DDN: Dual-domain Dynamic Normalization for Non-stationary Time Series Forecasting

Tao Dai[1,*], **Beiliang Wu**[1,*], **Peiyuan Liu**[2,†], **Naiqi Li**[2,†], **Xue Yuerong**[2], **Shu-Tao Xia**[2], **Zexuan Zhu**[1]
[1]College of Computer Science and Software Engineering, Shenzhen University, China
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
{daitao.edu, peiyuanliu.edu, linaiqi.thu}@gmail.com; xiast@sz.tsinghua.edu.cn

Abstract

Deep neural networks (DNNs) have recently achieved remarkable advancements in time series forecasting (TSF) due to their powerful ability of sequence dependence modeling. To date, existing DNN-based TSF methods still suffer from unreliable predictions for real-world data due to its non-stationarity characteristics, *i.e.,* data distribution varies quickly over time. To mitigate this issue, several normalization methods (e.g., SAN) have recently been specifically designed by normalization in a fixed period/window in the time domain. However, these methods still struggle to capture distribution variations, due to the complex time patterns of time series in the time domain. Based on the fact that wavelet transform can decompose time series into a linear combination of different frequencies, which exhibits distribution variations with time-varying periods, we propose a novel Dual-domain Dynamic Normalization (DDN) to dynamically capture distribution variations in both time and frequency domains. Specifically, our DDN tries to eliminate the non-stationarity of time series via both frequency and time domain normalization in a sliding window way. Besides, our DDN can serve as a plug-in-play module, and thus can be easily incorporated into other forecasting models. Extensive experiments on public benchmark datasets under different forecasting models demonstrate the superiority of our DDN over other normalization methods. Code is available at https://github.com/Hank0626/DDN.

1 Introduction

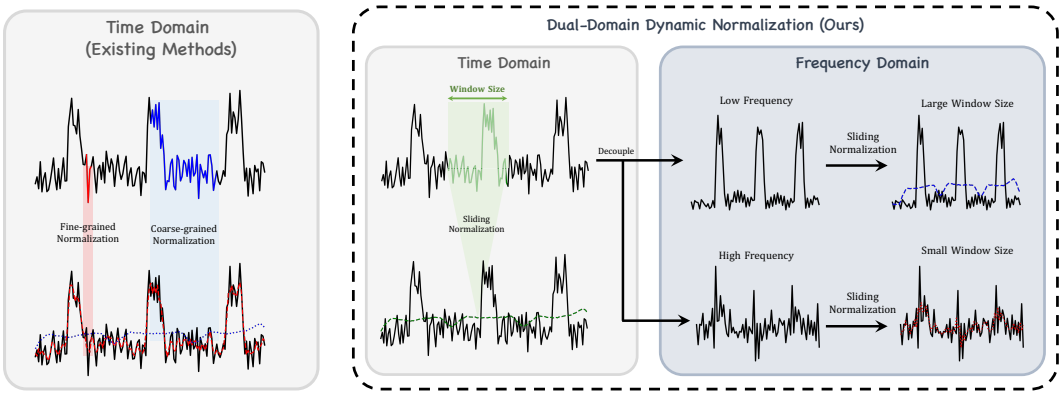
Deep neural networks (DNNs) with powerful dependency modeling capability have recently been widely used in time series forecasting (TSF) applications, including weather prediction [1], energy consumption estimation [2], and traffic flow forecasting [3]. Despite the great advancements of DNN-based TSF methods [4, 5, 6, 7], they still suffer from unreliable predictions for real-world data due to its non-stationary nature of real-world time series, *i.e.,* data distribution within the series varies quickly over time (*a.k.a*, distribution drift [8, 9, 10]). Such non-stationary challenge limits the real applications of DNN-based TSF methods.

To mitigate the problem of distribution drift, the classic reversible normalization [11] has recently been proposed with a two-stage pipeline of normalization and de-normalization. The former stage of normalization eliminates non-stationary factors for converting a non-stationary sequence into a stationary sequence, which has to acquire the mean and standard deviation of the sequence before. The latter stage of de-normalization reconstructs non-stationary information from the distribution prediction model or directly reuses the mean and standard deviation acquired in normalization.

[*]Equal contribution
[†]Correspondence to: Peiyuan Liu and Naiqi Li

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-3444



(a) (b)

Figure 1: (a) Existing methods with a fixed period/window normalization struggle to capture distribution variations. (b) Our method dynamically captures distribution variations in both time and frequency domains.

Later, several advanced variants of reversible normalization [12, 13, 14] have achieved impressive performance by further alleviating the non-stationary property of real-world time series.

Despite the great success of normalization methods, existing methods are limited in capturing distribution variations by performing normalization with a fixed period/window. As shown in Figure 1a, either existing coarse-grained (e.g., RevIN [11]) or fine-grained normalization (e.g., SAN [14]) in single time domain tends to produce sub-optimal performance. On the other hand, it is known that wavelet transform can decompose time series into a time-dependent sum of frequency components, which exhibits distribution variations with time-varying periods (see Figure 1b). Thus, making full use of such frequency information is helpful to capture distribution variations with time-varying periods and intensities. These observations motivate us to develop a more powerful normalization strategy to dynamically capture distribution variations.

In this paper, we propose a novel **Dual-domain Dynamic Normalization (DDN)** framework to dynamically capture distribution variations in both times and frequency domains in a sliding window way. Specifically, our DDN decomposes the original time series into different frequency components, including low-frequency and high-frequency components, based on Discrete Wavelet Transform (DWT) [15, 16]. Followed by performing sliding normalization in an individual frequency component with proper window size (see Figure 1b), which is helpful to capture distribution variations with time-varying periods and intensities. Besides, time domain normalization is developed to compute local sliding statistics [17], including sliding mean and sliding standard deviation. Unlike the previous works that process a coarse-grained level, our DDN leverages fine-grained a more informative sliding window to calculate distribution characteristics for every time step.

Our main contributions can be summarized as: **(i)** We propose a novel Dual-domain Dynamic Normalization (DDN) to dynamically capture distribution variations in both time and frequency domains with sliding statistics. Compared with previous works, our DDN is capable of dynamically reflecting the rapid variations to time series. **(ii)** Our DDN aims to eliminate non-stationary factors with frequency domain normalization and time domain normalization. Benefiting from the complementary properties of the time and frequency domain information, it allows our DDN to further clarify non-stationary factors and reconstruct non-stationary information. **(iii)** Extensive experiments demonstrate the effectiveness of our DDN, by achieving significant performance improvements across various baseline models on seven real-world datasets.

2 Related Works

2.1 Deep Models for Time Series Forecasting

Reviewing the development of time series forecasting based on deep models, early methods [18, 19, 20] often integrated cross-dimension information in embedding module, then modeling cross-time

information. In contrast, recent Sota methods indicate that two modeling ways can be better: CI (Channel Independent) and CD (Channel Dependent). The primary distinction between the two approaches lies in the former focusing only on cross-time features but the latter incorporating cross-dimension features. Theoretically, the latter can leverage more information and achieve higher prediction accuracy [21, 22, 23]. In practice, for relatively short input series, CD methods [24, 25] achieve comparable or even better performance than the CI methods. However, for longer input sequences, the situation is often the opposite[26, 4, 27]. In recent research, this difference can be attributed to the CD having higher capacity but often lacking robustness in predicting distributional drift than CI [28, 29], while longer series typically experience more severe distributional drift. The superior performance of CI highlights the importance of handling distribution drift, and it is a valuable direction in the current research on time series forecasting.

2.2 Stationary for Time Series Forecasting

RevIN [11] was the first work to apply reversible normalization for time series forecasting, which assumes that history and future sequences share the same distribution. It counts distribution statistics of historical sequence for both normalization and de-normalization. Due to its simplicity and impressive effectiveness, it has been widely used in recent works [30, 31]. However, RevIN overlooks the distributional differences between historical and future sequences. Building upon RevIN, Dish-TS [12] proposes different distribution characteristics for historical and future sequences, using a distribution forecasting model to predict mean and standard deviation. Concurrently, NST [13] employs a module to provide more consistent distribution with future distribution, which can refer to appendix B. Furthermore, SAN [14] notes that existing distribution assumptions may not adapt to the scenario that time series points rapidly change over time [32, 33] and proposes a more fine-grained method, which supposes the distribution characteristics of time points is different between slices but same within a slice. Nevertheless, SAN still stops at the slice level, rather than the time series point level. Meanwhile, existing works lack consideration of the discrepancies between low and high frequencies, leading to insufficient consideration of non-stationary information.

3 Methodology

In the realm of multivariate time series forecasting, we consider a historical sequence $\boldsymbol{X} \in \mathbb{R}^{M \times L}$ and aim to predict the corresponding future sequence $\boldsymbol{Y} \in \mathbb{R}^{M \times T}$, M is the number of channels. DDN is a model-agnostic plugin designed to align the distribution characteristics of $\boldsymbol{X}$ and accurately estimate the distribution of $\boldsymbol{Y}$. In this section, we will comprehensively outline the pipeline of the entire framework and elaborate on how to remove and reconstruct non-stationary factors of time series. To enhance clarity and facilitate understanding of subsequent chapters, the key notations used in this paper are summarized in Table 1, and the framework can be referred to in Figure 2.

Notation	Description
L, T	The time steps of the historical/future sequences
$\boldsymbol{x}^i, \boldsymbol{y}^i$	The i-th historical or future series
$\bar{\boldsymbol{x}}^i_*, \bar{\boldsymbol{y}}^i_*$	The $\boldsymbol{x}^i$ after normalization and it predicted series
$\boldsymbol{\mu}^i, \boldsymbol{\sigma}^i$	The i-th mean or standard deviation series of $\boldsymbol{x}^i$
$\boldsymbol{\mu}^i_y, \boldsymbol{\sigma}^i_y$	The i-th mean or standard deviation series of $\boldsymbol{y}^i$
$\boldsymbol{\mu}^i_*, \boldsymbol{\sigma}^i_*$	The distribution forecasting of $\boldsymbol{\mu}^i$ or $\boldsymbol{\sigma}^i$

Table 1: Summary of key mathematical notations

3.1 Overall Framework

As depicted in Figure 2, we first eliminate non-stationary factors via both the Frequency Domain Normalization (**FDN**) and the Time Domain Normalization (**TDN**). These processes output two stationary sequences and two sets of distribution characteristics. Two stationary sequences weighted to a sequence and input to the time series Forecasting Model (**FM**) for future sequence forecasting, while two sets of non-stationary factors input to Distribution Prediction Model (**DPM**) and predict future non-stationary factors. Finally, these factors are weighted together and incorporated with forecasting

https://doi.org/10.52202/079017-3444

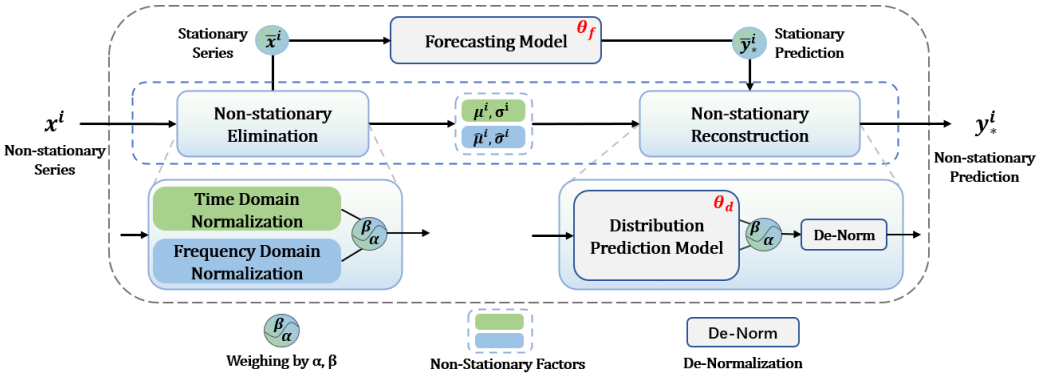


Figure 2: The comprehensive time series forecasting framework comprises a time series forecasting model and an auxiliary module designed for handling non-stationary factors. This auxiliary module consists of two sub-modules: one for eliminating non-stationary factors and another for reconstructing them. The non-stationary factor elimination sub-module includes Time Domain Normalization and Frequency Domain Normalization, while the non-stationary factor reconstruction sub-module incorporates a distribution prediction module.

output to reconstruct non-stationary factors by de-normalization. Here, θ_d and θ_f correspond to the parameters of DPM and FM, and the training strategy can be seen in the section 3.4.

3.2 Non-stationarity Elimination

For each series $\boldsymbol{x}^i$, we perform a sliding window along the temporal dimension to acquire distribution characteristics, then replicate padding that will align the length of sliding statistics to the original series. Finally, sliding mean $\boldsymbol{\mu}^i$ and sliding standard deviation $\boldsymbol{\sigma}^i$ represent to the distribution characteristics of $\boldsymbol{x}^i$. This process can be described as follows:

$$\begin{aligned} \mu_j^i = \frac{1}{2k+1}\sum_{-k}^{k} x_{j+t}^i, \quad \left(\sigma_j^i\right)^2 = \frac{1}{2k+1}\sum_{-k}^{k}\left(x_{j+t}^i - \mu_j^i\right)^2, \\ \boldsymbol{\mu}^i = \mathrm{Pad}(\{\mu_{k+1}^i, \cdots, \mu_{L-k}^i\}), \quad \boldsymbol{\sigma}^i = \mathrm{Pad}(\{\sigma_{k+1}^i, \cdots, \sigma_{L-k}^i\}). \end{aligned} \tag{1}$$

Here $2k+1$ is the size of the sliding window, and stride is 1. After that, the size of sliding statistics is $L-2k$. Where ${\mu_j}^i$ and ${\sigma_j}^i$ represent the mean value and standard deviation value of the j^{th} time point respectively, where $j \in \{k+1, \cdots, L-k\}$. To make sure each time point possesses corresponding sliding statistics. We copy the sliding statistics closest in time by $\mathrm{Pad}(\cdot)$ operation, the obtaining $\boldsymbol{\mu_i}$ and $\boldsymbol{\sigma_i}$ are used to achieve the transformation from non-stationary sequences to stationary sequences. The process is as follows:

$$\bar{\boldsymbol{x}}^i = \frac{1}{\boldsymbol{\sigma}^i + \epsilon} \odot \left(\boldsymbol{x}^i - \boldsymbol{\mu}^i\right), \quad \epsilon > 0. \tag{2}$$

Here, $\bar{\boldsymbol{x}}^i$ is the stationary series, ϵ is a positive number to prevent the denominator from zero, and $\odot$ denotes the element-wise product. By this sliding normalization, annotated as $\mathrm{SlidingNorm}(\cdot)$, we can acquire the non-stationary factors of each time point and convert non-stationary sequences to stationary sequences.

Frequency Domain Normalization. In this branch, to exhaustively unveil non-stationary factors and eliminate them accurately. Discrete Wavelet Transform (DWT) is conducted on $\boldsymbol{x}_i$ to separate the low-frequency component $\boldsymbol{x}_l^i$ and high-frequency component $\boldsymbol{x}_h^i$. Subsequently, we acquire and eliminate their non-stationary factors. The process is as follows:

$$\begin{aligned} \boldsymbol{x}_l^i, \boldsymbol{x}_h^i = \mathrm{DWT}_{\phi_{l,h}}(\boldsymbol{x}^i), \\ \bar{\boldsymbol{x}}_l^i, \boldsymbol{\mu}_l^i, \boldsymbol{\sigma}_l^i = \mathrm{SlidingNorm}(\boldsymbol{x}_l^i), \quad \bar{\boldsymbol{x}}_h^i, \boldsymbol{\mu}_h^i, \boldsymbol{\sigma}_h^i = \mathrm{SlidingNorm}(\boldsymbol{x}_h^i), \end{aligned} \tag{3}$$

Here, $\phi_{l,h}$ is a pair of learnable wavelet bases. $\bar{\boldsymbol{x}}_l^i$, $\boldsymbol{\mu}_l^i$, and $\boldsymbol{\sigma}_l^i$ represent the stationary sequence, sliding mean, and sliding standard deviation of the low-frequency component. While $\bar{\boldsymbol{x}}_h^i$, $\boldsymbol{\mu}_h^i$, and $\boldsymbol{\sigma}_h^i$

denote those of the high-frequency component. In practice, different types of DWT have different padding lengths and lead to different output lengths. To ensure a consistent and clear output length, Inverse Discrete Wavelet Transform (IDWT) performs to restore a definite size. The process is as follows:

$$\hat{\boldsymbol{x}}^i = \mathrm{IDWT}_{\phi_{l,h}}(\bar{\boldsymbol{x}}_l^i, \bar{\boldsymbol{x}}_h^i), \quad \hat{\boldsymbol{\mu}}^i = \mathrm{IDWT}_{\phi_{l,h}}(\boldsymbol{\mu}_l^i, \boldsymbol{\mu}_h^i),\ \hat{\boldsymbol{\sigma}}^i = \mathrm{IDWT}_{\phi_{l,h}}(\boldsymbol{\sigma}_l^i, \boldsymbol{\sigma}_h^i). \tag{4}$$

Where $\hat{\boldsymbol{x}}^i$, $\hat{\boldsymbol{\mu}}^i$, and $\hat{\boldsymbol{\sigma}}^i$ encompass the stationary sequences, sliding means, and sliding standard deviations of different frequency components. Through these operations, the output stationary sequence and distribution statistics maintain consistency with the dimensions of the input non-stationary sequence.

Time Domain Normalization. We conduct the same manners in the time domain without frequency decomposition. The process can be formulated as follows:

$$\bar{\boldsymbol{x}}^i, \boldsymbol{\mu}^i, \boldsymbol{\sigma}^i = \mathrm{SlidingNorm}(\boldsymbol{x}^i), \tag{5}$$

The wavelet transform in FDN typically involves padding, which can potentially distort the statistical distribution information of the decomposed sequences. To address this, we implement sliding normalization directly on the original sequence. Consequently, the resulting distribution information is utilized for predicting future distributions, while the output stationary sequence is weighted with the stationary sequence derived from FDN.

Stationary Sequences Weighting. Two stationary sequences from FDN and TDN will be weighted to a stationary output, which can be expressed as follows:

$$\bar{\boldsymbol{x}}^i = \bar{\boldsymbol{x}}^i \cdot \beta + \hat{\boldsymbol{x}}^i \cdot \alpha. \tag{6}$$

Here, α is a trainable parameter and $\beta = 1 - \alpha$. The weighted $\bar{\boldsymbol{x}}^i$ serves as the final stationary sequence, which is then inputted into FM for stable forecasting.

3.3 Non-stationarity Reconstruction

We acquire two sets of sliding statistics reflecting distribution variations after FDN and TDN. Later, we refer to the structure of existing distribution prediction works [34, 12] to predict future distribution. Initially, we calculate the mean value of each sliding statistic to compute the statistical differences. Subsequently, the difference and original series are inputted for future difference prediction. Ultimately, these predicted differences added to the mean value as future sliding statistics.

Frequency Domain Prediction. As shown in Figure 3, for the distribution statistics of FDN, we predict future statistics by distribution prediction model and formulate as:

$$\begin{aligned} \hat{\boldsymbol{\sigma}}_\Delta^i &= \mathrm{SP}\left(\hat{\boldsymbol{\sigma}}^i - \sigma_f^i, \boldsymbol{x}^i\right), \quad \hat{\boldsymbol{\sigma}}_*^i = \hat{\boldsymbol{\sigma}}_\Delta^i + \sigma_f^i, \\ \hat{\boldsymbol{\mu}}_\Delta^i &= \mathrm{MP}\left(\hat{\boldsymbol{\mu}}^i - \mu_f^i, \boldsymbol{x}^i - \mu_f^i\right), \quad \hat{\boldsymbol{\mu}}_*^i = \hat{\boldsymbol{\mu}}_\Delta^i + \mu_f^i. \end{aligned} \tag{7}$$

Here, μ_f^i and σ_f^i are the mean values of $\hat{\boldsymbol{\mu}}^i$ and $\hat{\boldsymbol{\sigma}}^i$. While $\hat{\boldsymbol{\mu}}_*^i$ and $\hat{\boldsymbol{\sigma}}_*^i$ denote the prediction of $\hat{\boldsymbol{\mu}}^i$ and $\hat{\boldsymbol{\sigma}}^i$. The MP is a mean prediction branch, and the SP is a standard deviation prediction branch. They are affiliated with DPM and adopt the same network structure.

Time Domain Prediction As the above frequency domain prediction process, for the distribution statistics of TDN, this prediction process can be formulated as follows:

$$\begin{aligned} \boldsymbol{\sigma}_\Delta^i &= \mathrm{SP}\left(\boldsymbol{\sigma}^i - \sigma_o^i, \boldsymbol{x}^i\right), \quad \boldsymbol{\sigma}_*^i = \boldsymbol{\sigma}_\Delta^i + \sigma_o^i, \\ \boldsymbol{\mu}_\Delta^i &= \mathrm{MP}\left(\boldsymbol{\mu}^i - \mu_o^i, \boldsymbol{x}^i - \mu_o^i\right), \quad \boldsymbol{\mu}_*^i = \boldsymbol{\mu}_\Delta^i + \mu_o^i. \end{aligned} \tag{8}$$

Here, μ_o^i and σ_o^i are the mean values of $\boldsymbol{\mu}^i$ and $\boldsymbol{\sigma}^i$, respectively. Likewise, $\boldsymbol{\mu}_*^i$ and $\boldsymbol{\sigma}_*^i$ denote the prediction of $\boldsymbol{\mu}^i$ and $\boldsymbol{\sigma}^i$. The SP and the MP are noted in frequency domain prediction.

https://doi.org/10.52202/079017-3444

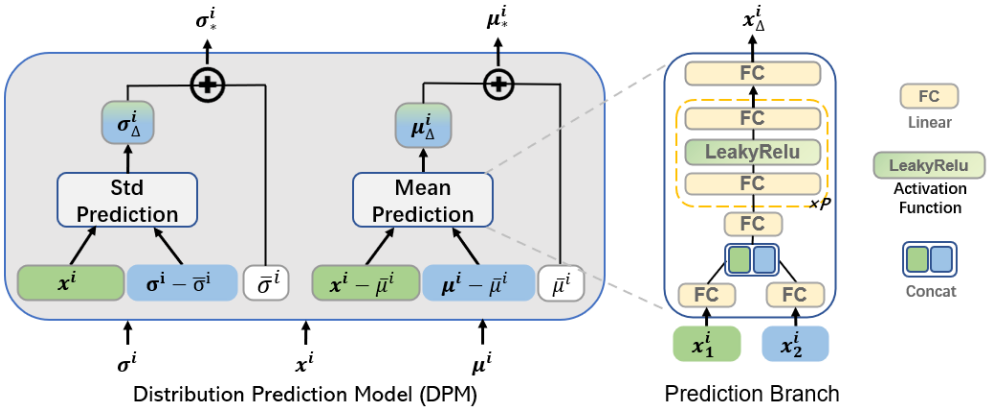


Figure 3: The architecture of the distribution prediction model primarily consists of two predictive branches: the Mean Prediction branch and the Standard Deviation (Std) Prediction branch. The specific network structure of these branches is illustrated in the Prediction Branch.

De-normalization. After the aforementioned distribution predictions, two sets of estimated distribution statistics will be gained. Which are weighted to reconstruct the non-stationary information of the output of the time series forecasting model. This process can be described as follows:

$$\begin{gathered}\boldsymbol{\mu}_*^i = \boldsymbol{\mu}_*^i \cdot \beta + \hat{\boldsymbol{\mu}}_*^i \cdot \alpha, \quad \boldsymbol{\sigma}_*^i = \boldsymbol{\sigma}_*^i \cdot \beta + \hat{\boldsymbol{\sigma}}_*^i \cdot \alpha, \quad \bar{\boldsymbol{x}}^i = \bar{\boldsymbol{x}}^i \cdot \beta + \hat{\boldsymbol{x}}^i \cdot \alpha. \\ \boldsymbol{y}_*^i = \bar{\boldsymbol{y}}_*^i \odot \left(\boldsymbol{\sigma}_*^i + \epsilon\right) + \boldsymbol{\mu}_*^i.\end{gathered} \tag{9}$$

Where $\bar{\boldsymbol{y}}_*^i$ is the output of the time series forecasting model, and $\boldsymbol{y}_*^i$ represents the predicted sequence after reconstructing non-stationary information. While α, β, and ϵ mentioned before.

3.4 Collaborative Training

Distribution prediction and future series forecasting are essentially a bi-level optimization problem [35, 36, 37], where distribution outputs significantly impact the future series output. To enhance the training effects of our models, we pre-train the DPM to yield a relatively well-trained DPM. This procedure can be formulated as follows:

$$\theta_d = \arg\min_{\text{MSE}} \left(\left(\boldsymbol{\mu}_*^i, \boldsymbol{\sigma}_*^i\right), \left(\boldsymbol{\mu}_y^i, \boldsymbol{\sigma}_y^i\right), \theta_d\right). \tag{10}$$

Where θ_d represents the parameters of the DPM, it is noteworthy that the wavelet bases $\phi_{l,h}$ and the weighted factor α belong to θ_d. We select the mean square error (MSE) as our loss function between the predicted distribution and the ground truth of the distribution, acquired from the future sequence through TDN. Subsequently, assuming a total training duration of T epochs, the parameters θ_d of the DPM will be frozen, then we train FM for T_1 epochs. Finally, DPM and FM will be subject to collaborative training during the remaining $T - T_1$ epochs. The process is as follows:

$$\begin{aligned}\theta_f &= \arg\min_{\text{MSE}} \left(\boldsymbol{y}_*^i, \boldsymbol{y}_y^i, \theta_f\right), \quad \text{if } t \leq T_1, \\ \{\theta_d, \theta_f\} &= \arg\min_{\text{MSE}} \left(\boldsymbol{y}_*^i, \boldsymbol{y}_y^i, \{\theta_d, \theta_f\}\right), \quad \text{otherwise.}\end{aligned} \tag{11}$$

Where t denotes the t^{th} epoch during the training process, and ϕ_f represents the parameters of the FM. We train DPM and FM concurrently to mitigate potential errors in the pretraining stage, as the ground truth of distribution is drived from future sequences based on distritution assumption. This ground truth is somewhat inconsistent with the actual situation and fails to account for high-frequency distribution changes. Consequently, we pretrain the DPM using assumption-based distribution ground truth, and then collaborative train it jointly based on the loss derived from the future sequences.

4 Experiments

In this section, we conduct comprehensive experiments on multiple real-world time series datasets to assess the effectiveness of our proposed reversible normalization method DDN.

Datasets We conduct extensive experiments on these seven popular real-world datasets [18], including **Electricity Transformer Temperature (ETT)** with its four subsets (ETTh1, ETTh2, ETTm1, ETTm2), **Weather**, **Electricity**, and **Traffic**. The setting of these datasets following original works [18, 20], and more descriptions about these datasets present in appendix A.1.

Baselines. DDN is a model-agnostic method that can be applied to any mainstream time series forecasting model. To demonstrate its versatility, we integrate DDN into several representative models, including the earlier proposed models **FEDformer** [19] and **Autoformer** [20], the CI model **DLinear** [27], and the CD model **iTransformer** [24].

Implementation details. Our experiments were conducted three times with a consistent random seed and averaged to mean values. The Mean Square Error (MSE) and Mean Absolute Error (MAE) are chosen as evaluation metrics, with MSE serving as the training loss. All models use the same prediction lengths $T = \{96, 192, 336, 720\}$. For the look-back window L, Autoformer [20] and FEDformer [19] use $L = 96$, while DLinear [27] and iTransformer [24] utilize $L = 336$ and $L = 720$ respectively. The wavelet bases initialize to the "coiflet" bases, the default size of our sliding window is set to 7 for information content and temporal locality balance, and α starts at zero. More implementation details of our experiments can be referred to appendix A.2.

4.1 Main Results

As illustrated, the DDN method significantly enhances the predictive performance of the four different baselines across nearly all datasets. For the MSE metric, this improvement is particularly evident in the three relatively large datasets: Weather, Electricity, and Traffic. Utilizing DDN, Autoformer achieves a relative error reduction of 19.2%, 24.7%, and 25.6%, respectively, while FEDformer achieves a relative error reduction of 13.1%, 16.2%, and 22.3%. Similarly, incorporating the DDN into the other models also results in substantial performance gains. Additionally, Autoformer, FEDformer, and DLinear do not employ reversible normalization in official implements. While iTransformer utilizes the RevIN [11] normalization technique based on static statistics. However, replacing the RevIN module in iTransformer with the DDN module still yields significant performance improvements. These results strongly demonstrate that DDN makes the baseline model more robust in forecasting.

Methods		Autoformer		+DDN		FEDformer		+DDN		DLinear		+DDN		iTransformer		+DDN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.458	0.448	**0.427**	**0.424**	**0.371**	0.411	0.385	**0.408**	0.377	0.399	**0.372**	**0.396**	0.392	0.422	**0.377**	**0.405**
	192	0.481	0.474	**0.472**	**0.452**	**0.420**	0.443	**0.415**	**0.452**	0.417	0.426	**0.406**	**0.416**	0.428	0.448	**0.414**	**0.430**
	336	0.508	0.485	**0.498**	**0.466**	**0.446**	0.459	0.458	**0.452**	0.464	0.461	**0.432**	**0.434**	0.467	0.475	**0.453**	**0.456**
	720	0.525	0.516	**0.502**	**0.483**	**0.482**	0.495	0.490	**0.479**	0.493	0.505	**0.462**	**0.474**	0.568	0.547	**0.553**	**0.530**
ETTm1	96	0.493	0.470	**0.354**	**0.390**	0.362	0.408	**0.313**	**0.364**	0.301	0.344	**0.288**	**0.342**	0.322	0.371	**0.301**	**0.355**
	192	0.546	0.498	**0.397**	**0.408**	0.395	0.427	**0.361**	**0.396**	0.335	0.366	**0.324**	**0.364**	0.353	0.392	**0.339**	**0.378**
	336	0.658	0.543	**0.429**	**0.433**	0.441	0.454	**0.417**	**0.430**	0.370	0.387	**0.356**	**0.385**	0.385	0.410	**0.370**	**0.396**
	720	0.626	0.532	**0.488**	**0.464**	0.488	0.481	**0.470**	**0.472**	0.425	0.421	**0.415**	**0.419**	0.441	0.443	**0.426**	**0.426**
Weather	96	0.247	0.320	**0.190**	**0.243**	0.246	0.328	**0.174**	**0.237**	0.175	0.237	**0.146**	**0.201**	0.177	0.228	**0.148**	**0.210**
	192	0.302	0.366	**0.231**	**0.282**	0.281	0.341	**0.233**	**0.294**	0.217	0.275	**0.190**	**0.247**	0.223	0.266	**0.191**	**0.252**
	336	0.362	0.394	**0.289**	**0.327**	0.337	0.376	**0.307**	**0.349**	0.263	0.314	**0.239**	**0.288**	0.287	0.310	**0.237**	**0.290**
	720	0.427	0.433	**0.369**	**0.375**	0.414	0.426	**0.399**	**0.405**	0.325	0.366	**0.311**	**0.343**	0.364	0.365	**0.301**	**0.336**
Electricity	96	0.195	0.309	**0.150**	**0.254**	0.185	0.300	**0.146**	**0.251**	0.140	0.237	**0.131**	**0.228**	0.133	0.229	**0.127**	**0.225**
	192	0.215	0.325	**0.173**	**0.275**	0.196	0.310	**0.168**	**0.268**	0.153	0.250	**0.148**	**0.246**	0.154	0.250	**0.146**	**0.246**
	336	0.237	0.344	**0.185**	**0.288**	0.215	0.330	**0.174**	**0.280**	0.168	0.267	**0.164**	**0.264**	0.170	0.266	**0.156**	**0.257**
	720	0.292	0.375	**0.201**	**0.304**	0.244	0.352	**0.216**	**0.312**	0.203	0.301	**0.201**	**0.299**	0.192	0.287	**0.179**	**0.282**
Traffic	96	0.654	0.403	**0.453**	**0.296**	0.579	0.363	**0.442**	**0.288**	0.411	0.283	**0.375**	**0.261**	0.348	0.254	**0.336**	**0.248**
	192	0.654	0.410	**0.462**	**0.304**	0.608	0.376	**0.462**	**0.300**	0.423	0.289	**0.396**	**0.272**	0.364	0.264	**0.347**	**0.254**
	336	0.629	0.391	**0.486**	**0.315**	0.620	0.385	**0.474**	**0.306**	0.437	0.297	**0.411**	**0.279**	0.381	0.272	**0.363**	**0.263**
	720	0.657	0.402	**0.529**	**0.344**	0.630	0.387	**0.512**	**0.329**	0.467	0.316	**0.448**	**0.298**	0.421	0.290	**0.412**	**0.286**

Table 2: Multivariate long-term forecasting results. The best results are highlighted in **bold**. More results can be found in Appendix D.1.

https://doi.org/10.52202/079017-3444

4.2 Comparison With Reversible Normalization Methods

Quantitative evaluation. In this part, we compare recent representative reversible normalization methods using the average MSE metric across four prediction lengths in $\{96, 192, 336, 720\}$. Detailed descriptions of these methods and links for access are provided in the Appendix B. As shown in Table 3, DDN not only significantly enhances the predictive performance of the baseline models, including RevIN [11], NTS [13], and Dish-TS [12], but also demonstrates superior results when compared to existing reversible normalization methods. For instance, taking the recent slice-level SAN [14] method as an example, DDN achieves substantial improvements even upon its solid foundation. Further comparative details are detailed in Appendix D.2.

Methods	Autoformer						FEDformer					
	+DDN	+RevIN	+NST	+Dish-TS	+SAN	IMP	+DDN	+RevIN	+NST	+Dish-TS	+SAN	IMP
ETTh1	**0.475**	0.519	0.521	0.521	0.518	3.7%	**0.437**	0.463	0.456	0.461	0.447	-1.6%
ETTh2	**0.403**	0.489	0.465	1.175	0.411	9.6%	**0.385**	0.465	0.481	1.004	0.404	9.8%
ETTm1	**0.417**	0.562	0.535	0.567	0.406	28.2%	0.390	0.415	0.411	0.422	**0.377**	7.6%
ETTm2	**0.283**	0.325	0.331	0.894	0.311	15.0%	**0.282**	0.310	0.315	0.759	0.287	6.6%
Weather	**0.270**	0.290	0.290	0.433	0.305	19.2%	0.278	0.268	**0.267**	0.398	0.279	13.1%
Electricity	**0.177**	0.219	0.213	0.231	0.204	24.7%	**0.176**	0.200	0.198	0.203	0.191	16.2%
Traffic	**0.483**	0.666	0.664	0.677	0.594	25.6%	**0.473**	0.647	0.649	0.652	0.572	22.3%

Table 3: Comparison between DDN and existing reversible normalization methods of varying granularities. IMP represents the relative percentage improvement of DDN over the original sequence. The best results are highlighted in **bold**.

Methods		DLinear				iTransformer			
		+SAN		+DDN		+SAN		+DDN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.152	0.210	**0.146**	**0.201**	0.150	**0.208**	**0.148**	0.210
	192	0.196	0.253	**0.190**	**0.247**	0.195	0.253	**0.191**	**0.252**
	336	0.246	0.296	**0.239**	**0.288**	0.248	0.295	**0.237**	**0.290**
	720	0.315	0.346	**0.311**	**0.343**	0.311	0.342	**0.301**	**0.336**
Electricity	96	0.137	0.234	**0.131**	**0.228**	0.130	0.229	**0.127**	**0.225**
	192	0.152	0.248	**0.148**	**0.246**	0.148	0.247	**0.146**	**0.246**
	336	0.167	**0.264**	**0.164**	0.264	0.158	0.259	**0.156**	**0.257**
	720	0.202	**0.296**	**0.201**	0.299	0.183	0.284	**0.179**	**0.282**
Traffic	96	0.412	0.290	**0.375**	**0.261**	0.363	0.269	**0.336**	**0.248**
	192	0.431	0.299	**0.396**	**0.272**	0.374	0.274	**0.347**	**0.254**
	336	0.447	0.308	**0.411**	**0.279**	0.389	0.281	**0.363**	**0.263**
	720	0.475	0.320	**0.448**	**0.298**	0.418	0.294	**0.412**	**0.286**

Table 4: Comparison of forecasting errors between the DDN and SAN. The best results are highlighted in **bold**.

Considering that early models often lacked robust generalizability as their naive modeling strategies, we additionally included comparisons with two recent representative models: DLinear [27] (modeling features with CI) and iTransformer [24] (modeling features with CD). These methods already have good non-stationary adaptability, so they can better reflect the performance upper limit of fine-grained normalization methods. As illustrated in table 4, DDN significantly outperforms the recent state-of-the-art model SAN in handling non-stationary information. Overall, DDN almost achieves the best performance compared to SAN in every forecasting case. Specifically, on the Traffic dataset, SAN [14] struggles to achieve satisfactory predictive performance and even performs worse than the original predictive model. In contrast, DDN demonstrates effects by a finer-grain non-stationarity processing.

Qualitative Evaluation. As shown in the figure 4, we compare DDN with reversible normalization methods at different granularities. It can be observed that there are significant differences among the various reversible normalization methods, which are related to their specific implementations. From subplots (a) and (b), it is evident that RevIN [11] reconstructs non-stationary information based on the distribution characteristics of historical sequences, making the distribution of the predicted sequence closer to that of the historical sequence rather than future sequence. However, as significant

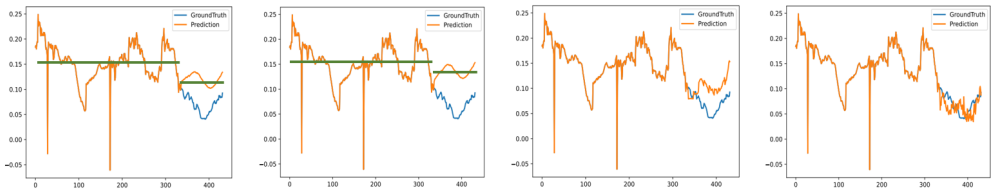


Figure 4: Comparison of reversible normalization methods, samples from DLiner [27] weather dataset forecasting. Green solid lines represent the mean of the historical and predicted sequences.

distribution differences between historical and future sequences, this approach may even degrade the predictive accuracy. Comparing subplots (c) and (d), although slice level SAN [14] significantly improves overall predictive performance, it still lags behind point level DDN in terms of fine-grained variations. Additionally, it is worth highlighting that the fine-grained capability of DDN enables the reconstructed sequence to exhibit rapid local fluctuations flexibly. More comparisons can be found in Appendix C.

4.3 Dual-domain Dynamic Normalization Analysis

To validate the effectiveness of FDN and TDN, we conducted ablation studies comparing the predictive performance when only using FDN or TDN for non-stationary processing. It is important to note that when using FDN only for non-stationary processing, the ground truth of DPM pretraining comes from the non-stationary factors of FDN on future sequences. Correspondingly, wavelet bases will be set to nonlearnable parameters when we use FDN only. Meanwhile, for a fair comparison, both FDN and TDN in DDN utilize the same prediction model and share parameters, thereby avoiding the misconception that the superior performance of DDN stems from a larger parameter space in the prediction model.

The results of the ablation studies, as shown in Table 5, indicate that both FDN and TDN achieved outstanding performance, with FDN often reaching comparable or even superior predictive results compared to TDN alone. Furthermore, when we use FDN and TDN simultaneously, the predictive performance of DDN approaches even surpasses the best performance of FDN or TDN alone. It validates the effectiveness of TDN in capturing fine-grained non-stationarity at the point level in the time domain while confirming the robustness of FDN separating frequency components with different rapid changes in the frequency domain, thus enabling more refined non-stationary processing.

Methods		DDN		DLinear TDN Only		FDN Only		DDN		iTransformer TDN Only		FDN Only	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.146**	**0.201**	0.150	0.203	0.149	0.206	**0.148**	**0.210**	0.153	0.212	0.153	0.216
	192	**0.190**	**0.247**	0.196	0.251	0.192	0.250	**0.191**	**0.252**	0.198	0.256	0.193	0.256
	336	**0.239**	**0.288**	0.247	0.294	0.242	0.293	**0.237**	**0.290**	0.249	0.302	0.245	0.305
	720	**0.311**	**0.343**	0.316	0.344	0.313	0.345	**0.301**	**0.336**	0.325	0.361	0.308	0.350
Electricity	96	**0.131**	**0.228**	0.133	0.230	0.132	0.229	0.127	**0.225**	0.130	0.227	**0.126**	0.226
	192	**0.148**	**0.246**	0.150	0.249	0.149	**0.246**	0.146	0.246	0.148	0.250	**0.144**	**0.244**
	336	**0.164**	**0.264**	0.165	**0.264**	0.166	**0.264**	**0.156**	**0.257**	0.160	0.259	**0.156**	**0.257**
	720	0.201	0.299	0.201	**0.295**	**0.200**	0.296	**0.179**	**0.282**	0.185	0.287	0.181	0.283
Traffic	96	**0.375**	**0.261**	0.378	0.263	**0.375**	0.263	**0.336**	**0.248**	0.338	0.250	0.338	0. 249
	192	**0.396**	**0.272**	0.399	0.278	0.298	0.274	**0.347**	**0.254**	0.352	0.257	0.351	0.256
	336	**0.411**	**0.279**	0.420	0.290	0.412	0.281	**0.363**	**0.263**	0.365	0.266	0.364	**0.263**
	720	**0.448**	**0.298**	0.460	0.310	0.450	0.300	0.412	0.286	0.418	0.288	**0.408**	**0.282**

Table 5: Ablation study of FDN and TDN. “TDN Only" and “FDN Only" indicate normalization using TDN only and FDN only, respectively. The best results are highlighted in **bold**.

https://doi.org/10.52202/079017-3444

5 Conclusion

In this work, we propose Dual-domain Dynamic Normalization (DDN), a novel method that dynamically captures non-stationary factors in time series forecasting, addressing sudden changes and distribution drifts in both time and frequency domains. Specifically, DDN employs sliding normalization in the time domain to eliminate and reconstruct non-stationary factors at a fine-grained level. In the frequency domain, it decomposes time series into high and low frequencies, effectively capturing rapid variations and sudden changes. As a model-agnostic auxiliary module, DDN significantly enhances the predictive performance of various forecasting models. Extensive experiments on seven real-world datasets validate the superiority of DDN, demonstrating its effectiveness in addressing distribution drift and improving the reliability of time series predictions.

6 Acknowledgments

This work is supported in part by the National Natural Science Foundation of China, under Grant (62302309, 62171248), Shenzhen Science and Technology Program (JCYJ20220818101014030, JCYJ20220818101012025), and the PCNL KEY project (PCL2023AS6-1).

References

[1] Zahra Karevan and Johan AK Suykens. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020. 1

[2] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017. 1

[3] Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, 2013. 1

[4] Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 3

[5] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 1

[6] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019. 1

[7] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017. 1

[8] Tonya Fields, George Hsieh, and Jules Chenou. Mitigating drift in time series data with noise augmentation. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 227–230. IEEE, 2019. 1

[9] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018. 1

[10] Wendi Li, Xiao Yang, Weiqing Liu, Yingce Xia, and Jiang Bian. Ddg-da: Data distribution generation for predictable concept drift adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4092–4100, 2022. 1

[11] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 7, 8, 21, 22

[12] Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7522–7529, 2023. 2, 3, 5, 8, 21, 22

[13] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022. 2, 3, 8, 21, 22

[14] Zhiding Liu, Mingyue Cheng, Zhi Li, Zhenya Huang, Qi Liu, Yanhu Xie, and Enhong Chen. Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3, 8, 9, 21, 22

[15] Pimwadee Chaovalit, Aryya Gangopadhyay, George Karabatis, and Zhiyuan Chen. Discrete wavelet transform-based time series analysis and mining. *ACM Computing Surveys (CSUR)*, 43(2):1–37, 2011. 2

[16] Shyh-Jier Huang and Cheng-Tao Hsieh. Coiflet wavelet transform applied to inspect power system disturbance-generated signals. *IEEE Transactions on Aerospace and Electronic Systems*, 38(1):204–210, 2002. 2

[17] Eric Zivot, Jiahui Wang, Eric Zivot, and Jiahui Wang. Rolling analysis of time series. *Modeling financial time series with S-Plus®*, pages 299–346, 2003. 2

[18] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021. 2, 7, 20

[19] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022. 2, 7, 21, 26

[20] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021. 2, 7, 20, 21, 26

[21] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science*, 10:96–112, 2016. 3

[22] Andreia Dionisio, Rui Menezes, and Diana A Mendes. Mutual information: a measure of dependency for nonlinear time series. *Physica A: Statistical Mechanics and its Applications*, 344(1-2):326–329, 2004. 3

[23] Stefan Frenzel and Bernd Pompe. Partial mutual information for coupling analysis of multivariate time series. *Physical review letters*, 99(20):204101, 2007. 3

[24] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023. 3, 7, 8, 20, 21

[25] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2022. 3

[26] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2022. 3

[27] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. 3, 7, 8, 9, 21, 24, 25

 https://doi.org/10.52202/079017-3444

[28] Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[29] Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *arXiv preprint arXiv:2304.05206*, 2023. 3

[30] Luo donghao and wang xue. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[31] Xue Wang, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. Card: Channel aligned robust blend transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023. 3

[32] Shohreh Deldari, Daniel V Smith, Hao Xue, and Flora D Salim. Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021*, pages 3124–3135, 2021. 3

[33] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*, 2021. 3

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[35] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067, 2021. 6

[36] El-Ghazali Talbi. A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for bi-level optimization*, pages 1–39. Springer, 2013. 6

[37] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. 6

[38] Graham Elliott, Thomas J Rothenberg, and James H Stock. Efficient tests for an autoregressive unit root, 1992. 20

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 20

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have clearly outlined the main contributions of this paper in the abstract and introduction sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

https://doi.org/10.52202/079017-3444

Justification: We have mentioned it in both the methods and experimental analysis sections.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We have clearly outlined the assumption and proof of this paper in the Introduction, Method, and Appendix sections.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have clearly outlined the experiment details in the Experiments and Appendix sections.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the code link in the abstract.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

 https://doi.org/10.52202/079017-3444

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have clearly outlined the experiment details in the Experiments and Appendix sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have conducted relevant analysis in both the experimental and appendix sections and the experimental results were obtained by averaging three times.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have clearly outlined the GPU version, programming languages and main libraries in the Appendix sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research conducted in the paper conform the ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

https://doi.org/10.52202/079017-3444

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have clearly outlined the links of these assets in the Appendix sections.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3444

A Experiments Setting

A.1 Dataset Details

Our comprehensive experiments are conducted on seven time series datasets. Consistent with the methodologies of [18, 20, 24], we partition all datasets chronologically into training, validation, and testing subsets. Specifically, for the ETT dataset, we adopted a 6:2:2 split ratio, while a 7:1:2 ratio was utilized for the other datasets. Detailed descriptions of these datasets are as follows:

(1) **ETT-small**[3] (Electricity Transformer Temperature) dataset: Comprises data from electricity transformers in two regions of China, collected between July 2016 and July 2018. It offers two different granularities: ETTh (1 hour) and ETTm (15 minutes). Each data point includes the value of oil temperature and six external power load features.

(2) **Weather**[4] dataset: Comprises 21 distinct meteorological measurements in Germany, recorded every 10 minutes throughout 2020. It features key indicators such as air temperature and visibility, providing an in-depth view of weather patterns.

(3) **Electricity**[5] dataset: Contains hourly electricity consumption data in kilowatt-hours (kWh) for 321 clients from 2012 to 2014, sourced from the UCI Machine Learning Repository.

(4) **Traffic**[6] dataset: Features hourly data on road occupancy rates from 862 detectors in the San Francisco Bay area freeways, covering 2015 to 2016.

We provide access to all datasets through https://github.com/thuml/iTransformer. Detailed statistics for these datasets, including time steps, channels, and ADF test [38] results (evaluate the stationarity of a time series; a smaller value indicates greater non-stationarity.), are presented in Table A.1.

Table 6: The statistics of datasets.

Datasets	Timesteps	Variates	Granularity	ADF
Electricity	26304	321	1 hour	-8.44
Weather	52696	21	10 min	-26.68
Traffic	17544	862	1 hour	-15.02
ETTh1	17420	7	1 hour	-5.91
ETTh2	17420	7	1 hour	-4.13
ETTm1	69680	7	15 min	-5.91
ETTm2	69680	7	15 min	-5.66

A.2 Setting Details

All experiments were conducted using PyTorch on a single NVIDIA 3090 24GB GPU. We utilize the ADAM optimizer [39] with an initial learning rate of $1e^{-4}$ for the distribution prediction model and employing Mean Squared Error (MSE) loss. The batch size, training epochs, and other baseline settings remain consistent with iTransformer [24]. The network for mean or standard deviation prediction comprises two feedforward Neural Network (FFN) layers, with dimensions of 512 for the first layer and 1024 for the second layer. We initialize the wavelet as Coiflet3, with α starting from 0. We conduct pre-training for 5 epochs and commence collaborative training from either the first or second epoch based on specific settings and datasets, aiming for improved training and model fitting.

[3] https://github.com/zhouhaoyi/ETDataset
[4] https://www.bgc-jena.mpg.de/wetter/
[5] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[6] https://pems.dot.ca.gov/

A.3 Baseline Methods

Our baseline methods are described as follows:

- Autoformer [20] is a transformer-based approach that adopts a decomposition strategy to learn complex temporal patterns in long-term prediction scenarios, decomposing time series into trend, cycle, and seasonal components, reflecting the long-term and seasonal aspects of the sequence data, respectively. The source code is available at https://github.com/thuml/Autoformer.
- FEDformer [19] is a hybrid Transformer-based model that integrates seasonal-trend decomposition and frequency enhancements. It can efficiently capture both global variations and intricate patterns. The source code is available at https://github.com/MAZiqing/FEDformer.
- DLinear [27] is a one-layer linear model challenging the efficacy of Transformer-based approaches in long-term time series forecasting, demonstrating superior performance on multiple datasets. The source code is available at https://github.com/cure-lab/LTSF-Linear.
- iTransformer [24] is a transformer-based model. The time series serve as variable tokens, utilizing self-attention mechanisms to capture correlations between multiple variables and using feedforward networks to encode the sequence representation. The source code is available at https://github.com/thuml/iTransformer.

B Reversible Normalization

Related reversible normalization methods are described in table 7, and they are described as follows:

- RevIN [11] introduces a data normalization method that addresses the limitations of simply eliminating non-stationary information, which can result in the loss of valuable data that the model needs to learn effectively. Unlike traditional methods that may lead to the model's inability to capture these critical non-stationary factors, this work proposes, for the first time, an explicit restoration of non-stationary information after the model's output. This ensures that while the model can learn without being affected by data drift, it also retains the essential non-stationary information. The source code is available at https://github.com/ts-kim/RevIN.
- NST [13] unlikes traditional time series forecasting methods that reduce non-stationarity by stabilizing the original data, this approach contradicts the importance of predicting sudden events in time series forecasting and overlooks the prevalence of non-stationary data in real-world scenarios, ultimately leading to overly stabilized modeling and prediction. To address this, this paper proposes a novel network architecture composed of sequence stabilization and inverse stabilization attention mechanisms. The source code is available at https://github.com/thuml/Nonstationary_Transformers.
- Dish-TS [12] notes that existing work on distribution shift in time series is often limited by distribution quantization and tends to overlook the potential distribution shift between the look back window and the horizon. To address this, this paper proposes using an MLP-based network to predict mean and standard deviation separately for the look back and horizon windows. The source code is available at https://github.com/weifantt/Dish-TS.
- SAN [14] indicates previous efforts on addressing non-stationarity have attempted to reduce it through normalization techniques. However, these methods typically overlook the distributional differences between input sequences and horizon sequences, assuming that all time points within the same instance share identical statistical properties. This overly idealistic approach can lead to suboptimal improvements. To address this issue, this paper introduces a novel slice-level adaptive normalization method. The source code is available at https://github.com/icantnamemyself/SAN.

Although NST [13] claims that non-stationary information can enhance feature diversity, previous methods have suffered from over-stationarization and inadequate consideration of non-stationary information utilization. However, it is noteworthy that, upon reviewing the current implementation

 https://doi.org/10.52202/079017-3444

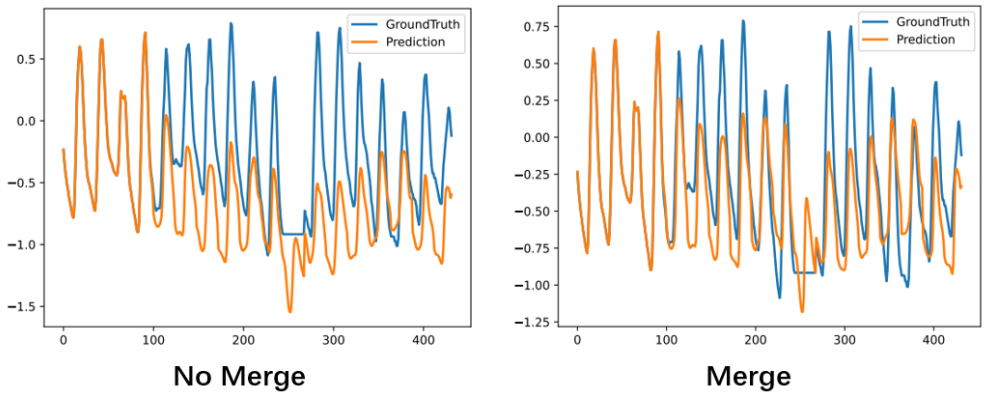


Figure 5: The comparison of NST [13], "Merge" means using non-stationary factors extraction module and merge to feature, 'No Merge" is the opposite.

Method	Granularity	Estimation
RevIN [11]	Series Level	Statistics
NST [13]	Input/Output Level	Prediction
Dish-TS [12]	Input/Output Level	Prediction
SAN [14]	Slice Level	Prediction
DDN (Ours)	Point Level	Prediction

Table 7: Comparative overview of non-stationary processing techniques in time series forecasting. "Granularity" and "Estimation" denote the normalization fineness and the prediction derivation method, respectively.

of the non-stationary transformer, the designed non-stationary information extraction module and the corresponding integration of this information into intermediate feature learning can essentially be seen as a mode that employs a distribution prediction network to estimate future non-stationary information and learns through the prediction network. Figure 5 presents a visual comparison of the effects with and without the integration of non-stationary information extraction. It is evident that the non-stationary information extraction and integration module fundamentally enables more accurate reconstruction of non-stationary information, such as mean value.

B.1 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) offers a nuanced approach to signal analysis by decomposing a time series into distinct frequency bands at multiple resolutions. This method is particularly adept at pinpointing both frequency and temporal aspects of a signal, making it invaluable for analyzing non-stationary time series. Initially, the general DWT of a time series $x(t)$ is expressed through the wavelet coefficients:

$$x_\phi(a,b) = \frac{1}{\sqrt{|a|}} \sum_t x(t) \cdot \phi\left(\frac{t-b}{a}\right) \tag{12}$$

where a and b denote the scaling and translation parameters, respectively, and ϕ is the mother wavelet function. Building upon this foundation, the DWT isolates the approximation coefficients (AC) and detail coefficients (DC), which capture distinct signal characteristics:

$$\begin{aligned} x^i_{ac} &= \mathrm{DWT}_{\phi_{\text{low}}}(x^i) = \sum_t x(t) \cdot \phi_{\text{low}}(t), \\ x^i_{dc} &= \mathrm{DWT}_{\phi_{\text{high}}}(x^i) = \sum_t x(t) \cdot \phi_{\text{high}}(t), \end{aligned} \tag{13}$$

where $\phi_{\text{low}}(t)$ represents the low-pass filter and $\phi_{\text{high}}(t)$ the high-pass filter. Together, these filters facilitate the separation of the input time series into x^i_{ac} and x^i_{dc}. The AC coefficients x^i_{ac} embody the low-frequency components that outline the overarching trends within the time series, whereas the DC coefficients x^i_{dc} encompass the high-frequency components, often associated with transient or noise elements in the signal.

The Inverse Discrete Wavelet Transform (IDWT) is then utilized to reconstruct the signal from its wavelet coefficients. The IDWT is the process that combines the AC and DC to form the original signal or an approximation of it. The reconstruction using IDWT can be written as:

$$x^i(t) = \mathrm{IDWT}(x^i_{ac}, x^i_{dc}) = \sum_a x^i_{ac}(a) \cdot \phi_{ac}(a,t) + \sum_b x^i_{dc}(b) \cdot \phi_{dc}(b,t) \tag{14}$$

where $\phi_{ac}(a,t)$ and $\phi_{dc}(b,t)$ are the reconstruction functions from the approximation and detail coefficients, respectively. The wavelet transform, with its ability to localize both frequency and time, provides a powerful framework for signal analysis, particularly for signals like time series that contain non-stationary elements.

https://doi.org/10.52202/079017-3444

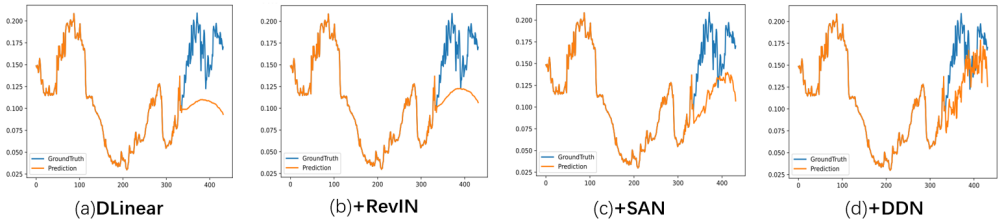


Figure 6: Comparison of reversible normalization methods, samples from DLiner [27] weather dataset forecasting.

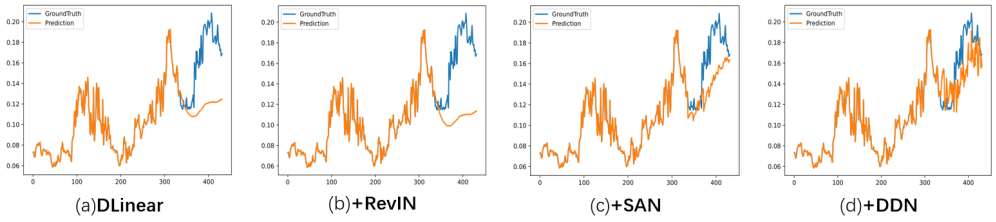


Figure 7: Comparison of reversible normalization methods, samples from DLiner [27] weather dataset forecasting.

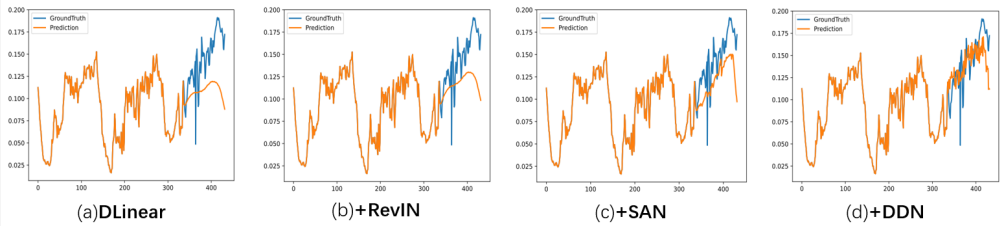


Figure 8: Comparison of reversible normalization methods, samples from DLiner [27] weather dataset forecasting.

C Visualization of Experiments

As illustrated in Figure 4, we present visual comparisons of predictions between different reversible normalization methods on the Weather dataset using DLinear [27]. The look-back window L is 336, and the prediction length T is 192. As discussed in Section 4.2, Figures 6, 7, and 8 highlight the necessity of a distribution change prediction module. Meanwhile, Figures 9 and 10 demonstrate the DDN method's superior capability in capturing details compared to the SAN method. This is particularly evident in the richer rapid changes present in its predictive output, allowing for dynamic adaptation and alignment with the rapidly changing data series in specific datasets.

To further substantiate this point, we conducted a visual comparison on the relatively smooth Traffic dataset in Figure 11, where these rapid changes also corresponded well with those in the original sequence, contrasting with the Weather dataset. This comparison underscores the enhanced detail and adaptability of the DDN method across different datasets.

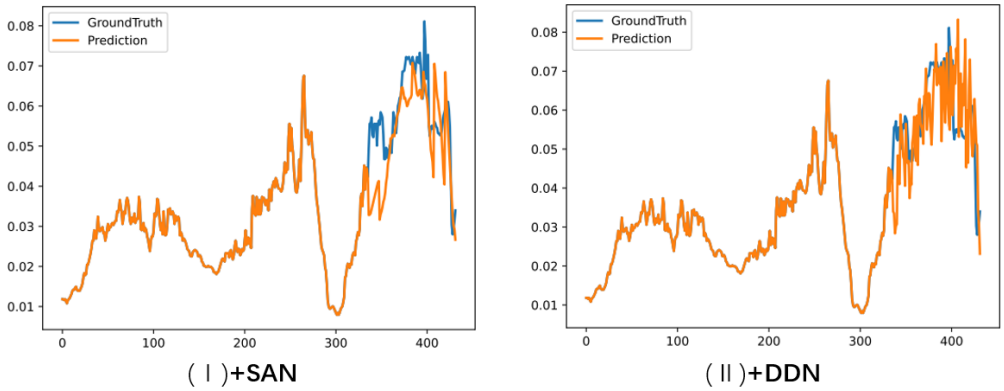


Figure 9: Comparison with slice level reversible normalization, samples from DLiner [27] weather dataset forecasting.

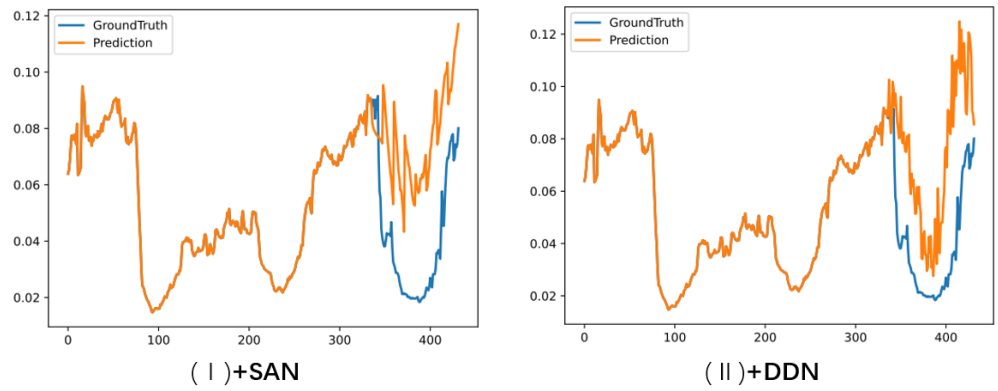


Figure 10: Comparison with slice level reversible normalization, samples from DLiner [27] weather dataset forecasting.

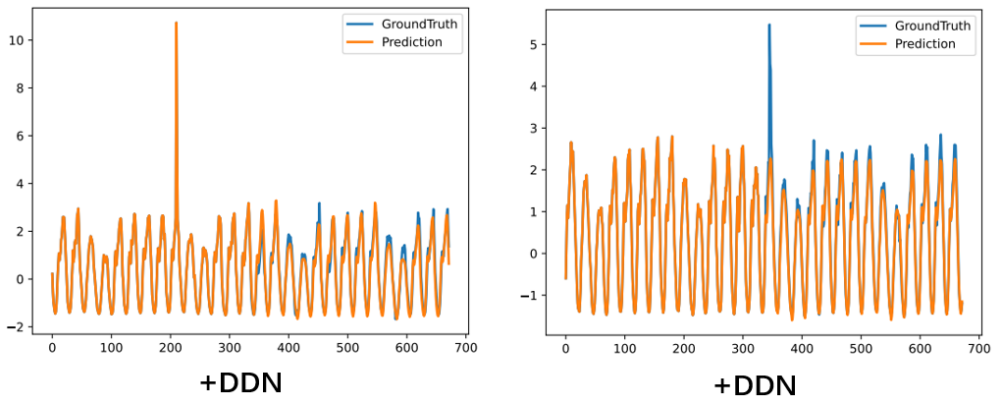


Figure 11: Our DDN reversible normalization method, samples from DLiner [27] Traffic dataset forecasting.

D Quantitative Evaluation Supplement

D.1 Multivariable Forecasting Results of ETT

As illustrated in table 8, we present the comprehensive multivariate forecasting results on the ETT dataset in Table 5, encompassing the hourly datasets ETTh1 and ETTh2, as well as the 15-

https://doi.org/10.52202/079017-3444

minute datasets ETTm1 and ETTm2. The data clearly indicate that DDN demonstrates substantial enhancements across these datasets when applied to various backbone models.

Methods		Autoformer		+DDN		FEDformer		+DDN		DLinear		+DDN		iTransformer		+DDN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.458	0.448	**0.427**	**0.424**	**0.371**	0.411	0.385	**0.408**	0.377	0.399	**0.372**	**0.396**	0.392	0.422	**0.377**	**0.405**
	192	0.481	0.474	**0.472**	**0.452**	**0.420**	0.443	**0.415**	**0.452**	0.417	0.426	**0.406**	**0.416**	0.428	0.448	**0.414**	**0.430**
	336	0.508	0.485	**0.498**	**0.466**	**0.446**	0.459	0.458	**0.452**	0.464	0.461	**0.432**	**0.434**	0.467	0.475	**0.453**	**0.456**
	720	0.525	0.516	**0.502**	**0.483**	**0.482**	0.495	0.490	**0.479**	0.493	0.505	**0.462**	**0.474**	0.568	0.547	**0.553**	**0.530**
ETTh2	96	0.384	0.420	**0.350**	**0.385**	0.341	0.382	**0.312**	**0.357**	0.292	0.356	**0.279**	**0.340**	0.315	0.366	**0.279**	**0.342**
	192	0.457	0.454	**0.398**	**0.413**	0.426	0.436	**0.384**	**0.403**	0.383	0.418	**0.341**	**0.379**	0.394	0.416	**0.341**	**0.384**
	336	0.468	0.473	**0.428**	**0.444**	0.481	0.479	**0.421**	**0.437**	0.473	0.477	**0.364**	**0.402**	0.430	0.445	**0.369**	**0.410**
	720	0.473	0.485	**0.437**	**0.460**	0.458	0.477	**0.424**	**0.450**	0.708	0.599	**0.396**	**0.434**	0.443	0.469	**0.406**	**0.447**
ETTm1	96	0.493	0.470	**0.354**	**0.390**	0.362	0.408	**0.313**	**0.364**	0.301	0.344	**0.288**	**0.342**	0.322	0.371	**0.301**	**0.355**
	192	0.546	0.498	**0.397**	**0.408**	0.395	0.427	**0.361**	**0.396**	0.335	0.366	**0.324**	**0.364**	0.353	0.392	**0.339**	**0.378**
	336	0.658	0.543	**0.429**	**0.433**	0.441	0.454	**0.417**	**0.430**	0.370	0.387	**0.356**	**0.385**	0.385	0.410	**0.370**	**0.396**
	720	0.626	0.532	**0.488**	**0.464**	0.488	0.481	**0.470**	**0.472**	0.425	0.421	**0.415**	**0.419**	0.441	0.443	**0.426**	**0.426**
ETTm2	96	0.261	0.329	**0.177**	**0.262**	0.191	0.283	**0.171**	**0.255**	0.169	0.263	**0.167**	**0.257**	0.187	0.279	**0.162**	**0.253**
	192	0.282	0.339	**0.240**	**0.304**	0.261	0.326	**0.240**	**0.298**	0.232	0.310	**0.225**	**0.298**	0.246	0.318	**0.217**	**0.291**
	336	0.350	0.378	**0.306**	**0.344**	0.327	0.365	**0.306**	**0.342**	0.303	0.361	**0.286**	**0.339**	0.300	0.354	**0.269**	**0.327**
	720	0.438	0.428	**0.409**	**0.421**	0.428	0.423	**0.413**	**0.410**	0.403	0.424	**0.371**	**0.391**	0.378	0.403	**0.350**	**0.380**

Table 8: Forecasting results comparison under different prediction lengths $T \in \{96, 192, 336, 720\}$ on ETT dataset. The input sequence length $L = 96$ for Autoformer and FEDformer, $L = 336$ for DLinear, and $L = 720$ for iTransformer. The **bold** values indicate best performance.

D.2 Comparison between DDN and Reversible Normalization Methods

This section comprehensively compares DDN with existing reversible normalization methods. As shown in Table 9, our method consistently achieves state-of-the-art performance across almost all datasets, with a slight under-performance on the ETTm1 dataset compared to SAN. This demonstrates the versatility and effectiveness of our approach.

D.3 Collaborative Training Ablation

Although recent works have shown that the two-stage training strategy effectively enhances the training of distribution prediction models, this training overly relies on the distribution ground truth obtained through distribution assumptions, which often contain inaccuracies. To address this, we introduce a collaborative training strategy that adjusts the distribution prediction model using the MSE loss between the actual sequence and the prediction during training. As illustrated in table 10, collaborative training generally yields superior training outcomes, and its lower bound in performance is comparable to that of the two-stage training strategy. This indicates that collaborative training can better mitigate the errors introduced by distribution assumptions and improve the accuracy of distribution prediction models.

D.4 Univariate Forecasting Results

Following the same settings as our main experiment, we present the univariate forecasting results of Autoformer [20] and FEDformer [19] across three datasets, including Weather, Electricity, and Traffic, in Table 11. Similar to the multivariate forecasting results, DDN consistently enhances the performance of mainstream forecasting models in nearly all cases. It demonstrates that DDN is applicable in both multivariate time series forecasting and univariate forecasting.

Methods		Autoformer										FEDformer									
		+DDN		+RevIN		+NST		+Dish-TS		+SAN		+DDN		+RevIN		+NST		+Dish-TS		+SAN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.427**	**0.424**	0.491	0.463	0.550	0.503	0.456	0.454	0.522	0.474	**0.385**	**0.408**	0.392	0.413	0.394	0.414	0.390	0.424	0.383	0.409
	192	**0.472**	**0.452**	0.513	0.478	0.530	0.492	0.495	0.480	0.498	0.472	**0.415**	**0.452**	0.443	0.444	0.441	0.442	0.441	0.458	0.431	0.438
	336	**0.498**	**0.466**	0.528	0.485	0.524	0.484	0.539	0.496	0.571	0.509	**0.458**	**0.452**	0.495	0.467	0.485	0.466	0.495	0.486	0.471	0.456
	720	**0.502**	**0.483**	0.543	0.510	0.510	0.491	0.563	0.522	0.555	0.514	**0.490**	**0.479**	0.520	0.498	0.505	0.496	0.519	0.509	0.504	0.488
ETTh2	96	0.350	0.385	0.411	0.410	0.394	0.398	1.100	0.670	**0.316**	**0.366**	**0.312**	**0.357**	0.380	0.402	0.381	0.403	0.806	0.589	0.300	0.355
	192	**0.398**	**0.413**	0.478	0.450	0.473	0.450	0.976	0.672	0.413	0.426	**0.384**	**0.403**	0.457	0.433	0.478	0.453	0.936	0.659	0.392	0.413
	336	**0.428**	**0.444**	0.545	0.493	0.528	0.490	1.521	0.783	0.446	0.457	**0.421**	**0.437**	0.515	0.479	0.561	0.499	1.039	0.702	0.459	0.462
	720	**0.437**	**0.460**	0.523	0.490	0.524	0.498	1.105	0.745	0.471	0.474	**0.424**	**0.450**	0.507	0.487	0.502	0.481	1.237	0.759	0.462	0.472
ETTm1	96	0.354	0.390	0.458	0.446	0.468	0.448	0.477	0.460	**0.343**	**0.378**	0.313	0.364	0.340	0.385	0.336	0.382	0.348	0.397	**0.311**	**0.355**
	192	0.397	0.408	0.560	0.491	0.526	0.468	0.545	0.488	**0.390**	**0.400**	0.361	0.396	0.390	0.411	0.386	0.409	0.406	0.428	**0.351**	**0.383**
	336	0.429	0.433	0.607	0.508	0.786	0.559	0.650	0.533	**0.415**	**0.418**	0.417	0.430	0.432	0.436	0.438	0.441	0.438	0.450	**0.390**	**0.407**
	720	0.488	0.464	0.623	0.526	0.564	0.501	0.595	0.518	**0.476**	**0.453**	0.470	0.472	0.497	0.466	0.483	0.460	0.497	0.481	**0.456**	**0.444**
ETTm2	96	**0.177**	**0.262**	0.233	0.307	0.253	0.323	0.976	0.57	0.236	0.317	**0.171**	**0.255**	0.192	0.272	0.191	0.272	0.394	0.395	0.175	0.266
	192	**0.240**	**0.304**	0.288	0.337	0.289	0.335	0.532	0.485	0.260	0.329	**0.240**	**0.29**8	0.270	0.320	0.270	0.321	0.552	0.472	0.246	0.315
	336	**0.306**	**0.344**	0.345	0.370	0.339	0.365	0.795	0.592	0.330	0.376	**0.306**	**0.342**	0.348	0.367	0.353	0.371	0.808	0.601	0.315	0.362
	720	**0.409**	**0.421**	0.434	0.419	0.426	0.432	1.271	0.768	0.417	0.428	**0.413**	**0.410**	0.430	0.415	0.445	0.422	1.282	0.771	0.412	0.422
Weather	96	**0.190**	**0.243**	0.212	0.257	0.211	0.254	0.268	0.338	0.194	0.256	**0.174**	**0.237**	0.187	0.234	0.187	0.234	0.244	0.317	0.179	0.239
	192	**0.231**	**0.282**	0.264	0.300	0.265	0.301	0.376	0.421	0.258	0.316	**0.233**	**0.294**	0.235	0.272	0.235	0.272	0.320	0.980	0.234	0.296
	336	**0.289**	**0.327**	0.309	0.329	0.303	0.324	0.475	0.486	0.329	0.367	0.307	0.349	**0.287**	**0.307**	0.289	0.308	0.424	0.452	0.304	0.384
	720	0.369	0.375	0.377	0.367	**0.366**	**0.357**	0.612	0.560	0.440	0.438	0.399	0.405	0.361	0.353	**0.359**	**0.352**	0.604	0.553	0.400	0.404
Electricity	96	**0.150**	**0.254**	0.179	0.286	0.179	0.285	0.197	0.290	0.172	0.281	**0.146**	**0.251**	0.172	0.278	0.172	0.279	0.175	0.284	0.164	0.272
	192	**0.173**	**0.275**	0.216	0.316	0.209	0.309	0.215	0.318	0.195	0.300	**0.168**	**0.268**	0.185	0.289	0.187	0.291	0.188	0.296	0.179	0.286
	336	**0.185**	**0.288**	0.233	0.331	0.246	0.335	0.244	0.343	0.211	0.316	**0.174**	**0.280**	0.200	0.304	0.202	0.307	0.209	0.319	0.191	0.299
	720	**0.201**	**0.304**	0.246	0.341	0.252	0.354	0.286	0.370	0.236	0.335	**0.216**	**0.312**	0.243	0.337	0.230	0.326	0.239	0.343	0.230	0.334
Traffic	96	**0.453**	**0.296**	0.643	0.354	0.654	0.354	0.652	0.363	0.569	0.350	**0.422**	**0.288**	0.613	0.347	0.612	0.348	0.613	0.350	0.536	0.330
	192	**0.462**	**0.304**	0.659	0.373	0.643	0.367	0.669	0.374	0.594	0.364	**0.462**	**0.300**	0.637	0.356	0.641	0.357	0.644	0.362	0.565	0.345
	336	**0.486**	**0.315**	0.662	0.371	0.665	0.363	0.683	0.376	0.591	0.363	**0.474**	**0.306**	0.652	0.363	0.654	0.363	0.659	0.370	0.580	0.354
	720	**0.529**	**0.344**	0.700	0.384	0.667	0.373	0.703	0.392	0.623	0.380	**0.512**	**0.329**	0.686	0.382	0.688	0.380	0.693	0.388	0.607	0.367

Table 9: Comparison of forecasting errors between different reversible normalization methods. The **bold** values indicate best performance.

Methods		DLinear				iTransformer			
		Co-train		Wo Co-train		Co-train		Wo Co-train	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.146**	**0.201**	0.151	0.209	**0.148**	**0.210**	0.149	0.212
	192	**0.190**	**0.247**	0.193	0.251	**0.191**	**0.252**	**0.191**	**0.252**
	336	**0.239**	**0.288**	0.241	0.291	**0.237**	0.290	**0.236**	**0.289**
	720	**0.311**	**0.343**	0.308	0.343	**0.301**	**0.336**	**0.301**	0.337
Electricity	96	**0.131**	**0.228**	0.135	0.231	**0.127**	**0.225**	0.130	0.229
	192	**0.148**	**0.246**	0.150	0.247	**0.146**	**0.246**	0.147	**0.246**
	336	**0.164**	**0.264**	0.167	0.266	**0.156**	**0.257**	0.157	0.258
	720	**0.201**	**0.299**	0.210	0.309	**0.179**	**0.282**	0.184	0.286
Traffic	96	**0.375**	**0.261**	0.395	0.279	**0.336**	**0.248**	0.344	0.254
	192	**0.396**	**0.272**	0.412	0.286	**0.347**	**0.254**	0.354	0.261
	336	**0.411**	**0.279**	0.425	0.291	**0.363**	**0.263**	0.370	0.268
	720	**0.448**	**0.298**	0.452	0.308	**0.412**	**0.286**	0.410	0.288

Table 10: Comparison between the collaborative training strategy and the conventional two-stage training strategy. "Co-train" denotes the use of the collaborative training strategy, while "Wo Co-train" signifies the approach where the distribution prediction model's parameters are frozen after pre-training, and only the time series forecasting model is trained. The **bold** values indicate best performance.

https://doi.org/10.52202/079017-3444

Methods		Autoformer		+DDN		FEDformer		+DDN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.004	0.047	**0.003**	**0.044**	**0.002**	**0.037**	0.003	0.040
	192	0.003	0.045	**0.003**	**0.039**	0.005	0.058	**0.004**	**0.052**
	336	0.008	0.068	**0.003**	**0.041**	0.003	0.045	**0.002**	**0.031**
	720	0.058	0.176	**0.003**	**0.044**	0.011	0.080	**0.004**	**0.052**
Electricity	96	0.442	0.490	**0.346**	**0.424**	0.302	0.413	**0.213**	**0.330**
	192	0.555	0.550	**0.342**	**0.423**	0.377	0.459	**0.252**	**0.357**
	336	0.617	0.620	**0.380**	**0.447**	0.673	0.636	**0.313**	**0.401**
	720	0.645	0.624	**0.463**	**0.505**	0.575	0.575	**0.410**	**0.454**
Traffic	96	0.265	0.375	**0.153**	**0.239**	0.179	0.282	**0.137**	**0.217**
	192	0.266	0.372	**0.156**	**0.242**	0.211	0.316	**0.142**	**0.220**
	336	0.284	0.371	**0.164**	**0.256**	0.369	0.458	**0.140**	**0.222**
	720	0.260	0.369	**0.187**	**0.279**	0.300	0.407	**0.157**	**0.242**

Table 11: Univariate forecasting results. The **bold** values indicate best performance.