and minima and taking their difference. First, the local maxima and minima of an output sequence for timesteps t_{out} are determined by checking all indices $i \in [l_{\text{in}} + order, l_{\text{in}} + l_{\text{out}} - 1 - order]$ against all neighboring indices $k \in [i - order, i + order]$ with $k \neq i$:

$$lmax_{t_{\text{out}}} = \{y^{P,s^P_n}_{t_i} \mid y^{P,s^P_n}_{t_i} > y^{P,s^P_n}_{t_k}\} \tag{3}$$

$$lmin_{t_{\text{out}}} = \{y^{P,s^P_n}_{t_i} \mid y^{P,s^P_n}_{t_i} < y^{P,s^P_n}_{t_k}\}, \tag{4}$$

where $y^{P,s^P_n}_{t_i}, y^{P,s^P_n}_{t_k} \in y^{P,s^P_n}_{t_{\text{out}}}$, with $y^{P,s^P_n}_{t_{\text{out}}}$ representing either the predicted sequence $\hat{y}^{P,s^P_n}_{t_{\text{out}}}$ or the true sequence $\tilde{y}^{P,s^P_n}_{t_{\text{out}}}$.

The variable *order* indicates the number of values before and after timestep t_i that are compared to determine if $y_{t_i}^{P,s_n^P}$ corresponds to a local maximum or minimum, which is set to 5 in this study. After obtaining $lmax_{t_{\text{out}}}$ and $lmin_{t_{\text{out}}}$ for all sequences in the test data, their average is derived for each scale variant:

$$lmax = \frac{1}{|\mathcal{L}_{lmax}|} \sum_{t_{\text{out}} \in \mathcal{T}_{\text{out}}} \sum_{v_{lmax} \in lmax_{t_{\text{out}}}} v_{lmax} \tag{5}$$

$$lmin = \frac{1}{|\mathcal{L}_{lmin}|} \sum_{t_{\text{out}} \in \mathcal{T}_{\text{out}}} \sum_{v_{lmin} \in lmin_{t_{\text{out}}}} v_{lmin}, \tag{6}$$

where $|\mathcal{L}_{lmax}|$ and $|\mathcal{L}_{lmin}|$ are the total numbers of local maxima and minima obtained from all output sequences. Using this result, the range r^{P,s_n^P} for each periodicity dataset scale is calculated:

$$r^{P,s_n^P} = lmax - lmin. \tag{7}$$

As a result, the predicted and true periodicity ranges for each dataset scale, namely $\hat{r}^P = \{\hat{r}^{P,s_n^P} \mid s_n^P \in S^P\}$ and $\tilde{r}^P = \{\tilde{r}^{P,s_n^P} \mid s_n^P \in S^P\}$, are obtained.

Trend Prediction Capability Test Meanwhile, some models lack the ability to produce reasonable predictions for linear trend data. These models often fail to capture differences in linear trend slopes, thereby producing similar results regardless of the slope. Defining trend parameters for such models and applying them during downstream forecasting can degrade forecasting accuracy. To avoid this issue, the model is evaluated to determine if it is capable of predicting linear trend data. For each trend dataset, the linear trend slopes derived by a model for different scale variants are tested under the following conditions:

(i) The average absolute difference between the predicted and true trend slopes is less than or equal to γ_a:

$$\frac{1}{|S^T|} \sum_{s_n^T \in S^T} |\hat{a}^{T,s_n^T} - \tilde{a}^{T,s_n^T}| \leq \gamma_a;$$

(ii) The average absolute difference between the predicted and true trend intercepts is less than or equal to γ_b:

$$\frac{1}{|S^T|} \sum_{s_n^T \in S^T} |\hat{b}^{T,s_n^T} - \tilde{b}^{T,s_n^T}| \leq \gamma_b;$$

(iii) The absolute difference between the range of the predicted and true trend intercepts is less than or equal to $\gamma_{\Delta b}$:

$$|(\max(\hat{b}^T) - \min(\hat{b}^T)) - (\max(\tilde{b}^T) - \min(\tilde{b}^T))| \leq \gamma_{\Delta b},$$

where $b^T = \{\hat{b}^T, \tilde{b}^T\}$ indicates the intercepts averaged for each dataset and scale variants as with Equation (2). The variables γ_a, γ_b, and $\gamma_{\Delta b}$ are thresholds for classifying a model based on its linear trend prediction capability under each condition, all set to 0.01 in this study. A model is considered to be capable of predicting positive or negative linear trend patterns if at least one of these conditions is met for the corresponding trend dataset. Otherwise, the model is considered incapable of predicting linear trend data.

Parameter Definition For each dataset, parameters are determined from the linear fitting results of the characteristics retrieved from the predicted and true data. Considering the predicted characteristics of each scale variant as explanatory variables and the true characteristics of each scale variant as response variables, linear fitting using the Theil-Sen Estimator is applied to the trend and periodicity datasets as follows:

$$\text{TheilSen}(\tilde{a}^T) = \alpha^T \cdot \hat{a}^T + \beta^T \tag{8}$$

$$\text{TheilSen}(\tilde{r}^P) = \alpha^P \cdot \hat{r}^P + \beta^P. \tag{9}$$

From the fitting results, the slope and intercept are determined as parameters of the corresponding dataset. That is, parameters α^{T+} and β^{T+} for the positive slope trend dataset, α^{T-} and β^{T-} for the negative slope trend dataset, and α^P and β^P for the periodicity dataset are defined. Note that if a model is considered incapable of predicting the positive or negative slope trend from the previously described evaluation, NaN values are set as parameters for the corresponding trend dataset.

Model Prediction Characteristics-Aware Data Adjustment

In the downstream few-shot time series forecasting task, the proposed module adjusts the trend and periodicity components of the input data using the module parameters defined for the backbone model. First, the linear trend and sinusoidal periodicity components are extracted from the input data. These components are then adjusted using the module parameters. Finally, the adjusted components are merged with the remaining component to construct the model input. In cases where the model is incapable of predicting linear trend patterns, the trend component adjustment is omitted, and an alternative procedure is applied to the model output.

Input Data Decomposition After receiving an input sequence for forecasting, the proposed module initially decomposes it into the linear trend component $x_{t_{\text{in}}}^T$, sinusoidal periodicity component $x_{t_{\text{in}}}^P$, and residual component $x_{t_{\text{in}}}^R$. First, the trend component is obtained by deriving the linear fit using the Theil-Sen Estimator for the input sequence and separating this result from the sequence. Assuming an input sequence $x_{t_{\text{in}}}$ for timesteps $t_{\text{in}} = (t_0, \dots, t_{l_{\text{in}}-1})$, its trend component $x_{t_{\text{in}}}^T$ can be obtained as follows:

$$\begin{aligned} x_{t_{\text{in}}}^T &= \text{TheilSen}(x_{t_{\text{in}}}) \\ &= a_{t_{\text{in}}}^T \cdot t_{\text{in}} + b_{t_{\text{in}}}^T, \end{aligned} \tag{10}$$

where $a_{t_{\text{in}}}^T$ and $b_{t_{\text{in}}}^T$ are the slope and intercept of the fitted linear trend data, respectively. The derived trend component is then subtracted from the input sequence:

$$x_{t_{\text{in}}}^{R'} = x_{t_{\text{in}}} - x_{t_{\text{in}}}^T, \tag{11}$$

where $x_{t_{\text{in}}}^{R'}$ is the input sequence with the trend component removed.

Next, the periodicity component is obtained by isolating the strongest frequency component from $x_{t_{\text{in}}}^{R'}$. To identify the frequency component to extract, the Discrete Fourier Transform (DFT) for real inputs calculated with the Fast Fourier Transform (FFT) algorithm (Cooley and Tukey 1965) is used. Given a value $x^{R'}_{t_i} \in x^{R'}_{t_{\text{in}}}$ at timestep t_i, the FFT for real inputs is applied as follows:

$$\text{rFFT}(x^{R'}_{t_i}) = f_k. \tag{12}$$

$f_k \in f$ is a frequency component value with an index $0 \leq k < k_{\text{max}}$, where k_{max} can be expressed as:

$$k_{\text{max}} = \begin{cases} \frac{t_i}{2} + 1 & \text{if } t_i = 2m \\ \frac{t_i+1}{2} & \text{if } t_i = 2m+1, \end{cases} \tag{13}$$

assuming m is an arbitrary natural number. From the obtained component, the strongest frequency value $f_{\max}$ is identified:

$$f_{\max} = \begin{cases} \max_{k' \geq 0}(|f_{k'}|) & \text{if } \max_{k' \geq 0}(|f_{k'}|) \neq |f_0| \\ \max_{k' \geq 1}(|f_{k'}|) & \text{if } \max_{k' \geq 0}(|f_{k'}|) = |f_0|. \end{cases} \tag{14}$$

As indicated in the equation above, f_0 is not used as $f_{\max}$ since this represents the zero-frequency term. After obtaining $f_{\max}$, all frequency component values except this value are set to 0:

$$f'_k = \begin{cases} f_k & \text{if } f_k = f_{\max} \\ 0 & \text{if } f_k \neq f_{\max}. \end{cases} \tag{15}$$

The retrieved frequency component value $f'_k \in f'$ is converted back to the time domain using the inverse FFT for real inputs:

$$x^P_{t_i} = \text{irFFT}(f'_k), \tag{16}$$

which results in $x^P_{t_i} \in x^P_{t_{\text{in}}}$, a value of the desired periodicity component. The derived periodicity component $x^P_{t_{\text{in}}}$ is subtracted from $x^{R'}_{t_{\text{in}}}$:

$$x^R_{t_{\text{in}}} = x^{R'}_{t_{\text{in}}} - x^P_{t_{\text{in}}}. \tag{17}$$

The remaining component $x^R_{t_{\text{in}}}$ is considered as the residual component of the input sequence.

Component Adjustment Among the three components of the input sequence, the trend component $x^T_{t_{\text{in}}}$ and the periodicity component $x^P_{t_{\text{in}}}$ undergo an adjustment procedure using the module parameters. For the trend component, x in $y = \alpha^T x + \beta^T$ is replaced with the slope $a^T_{t_{\text{in}}}$ retrieved from Equation (10) to derive the adjusted slope $a^{T}_{t_{\text{in}}}{}'$:

$$a^{T}_{t_{\text{in}}}{}' = \begin{cases} \alpha^{T+} \cdot a^T_{t_{\text{in}}} + \beta^{T+} & \text{if } a^T_{t_{\text{in}}} \geq 0 \\ \alpha^{T-} \cdot a^T_{t_{\text{in}}} + \beta^{T-} & \text{if } a^T_{t_{\text{in}}} < 0. \end{cases} \tag{18}$$

As shown in the equation, either α^{T+} and β^{T+}, the parameters for the positive slope trend, or α^{T-} and β^{T-}, the parameters for the negative slope trend, are applied based on the sign of $a^T_{t_{\text{in}}}$. Using this result, the adjusted trend component $x^{T}_{t_{\text{in}}}{}'$ is derived by replacing the slope $a^T_{t_{\text{in}}}$ in Equation (10) with the adjusted slope $a^{T}_{t_{\text{in}}}{}'$:

$$x^{T}_{t_{\text{in}}}{}' = a^{T}_{t_{\text{in}}}{}' \cdot t_{\text{in}} + b^T_{t_{\text{in}}}. \tag{19}$$

	Dim.	Freq.	Data Points		
			Train (5%)	Valid	Test
ETTh1	7	60	241	2,689	2,689
ETTm1	7	15	1,537	11,329	11,329
Weather	21	10	1,653	5,079	10,349

Table 1: Specifications of each dataset. “Dim.” refers to the number of features, and “Freq.” indicates the observation interval in minutes.

Note that if α^T and β^T are set to NaN, indicating that the forecasting model is incapable of predicting linear trend patterns, this adjustment procedure is not performed.

To adjust the periodicity component $x^P_{t_{\text{in}}}$, the range is computed as follows:

$$r^P_{t_{\text{in}}} = \max(x^P_{t_{\text{in}}}) - \min(x^P_{t_{\text{in}}}). \tag{20}$$

The derived range $r^P_{t_{\text{in}}}$ is adjusted using the module parameters as in Equation (18):

$$r^{P}_{t_{\text{in}}}{}' = \alpha^P \cdot r^P_{t_{\text{in}}} + \beta^P. \tag{21}$$

Finally, the adjusted periodicity component $x^{P}_{t_{\text{in}}}{}'$ is calculated as follows:

$$x^{P}_{t_{\text{in}}}{}' = x^P_{t_{\text{in}}} \cdot \frac{r^{P}_{t_{\text{in}}}{}'}{r^P_{t_{\text{in}}}}. \tag{22}$$

After obtaining the adjusted trend component $x^{T}_{t_{\text{in}}}{}'$ and periodicity component $x^{P}_{t_{\text{in}}}{}'$, these are combined with the residual component $x^R_{t_{\text{in}}}$ to produce the adjusted input sequence:

$$x_{t_{\text{in}}}{}' = x^{T}_{t_{\text{in}}}{}' + x^{P}_{t_{\text{in}}}{}' + x^R_{t_{\text{in}}}. \tag{23}$$

The resulting sequence $x_{t_{\text{in}}}{}'$ serves as the model input for prediction.

If the trend adjustment procedure is skipped due to the model lacking the ability to predict linear trend patterns, the model output trend slope $a^T_{t_{\text{out}}}$ is substituted with the input trend slope $a^T_{t_{\text{in}}}$. In detail, the trend component of the output sequence is extracted similarly to Equation (10), and its slope $a^T_{t_{\text{out}}}$ is replaced with the input slope $a^T_{t_{\text{in}}}$.

Experiments

Experimental Setup

Datasets For the downstream few-shot time series forecasting task, three real-world datasets are used. Table 1 summarizes the specifications of each dataset. ETTh1 and ETTm1 are included in the Electricity Transformer Temperature (ETT) dataset (Zhou et al. 2021), which contains power load data from electricity transformers placed in a county in China. Weather (Wu et al. 2021) contains climate data provided by the Max Planck Institute for Biogeochemistry. Based on the experimental setup of Zhou et al. (Zhou et al. 2023), the ratio of the training, validation, and test data is initially set to 3:1:1 for the ETT datasets and 7:1:2 for the Weather dataset, and then only 5% of the training data is used for training.

	GPT4TS	Timer-XL	Time-MoE
α^{T+}	0.87367	NaN	NaN
β^{T+}	-0.00004	NaN	NaN
α^{T-}	0.81741	NaN	NaN
β^{T-}	-0.00039	NaN	NaN
α^{P}	0.98892	0.99606	0.77853
β^{P}	0.01455	0.00067	0.09516

Table 2: Parameters determined for each model during module optimization.

Baselines In the experiments, three time series foundation models, GPT4TS (Zhou et al. 2023), Timer-XL (Liu et al. 2025), and Time-MoE (Shi et al. 2025), are used as backbones for the proposed module and as baselines for comparison. GPT4TS is implemented based on an existing repository[1], while Timer-XL and Time-MoE are implemented using publicly available pre-trained checkpoints[2][3]. Note that, for Timer-XL and Time-MoE, the datasets used for the downstream forecasting phase are not among those used to pre-train these models.

Experimental Settings The forecasting framework is implemented based on the Time Series Library repository[4], with some modifications to support the required functions. Notably, for Timer-XL and Time-MoE, the model output processing is modified to allow the prediction length to be flexibly changed. In this experiment, the input and prediction length of the model are both set to 96. Other major settings include the batch size set to 32, the maximum training epochs set to 10, and the random seed set to 2021. As evaluation metrics, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used.

Results

Module Optimization Table 2 shows the parameters obtained for each model during module optimization. Remarkably, NaN values were set as the trend parameters for Timer-XL and Time-MoE, indicating that these models are incapable of predicting linear trend patterns. In contrast, GPT4TS had parameters defined for both the positive and negative slope trend data, with slightly different values of α^T and β^T. For the periodicity parameters, all models had α near 1.0 and β near 0.0, but their values were considerably different across models. These results suggest that existing time series foundation models can be distinguished to a certain extent with the proposed parameters.

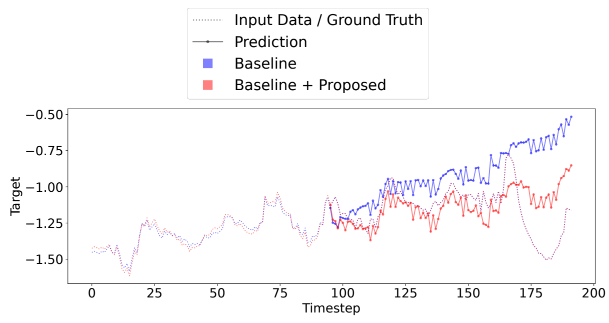
Downstream Forecasting Task Table 3 presents the forecasting results for each dataset. Overall, models with the proposed module performed comparably to or better than the baselines in prediction accuracy. Specifically, for GPT4TS,

[1]https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All

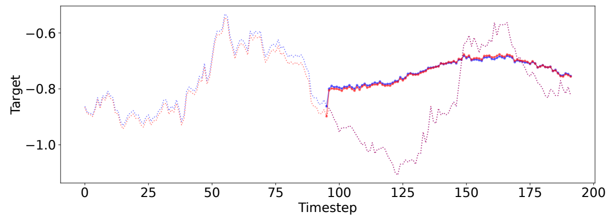
[2]https://huggingface.co/thuml/timer-base-84m

[3]https://huggingface.co/Maple728/TimeMoE-50M

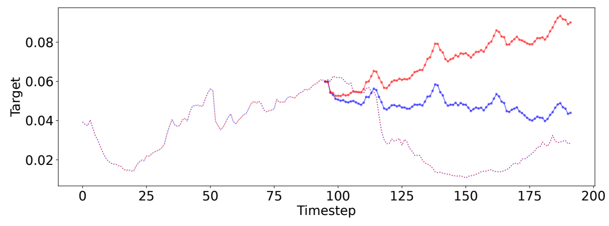
[4]https://github.com/thuml/Time-Series-Library



(a) Example of predictions for the ETTh1 dataset using Time-MoE as the baseline. The module improved the MSE of the baseline from 0.12581 to 0.02835.



(b) Example of predictions for the ETTm1 dataset using GPT4TS as the baseline. The module had little effect on the MSE of the baseline, only changing it from 0.02897 to 0.02873.



(c) Example of predictions for the Weather dataset using Timer-XL as the baseline. The module worsened the MSE of the baseline from 0.00063 to 0.00249.

Figure 3: Examples of predictions on each dataset by different models.

the accuracy improved by a small margin for the ETTm1 dataset, but it was slightly worse for the other two datasets. For Timer-XL and Time-MoE, the proposed module increased the prediction accuracy, especially for the ETTm1 dataset, yet the accuracy was relatively worse for the Weather dataset. These results suggest that the proposed module can partially improve the accuracy of forecasting by existing time series foundation models.

Figure 3 displays examples of input sequences and the corresponding predicted and true output sequences. Each figure illustrates cases where the proposed module improved, did not affect, and worsened the MSE of the model. Figure 3a shows predictions on an input sequence from ETTh1 by Time-MoE alone and with the proposed module. In this case, the configuration incorporating the proposed module was successful in capturing the peaks of the ground truth, indicating the ability to better capture the periodicity. Figure 3b displays predictions on ETTm1 by GPT4TS

		GPT4TS		Timer-XL		Time-MoE	
Dataset	Metric	Baseline	+ Prop.	Baseline	+ Prop.	Baseline	+ Prop.
ETTh1	MSE	**0.08964**	0.09449	0.19290	**0.13405**	0.14405	**0.11891**
	MAE	**0.22709**	0.23383	0.35363	**0.29293**	0.28905	**0.26266**
ETTm1	MSE	0.03439	**0.03388**	0.20506	**0.09367**	0.26044	**0.06055**
	MAE	0.13864	**0.13796**	0.36994	**0.24612**	0.41072	**0.18370**
Weather	MSE	**0.00119**	0.00120	**0.00123**	0.00220	**0.00181**	0.00252
	MAE	**0.02537**	0.02567	**0.02607**	0.03452	**0.03044**	0.03723

Table 3: 5% few-shot time series forecasting results on each dataset. "+ Prop." represents the model incorporating the proposed module. Values shown in bold indicate the best result among the same model and dataset.

and the same model with the proposed module. The two predictions were similar, suggesting that the proposed module does not always contribute to significantly affecting the model performance. Figure 3c presents the predictions on the Weather dataset by Timer-XL and its variants with the proposed module. In this example, the model with the proposed module predicted farther off from the ground truth than the baseline, decreasing the prediction performance. Overall, these results show that the proposed plug-and-play module can improve the prediction performance of existing models in certain cases, while also revealing the need for further refinement.

Ablation Study on Trend Adjustment As an ablation study, the prediction performance of the proposed module and its variant without the trend replacement procedure for models incapable of predicting linear trend patterns is evaluated. Table 4 compares the proposed module with and without this procedure. In this experiment, Timer-XL and Time-MoE were used as backbones since they were considered unable to predict linear trend data during module optimization. For both models, applying the proposed module with trend replacement improved the accuracy on ETTh1 and ETTm1, but the accuracy was relatively low on the Weather dataset. These results suggest that substituting the output trend slope with the input trend slope can be an effective strategy for models that are not capable of predicting linear trend data.

Dataset	Metric	+ Prop.	+ Prop. w/o TRep
ETTh1	MSE	**0.13405**	0.19250
	MAE	**0.29293**	0.35331
ETTm1	MSE	**0.09367**	0.20557
	MAE	**0.24612**	0.37080
Weather	MSE	0.00220	**0.00124**
	MAE	0.03452	**0.02609**

(a) Forecasting results of Timer-XL with different configurations of the proposed method.

Dataset	Metric	+ Prop.	+ Prop. w/o TRep
ETTh1	MSE	**0.11891**	0.15111
	MAE	**0.26266**	0.29376
ETTm1	MSE	**0.06055**	0.27126
	MAE	**0.18370**	0.41387
Weather	MSE	0.00252	**0.00173**
	MAE	0.03723	**0.03032**

(b) Forecasting results of Time-MoE with different configurations of the proposed method.

Table 4: 5% few-shot time series forecasting results by models incapable of linear trend prediction using different configurations of the proposed module. "+ Prop. w/o TRep" denotes the proposed module without the trend replacement procedure.

Conclusion and Future Work

In this paper, we propose a plug-and-play module for few-shot time series forecasting that considers the prediction characteristics of the time series foundation model used for forecasting. The proposed module captures the trend and periodicity prediction characteristics of the backbone model as parameters obtained from module optimization using specialized datasets, and employs them to adjust the corresponding components of the input sequence during the downstream forecasting phase. Experiments on real-world datasets demonstrate that installing the proposed module into existing models leads to comparable or better prediction performance than using the models alone.

As future work, refining methods for defining module parameters and adjusting the input data may further enhance the forecasting ability of the proposed module. In determining the parameters, considering additional time series characteristics such as nonlinear trend or periodicity with multiple frequencies as the module optimization dataset can expand the ability of the module to capture the prediction characteristics of diverse models. For input data adjustment, refining the adjustment procedures, especially those specialized for models incapable of predicting linear trend patterns, may enable the module to generate data more consistent with the prediction characteristics of the model. In addition, carrying out experiments under diverse settings, such as longer prediction lengths, could provide additional insights into the forecasting performance of the module. Furthermore, comparing the prediction accuracy of the proposed module against existing modules can further verify its superiority. It would also be worthwhile to evaluate how the proposed module affects the forecasting performance when used along with existing modules.

References

Cooley, J. W.; and Tukey, J. W. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90): 297–301.

Dai, T.; Wu, B.; Liu, P.; Li, N.; Yuerong, X.; Xia, S.-T.; and Zhu, Z. 2024. DDN: Dual-domain Dynamic Normalization for Non-stationary Time Series Forecasting. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 37, 108490–108517.

Fan, J.; Xiang, J.; Liu, J.; Wang, Z.; and Wu, H. 2025. Long-term time series forecasting based on Siamese network: a perspective on few-shot learning. *International Journal of Machine Learning and Cybernetics*, 16(2): 999–1014.

Huang, Q.; Zhou, Z.; Yang, K.; Yi, Z.; Wang, X.; and Wang, Y. 2025. TimeBase: The Power of Minimalism in Efficient Long-term Time Series Forecasting. In *Proceedings of the Forty-second International Conference on Machine Learning*.

Iwata, T.; and Kumagai, A. 2020. Few-shot Learning for Time-series Forecasting. arXiv:2009.14379.

Jiang, Y.; Chen, Y.; Li, X.; Chao, Q.; Liu, S.; and Cong, G. 2025. FSTLLM: Spatio-Temporal LLM for Few Shot Time Series Forecasting. In *Proceedings of the Forty-second International Conference on Machine Learning*.

Kang, B. G.; Lee, D.; Kim, H.; Chung, D.; and Yoon, S. 2024. Introducing Spectral Attention for Long-Range Dependency in Time Series Forecasting. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 37, 136509–136544.

Liang, Y.; Wen, H.; Nie, Y.; Jiang, Y.; Jin, M.; Song, D.; Pan, S.; and Wen, Q. 2024. Foundation Models for Time Series Analysis: A Tutorial and Survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6555–6565.

Liu, Y.; Qin, G.; Huang, X.; Wang, J.; and Long, M. 2025. Timer-XL: Long-Context Transformers for Unified Time Series Forecasting. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

Lu, M.; and Xu, X. 2024. TRNN: An efficient time-series recurrent neural network for stock price prediction. *Information Sciences*, 657: 119951.

Ouyang, P.; Chen, D.; Yang, T.; Feng, S.; Jin, Z.; and Xu, M. 2025. FAF: A Feature-Adaptive Framework for Few-Shot Time Series Forecasting. arXiv:2506.19567.

Sen, P. K. 1968. Estimates of the Regression Coefficient Based on Kendall's Tau. *Journal of the American Statistical Association*, 63(324): 1379–1389.

Shi, X.; Wang, S.; Nie, Y.; Li, D.; Ye, Z.; Wen, Q.; and Jin, M. 2025. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

Theil, H. 1950. A rank-invariant method of linear and polynomial regression analysis. In *Proceedings of the Royal Netherlands Academy of Sciences*, volume 53, 386–392.

Wang, C.; Qi, Q.; Wang, J.; Sun, H.; Zhuang, Z.; Wu, J.; Zhang, L.; and Liao, J. 2025. ChatTime: A Unified Multimodal Time Series Foundation Model Bridging Numerical and Textual Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12): 12694–12702.

Wellens, A. P.; Boute, R. N.; and Udenio, M. 2024. Simplifying tree-based methods for retail sales forecasting with explanatory variables. *European Journal of Operational Research*, 314(2): 523–539.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 34, 22419–22430.

Xu, J.; and Li, K. 2022. Automated Few-Shot Time Series Forecasting based on Bi-level Programming. arXiv:2203.03328.

Ye, W.; Deng, S.; Zou, Q.; and Gui, N. 2024. Frequency Adaptive Normalization For Non-stationary Time Series Forecasting. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 37, 31350–31379.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11106–11115.

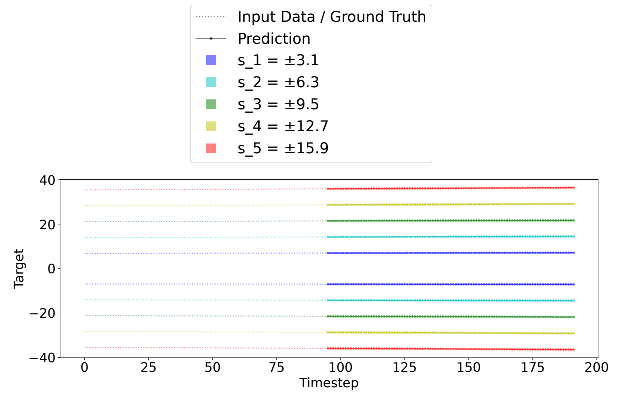
Zhou, T.; Niu, P.; Wang, X.; Sun, L.; and Jin, R. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, 43322–43355.

Appendix

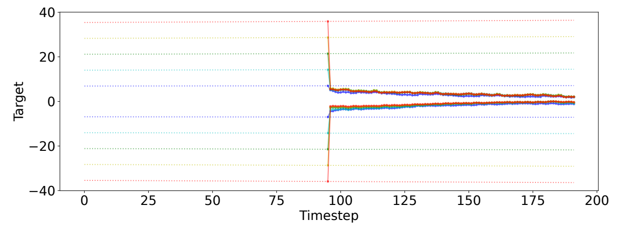
Additional Experiment Results

Predictions on Module Optimization Datasets Figures 4 and 5 show prediction examples by each model on the linear trend and sinusoidal periodicity module optimization datasets, respectively. For the trend datasets, GPT4TS succeeded in making predictions that were consistent with each input slope and intercept. In contrast, Timer-XL and Time-MoE generated similar predictions regardless of input slopes and intercepts, indicating that these models struggle to predict linear trend patterns. For the periodicity datasets, all models were able to accurately capture data fluctuations, although the noisiness of the predictions varied across models and periodicity ranges.

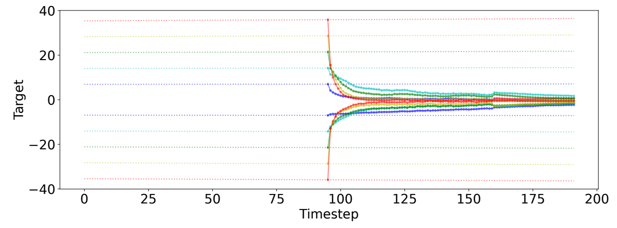
Trend Prediction Capability Test Results Table 5 shows the results of the evaluation conducted during module optimization to determine if a model is capable of predicting linear trend patterns. For both the positive and negative slope datasets, GPT4TS fulfilled most of the conditions, indicating that the model has the ability to predict linear trend data. However, Timer-XL and Time-MoE did not satisfy all of the conditions and were therefore determined to be incapable of predicting linear trend data. These results are consistent with the visualized linear trend predictions in Figure 4, suggesting that this evaluation effectively distinguishes models that can and cannot predict linear trend patterns.



(a) Example of predictions on the trend dataset by GPT4TS.



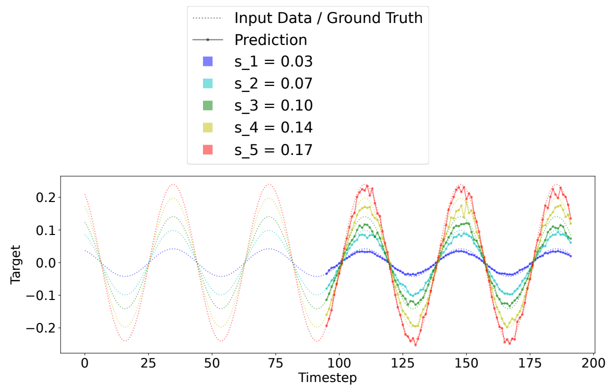
(b) Example of predictions on the trend dataset by Timer-XL.



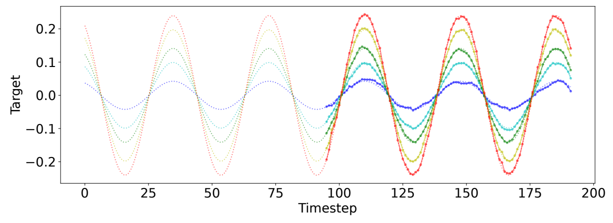
(c) Example of predictions on the trend dataset by Time-MoE.

Figure 4: Examples of predictions on each linear trend module optimization dataset by different models.

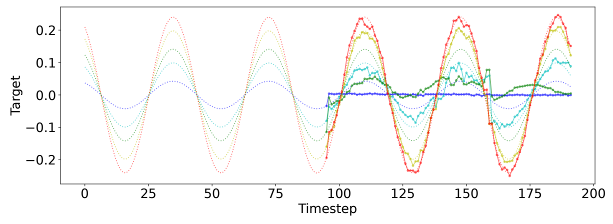
Visualization of Parameters Figures 6 and 7 are visualizations of the parameters from module optimization, determined according to Equations (8) and (9). In Figure 6, each data point is plotted with the average predicted slopes $\hat{a}^T$ on the horizontal axis and the average true slopes $\tilde{a}^T$ on the vertical axis, and the dotted line indicates their linear fit. Since the parameters were not defined for Timer-XL and Time-MoE due to their inability to predict linear trend data, only the visualizations for GPT4TS are displayed. Similarly, in Figure 7, each data point is plotted based on the average predicted and true ranges $\hat{r}^P$ and $\tilde{r}^P$, and their linear fit is visualized. In most cases, all data lay along the linear fit line, indicating that the relationship between the predicted and true data characteristics can be effectively expressed as a linear equation. On the other hand, the cases shown in Figures 6a and 7c each had an outlier, but these did not affect the linear fit result. This implies that using the Theil-Sen Estimator for linear fitting effectively prevents irrelevant parameters from being defined by outliers.



(a) Example of predictions on the periodicity dataset by GPT4TS.



(b) Example of predictions on the periodicity dataset by Timer-XL.



(c) Example of predictions on the periodicity dataset by Time-MoE.

Figure 5: Examples of predictions on each sinusoidal periodicity module optimization dataset by different models.

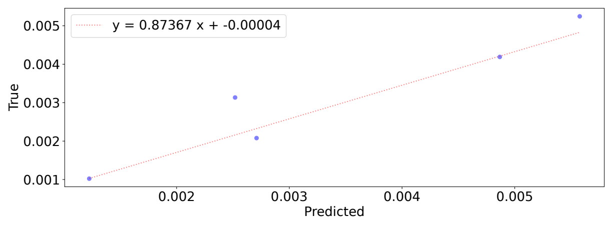
	GPT4TS	Timer-XL	Time-MoE
(i)	✓ (0.00049)	✗ (0.03205)	✗ (0.02462)
(ii)	✓ (0.01336)	✗ (21.10964)	✗ (24.51384)
(iii)	✓ (0.00854)	✗ (33.88346)	✗ (30.90542)
Capability	✓	✗	✗

(a) Trend prediction capability test results on each model for the positive slope trend dataset.

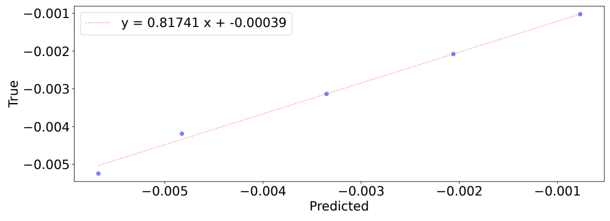
	GPT4TS	Timer-XL	Time-MoE
(i)	✓ (0.00031)	✗ (0.03574)	✗ (0.03199)
(ii)	✓ (0.00839)	✗ (22.95760)	✗ (22.34476)
(iii)	✗ (0.02714)	✗ (33.34745)	✗ (29.30762)
Capability	✓	✗	✗

(b) Trend prediction capability test results on each model for the negative slope trend dataset.

Table 5: The results of the trend prediction capability test on each model for each condition. "✓" means the model met the corresponding condition, and "✗" means the model did not meet the condition. The numbers in parentheses in rows (i), (ii), and (iii) indicate the average absolute slope difference, average absolute intercept difference, and maximum and minimum intercept deviation difference of the predicted and true data, respectively.

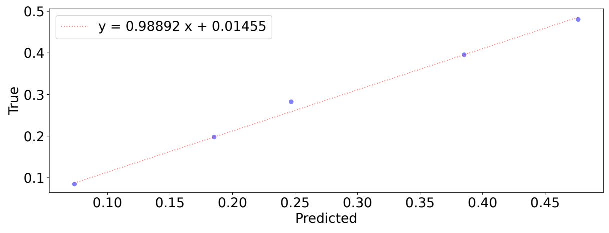


(a) Parameters of GPT4TS for the positive slope trend dataset.

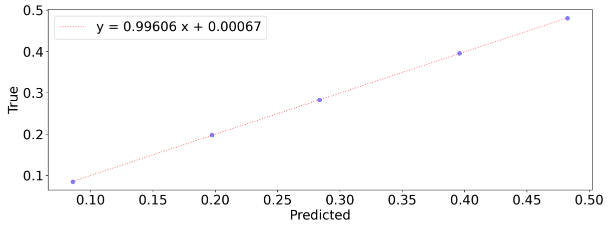


(b) Parameters of GPT4TS for the negative slope trend dataset.

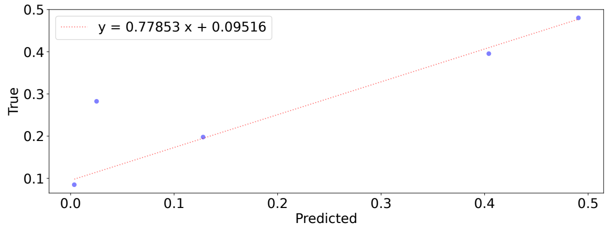
Figure 6: Visualization of module parameters obtained from the trend datasets.



(a) Parameters of GPT4TS for the periodicity dataset.



(b) Parameters of Timer-XL for the periodicity dataset.



(c) Parameters of Time-MoE for the periodicity dataset.

Figure 7: Visualization of module parameters obtained from the periodicity dataset.