Modeling Temporal Dependencies within the Target for Long-Term Time Series Forecasting

Qi Xiong, Kai Tang, Minbo Ma, Ji Zhang, Jie Xu, and Tianrui Li* *Senior Member, IEEE*

***Abstract*—Long-term time series forecasting (LTSF) is a critical task across diverse domains. Despite significant advancements in LTSF research, we identify a performance bottleneck in existing LTSF methods caused by the inadequate modeling of Temporal Dependencies within the Target (TDT). To address this issue, we propose a novel and generic temporal modeling framework, Temporal Dependency Alignment (TDAlign), that equips existing LTSF methods with TDT learning capabilities. TDAlign introduces two key innovations: 1) a loss function that aligns the change values between adjacent time steps in the predictions with those in the target, ensuring consistency with variation patterns, and 2) an adaptive loss balancing strategy that seamlessly integrates the new loss function with existing LTSF methods without introducing additional learnable parameters. As a plug-and-play framework, TDAlign enhances existing methods with minimal computational overhead, featuring only linear time complexity and constant space complexity relative to the prediction length. Extensive experiments on six strong LTSF baselines across seven real-world datasets demonstrate the effectiveness and flexibility of TDAlign. On average, TDAlign reduces baseline prediction errors by 1.47% to 9.19% and change value errors by 4.57% to 15.78%, highlighting its substantial performance improvements.**

***Index Terms*—Time series analysis, long-term forecasting, temporal dependency.**

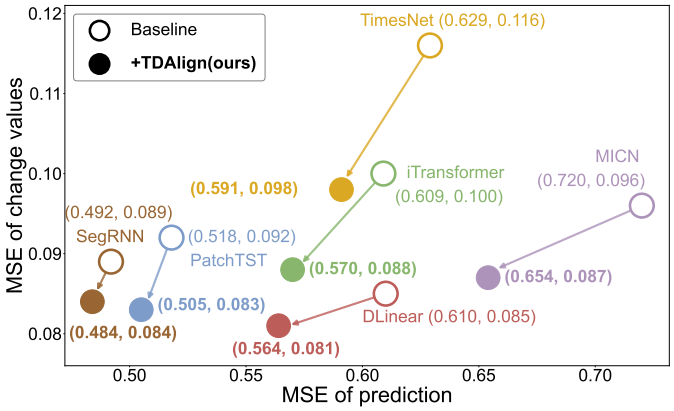


Fig. 1: Performance comparisons between methods with and without our proposed TDAlign across six baseline methods. The x-axis presents the Mean Squared Error (MSE) scores, while the y-axis displays the MSE of change values between adjacent time steps within the prediction. The results are averaged over seven datasets. For MSE, a smaller value indicates better performance.

I. Introduction

Time series forecasting refers to predicting the target future time series based on the input historical time series. It has garnered significant attention due to its crucial applications in various real-world domains, such as stock price prediction [1], [2], traffic planning [3], [4], energy demand analysis [5], [6], and extreme weather warning [7], [8]. In the early stage, numerous studies [9] focused on short-term time series forecasting, typically predicting 48 time steps or fewer (e.g., hours or days) [10], [11], [12]. However, these methods often fail to meet the predictive capacity required for long-term time series forecasting (LTSF) in many practical scenarios, where anticipating future trends over extended periods, such as 720 time steps, is essential for significantly ahead in resource allocation, risk management, and decision-making. In response, recent studies present various neural networks to capture long-range temporal dependencies from input historical time series, including Multi-Layer Perceptrons (MLPs) [13], [14], [15], Convolutional Neural Networks (CNNs) [16], [17], [18], Transformer [19], [20], [21], and Recurrent Neural Networks (RNNs) [22].

Despite substantial progress, our in-depth analysis of temporal relations between adjacent time steps reveals that existing LTSF methods lack adequate modeling of *temporal dependencies within the target* (TDT), which significantly limits the method's capacity to generate accurate and realistic predictions (see the analysis in Section III-B for details). Prior efforts such as [15], [23], [22] introduce position/timestamp embeddings (e.g., minute, hour, week, month, and year) to preserve the temporal ordering information within the target. While modeling temporal feature discrepancy, such implicit learning cannot capture dynamic temporal patterns that evolve over time [14], [24]. The autoregressive decoding strategy is another relevant solution that feeds the former prediction into the next step operation, naturally learning sequential relationships during the step-by-step inference process [25], [26]. However, it suffers two major drawbacks: error accumulation and computational overhead, where both forecast bias and inference time rapidly increase as the forecast horizon expands [27], [28].

Qi Xiong, Kai Tang, Minbo Ma, Ji Zhang, and Tianrui Li (the corresponding author) are with the School of Computing and Artificial Intelligence, Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Manufacturing Industry Chains Collaboration and Information Support Technology Key Laboratory of Sichuan Province, Southwest Jiaotong University, Chengdu 611756, China (e-mails: xiongqi@my.swjtu.edu.cn; tangkailh@outlook.com; minboma@my.swjtu.edu.cn; jizhang.jim@gmail.com; trli@swjtu.edu.cn).

Jie Xu is with the School of Computing, The University of Leeds, LS2 9JT Leeds, U.K. (e-mail: j.xu@leeds.ac.uk).

In this work, we propose Temporal Dependency Alignment (TDAlign), a generic framework that enables existing methods to capture temporal dependencies within the target time series via an additional learning objective. We first formulate the TDT with the change values between adjacent time steps to facilitate capturing intrinsic temporal patterns of the target. Building upon this, we introduce a novel loss function to align the change values within the prediction with those within the target time series, thereby guiding the model to effectively learn intricate temporal relations. Finally, we design an adaptive loss balancing strategy that dynamically adjusts the relative importance between the conventional forecasting objective and the TDT learning objective, enabling TDAlign to flexibly integrate with existing LTSF methods without introducing additional learnable parameters (Section III-C3). Computational complexity analysis demonstrates that TDAlign incurs negligible computational overhead, with time complexity of $\mathcal{O}(H)$ and space complexity of $\mathcal{O}(1)$, where H represents the length of the prediction (Section III-D).

Flexibility and Effectiveness. Remarkably, our proposed TDAlign is orthogonal to existing LTSF methods without introducing additional learnable parameters. We evaluate TDAlign using a broad spectrum of strong baseline methods, including the MLP-based method DLinear [14], CNN-based methods MICN [17] and TimesNet [18], Transformer-based methods PatchTST [20] and iTransformer [29], and the RNN-based method SegRNN [22]. Experimental results on seven widely used LTSF benchmark datasets demonstrate that TDAlign consistently improves the forecasting performance of these six baseline methods, showcasing its strong flexibility and effectiveness (Section IV-B). As shown in Figure 1, TDAlign achieves significant error reductions across seven datasets: (1) For change values between adjacent time steps within the prediction, absolute error reductions range from **4.57%** to **15.78%**. (2) For the prediction, absolute error reductions range from **1.47%** to **9.19%**.

In summary, our contributions are as follows:

- We provide a new perspective from the temporal relations between adjacent time steps within the target to analyze the performance bottleneck of existing LTSF methods, revealing their inadequate temporal dependency modeling.
- We propose TDAlign, a novel and generic framework for learning TDT. TDAlign is orthogonal to existing methods and introduces no additional learnable parameters, incurring minimal computational overhead.
- We conduct extensive experiments on six state-of-the-art methods across seven real-world datasets. The results demonstrate that TDAlign consistently improves the forecasting performance of these methods by a significant margin, showcasing its flexibility and effectiveness.

II. Related Work

A. Long-Term Time Series Forecasting

Long-term time series forecasting (LTSF) has been extensively studied in recent years. In contrast to traditional short-term forecasting, which typically predicts 48 steps or fewer [10], [11], [12], LTSF extends forecast horizons, such as 720 steps [27]. The task poses substantial challenges for existing neural time series architectures, particularly in capturing long-term dependencies and managing model complexity. Consequently, despite their proficiency in handling sequential data [25], [30], the Recurrent Neural Network (RNN) has gradually lost prominence in the LTSF domain due to the rapid increase in error and inference time as the forecast horizon expands [31], [32]. Recently, SegRNN [22] has revitalized the RNN architecture in the LTSF field by replacing segment-wise iterations instead of traditional point-wise processing, while leveraging non-autoregressive strategies to enable efficient parallel multi-step forecasting.

The Transformer architecture [33], benefiting from the attention mechanism's capacity to capture long-term dependencies in sequential data, has garnered significant attention in the LTSF domain [34], [35], [36]. Early efforts to adapt the vanilla Transformer for LTSF tasks focused on designing novel mechanisms to reduce the computational complexity of the original attention mechanism. For example, LogSparse [37] proposes convolutional self-attention by employing causal convolutions to produce queries and keys in the self-attention layer. Informer [27] introduces a ProbSparse self-attention with distilling techniques to extract dominant queries efficiently and a generative-style decoder for parallel multi-step forecasting. Pyraformer [23] designs a hierarchical pyramidal attention module to capture temporal dependencies of different ranges with linear time and memory complexity. These works use point-wise representations, which are proven insufficient to capture local semantic information in time series. To further enhance efficiency and forecasting performance, recent advancements have been devoted to exploring patch-wise and series-wise dependencies. For instance, Crossformer [21] and PatchTST [20] leverage patch-based techniques, commonly employed in Natural Language Processing [38] and Computer Vision [39], to divide time series data into subseries-level patches, capturing patch-wise dependency. iTransformer [29] embeds independent time series as tokens and repurposes the Transformer architecture by applying attention mechanisms to these variate tokens, capturing series-wise dependency.

Besides the Transformer architecture, the Convolutional Neural Network (CNN) and the Multi-layer Perceptron (MLP) have also emerged as prominent architectures, demonstrating impressive results in the LTSF field. CNN-based methods, such as SCINet [16], MICN [17], and TimesNet [18], excel at capturing local and multi-scale temporal features. Following the success of DLinear [14], which highlighted the efficacy of parsimonious models in addressing complex time series data, MLP-based methods have gained increasing attention, as exemplified by RLinear [13], TiDE [15] and TSMixer [40].

B. Modeling TDT in Time Series Forecasting

Modeling the temporal dependencies within the target refers to the method's ability to effectively capture the internal correlations across multiple steps in the target time series, which fundamentally influence and reflect the overall forecasting performance. Existing research predominantly explores these complex temporal relations through two primary approaches.

Early studies adopted the autoregressive decoding strategy to forecast multiple steps, such as LSTM [25], GRU [26], and DeepAR [30]. These methods utilize the most recently forecasted value as part of the input for predicting the next step. During the recursive process, models naturally learn the dependencies between consecutive steps. However, despite their impressive success in short-term time series forecasting [41], autoregressive methods are unsuitable for long-term time series forecasting tasks due to error accumulation and slow inference speeds inherent in step-by-step prediction [27], [28].

On the other hand, numerous methods attempt to preserve the temporal order information within the target time series by incorporating auxiliary features such as position or timestamp embeddings. For instance, Pyraformer [23] combines zero placeholders, positional embeddings, and timestamp embeddings to represent prediction tokens. SegRNN [22] leverages relative positional embeddings to indicate the sequential order of multiple segments. Methods can implicitly learn some internal correlations with temporal order information, where the model assigns distinct weights to different positions or timestamps [42], [43]. However, such implicit learning is insufficient for effectively capturing intricate and dynamic temporal patterns [14], [24].

The modeling of TDT remains inadequate in existing long-term time series forecasting methods. In this work, we aim to address this limitation and substantially enhance the forecasting performance. We highlight that our proposed framework is orthogonal to existing methods, enabling end-to-end training for non-autoregressive forecasting.

III. METHOD

In this section, we start with the definition of the LTSF tasks and the introduction of frequently used notions. Subsequently, we provide an in-depth analysis of existing LTSF methods and reveal their inadequate modeling of TDT. Following this, we present a detailed exposition of our proposed Temporal Dependency Alignment (TDAlign) framework, designed to address this limitation. Finally, we conduct a complexity analysis of TDAlign to demonstrate its efficiency.

A. Preliminaries

In the following, *input* refers to the historical time series, while *target* and *prediction* represent the ground truth and predicted values of the future time series, respectively. Let $x_t \in \mathbb{R}^N$ denote the observation at time step t of N variables. In LTSF tasks, given the input $X = \{x_{t-L+1}, x_{t-L+2}, \cdots, x_t\} \in \mathbb{R}^{L\times N}$, the model $\mathcal{F}$ with learnable parameters θ aims to predict the target $Y = \{x_{t+1}, x_{t+2}, \ldots, x_{t+H}\}$, where the corresponding prediction is $\widehat{Y} = \{\hat{x}_{t+1}, \hat{x}_{t+2}, \cdots, \hat{x}_{t+H}\} \in \mathbb{R}^{H\times N}$. Here, L is the length of the input, while H is the length of the target and the prediction. Typically, H is much larger in LTSF tasks than in traditional short-term time series forecasting tasks, such as 720 time steps. This forecast behavior can be formally summarized as $\mathcal{F}_\theta : X \to \widehat{Y}$. Our ultimate goal is to train a model that minimizes the discrepancy between the prediction $\widehat{Y}$ and the target Y. Frequently used notations are listed in Table I.

TABLE I: Mathematical Notation.

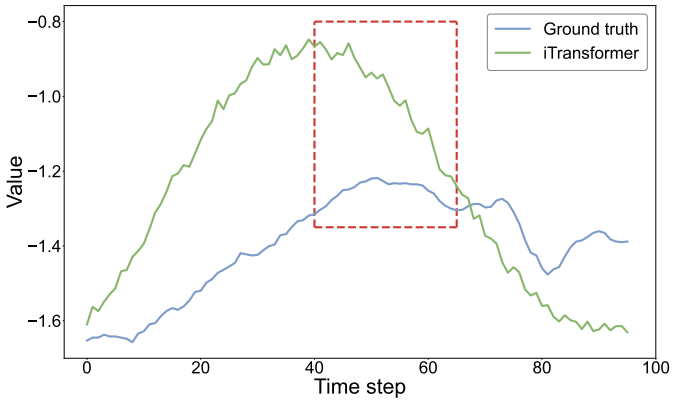
Notions	Descriptions
N	The number of variables.
$x_t \in \mathbb{R}^N$	The observation at the t-th time step of N variables.
L	The length of the historical input.
H	The length of the future target and prediction.
$X \in \mathbb{R}^{L\times N}$	The input of N variables over L time steps.
$Y, \widehat{Y} \in \mathbb{R}^{H\times N}$	The target and prediction of N variables over H time steps.
$d_t, \widehat{d}_t \in \mathbb{R}^N$	The temporal dependency at the t-th time step within the target and prediction of N variables.
$D, \widehat{D} \in \mathbb{R}^{H\times N}$	The temporal dependencies within the target and prediction of N variables over H time steps.
$\mathrm{sgn}(\cdot)$	The sign function.
$\mathbb{1}(\cdot)$	The indicator function.
$\mathcal{L}_Y$	The loss between Y and $\widehat{Y}$.
$\mathcal{L}_D$	The loss between D and $\widehat{D}$.
ρ	The sign inconsistency ratio between D and $\widehat{D}$.

B. Motivation

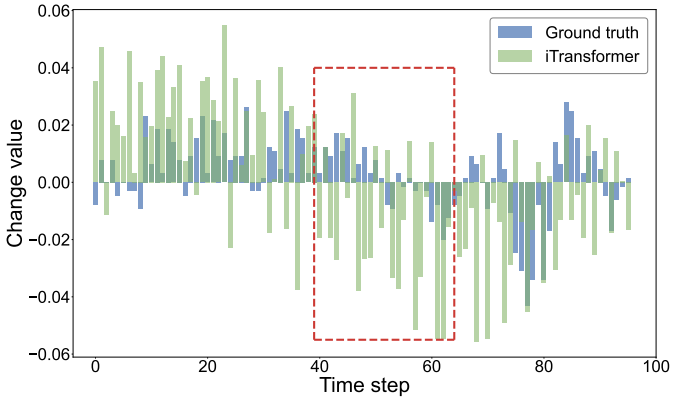
An intrinsic feature of time series is that consecutive time steps typically exhibit strong correlations [44]. This inherent characteristic suggests that forecasting models should adequately account for temporal dependencies across multiple steps, encompassing both the input and target [45]. Inspired by this insight, we assess the effort of existing methods in capturing temporal variations.

However, through an investigation of existing LTSF methods, we find that they predominantly focus on extracting temporal patterns within the input but fail to adequately model Temporal Dependencies within the Target (TDT), ultimately leading to suboptimal forecasting performance. We present a forecasting instance from the state-of-the-art method iTransformer [29] on the Weather dataset in Figure 2. As shown in Figure 2(a), there are significant discrepancies between predicted and ground truth values at most time steps. Delving deeper into the underlying causes of these discrepancies, we analyze the relations between consecutive time steps. Figure 2(b) illustrates that the method not only exhibits substantial deviations in change values between adjacent time steps but even fails to accurately predict change directions, particularly in the regions marked by red dashed boxes. These observations demonstrate that the method lacks adequate modeling of temporal relations across multiple steps in the target, resulting in unrealistic and inaccurate predictions.

The inadequate modeling of TDT arises from the reliance on the non-autoregressive decoding strategy and a single loss function. Specifically, the autoregressive decoding strategy is unsuitable for LTSF tasks due to error accumulation and slow inference speeds inherent in step-by-step forecasting [28]. To avoid these issues, methods that adopt the non-autoregressive decoding strategy have become dominant in LTSF tasks [14], simultaneously generating all prediction steps [20], [22]. However, these methods drop the dependency information in the target [46], [47]. Besides, regardless of the architectures and parameterizations, the vast majority of methods [18], [29] optimize a single conventional forecasting



(a) Comparison of the target and prediction.



(b) Comparison of change values between adjacent time steps within the target and prediction.

Fig. 2: An illustration of a forecasting instance produced by the iTransformer [29] on the Weather dataset. $L = 96$, $H = 96$.

objective $\mathcal{L}_Y$, whose general formulation can be expressed as:

$$\mathcal{L}_Y = \frac{1}{H}\sum_{i=1}^{H} \ell(x_{t+i}, \hat{x}_{t+i}) \tag{1}$$

where $\ell(\cdot,\cdot)$ denotes the error evaluated at a single time step. Popular choices for $\mathcal{L}_Y$ include $L1$ loss, $L2$ loss and other variants [48]. However, methods merely optimize the model towards the minimum of the average of step-wise prediction error with $\mathcal{L}_Y$, overlooking the temporal relations when generating prediction steps [49], [50].

C. Temporal Dependency Alignment

Motivated by the analysis of existing LTSF methods' inadequate capture of change patterns in Section III-B, we propose a generic and novel Temporal Dependency Alignment (TDAlign) framework that can plug into existing methods to model TDT. An overview of the proposed TDAlign framework is shown in Figure 3.

1) Temporal Dependencies within the Target: A fair formulation of temporal dependencies should benefit not only the learning of these dependencies but also the evaluation used to assess the learning effects. We consider two key insights: (1) In time series, temporally proximate time steps typically exhibit stronger correlations compared to those with far temporal distance [44], [51], [45]; (2) The autoregressive decoding strategy enables methods to naturally learn sequential dependencies, where each prediction step is generated conditioned on its immediately preceding prediction step [25], [26].

Inspired by these insights, we formulate Temporal Dependencies within the Target (TDT) using first-order differencing values [52] of the target. Specifically, we formulate TDT as $D = \{d_{t+1}, d_{t+2}, \cdots, d_{t+H}\} \in \mathbb{R}^{H\times N}$, where each term $d_{t+i} \in \mathbb{R}^N$ equals to the change value between adjacent time steps, calculated as:

$$d_{t+i} = x_{t+i} - x_{t+i-1}, 1 \le i \le H. \tag{2}$$

We formulate Temporal Dependencies within the Prediction (TDP) as $\widehat{D} = \left\{\hat{d}_{t+1}, \hat{d}_{t+2}, \cdots, \hat{d}_{t+H}\right\} \in \mathbb{R}^{H\times N}$, where $\hat{d}_{t+i}$ is computed as:

$$\hat{d}_{t+i} = \begin{cases} \hat{x}_{t+i} - x_t & \text{if } i = 1, \\ \hat{x}_{t+i} - \hat{x}_{t+i-1} & \text{if } 2 \le i \le H. \end{cases} \tag{3}$$

Despite its simplicity, using first-order differencing values (i.e., change values between adjacent time steps) to formulate temporal dependencies demonstrates greater generality and effectiveness compared to variants that use higher-order differencing values or change values between non-adjacent time steps (see the experiments in Section IV-C for details).

2) A Novel Loss Function for TDAlign: In order to effectively capture intricate temporal patterns of the target, a direct approach is to guide the model to align TDP with TDT. To achieve this, we propose a novel loss function, $\mathcal{L}_D$, which measures the average error between TDP and TDT. Consistent with the conventional forecasting objective $\mathcal{L}_Y$ (Equation 1), the TDT learning objective $\mathcal{L}_D$ is computed using either MSE or MAE and its general formulation is expressed as:

$$\mathcal{L}_D = \frac{1}{H}\sum_{i=1}^{H} \ell(d_{t+i}, \hat{d}_{t+i}), \tag{4}$$

where $\ell(\cdot,\cdot)$ denotes the error evaluated at a single time step. Notably, a key distinction is that each term in $\mathcal{L}_Y$ involves individual time steps, whereas each term in $\mathcal{L}_D$ involves two consecutive time steps.

3) Adaptive Loss Balancing: To enable TDAlign to integrate with the forecasting model, an intuitive solution is to set a weight λ for balancing the learning procedure, which can be expressed as:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_Y + (1-\lambda)\cdot \mathcal{L}_D. \tag{5}$$

While manually tuning λ to determine its optimal value is a straightforward option, this approach lacks flexibility and is labor-intensive. Therefore, we propose an adaptive loss balancing strategy using a dynamic weight ρ, which is calculated as follows:

$$\rho = \frac{1}{H}\sum_{i=1}^{H} \mathbb{1}(\text{sgn}(d_{t+i}) \neq \text{sgn}(\hat{d}_{t+i})) \tag{6}$$

where $\mathbb{1}(\cdot)$ is the indicator function that returns 1 if the condition $\text{sgn}(d_{t+i}) \neq \text{sgn}(\hat{d}_{t+i})$ is true and 0 otherwise, and

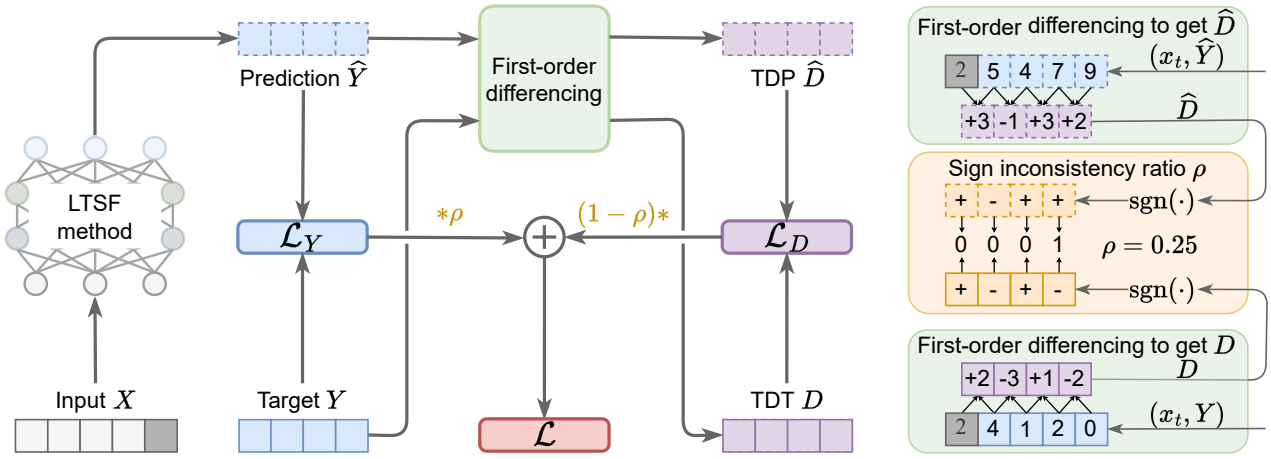


Fig. 3: Overview of our proposed Temporal Dependency Alignment (TDAlign) framework. First, TDAlign formulates temporal dependencies within the target (TDT)—denoted as D, and temporal dependencies within the prediction (TDP)—denoted as $\widehat{D}$, using the first-order differencing values of the target Y and the prediction $\widehat{Y}$, respectively (in green boxes). Then, TDAlign devises a loss L_D that guides the model to align $\widehat{D}$ with D for capturing temporal patterns. Finally, TDAlign calculates a balance weight ρ (in the orange box) to adaptively control the relative contributions of the conventional forecasting loss L_Y and the devised L_D, thereby improving existing methods without introducing additional learnable parameters.

$\mathrm{sgn}(\cdot)$ denotes the sign function. Consequently, ρ represents the sign inconsistency ratio between TDP and TDT, with values ranging from 0 to 1.

The dynamic weight ρ offers several advantages: (1) Simplicity and informativeness. ρ is directly computed from TDP and TDT without requiring additional data or complex preprocessing. Moreover, the sign indicates the change direction between adjacent time steps. Therefore, ρ can also serve as a metric to quantify the method's ability to capture TDT, as detailed in Section IV-A3. (2) Effectiveness. ρ dynamically adjusts the relative importance between the conventional forecasting objective $\mathcal{L}_Y$ (Equation 1) and the TDT learning objective $\mathcal{L}_D$ (Equation 4) based on the model's current learning state. The effectiveness of ρ is further validated by experiments detailed in Section IV-C. (3) Flexibility. The formulation of ρ introduces no additional learnable parameters, enabling TDAlign to serve as a model-agnostic framework that seamlessly integrates with various LTSF methods.

4) Training: Formally, when integrating with existing LTSF methods, the overall loss function of TDAlign is expressed as:

$$\mathcal{L} = \rho \cdot \mathcal{L}_Y + (1-\rho) \cdot \mathcal{L}_D. \tag{7}$$

The complete training procedure of TDAlign is described in **Algorithm** 1.

Algorithm 1: Temporal Dependencies Alignment

Input: The input $X \in \mathbb{R}^{L \times N}$, the target $Y \in \mathbb{R}^{H \times N}$, the forecasting model $\mathcal{F}$ with parameters θ.
Output: Updated model parameters θ^*
while *not converged* **do**
  Predict $\widehat{Y} \leftarrow \mathcal{F}_\theta(X)$;
  Construct D from Y using Equation 2 and compute $\widehat{D}$ from $\widehat{Y}$ using Equation 3;
  Compute $\mathcal{L}_Y$ and $\mathcal{L}_D$ according to Equation 1 and Equation 4 respectively;
  Compute the weight ρ using Equation 6;
  Compute the overall loss $\mathcal{L} \leftarrow \rho \cdot \mathcal{L}_Y + (1-\rho) \cdot \mathcal{L}_D$;
  Update parameters $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$;
end

D. Computational Complexity Analysis

We analyze the additional computational complexity introduced by TDAlign in terms of both time and space.

For time complexity, TDAlign requires several additional computations. The calculation of D (Equation 2) and $\widehat{D}$ (Equation 3) each incurs a time complexity of $O(H)$. The computation of $\mathcal{L}_D$ (Equation 4) and ρ (Equation 6) each contributes $O(H)$ to the time complexity. The final weighted sum of $\mathcal{L}_Y$ and $\mathcal{L}_D$ (Equation 7) adds a constant time $O(1)$. In total, the additional time complexity of TDAlign is $O(4H+1)$, simplifying to $O(H)$ in asymptotic terms, where H denotes the length of the prediction.

Regarding space complexity, TDAlign introduces only a constant overhead. It requires storing a few scalar values such as $\mathcal{L}_D$, ρ, and some intermediate results. Importantly, D and $\widehat{D}$ are computed on-the-fly and discarded after their use in calculating $\mathcal{L}_D$ and ρ, ensuring no contribution to the persistent space complexity. As a result, the additional space complexity of TDAlign is $O(1)$.

In conclusion, TDAlign is an efficient plug-and-play framework, introducing minimal computational overhead that scales linearly with prediction length in time complexity, while maintaining constant space complexity.

TABLE II: Statistics of benchmark datasets.

Dataset	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Electricity	ILI
Variables	7	7	7	7	21	321	7
Timesteps	17420	17420	69680	69680	52696	26304	966
Granularity	1 hour	1 hour	15 minutes	15 minutes	10 minutes	1 hour	1 week

TABLE III: Input and prediction lengths (L and H) of various baseline methods on different datasets.

Model	ILI dataset $L \rightarrow H$	Other datasets $L \rightarrow H$
SegRNN	$60 \rightarrow \{24, 36, 48, 60\}$	$720 \rightarrow \{96, 192, 336, 720\}$
DLinear, PatchTST	$104 \rightarrow \{24, 36, 48, 60\}$	$336 \rightarrow \{96, 192, 336, 720\}$
iTransformer, MICN, TimesNet	$36 \rightarrow \{24, 36, 48, 60\}$	$96 \rightarrow \{96, 192, 336, 720\}$

IV. EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the effectiveness and generalizability of our proposed TDAlign. We begin with a detailed description of the experimental setup. Subsequently, we present the main experimental results and statistics of improvements achieved by integrating TDAlign into baseline methods. Furthermore, we conduct ablation studies to evaluate the effectiveness of various components of TDAlign and assess the impact of the formulation of temporal dependencies. Finally, we showcase several long-term forecasting instances to demonstrate that TDAlign produces more accurate and realistic predictions compared to baseline methods.

A. Experimental Setup

1) Datasets: We conduct experiments on seven widely used LTSF benchmark datasets. These multivariate datasets cover various real-world application scenarios, including:

- ETT (Electricity Transformer Temperature)[1]: This dataset contains the oil temperature and six power load features from electricity transformers from July 2016 to July 2018. It can be further divided into four sub-datasets (ETTh1, ETTh2, ETTm1, and ETTm2) based on data collection granularity and regions.
- Electricity[2]: This dataset collects the electricity consumption of 321 clients from 2012 to 2014.
- Weather[3]: This dataset records various climate indicators such as temperature, humidity, and wind speed for 2020 in Germany.
- ILI (Influenza-Like Illness)[4]: This dataset describes the ratio of influenza-like illness patients versus the total patients from the Centers for Disease Control and Prevention of the United States from 2002 to 2021.

The statistics of these datasets is summarized in Table II.

2) Baselines: We consider six open-source state-of-the-art LTSF methods as baseline methods, comprising a wide variety of architectures:

- DLinear[14][5]: An MLP-based method that employs two linear layers to extract trend and seasonal features.
- MICN[17][6]: A CNN-based method that adopts downsampled convolution and isometric convolution to capture local and global temporal features.
- TimesNet[18][7]: A CNN-based method that converts 1D time series into 2D tensors and captures temporal intraperiod- and interperiod- variations using deep convolutional neural networks.
- PatchTST[20][8]: A Transformer-based method that leverages the patching structure to capture local semantic information, advancing both the performance and efficiency of Transformers for LTSF.
- iTransformer[29][9]: A Transformer-based method that embeds time series as variate tokens, captures multivariate correlations through self-attention and extracts temporal dependencies using Transformer's feed-forward network.
- SegRNN[22][10]: An RNN-based method that reintroduces RNN architectures to LTSF through segment-wise iterations and parallel multi-step forecasting. During the decoding phase, it incorporates position encodings to implicitly learn correlations between segments.

3) Evaluation Metrics: Following the evaluation metrics employed in baseline methods, we assess the forecasting error of the prediction using MSE and MAE. Beyond the conventional evaluation, we extend our assessment to the forecasting error of TDP, also using MSE and MAE but denoted as MSE_D and MAE_D for distinction. Additionally, we also use ρ (Equation 6) to evaluate the forecasting error of TDP, as defined in Equation 6. As described in Section III-C3, ρ represents the sign inconsistency ratio between TDP and TDT, and can also indicate the method's error rate in predicting the change direction between adjacent time steps. Therefore, unless otherwise specified, MSE and MAE refer to the errors between Y and $\widehat{Y}$, whereas MSE_D, MAE_D, and ρ specifically represent the errors between D and $\widehat{D}$. All these metrics are non-negative, with lower values indicating better performance.

[1]https://github.com/zhouhaoyi/ETDataset
[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[3]https://www.bgc-jena.mpg.de/wetter/
[4]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
[5]https://github.com/cure-lab/LTSF-Linear
[6]https://github.com/wanghq21/MICN
[7]https://github.com/thuml/TimesNet
[8]https://github.com/yuqinie98/PatchTST
[9]https://github.com/thuml/iTransformer
[10]https://github.com/lss-1138/SegRNN

TABLE IV: Comparison of forecasting errors between baseline methods and TDAlign. Results are averaged over 5 runs, with better performance highlighted in bold. "Avg error" represents the average error of each baseline method over 7 datasets.

Architecture		CNN-based								MLP-based			
Method		MICN		+TDTAlign		TimesNet		+TDTAlign		DLinear		+TDTAlign	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.417±0.014	0.442±0.011	**0.375±0.001**	**0.398±0.001**	0.406±0.011	0.424±0.007	**0.384±0.003**	**0.396±0.001**	0.376±0.005	0.399±0.006	**0.362±0.000**	**0.384±0.000**
	192	0.483±0.010	0.485±0.005	**0.416±0.006**	**0.426±0.002**	0.458±0.012	0.453±0.007	**0.438±0.012**	**0.429±0.007**	0.407±0.003	0.416±0.003	**0.400±0.000**	**0.407±0.000**
	336	0.581±0.037	0.559±0.021	**0.445±0.003**	**0.448±0.002**	0.508±0.017	0.479±0.010	**0.474±0.011**	**0.449±0.007**	0.490±0.041	0.485±0.033	**0.431±0.000**	**0.427±0.000**
	720	0.689±0.092	0.633±0.049	**0.533±0.008**	**0.524±0.005**	0.516±0.010	0.493±0.006	**0.499±0.012**	**0.481±0.009**	0.498±0.021	0.508±0.015	**0.452±0.000**	**0.473±0.000**
ETTh2	96	0.313±0.012	0.369±0.013	**0.294±0.002**	**0.347±0.002**	0.321±0.006	0.363±0.003	**0.305±0.007**	**0.348±0.005**	0.294±0.006	0.357±0.005	**0.278±0.000**	**0.334±0.000**
	192	0.422±0.015	0.445±0.011	**0.384±0.005**	**0.406±0.003**	0.411±0.011	0.417±0.007	**0.390±0.006**	**0.397±0.004**	0.376±0.005	0.412±0.005	**0.354±0.000**	**0.386±0.000**
	336	0.693±0.090	0.584±0.046	**0.446±0.008**	**0.451±0.003**	0.467±0.016	0.460±0.011	**0.429±0.012**	**0.433±0.009**	0.458±0.010	0.466±0.005	**0.411±0.000**	**0.430±0.000**
	720	0.853±0.046	0.667±0.023	**0.635±0.025**	**0.563±0.009**	0.466±0.025	0.469±0.015	**0.422±0.012**	**0.437±0.007**	0.714±0.019	0.601±0.006	**0.582±0.000**	**0.531±0.000**
ETTm1	96	0.312±0.003	0.363±0.004	**0.301±0.002**	**0.339±0.001**	0.335±0.004	0.375±0.002	**0.331±0.004**	**0.361±0.003**	0.301±0.001	0.345±0.002	**0.292±0.000**	**0.333±0.000**
	192	0.358±0.004	0.393±0.006	**0.348±0.003**	**0.369±0.004**	0.395±0.011	0.405±0.007	**0.376±0.006**	**0.385±0.005**	0.336±0.001	0.367±0.001	**0.331±0.000**	**0.356±0.000**
	336	0.401±0.006	0.424±0.007	**0.382±0.005**	**0.396±0.006**	0.420±0.009	0.423±0.003	**0.402±0.002**	**0.401±0.002**	0.371±0.003	0.389±0.004	**0.365±0.000**	**0.376±0.000**
	720	0.470±0.013	0.473±0.012	**0.451±0.008**	**0.445±0.009**	0.486±0.010	0.457±0.004	**0.463±0.002**	**0.434±0.002**	0.426±0.002	0.422±0.003	**0.420±0.000**	**0.411±0.000**
ETTm2	96	0.176±0.000	0.273±0.001	**0.171±0.001**	**0.259±0.001**	0.187±0.002	0.266±0.001	**0.181±0.002**	**0.257±0.001**	0.170±0.002	0.264±0.002	**0.164±0.000**	**0.247±0.000**
	192	0.266±0.011	0.342±0.011	**0.233±0.003**	**0.303±0.003**	0.254±0.002	0.309±0.001	**0.246±0.002**	**0.299±0.001**	0.229±0.005	0.306±0.007	**0.221±0.000**	**0.289±0.000**
	336	0.321±0.025	0.377±0.023	**0.294±0.006**	**0.345±0.005**	0.316±0.005	0.346±0.003	**0.309±0.002**	**0.338±0.001**	0.288±0.005	0.349±0.007	**0.278±0.000**	**0.332±0.000**
	720	0.466±0.033	0.468±0.021	**0.391±0.013**	**0.408±0.010**	0.425±0.005	0.408±0.002	**0.417±0.006**	**0.401±0.003**	0.438±0.072	0.442±0.042	**0.377±0.000**	**0.399±0.000**
Electricity	96	0.161±0.002	0.268±0.002	**0.156±0.001**	**0.258±0.001**	**0.168±0.003**	0.272±0.003	0.171±0.003	**0.267±0.003**	**0.140±0.000**	0.237±0.000	0.141±0.000	**0.235±0.000**
	192	0.176±0.002	0.283±0.002	**0.173±0.003**	**0.272±0.002**	0.186±0.003	0.287±0.003	**0.182±0.003**	**0.278±0.003**	0.154±0.000	0.250±0.000	**0.154±0.000**	**0.247±0.000**
	336	0.196±0.004	0.306±0.005	**0.184±0.004**	**0.284±0.004**	**0.200±0.005**	**0.301±0.005**	0.236±0.024	0.313±0.016	**0.169±0.000**	0.268±0.000	0.170±0.000	**0.263±0.000**
	720	0.216±0.004	0.324±0.005	**0.212±0.006**	**0.308±0.004**	**0.231±0.008**	**0.324±0.006**	0.292±0.008	0.354±0.007	0.204±0.000	0.301±0.000	**0.204±0.000**	**0.294±0.000**
ILI	24	2.760±0.025	1.151±0.008	**2.546±0.038**	**1.059±0.010**	2.336±0.359	0.939±0.044	**2.112±0.192**	**0.901±0.035**	2.265±0.064	1.048±0.027	**2.174±0.003**	**1.001±0.001**
	36	2.761±0.049	1.140±0.013	**2.615±0.046**	**1.072±0.011**	2.493±0.074	0.994±0.017	**2.167±0.116**	**0.905±0.024**	2.251±0.107	1.064±0.049	**2.091±0.003**	**0.995±0.001**
	48	2.778±0.056	1.133±0.015	**2.680±0.043**	**1.078±0.009**	2.379±0.069	0.948±0.019	**2.168±0.040**	**0.889±0.009**	2.302±0.021	1.082±0.007	**2.064±0.006**	**1.000±0.001**
	60	2.879±0.047	1.150±0.010	**2.708±0.028**	**1.085±0.006**	2.196±0.100	0.962±0.024	**2.113±0.087**	**0.895±0.020**	2.435±0.024	1.120±0.009	**2.124±0.009**	**1.029±0.003**
Weather	96	0.170±0.003	0.235±0.006	**0.158±0.001**	**0.198±0.001**	0.172±0.002	0.222±0.001	**0.170±0.001**	**0.214±0.002**	0.175±0.001	0.238±0.003	**0.173±0.000**	**0.217±0.000**
	192	0.222±0.006	0.284±0.007	**0.207±0.001**	**0.247±0.001**	0.235±0.004	0.273±0.004	**0.225±0.003**	**0.261±0.002**	0.216±0.001	0.274±0.001	**0.213±0.000**	**0.255±0.000**
	336	0.278±0.014	0.339±0.020	**0.252±0.004**	**0.282±0.003**	0.285±0.003	0.306±0.002	**0.281±0.003**	**0.300±0.003**	0.262±0.002	0.313±0.003	**0.258±0.000**	**0.292±0.000**
	720	0.336±0.014	0.376±0.017	**0.311±0.002**	**0.328±0.003**	0.360±0.002	0.355±0.001	**0.355±0.002**	**0.349±0.001**	0.325±0.001	0.364±0.001	**0.320±0.000**	**0.342±0.000**
Avg error		0.720	0.510	**0.654**	**0.461**	0.629	0.455	**0.591**	**0.435**	0.610	0.467	**0.564**	**0.439**

Architecture		Transformer-based								RNN-based			
Method		PatchTST		+TDTAlign		iTransformer		+TDTAlign		SegRNN		+TDTAlign	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.380±0.001	0.402±0.001	**0.373±0.001**	**0.396±0.001**	0.387±0.001	0.405±0.001	**0.375±0.001**	**0.389±0.000**	0.351±0.001	0.382±0.000	**0.349±0.000**	**0.380±0.000**
	192	0.412±0.000	0.420±0.000	**0.409±0.001**	**0.417±0.001**	0.441±0.002	0.436±0.001	**0.426±0.001**	**0.420±0.001**	0.391±0.001	0.407±0.001	**0.384±0.001**	**0.403±0.000**
	336	0.436±0.003	0.437±0.003	**0.433±0.001**	**0.431±0.001**	0.491±0.004	0.461±0.003	**0.469±0.001**	**0.442±0.001**	0.423±0.006	0.426±0.004	**0.409±0.002**	**0.418±0.001**
	720	0.453±0.003	0.469±0.002	**0.433±0.002**	**0.453±0.002**	0.511±0.010	0.494±0.006	**0.460±0.001**	**0.460±0.001**	0.446±0.007	0.456±0.005	**0.445±0.010**	**0.452±0.006**
ETTh2	96	**0.275±0.001**	0.338±0.001	0.277±0.001	**0.332±0.000**	0.300±0.002	0.351±0.001	**0.290±0.002**	**0.337±0.001**	0.274±0.001	0.329±0.001	**0.273±0.003**	**0.327±0.001**
	192	**0.340±0.001**	0.380±0.001	0.347±0.002	**0.376±0.001**	0.380±0.002	0.399±0.001	**0.367±0.001**	**0.386±0.000**	0.340±0.002	0.373±0.001	**0.333±0.003**	**0.368±0.002**
	336	**0.364±0.003**	0.398±0.001	0.367±0.004	**0.395±0.002**	0.423±0.002	0.432±0.001	**0.417±0.001**	**0.424±0.001**	0.386±0.006	0.409±0.004	**0.368±0.009**	**0.397±0.004**
	720	0.390±0.002	0.427±0.002	**0.388±0.003**	**0.420±0.002**	0.431±0.004	0.448±0.002	**0.419±0.001**	**0.438±0.000**	0.406±0.004	0.432±0.003	**0.384±0.003**	**0.417±0.002**
ETTm1	96	0.290±0.001	0.342±0.001	**0.287±0.004**	**0.327±0.003**	0.341±0.003	0.376±0.002	**0.329±0.001**	**0.355±0.001**	0.284±0.001	0.335±0.001	**0.278±0.001**	**0.333±0.001**
	192	**0.332±0.002**	0.369±0.001	0.336±0.006	**0.358±0.001**	0.381±0.001	0.394±0.001	**0.381±0.001**	**0.381±0.001**	0.320±0.001	0.360±0.001	**0.317±0.001**	**0.359±0.001**
	336	0.366±0.001	0.391±0.001	**0.360±0.006**	**0.378±0.003**	0.420±0.001	0.419±0.001	**0.418±0.002**	**0.407±0.001**	0.351±0.001	0.383±0.001	**0.346±0.002**	**0.381±0.001**
	720	0.417±0.003	0.423±0.002	**0.413±0.005**	**0.410±0.002**	**0.489±0.001**	0.457±0.001	0.489±0.002	**0.448±0.001**	0.410±0.003	0.417±0.001	**0.399±0.002**	**0.414±0.001**
ETTm2	96	**0.164±0.001**	0.253±0.001	0.164±0.003	**0.246±0.002**	0.184±0.000	0.269±0.001	**0.183±0.000**	**0.263±0.000**	0.158±0.000	0.241±0.000	**0.156±0.000**	**0.240±0.000**
	192	0.221±0.001	0.292±0.001	**0.220±0.002**	**0.286±0.002**	0.252±0.001	0.313±0.001	**0.250±0.000**	**0.306±0.000**	0.215±0.001	0.283±0.000	**0.211±0.001**	**0.280±0.001**
	336	0.276±0.001	0.329±0.001	**0.269±0.004**	**0.321±0.002**	0.315±0.001	0.352±0.001	**0.311±0.000**	**0.345±0.000**	0.263±0.001	0.317±0.000	**0.260±0.002**	**0.315±0.002**
	720	0.367±0.001	0.384±0.001	**0.361±0.002**	**0.379±0.003**	0.413±0.001	0.406±0.001	**0.412±0.001**	**0.401±0.001**	0.333±0.001	0.369±0.001	**0.329±0.002**	**0.367±0.001**
Electricity	96	0.130±0.000	0.223±0.000	**0.130±0.000**	**0.216±0.000**	0.148±0.000	0.240±0.000	**0.148±0.000**	**0.234±0.000**	0.129±0.000	0.220±0.000	**0.125±0.000**	**0.214±0.000**
	192	0.149±0.001	0.241±0.001	**0.147±0.000**	**0.232±0.000**	0.165±0.001	0.256±0.001	**0.164±0.001**	**0.248±0.001**	0.150±0.000	0.240±0.000	**0.148±0.001**	**0.237±0.001**
	336	0.166±0.000	0.260±0.000	**0.162±0.000**	**0.247±0.000**	0.178±0.001	0.270±0.001	**0.177±0.001**	**0.263±0.001**	0.167±0.000	0.258±0.000	**0.166±0.000**	**0.257±0.000**
	720	0.204±0.001	0.293±0.001	**0.197±0.000**	**0.279±0.000**	0.214±0.006	0.302±0.005	**0.210±0.001**	**0.292±0.000**	0.201±0.001	0.290±0.000	**0.200±0.001**	**0.289±0.001**
ILI	24	2.010±0.112	0.896±0.048	**1.826±0.083**	**0.826±0.029**	2.356±0.022	0.952±0.008	**2.173±0.182**	**0.901±0.026**	**1.748±0.049**	**0.764±0.015**	1.756±0.090	0.790±0.026
	36	1.886±0.062	0.888±0.032	**1.783±0.087**	**0.846±0.032**	2.238±0.034	0.960±0.006	**1.982±0.048**	**0.871±0.010**	1.700±0.063	**0.772±0.018**	**1.636±0.061**	0.773±0.009
	48	1.780±0.108	0.898±0.027	**1.767±0.144**	**0.859±0.039**	2.268±0.057	0.971±0.015	**2.019±0.011**	**0.872±0.003**	1.735±0.021	**0.776±0.009**	**1.719±0.068**	0.799±0.015
	60	1.794±0.056	0.905±0.011	**1.780±0.068**	**0.862±0.017**	2.289±0.026	0.987±0.004	**2.058±0.020**	**0.889±0.005**	1.709±0.051	**0.788±0.015**	**1.698±0.040**	0.795±0.009
Weather	96	0.151±0.000	0.199±0.000	**0.150±0.000**	**0.185±0.000**	0.175±0.001	0.216±0.001	**0.168±0.001**	**0.202±0.001**	0.142±0.000	**0.182±0.000**	**0.142±0.000**	0.183±0.001
	192	0.195±0.001	0.241±0.001	**0.194±0.000**	**0.228±0.000**	0.224±0.001	0.257±0.001	**0.219±0.001**	**0.247±0.001**	0.186±0.000	0.227±0.000	**0.186±0.000**	**0.226±0.000**
	336	0.248±0.001	0.282±0.001	**0.247±0.001**	**0.269±0.000**	0.281±0.001	0.299±0.001	**0.277±0.001**	**0.291±0.000**	0.236±0.000	0.268±0.000	**0.235±0.000**	**0.267±0.000**
	720	**0.321±0.001**	0.335±0.001	0.322±0.000	**0.324±0.000**	0.358±0.001	0.350±0.001	**0.356±0.001**	**0.343±0.000**	0.309±0.000	**0.319±0.000**	**0.308±0.001**	0.320±0.001
Avg error		0.518	0.418	**0.505**	**0.404**	0.609	0.445	**0.570**	**0.423**	0.492	0.394	**0.484**	**0.393**

4) Implementation Details: We conduct fair experiments to evaluate and demonstrate the effectiveness and flexibility of our proposed TDAlign. First, we strictly adhere to the consistent preprocessing procedures and data loading parameters for each baseline method to ensure result comparability. Based on the statistics of the training set, we apply z-score normalization to standardize the data. For dataset partitioning, we use a 6:2:2 ratio to split the training, validation, and test sets for the four ETT datasets, while the other datasets follow a 7:2:1 splitting ratio. Second, we fully adopt the parameter configurations for baseline methods as described in their official code repositories. Although all baseline methods maintain a consistent

TABLE V: Statistics of improvements achieved by integrating TDAlign into the baseline method, averaged over seven datasets. Values in parentheses indicate the count of cases showing improvement. For each metric, "Avg improvement" represents the average improvement over six baseline methods, and "Count" indicates the total number of cases showing improvement.

Model	MSE	MAE	MSE_D	MAE_D	ρ
MICN	9.19% (28)	9.72% (28)	9.60% (26)	7.70% (28)	9.03% (28)
TimesNet	6.10% (25)	4.38% (26)	15.78% (27)	11.99% (28)	11.32% (28)
DLinear	7.42% (24)	6.13% (28)	4.57% (18)	4.80% (26)	4.56% (26)
PatchTST	2.58% (21)	3.56% (28)	9.05% (24)	8.71% (28)	9.73% (28)
iTransformer	6.49% (25)	4.95% (28)	11.68% (26)	11.45% (28)	12.17% (28)
SegRNN	1.47% (25)	0.20% (22)	5.72% (24)	3.79% (26)	6.11% (28)
Avg improvement (Count)	5.54% (148)	4.82% (160)	9.40% (145)	8.07% (164)	8.82% (166)

prediction length for each dataset, their input lengths vary. In Table III, we summarize the specific configurations of input and prediction lengths of each baseline method on different datasets. Third, it is worth emphasizing that we do not employ the *Drop Last* operation during testing for all experiments, as suggested by [53]. Finally, all experiments are implemented using PyTorch and conducted on the NVIDIA RTX 4090 GPU with 24GB memory. Each experiment is repeated five times with different random seeds, and the mean and standard deviation of the results are reported.

B. Main Results and Analysis

We conducted extensive experiments encompassing seven real-world benchmark datasets, six baseline methods, and four different prediction lengths for each dataset, resulting in 168 experimental configurations.

Table IV presents a detailed comparison of the forecasting errors (MSE and MAE) between baseline methods and TDAlign. As shown in this table, first, we can clearly observe that our proposed TDAlign framework improves baseline methods by a significant margin in most cases, showcasing its effectiveness and generalizability. Notably, TDAlign improves MICN's performance across all prediction lengths and datasets. Second, TDAlign contributes to more stable performance, where it shows smaller standard deviations compared to baseline methods in most cases. For instance, DLinear shows non-zero standard deviations across six datasets except for the Electricity dataset, whereas TDAlign demonstrates zero standard deviations on six datasets and exhibits non-zero but significantly smaller standard deviations on the ILI dataset compared to DLinear. Third, implicit learning is insufficient to capture TDT, as demonstrated by SegRNN [22], which maintains sequential order through position embeddings yet still benefits from TDAlign.

Table V presents statistics of improvements achieved by TDAlign with six baseline methods. For each baseline method and evaluation metric, we report the average percentage improvement over seven datasets and the count of cases showing improvement. The statistical results demonstrate that our proposed TDAlign framework consistently boosts the forecasting performance of six baseline methods in terms of both the TDP and prediction by a large margin. For TDP, the improvements range from 4.56% to 12.17% in ρ, 4.57% to 15.78% in MSE_D, and 3.79% to 11.90% in MAE_D. For the prediction, the improvements range from 1.47% to 9.19% in MSE and 0.20% to 9.72% in MAE. Moreover, out of the 168 total cases, TDAlign demonstrates improvements in 166 cases for ρ, 145 for MSE_D, 164 for MAE_D, 148 for MSE, and 160 for MAE.

TABLE VI: Analysis of the impact of $\mathcal{L}_D$ and ρ on the ETTh1 dataset, with the iTransformer [29] as the baseline method. The best performances are highlighted in bold.

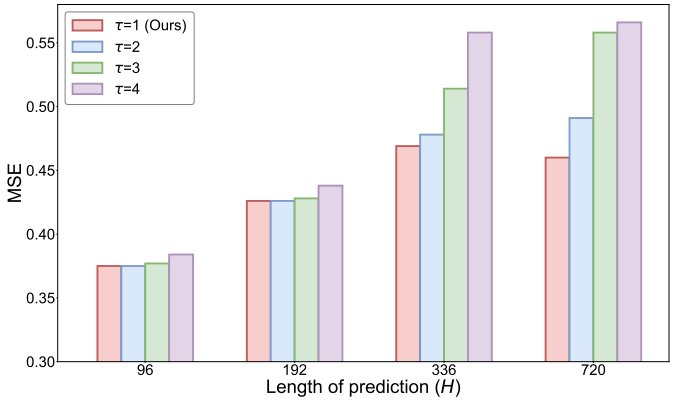
H	Setting	MSE	MAE
96	① baseline only	0.387±0.001	0.405±0.001
	② + $\mathcal{L}_D$	0.377±0.002	0.391±0.000
	③ + ρ	0.387±0.001	0.399±0.001
	④ + $\mathcal{L}_D$ + α	0.378±0.001	0.391±0.001
	⑤ + $\mathcal{L}_D$ + ρ (TDAlign)	**0.375±0.001**	**0.389±0.000**
192	① baseline only	0.441±0.002	0.436±0.001
	② + $\mathcal{L}_D$	0.430±0.001	0.422±0.001
	③ + ρ	0.438±0.001	0.428±0.001
	④ + $\mathcal{L}_D$ + α	0.430±0.002	0.423±0.001
	⑤ + $\mathcal{L}_D$ + ρ (TDAlign)	**0.426±0.001**	**0.420±0.001**
336	① baseline only	0.491±0.004	0.461±0.003
	② + $\mathcal{L}_D$	0.473±0.001	0.445±0.001
	③ + ρ	0.484±0.002	0.454±0.002
	④ + $\mathcal{L}_D$ + α	0.473±0.001	0.445±0.001
	⑤ + $\mathcal{L}_D$ + ρ (TDAlign)	**0.469±0.001**	**0.442±0.001**
720	① baseline only	0.511±0.010	0.494±0.006
	② + $\mathcal{L}_D$	0.470±0.009	0.466±0.007
	③ + ρ	0.482±0.010	0.476±0.006
	④ + $\mathcal{L}_D$ + α	0.471±0.009	0.466±0.007
	⑤ + $\mathcal{L}_D$ + ρ (TDAlign)	**0.460±0.001**	**0.460±0.001**

C. Ablation Studies

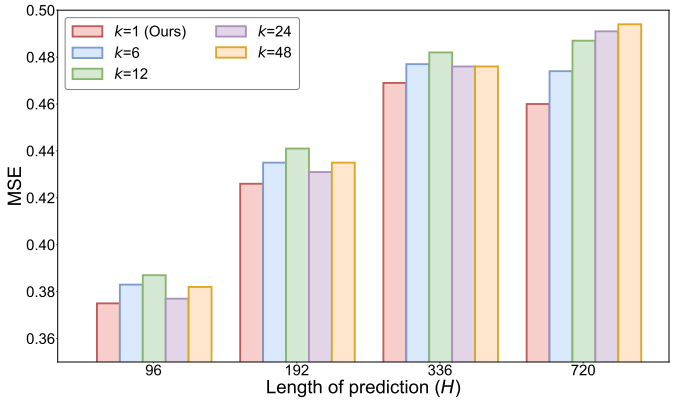
We conduct ablation experiments on the ETTh1 dataset to evaluate the effectiveness of devised components of TDAlign, using the iTransformer [29] as the baseline method.

1) Effectiveness of $\mathcal{L}_D$ and ρ: Our TDAlign contains two key components, including a new loss function $\mathcal{L}_D$ for guiding the model to capture TDT, as well as an adaptive loss balancing strategy using a weight ρ for dynamically adjusting the relative importance between the conventional optimization objective and the TDT learning objective. We analyze their impact through experiments with five different settings:

- baseline only, where the overall loss is $\mathcal{L} = \mathcal{L}_Y$.
- baseline with $\mathcal{L}_D$, where the overall loss is $\mathcal{L} = \mathcal{L}_Y + \mathcal{L}_D$.
- baseline with ρ, where the overall loss is $\mathcal{L} = \rho \cdot \mathcal{L}_Y$.
- baseline with $\mathcal{L}_D$ and α (i.e., replacing ρ by a learning weight α), where the overall loss is $\mathcal{L} = \alpha \cdot \mathcal{L}_Y + (1 - \alpha) \cdot \mathcal{L}_D$.
- baseline with TDAlign, where the overall loss is $\mathcal{L} = \rho \cdot \mathcal{L}_Y + (1 - \rho) \cdot \mathcal{L}_D$.



(a) Using τ-order differencing values to formulate TDT and TDP.



(b) Using change values between time steps separated by the interval k to formulate TDT and TDP.

Fig. 4: Ablation of the formulation of temporal dependencies on the ETTh1 dataset, with the iTransformer [29] as the baseline method.

The results are presented in Table VI, from which we observe that: First, TDAlign significantly improves the forecasting performance of iTransformer across all four prediction lengths, as shown by the comparison between ① and ⑤. Second, learning TDT is crucial for improving forecasting performance. Specifically, ② enables the model to account for the change value between adjacent time steps when forecasting, while ③ enables the model to consider the change direction between adjacent time steps when forecasting. Both settings have superior forecasting performance compared to the baseline method, which overlooks internal correlations across multiple steps. Third, our proposed new loss function has a more significant impact on TDAlign's performance than our adaptive loss balancing strategy, where ② outperform ③ by a large margin. Fourth, our proposed adaptive loss balancing strategy effectively controls the relative importance between the conventional optimization objective and the TDT learning objective. Specifically, by incorporating ρ, ⑤ outperforms ②, which only incorporates $\mathcal{L}_D$, and ④, where ρ is replaced by a learning weight α.

2) Comparison of the formulation of temporal dependencies: As shown in Figure 4, we evaluate our approach of using first-order differencing values (i.e., change values between adjacent time steps) to formulate temporal dependencies by comparing it with two variants. The first variant uses τ-order differencing values to formulate temporal dependencies. Taking the TDT as an example, its formulation is given by $D = \left\{d_{t+1}^{(\tau)}, d_{t+2}^{(\tau)}, \cdots, d_{t+H}^{(\tau)}\right\} \in \mathbb{R}^{H\times N}$, where $d_{t+i}^{(\tau)} = d_{t+i}^{(\tau-1)} - d_{t+i-1}^{(\tau-1)}$ and d_{t+i}^{1} corresponds to d_{t+i} defined in Equation 2. We experiment with $\tau \in \{1, 2, 3, 4\}$ The second variant uses change values between time steps separated by the interval k to formulate temporal dependencies. Taking the TDT as an example, its formulation is given by $D = \{d_{t+1}, d_{t+2}, \cdots, d_{t+H}\} \in \mathbb{R}^{H\times N}$, where $d_{t+i} = d_{t+i+k} - d_{t+i}$. We experiment with $k \in \{1, 6, 12, 24, 48\}$.

The results in Figure 4(a) show that using first-order differencing values (i.e., $\tau = 1$) achieves the best performance, with performance degrading as τ increases. Although higher-order differencing values represent more complex temporal dependencies, they may not provide substantial additional information compared to first-order differencing values. Instead, they tend to overshadow lower-order dependency information and over-constrain the model's learning, thereby compromising its learning and generalization ability. Moreover, since each $d_{t+i}^{(\tau)}$ involves more time steps, higher-order differencing values become more sensitive to noise, leading to degraded model performance. The results in Figure 4(b) indicate that using change values between adjacent time steps (i.e., $k = 1$) yields the best performance. This phenomenon can be attributed to the fact that as k increases, the correlation between x_{t+i} and x_{t+i+k} typically weakens. Learning unreliable temporal dependency information adversely affects model performance. These results demonstrate that using first-order differencing values (i.e., change values between adjacent time steps) to formulate temporal dependencies is simple yet effective.

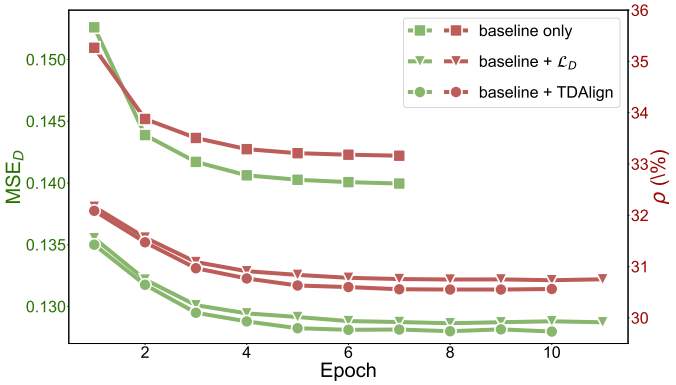
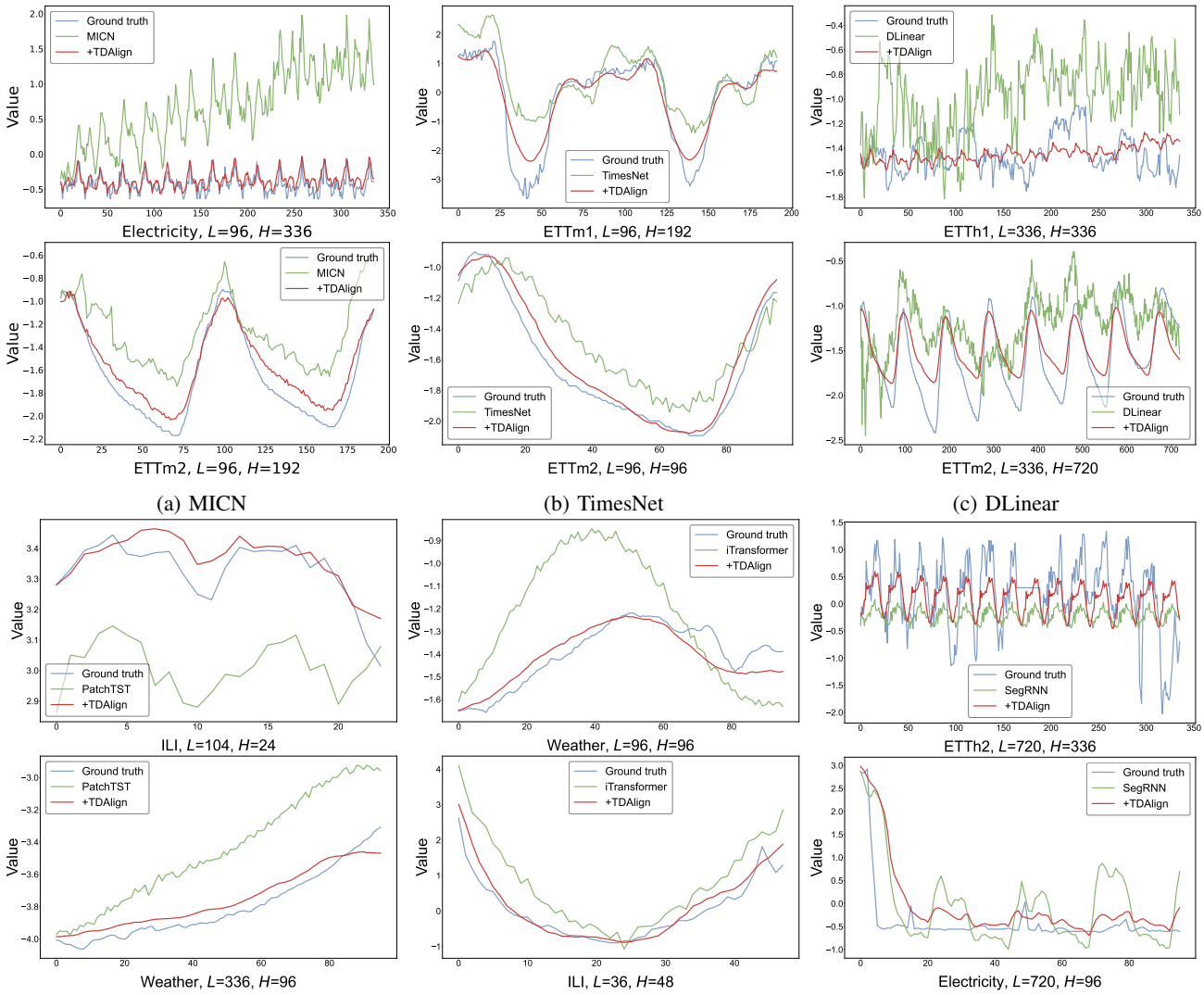


Fig. 5: Loss curves on the validation set of the ETTh1 dataset, with iTransformer [29] as the baseline method.

D. Visualization

For an intuitive understanding of the training process of TDAlign, Figure 5 displays the loss curves of the MSE_D and ρ metrics on the ETTh1 dataset's validation set. We employ iTransformer [29] as the baseline method, with both input and prediction lengths set at 96 steps. These validation loss curves are from experiments with three different settings:



(a) MICN (b) TimesNet (c) DLinear

(d) PatchTST (e) iTransformer (f) SegRNN

Fig. 6: Comparison of long-term time series forecasting results between our TDAlign and six baseline models.

- baseline only, where the overall loss is $\mathcal{L} = \mathcal{L}_Y$.
- baseline with $\mathcal{L}_D$, where the overall loss is $\mathcal{L} = \mathcal{L}_Y + \mathcal{L}_D$.
- baseline with TDAlign, where the overall loss is $\mathcal{L} = \rho \cdot \mathcal{L}_Y + (1 - \rho) \cdot \mathcal{L}_D$.

From these curves, we observe the following: First, the training process of TDAlign is stable, where loss curves of ③ decrease smoothly. Second, explicitly modeling TDT significantly improves the prediction accuracy of both change directions and change values between adjacent time steps, as shown by the lower MSE_D and ρ of ② compared to ①. Third, the devised adaptive loss balancing strategy further enhances TDAlign's performance, as indicated by the further reduction in MSE_D and ρ of ③ compared to ②.

In order to better illustrate the effectiveness of TDAlign, Figure 6 presents a comprehensive comparison of forecasting results between TDAlign and six baseline methods. Overall, baseline methods produce inaccurate predictions with significant deviations from the ground truth. A closer examination of their forecasting results reveals that these methods struggle to accurately capture both the change values and the change directions. In contrast, TDAlign proposes a TDT learning objective for capturing variation patterns of time series. Moreover, it dynamically adjusts the importance of this TDT learning objective relative to the conventional forecasting objective based on the model's learning state. Consequently, TDAlign demonstrates remarkable improvements over baseline methods, consistently generating more accurate and realistic forecasting results that align with the ground truth.

V. Conclusion

In this study, we investigated existing LTSF methods, revealing that their forecasting performance is limited by the inadequate modeling of TDT. To alleviate this problem, we proposed the TDAlign framework, which can plug into existing LTSF methods without introducing additional learnable parameters. In addition, we conducted extensive experiments on seven widely used benchmark datasets, and the results demonstrate that TDAlign consistently enhances the forecasting performance of six state-of-the-art LTSF methods by a

significant margin, showcasing its flexibility and effectiveness. We hope that TDAlign can serve as a fundamental component for time series forecasting and offer a new perspective for improving forecasting performance.

Despite its promising performance, TDAlign currently has limitations in handling irregularly sampled time series and time series with missing values. Future research could focus on extending TDAlign to address these challenges, potentially broadening its applicability to more diverse time series types.

REFERENCES

[1] S. Feng, C. Miao, K. Xu, J. Wu, P. Wu, Y. Zhang *et al.*, "Multi-scale attention flow for probabilistic time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 5, pp. 2056–2068, 2024.

[2] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning : A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, p. 106181, 2020.

[3] L. Yang, Z. Luo, S. Zhang, F. Teng, and T. Li, "Continual learning for smart city: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, pp. 7805–7824, 2024.

[4] S. Zhang, Z. Luo, L. Yang, F. Teng, and T. Li, "A survey of route recommendations: Methods, applications, and opportunities," *Information Fusion*, vol. 108, p. 102413, 2024.

[5] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, "A multi-view multi-task learning framework for multi-variate time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7665–7680, 2023.

[6] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902–924, 2017.

[7] A. G. Salman, B. Kanigoro, and Y. Heryadi, "Weather forecasting using deep learning techniques," in *Proceedings of the International Conference on Advanced Computer Science and Information Systems*, 2015, pp. 281–285.

[8] Y. Gong, T. He, M. Chen, B. Wang, L. Nie, and Y. Yin, "Spatio-temporal enhanced contrastive and contextual learning for weather forecasting," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[9] M. S. Al-Musaylh, R. C. Deo, J. F. Adamowski, and Y. Li, "Short-term electricity demand forecasting with mars, svr and arima models using aggregated demand data in queensland, australia," *Advanced Engineering Informatics*, vol. 35, pp. 1–16, 2018.

[10] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[11] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proceedings of the International Conference on Learning Representations*, 2018.

[12] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017, pp. 2627–2633.

[13] Z. Li, S. Qi, Y. Li, and Z. Xu, "Revisiting long-term time series forecasting: An investigation on linear mapping," *arXiv preprint arXiv:2305.10721*, 2023.

[14] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, Jun. 2023, pp. 11 121–11 128.

[15] A. Das, W. Kong, A. Leach, R. Sen, and R. Yu, "Long-term forecasting with tide: Time-series dense encoder," *arXiv preprint arXiv:2304.08424*, 2023.

[16] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma *et al.*, "Scinet: Time series modeling and forecasting with sample convolution and interaction," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 5816–5828.

[17] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "Micn: Multi-scale local and global context modeling for long-term series forecasting," in *Proceedings of the International Conference on Learning Representations*, 2023.

[18] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *Proceedings of the International Conference on Learning Representations*, 2023.

[19] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang *et al.*, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[20] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proceedings of the International Conference on Learning Representations*, 2023.

[21] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Proceedings of the International Conference on Learning Representations*, 2023.

[22] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang, "Segrnn: Segment recurrent neural network for long-term time series forecasting," *arXiv preprint arXiv:2308.11200*, 2023.

[23] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu *et al.*, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proceedings of the International Conference on Learning Representations*, 2021.

[24] J. Zhang, J. Wang, W. Qiang, F. Xu, C. Zheng, F. Sun *et al.*, "Intriguing properties of positional encoding in time series forecasting," *CoRR*, vol. abs/2404.10337, 2024.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014, pp. 1724–1734.

[27] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, May 2021, pp. 11 106–11 115.

[28] L. Shen and J. Kwok, "Non-autoregressive conditional diffusion models for time series prediction," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2023, pp. 31 016–31 029.

[29] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma *et al.*, "itransformer: Inverted transformers are effective for time series forecasting," in *Proceedings of the International Conference on Learning Representations*, 2024.

[30] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[31] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proceedings of the International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 95–104.

[32] Y. Tan, L. Xie, and X. Cheng, "Neural differential recurrent neural network with adaptive time steps," *arXiv preprint arXiv:2306.01674*, 2023.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez *et al.*, "Attention is all you need," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 30, 2017.

[34] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 22 419–22 430.

[35] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2022, pp. 27 268–27 286.

[36] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 9881–9893.

[37] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang *et al.*, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[38] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[39] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, 2022, pp. 16 000–16 009.

[40] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, and J. Kalagnanam, "Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 459–469.

[41] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 17 105–17 115.

[42] Z. Shao, Z. Zhang, F. Wang, W. Wei, and Y. Xu, "Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting," in *Proceedings of the ACM International Conference on Information & Knowledge Management*, 2022, pp. 4454–4458.

[43] C. Wang, Q. Qi, J. Wang, H. Sun, Z. Zhuang, J. Wu *et al.*, "Rethinking the power of timestamps for robust time series forecasting: A global-local fusion perspective," in *Proceedings of the Advances in Neural Information Processing Systems*, 2024.

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

[45] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[46] F. Huang, T. Tao, H. Zhou, L. Li, and M. Huang, "On the learning of non-autoregressive transformers," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2022, pp. 9356–9376.

[47] Y. Xiao, L. Wu, J. Guo, J. Li, M. Zhang, T. Qin *et al.*, "A survey on non-autoregressive generation for neural machine translation and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 10, pp. 11 407–11 427, 2023.

[48] A. Jadon, A. Patil, and S. Jadon, "A comprehensive survey of regression-based loss functions for time series forecasting," in *Proceedings of the International Conference on Data Management, Analytics & Innovation*. Springer, 2024, pp. 117–147.

[49] W. Xue, X. Mou, L. Zhang, and X. Feng, "Perceptual fidelity aware mean squared error," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 705–712.

[50] V. Le Guen and N. Thome, "Shape and time distortion loss for training deep time series forecasting models," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[51] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Australia: OTexts, 2018.

[52] D. A. Dickey and S. G. Pantula, "Determining the order of differencing in autoregressive processes," *Journal of Business & Economic Statistics*, vol. 5, no. 4, pp. 455–461, 1987.

[53] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang *et al.*, "Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods," *Proceedings of the VLDB Endowment*, vol. 17, no. 9, pp. 2363–2377, 2024.