Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting

Yong Liu[*], **Haixu Wu**[*], **Jianmin Wang, Mingsheng Long**[✉]
School of Software, BNRist, Tsinghua University, China
{liuyong21,whx20}@mails.tsinghua.edu.cn, {jimwang,mingsheng}@tsinghua.edu.cn

Abstract

Transformers have shown great power in time series forecasting due to their global-range modeling ability. However, their performance can degenerate terribly on non-stationary real-world data in which the joint distribution changes over time. Previous studies primarily adopt stationarization to attenuate the non-stationarity of original series for better predictability. But the stationarized series deprived of inherent non-stationarity can be less instructive for real-world bursty events forecasting. This problem, termed *over-stationarization* in this paper, leads Transformers to generate indistinguishable temporal attentions for different series and impedes the predictive capability of deep models. To tackle the dilemma between series predictability and model capability, we propose *Non-stationary Transformers* as a generic framework with two interdependent modules: Series Stationarization and De-stationary Attention. Concretely, Series Stationarization unifies the statistics of each input and converts the output with restored statistics for better predictability. To address the over-stationarization problem, De-stationary Attention is devised to recover the intrinsic non-stationary information into temporal dependencies by approximating distinguishable attentions learned from raw series. Our Non-stationary Transformers framework consistently boosts mainstream Transformers by a large margin, which reduces MSE by 49.43% on Transformer, 47.34% on Informer, and 46.89% on Reformer, making them the state-of-the-art in time series forecasting. Code is available at this repository: https://github.com/thuml/Nonstationary_Transformers.

1 Introduction

Time series forecasting has become increasingly ubiquitous in real-world applications, such as weather forecasting, energy consumption planning, and financial risk assessment. Recently, Transformers [34] have achieved progressive breakthrough on extensive areas [12, 13, 10, 24]. Especially in time series forecasting, credited to their stacked structure and the capability of attention mechanisms, Transformers can naturally capture the temporal dependencies from deep multi-level features [39, 19, 22, 37], thereby fitting the series forecasting task perfectly.

Despite the remarkable architectural design, it is still challenging for Transformers to predict real-world time series because of the non-stationarity of data. Non-stationary time series is characterized by the continuous change of statistical properties and joint distribution over time, which makes the time series less predictable [6, 16]. Besides, it is a fundamental problem to make deep models generalize well on a varying distribution [28, 21, 5]. In previous work, it is generally acknowledged to pre-process the time series by stationarization [26, 29, 17], which can attenuate the non-stationarity of raw time series for better predictability and provide more stable data distribution for deep models.

[*]Equal Contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

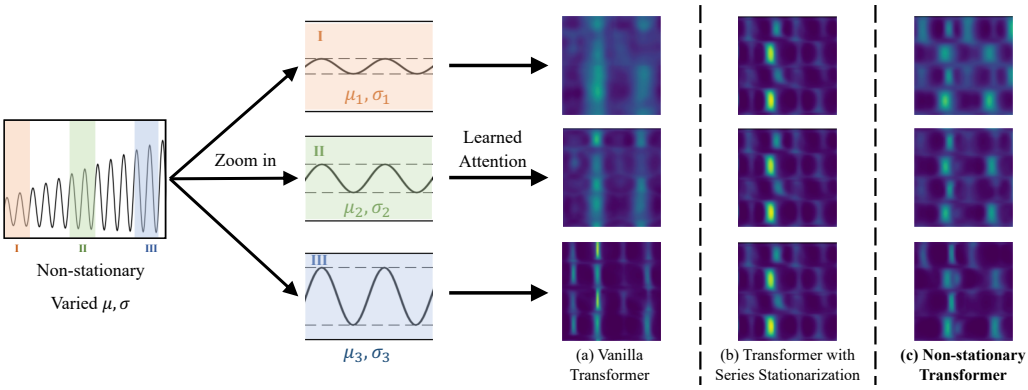

Figure 1: Visualization of learned temporal attentions for different series with varied mean μ and standard deviation σ. (a) is from the vanilla Transformer [34] trained on raw series. (b) is from the Transformer trained on stationarized series, which presents similar attentions. (c) is from Non-stationary Transformers, which involves De-stationary Attention to avoid over-stationarization.

However, non-stationarity is the inherent property of real-world time series and also good guidance for discovering temporal dependencies for forecasting. Experimentally, we observe that training on the stationarized series will undermine the distinction of attentions learned by Transformers. While vanilla Transformers [34] can capture distinct temporal dependencies from different series in Figure 1(a), Transformers trained on the stationarized series tend to generate indistinguishable attentions in Figure 1(b). This problem, named by the *over-stationarization*, will bring unexpected side-effect that makes Transformers fail to capture eventful temporal dependencies, limit the model's predictive ability, and even induce the model to generate outputs with huge non-stationarity deviation from the ground truth. Thus, *how to attenuate time series non-stationarity towards better predictability and mitigate the over-stationarization problem for model capability simultaneously* is the key problem to further improve the performance of forecasting.

In this paper, we explore the effect of stationarization in time series forecasting and propose *Non-stationary Transformers* as a general framework, which empowers Transformer [34] and its efficient variants [19, 39, 37] with great predictive ability for real-world time series. The proposed framework involves two interdependent modules: Series Stationarization to increase the predictability of non-stationary series and De-stationary Attention to alleviate over-stationarization. Technically, Series Stationarization adopts a simple but effective normalization strategy to unify the key statistics of each series without extra parameters. And De-stationary Attention approximates the attention of unstationarized data and compensates the intrinsic non-stationarity of raw series. Benefiting from the above designs, Non-stationary Transformers can take advantage of the great predictability of stationarized series and crucial temporal dependencies discovered from original non-stationary data. Our method achieves state-of-the-art performance on six real-world benchmarks and can generalize to various Transformers for further improvement. The contributions lie in three folds:

- We refine that the predictive capability of non-stationary series is essential in real-world forecasting. By detailed analysis, we find out that current stationarization approaches will lead to the over-stationarization problem, limiting the predictive capability of Transformers.
- We propose Non-stationary Transformers as a generic framework, including Series Stationarization to make the series more predictable and De-stationary Attention to avoid the over-stationarization problem by re-incorporating the non-stationarity of original series.
- Non-stationary Transformers consistently boosts four mainstream Transformers by a large margin and achieves state-of-the-art performance on six real-world benchmarks.

2 Related Work

2.1 Deep Models for Time Series Forecasting

In recent years, deep models with elaboratively designed architectures have achieved great progress in time series forecasting. RNN-based models [35, 38, 25, 31, 32] are proposed for application in an

autoregressive manner for sequence modeling, but the recurrent structure can suffer from modeling long-term dependency. Soon afterward, Transformer [34] emerges and shows great power in sequence modeling. To overcome the quadratic computation growth on sequence length, subsequent works aim to reduce Self-Attention's complexity. Especially in time series forecasting, Informer [39] extends Self-Attention with KL-divergence criterion to select dominant queries. Reformer [19] introduces local-sensitive hashing (LSH) to approximate attention by allocated similar queries. Not only improved by reduced complexity, the following models further develop delicate building blocks for time series forecasting. Autoformer [37] fuses the decomposition blocks into a canonical structure and develops Auto-Correlation to discover series-wise connections. Pyraformer [23] designs pyramid attention module (PAM) to capture temporal dependencies with different hierarchies. Other deep but Transformer-free models also achieve remarkable performance. N-BEATS [27] proposes the explicit decomposition of trend and seasonal terms with strong interpretability. N-HiTS [9] introduces hierarchical layout and multi-rate sampling for tackling time series with respective frequency bands. In this paper, different from previous works focusing on architectural design, we analyze the series forecasting task from the basic view of stationarity, which is an essential property of time series [6, 16]. It is also notable that as a general framework, our proposed Non-stationary Transformers can be easily applied to various Transformer-based models.

2.2 Stationarization for Time Series Forecasting

While stationarity is important to the predictability of time series [6, 16], real-world series always present non-stationarity. To tackle this problem, the classical statistical method ARIMA [7, 8] stationarizes the time series through differencing. As for deep models, since the distribution-varying problem accompanied by non-stationarity makes deep forecasting even more intractable, stationarization methods are widely explored and always adopted as the pre-processing for deep model inputs. Adaptive Norm [26] applies z-score normalization for each series fragment by global statistics of a sampled set. DAIN [29] employs a nonlinear neural network to adaptively stationarize time series with observed training distribution. RevIN [17] introduces a two-stage instance normalization [33] that transforms model input and output respectively to reduce the discrepancy of each series. In contrast, we find out that directly stationarizing time series will damage the model's capability of modeling specific temporal dependency. Therefore, unlike previous methods, in addition to the stationarization, Non-stationary Transformers further develops De-stationary Attention to bring the intrinsic non-stationarity of the raw series back to attention.

3 Non-stationary Transformers

As aforementioned, stationarity is an important element of time series predictability. Previous "direct stationarization" designs can attenuate non-stationarity of series for better predictability, but they obviously neglect inherent properties of real-world series, which will result in the over-stationarization problem as stated in Figure 1. To deal with the dilemma, we go beyond previous works and propose *Non-stationary Transformers* as a generic framework. Our model involves two complementary parts: Series Stationarization to attenuate time series non-stationarity and De-stationary Attention to re-incorporate non-stationary information of raw series. Empowered by these designs, Non-stationary Transformers can improve data predictability and maintain model capability simultaneously.

3.1 Series Stationarization

Non-stationary time series make the forecasting task intractable for deep models because it is hard for them to generalize well on series with changed statistics during inference, typically varied mean and standard deviation. The pilot work, RevIN [17] applies instance normalization with learnable affine parameters to each input and restores the statistics to the corresponding output, which makes each series follow a similar distribution. Experimentally, we find that this design also works well without learnable parameters. Thus, we propose a more straightforward but effective design to wrap Transformers as the base model without extra parameters, naming by Series Stationarization. As is shown in Figure 2, it contains two corresponding operations: Normalization module at first to deal with the non-stationary series caused by varied mean and standard deviation, and De-normalization module at the end to transform the model outputs back with original statistics. Here are the details.

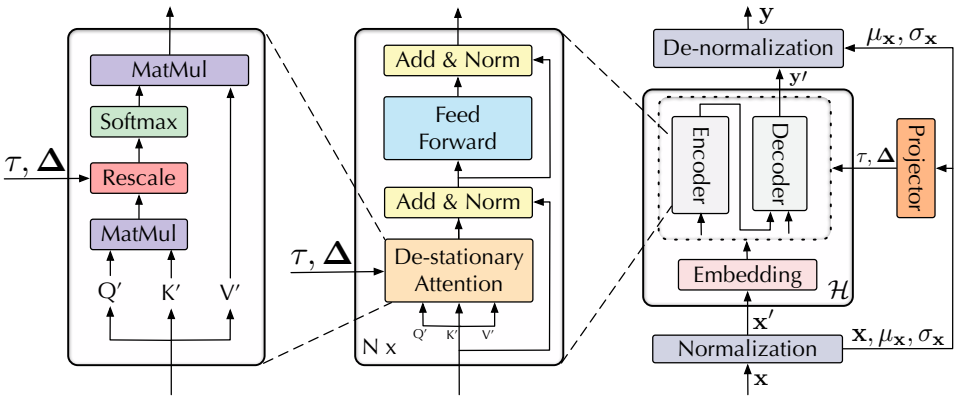

Figure 2: Non-stationary Transformers. Series Stationarization is adopted as a wrapper on the base model to normalize each incoming series and de-normalize the output. De-stationary Attention replaces the original Attention mechanism to approximate attention learned from unstationarized series, which rescales current temporal dependency weights with learned de-stationary factors $\tau, \mathbf{\Delta}$.

Normalization module To attenuate the non-stationarity of each input series, we conduct normalization on the temporal dimension by a sliding window over time. For each input series $\mathbf{x} = [x_1, x_2, ..., x_S]^\top \in \mathbb{R}^{S \times C}$, we transform it by translation and scaling operations and obtain $\mathbf{x}' = [x'_1, x'_2, ..., x'_S]^\top \in \mathbb{R}^{S \times C}$, where S and C denote the sequence length and variable number respectively. The Normalization module can be formulated as follows:

$$\mu_{\mathbf{x}} = \frac{1}{S}\sum_{i=1}^{S} x_i,\ \sigma_{\mathbf{x}}^2 = \frac{1}{S}\sum_{i=1}^{S}(x_i - \mu_{\mathbf{x}})^2,\ x'_i = \frac{1}{\sigma_{\mathbf{x}}} \odot (x_i - \mu_{\mathbf{x}}), \tag{1}$$

where $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}} \in \mathbb{R}^{C \times 1}$, $\frac{1}{\sigma_{\mathbf{x}}}$ means the element-wise division and $\odot$ is the element-wise product. Note that Normalization module decreases the distributional discrepancy among each input time series, making the distribution of the model input more stable.

De-normalization module As shown in Figure 2, after the base model $\mathcal{H}$ predicting the future value with length-O, we adopt De-normalization to transform the model output $\mathbf{y}' = [y'_1, y'_2, ..., y'_O]^\top \in \mathbb{R}^{O \times C}$ with $\sigma_{\mathbf{x}}$ and $\mu_{\mathbf{x}}$ and obtain $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_O]^\top$ as the eventual forecasting results. The De-normalization module can be formulated as follows:

$$\mathbf{y}' = \mathcal{H}(\mathbf{x}'),\ \hat{y}_i = \sigma_{\mathbf{x}} \odot (y'_i + \mu_{\mathbf{x}}). \tag{2}$$

By means of the two-stage transformation, the base models will receive stationarized inputs, which follow a stable distribution and are easier to generalize. This design also makes the model equivariant to translational and scaling perturbance of time series, thereby benefiting real-world series forecasting.

3.2 De-stationary Attention

While the statistics of each time series are explicitly restored to the corresponding prediction, the non-stationarity of the original series cannot be fully recovered only by De-normalization. For instance, Series Stationarization can generate the same stationarized input $\mathbf{x}'$ from distinct time series $\mathbf{x}_1, \mathbf{x}_2$ (i.e. $\mathbf{x}_2 = \alpha\mathbf{x}_1 + \beta$), and the base model will get identical attention that fails to capture crucial temporal dependencies entangled with non-stationarity (Figure 1). In other words, the undermined effects caused by over-stationarization happen inside the deep model, especially in the calculation of attention. Furthermore, non-stationary time series are fragmented and normalized into several series chunks with the same mean and variance, which follow more similar distributions than the raw data before stationarization. Thus, the model is more likely to generate over-stationary and uneventful outputs, which is irreconcilable with the natural non-stationarity of the original series.

To tackle the over-stationarization problem caused by Series Stationarization, we propose a novel De-stationary Attention mechanism, which can approximate the attention that is obtained without stationarization and discover the particular temporal dependencies from original non-stationary data.

Analysis of the plain model As mentioned above, the over-stationarization problem is caused by the vanishment of inherent non-stationarity information, which will make the base model fail to capture eventful temporal dependencies for forecasting. Therefore, we try to approximate the attention learned from the original non-stationary series. We start from the formula of Self-Attention [34]:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{S \times d_k}$ are length-S queries, keys and values of d_k-dimension respectively, and $\mathrm{Softmax}(\cdot)$ is conducted row by row. To simplify the analysis, we assume the embedding and feed-forward layers f to hold the linear properties[2] and f is conducted separately on each time point, that is, each query token in $\mathbf{Q} = [q_1, q_2, ..., q_S]^\top$ can be calculated as $q_i = f(x_i)$ with respect to the input series $\mathbf{x} = [x_1, x_2, \cdots, x_S]^\top$. Since it is a convention to conduct normalization on each time series variable to avoid certain variable that dominates the scale, we can further assume each variable of series $\mathbf{x}$ shares the same variance, and thus original $\sigma_\mathbf{x} \in \mathbb{R}^{C \times 1}$ is reduced to a scalar. After Normalization module, the model receives the stationarized input $\mathbf{x}' = (\mathbf{x} - \mathbf{1}\mu_\mathbf{x}^\top)/\sigma_\mathbf{x}$, where $\mathbf{1} \in \mathbb{R}^{S \times 1}$ is an all-ones vector. Based on the linear property assumption, it can be proved that the Attention layer will receive $\mathbf{Q}' = [f(x_1'), ..., f(x_S')]^\top = (\mathbf{Q} - \mathbf{1}\mu_\mathbf{Q}^\top)/\sigma_\mathbf{x}$, where $\mu_\mathbf{Q} \in \mathbb{R}^{d_k \times 1}$ is the mean of $\mathbf{Q}$ along the temporal dimension (See Appendix A for a detailed proof). And so is the corresponding transformed $\mathbf{K}', \mathbf{V}'$. Without Series Stationarization, the input of $\mathrm{Softmax}(\cdot)$ in Self-Attention should be $\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}$, while now the attention is calculated based on $\mathbf{Q}', \mathbf{K}'$:

$$\begin{aligned} \mathbf{Q}'\mathbf{K}'^\top &= \frac{1}{\sigma_\mathbf{x}^2}\left(\mathbf{Q}\mathbf{K}^\top - \mathbf{1}(\mu_\mathbf{Q}^\top\mathbf{K}^\top) - (\mathbf{Q}\mu_\mathbf{K})\mathbf{1}^\top + \mathbf{1}(\mu_\mathbf{Q}^\top\mu_\mathbf{K})\mathbf{1}^\top\right), \\ \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) &= \mathrm{Softmax}\left(\frac{\sigma_\mathbf{x}^2\,\mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}(\mu_\mathbf{Q}^\top\mathbf{K}^\top) + (\mathbf{Q}\mu_\mathbf{K})\mathbf{1}^\top - \mathbf{1}(\mu_\mathbf{Q}^\top\mu_\mathbf{K})\mathbf{1}^\top}{\sqrt{d_k}}\right). \end{aligned} \tag{4}$$

We find that $\mathbf{Q}\mu_\mathbf{K} \in \mathbb{R}^{S \times 1}$ and $\mu_\mathbf{Q}^\top\mu_\mathbf{K} \in \mathbb{R}$, and they are repeatedly operated on each column and element of $\sigma_\mathbf{x}^2\mathbf{Q}'\mathbf{K}'^\top \in \mathbb{R}^{S \times S}$ respectively. Since $\mathrm{Softmax}(\cdot)$ is invariant to the same translation on the row dimension of input, we have the following equation:

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) = \mathrm{Softmax}\left(\frac{\sigma_\mathbf{x}^2\,\mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}\mu_\mathbf{Q}^\top\mathbf{K}^\top}{\sqrt{d_k}}\right). \tag{5}$$

Equation 5 deduces a direct expression of the attention $\mathrm{Softmax}\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}\right)$ learned from raw series $\mathbf{x}$. Except for the current $\mathbf{Q}', \mathbf{K}'$ from stationarized series $\mathbf{x}'$, this expression also requires the non-stationary information $\sigma_\mathbf{x}, \mu_\mathbf{Q}, \mathbf{K}$ that are eliminated by Series Stationarization.

De-stationary Attention To recover the original attention on non-stationary series, we attempt to bring the vanished non-stationary information back to its calculation. Based on Equation 5, the key is to approximate the positive scaling scalar $\tau = \sigma_\mathbf{x}^2 \in \mathbb{R}^+$ and shifting vector $\mathbf{\Delta} = \mathbf{K}\mu_\mathbf{Q} \in \mathbb{R}^{S \times 1}$, which are defined as *de-stationary factors*. Since the strict linear property hardly holds for a deep model, other than estimating and utilizing real factors with great effort, we try to learn de-stationary factors directly from the statistics of unstationarized $\mathbf{x}, \mathbf{Q}$ and $\mathbf{K}$ by a simple but effective multi-layer perceptron layer. As we can only discover limited non-stationary information from current $\mathbf{Q}', \mathbf{K}'$, the unique and reasonable source to compensate non-stationarity is the original $\mathbf{x}$ without being normalized. Thus, as a direct deep learning implementation of Equation 5, we apply a multi-layer perceptron as the projector to learn de-stationary factors $\tau, \mathbf{\Delta}$ from the statistics $\mu_\mathbf{x}, \sigma_\mathbf{x}$ of unstationarized $\mathbf{x}$ individually. And the De-stationary Attention is calculated as follows:

$$\begin{aligned} \log\tau = \mathrm{MLP}(\sigma_\mathbf{x}, \mathbf{x}), \mathbf{\Delta} = \mathrm{MLP}(\mu_\mathbf{x}, \mathbf{x}), \\ \mathrm{Attn}(\mathbf{Q}', \mathbf{K}', \mathbf{V}', \tau, \mathbf{\Delta}) = \mathrm{Softmax}\left(\frac{\tau\,\mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}\mathbf{\Delta}^\top}{\sqrt{d_k}}\right)\mathbf{V}', \end{aligned} \tag{6}$$

where the de-stationary factors τ and $\mathbf{\Delta}$ are shared by De-stationary Attention of all layers (Figure 2). De-stationary Attention mechanism learns the temporal dependencies from both stationarized series

[2]Function f has the linear property if it satisfies that $f(ax + by) = af(x) + bf(y)$, where a, b are scalar constants and x, y are vector variables.

$\mathbf{Q}'$, $\mathbf{K}'$ and non-stationary series $\mathbf{x}$, $\mu_{\mathbf{x}}$, $\sigma_{\mathbf{x}}$, and multiplies by the stationarized values $\mathbf{V}'$. Therefore, it can benefit from the predictability of stationarized series and maintain the inherent temporal dependencies of raw series simultaneously.

Overall architecture Following the prior use of Transformers [39, 37] in time series forecasting, we adopt the standard Encoder-Decoder structure (Figure 2), where the encoder is to extract information from past observations, and the decoder is to aggregate past information and refine the prediction from simple initialization. The canonical Non-stationary Transformer is wrapped by Series Stationarization to both the input and output of vanilla Transformer [34], and replacing the Self-Attention by our proposed De-stationary Attention, which can boost the non-stationary series predictive capability of the base model. For the Transformer variants [19, 39, 37], we transform the terms inside $\mathrm{Softmax}(\cdot)$ with the de-stationary factors τ, $\mathbf{\Delta}$ to re-integrate the non-stationary information (See Appendix E.2 for the implementation details).

4 Experiments

We conduct extensive experiments to evaluate the performance of Non-stationary Transformers on six real-world time series forecasting benchmarks and further validate the generality of the proposed framework on various mainstream Transformer variants.

Datasets Here are the descriptions of the datasets: (1) **Electricity** [1] records the hourly electricity consumption of 321 clients from 2012 to 2014. (2) **ETT** [39] contains the time series of oil de-stationary factors and power load collected by electricity transformers from July 2016 to July 2018. ETTm1 /ETTm2 are recorded every 15 minutes, and ETTh1/ETTh2 are recorded every hour. (3) **Exchange** [20] collects the panel data of daily exchange rates from 8 countries from 1990 to 2016. (4) **ILI** [2] collects the ratio of influenza-like illness patients versus the total patients in one week, which is reported weekly by Centers for Disease Control and Prevention of the United States from 2002 and 2021. (5) **Traffic** [3] contains hourly road occupancy rates measured by 862 sensors on San Francisco Bay area freeways from January 2015 to December 2016. (6) **Weather** [4] includes meteorological time series with 21 weather indicators collected every 10 minutes from the Weather Station of the Max Planck Biogeochemistry Institute in 2020.

Especially, in this paper, we adopt the Augmented Dick-Fuller (ADF) test statistic [14] as the metric to quantitatively measure the *degree of stationarity*. A smaller ADF test statistic indicates a higher degree of stationarity, which means the distribution is more stable. Table 1 summarizes the overall statistics of the datasets and lists them in ascending order by degree of stationarity. We follow the standard protocol that divides each dataset into the training, validation, and testing subsets according to the chronological order. The split ratio is 6:2:2 for the ETT dataset and 7:1:2 for others.

Table 1: Summary of datasets. Smaller ADF test statistic indicates more stationary dataset.

Dataset	Variable Number	Sampling Frequency	Total Observations	ADF Test Statistic
Exchange	8	1 Day	7,588	-1.889
ILI	7	1 Week	966	-5.406
ETTm2	7	15 Minutes	69,680	-6.225
Electricity	321	1 Hour	26,304	-8.483
Traffic	862	1 Hour	17,544	-15.046
Weather	21	10 Minutes	52,695	-26.661

Baselines We evaluate the vanilla Transformer [34] equipped by the Non-stationary Transformers framework in both multivariate and univariate settings to demonstrate its effectiveness. For multivariate forecasting, we include six state-of-the-art deep forecasting models: Autoformer [37], Pyraformer [23], Informer [39], LogTrans [22], Reformer [19] and LSTNet [20]. For univariate forecasting, we include seven competitive baselines: N-HiTS [9], N-BEATS [27], Autoformer [37], Pyraformer [23], Informer [39], Reformer [19] and ARIMA [7]. In addition, we adopt the proposed framework on both the canonical and efficient variants of Transformers: Transformer [34], Informer [39], Reformer [19] and Autoformer [37] to validate the generality of our framework.

Implementation details All the experiments are implemented with PyTorch [30] and conducted on a single NVIDIA TITAN V 12GB GPU. Each model is trained by ADAM [18] using L2 loss with

the initial learning rate of 10^{-4} and batch size of 32. Each Transformer-based model contains two encoder layers and one decoder layer. Considering the efficiency of hyperparameters search, we use two-layer perceptron projector with the hidden dimension varying in $\{64, 128, 256\}$ in De-stationary Attention. We repeat each experiment three times with different random seeds and report the test MSE/MAE under different prediction lengths, and the standard deviations are also provided in the Appendix C.2. A lower MSE/MAE indicates better performance.

4.1 Main Results

Forecasting results As for multivariate forecasting results, the vanilla Transformer equipped with our framework consistently achieves state-of-the-art performance in all benchmarks and prediction lengths (Table 2). Notably, Non-stationary Transformer outperforms other deep models impressively on datasets characterized by high non-stationarity: under the prediction length of 336, we achieve **17%** MSE reduction ($0.509 \rightarrow 0.421$) on Exchange and **25%** ($2.669 \rightarrow 2.010$) on ILI compared to previous state-of-the-art results, which indicates that the potential of deep model is still constrained on non-stationary data. We also list the univariate results of two typical datasets with different stationarity in Table 3. Non-stationary Transformer still realizes remarkable forecasting performance.

Table 2: Forecasting results comparison under different prediction lengths $O \in \{96, 192, 336, 720\}$. The input sequence length is set to 36 for ILI and 96 for the others. Additional results (ETTm1, ETTh1, ETTh2) can be found in Appendix C.1.

Models		**Ours**		Autoformer [37]		Pyraformer [23]		Informer [39]		LogTrans [22]		Reformer [19]		LSTNet [20]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	**0.111**	**0.237**	0.197	0.323	0.852	0.780	0.847	0.752	0.968	0.812	1.065	0.829	1.551	1.058
	192	**0.219**	**0.335**	0.300	0.369	0.993	0.858	1.204	0.895	1.040	0.851	1.188	0.906	1.477	1.028
	336	**0.421**	**0.476**	0.509	0.524	1.240	0.958	1.672	1.036	1.659	1.081	1.357	0.976	1.507	1.031
	720	**1.092**	**0.769**	1.447	0.941	1.711	1.093	2.478	1.310	1.941	1.127	1.510	1.016	2.285	1.243
ILI	24	**2.294**	**0.945**	3.483	1.287	5.800	1.693	5.764	1.677	4.480	1.444	4.400	1.382	6.026	1.770
	36	**1.825**	**0.848**	3.103	1.148	6.043	1.733	4.755	1.467	4.799	1.467	4.783	1.448	5.340	1.668
	48	**2.010**	**0.900**	2.669	1.085	6.213	1.763	4.763	1.469	4.800	1.468	4.832	1.465	6.080	1.787
	60	**2.178**	**0.963**	2.770	1.125	6.531	1.814	5.264	1.564	5.278	1.560	4.882	1.483	5.548	1.720
ETTm2	96	**0.192**	**0.274**	0.255	0.339	0.409	0.488	0.365	0.453	0.768	0.642	0.658	0.619	3.142	1.365
	192	**0.280**	**0.339**	0.281	0.340	0.673	0.641	0.533	0.563	0.989	0.757	1.078	0.827	3.154	1.369
	336	**0.334**	**0.361**	0.339	0.372	1.210	0.846	1.363	0.887	1.334	0.872	1.549	0.972	3.160	1.369
	720	**0.417**	**0.413**	0.422	0.419	4.044	1.526	3.379	1.388	3.048	1.328	2.631	1.242	3.171	1.368
Electricity	96	**0.169**	**0.273**	0.201	0.317	0.498	0.299	0.274	0.368	0.258	0.357	0.312	0.402	0.680	0.645
	192	**0.182**	**0.286**	0.222	0.334	0.828	0.312	0.296	0.386	0.266	0.368	0.348	0.433	0.725	0.676
	336	**0.200**	**0.304**	0.231	0.338	1.476	0.326	0.300	0.394	0.280	0.380	0.350	0.433	0.828	0.727
	720	**0.222**	**0.321**	0.254	0.361	4.090	0.372	0.373	0.439	0.283	0.376	0.340	0.420	0.957	0.811
Traffic	96	**0.612**	**0.338**	0.613	0.388	0.684	0.393	0.719	0.391	0.684	0.384	0.732	0.423	1.107	0.685
	192	**0.613**	**0.340**	0.616	0.382	0.692	0.394	0.696	0.379	0.685	0.390	0.733	0.420	1.157	0.706
	336	**0.618**	**0.328**	0.622	0.337	0.699	0.396	0.777	0.420	0.733	0.408	0.742	0.420	1.216	0.730
	720	**0.653**	**0.355**	0.660	0.408	0.712	0.404	0.864	0.472	0.717	0.396	0.755	0.423	1.481	0.805
Weather	96	**0.173**	**0.223**	0.266	0.336	0.354	0.392	0.300	0.384	0.458	0.490	0.689	0.596	0.594	0.587
	192	**0.245**	**0.285**	0.307	0.367	0.673	0.597	0.598	0.544	0.658	0.589	0.752	0.638	0.560	0.565
	336	**0.321**	**0.338**	0.359	0.395	0.634	0.592	0.578	0.523	0.797	0.652	0.639	0.596	0.597	0.587
	720	**0.414**	**0.410**	0.419	0.428	0.942	0.723	1.059	0.741	0.869	0.675	1.130	0.792	0.618	0.599

Framework generality We apply our framework to four mainstream Transformers and report the performance promotion of each model (Table 4). Our method consistently improves the forecasting ability of different models. Overall, it achieves averaged **49.43%** promotion on Transformer, **47.34%** on Informer, **46.89%** on Reformer and **10.57%** on Autoformer, making each of them surpass previous state-of-the-art. Compared to native blocks of the models, there is hardly any parameter and computation increase by applying our framework (See Appendix C.5 for details), and thereby their computational complexities can be preserved. It validates that Non-stationary Transformer is an effective and lightweight framework that can be widely applied to Transformer-based models and enhances their non-stationary predictability to achieve state-of-the-art performance.

Table 3: Univariate results under different prediction lengths $O \in \{96, 192, 336, 720\}$ on two typical datasets with strong non-stationary. The input sequence length is set to 96.

Models		**Ours**		N-HiTS [9]		N-BEATS [27]		Autoformer [37]		Pyraformer [23]		Informer [39]		Reformer [19]		ARIMA [6]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	**0.104**	**0.235**	0.114	0.248	0.156	0.299	0.241	0.387	0.290	0.439	0.591	0.615	1.327	0.944	0.112	0.245
	192	**0.230**	**0.375**	0.250	0.387	0.669	0.665	0.273	0.403	0.594	0.644	1.183	0.912	1.258	0.924	0.304	0.404
	336	**0.432**	**0.509**	0.434	0.516	0.611	0.605	0.508	0.539	0.962	0.824	1.367	0.984	2.179	1.296	0.736	0.598
	720	**0.782**	**0.682**	1.061	0.773	1.111	0.860	0.991	0.768	1.285	0.958	1.872	1.072	1.280	0.953	1.871	0.935
ETTm2	96	0.069	0.193	0.092	0.232	0.082	0.219	**0.065**	**0.189**	0.074	0.208	0.088	0.225	0.131	0.288	0.211	0.362
	192	**0.109**	**0.249**	0.128	0.276	0.120	0.268	0.118	0.256	0.116	0.252	0.132	0.283	0.186	0.354	0.261	0.406
	336	**0.139**	**0.286**	0.165	0.314	0.226	0.370	0.154	0.305	0.143	0.295	0.180	0.336	0.220	0.381	0.317	0.448
	720	**0.180**	**0.331**	0.243	0.397	0.188	0.338	0.182	0.335	0.197	0.338	0.300	0.435	0.267	0.430	0.366	0.487

Table 4: Performance promotion by applying the proposed framework to Transformer and its variants. We report the averaged MSE/MAE of all prediction lengths (stated in Table 2) and the relative MSE reduction ratios (Promotion) by our framework. Full results (under all prediction lengths and promotion on ETSformer [36], FEDformer [40]) can be found in Appendix C.2.

Dataset	Exchange		ILI		ETTm2		Electricity		Traffic		Weather	
Model	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Transformer	1.425	0.915	4.864	1.460	1.501	0.869	0.277	0.372	0.665	0.363	0.657	0.573
+ Ours	**0.457**	**0.449**	**2.077**	**0.914**	**0.306**	**0.347**	**0.193**	**0.296**	**0.628**	**0.345**	**0.288**	**0.314**
Promotion	67.93%		57.30%		79.61%		30.32%		5.56%		56.16%	
Informer	1.550	0.998	5.137	1.544	1.410	0.823	0.311	0.397	0.764	0.416	0.634	0.548
+ Ours	**0.496**	**0.460**	**2.125**	**0.928**	**0.460**	**0.434**	**0.226**	**0.330**	**0.719**	**0.409**	**0.275**	**0.302**
Promotion	68.00%		58.63%		67.38%		27.33%		5.89%		56.78%	
Reformer	1.280	0.932	4.724	1.443	1.479	0.915	0.338	0.429	0.741	0.423	0.803	0.656
+ Ours	**0.462**	**0.468**	**2.865**	**1.065**	**0.493**	**0.441**	**0.206**	**0.308**	**0.682**	**0.372**	**0.286**	**0.308**
Promotion	63.91%		39.35%		66.67%		39.05%		7.96%		64.38%	
Autoformer	0.613	0.539	3.006	1.161	0.324	0.368	0.227	0.338	0.628	0.379	0.338	0.382
+ Ours	**0.487**	**0.491**	**2.545**	**1.039**	**0.305**	**0.345**	**0.216**	**0.315**	**0.619**	**0.364**	**0.286**	**0.310**
Promotion	20.55%		15.34%		5.86%		4.85%		1.43%		15.38%	

4.2 Ablation Study

Quality evaluation To explore the role of each module in our proposed framework, we compare the prediction results on ETTm2 obtained by three models: vanilla Transformer, Transformer with only Series Stationarization, and our Non-stationary Transformer. In Figure 3, we find out that the two modules strengthen the non-stationary forecasting ability of Transformer from different perspectives. Series Stationarization focuses on the alignment of statistical properties among each series input that benefits Transformer a lot to generalize on out-of-distribution data. However, as is shown in Figure 3(b), the over-stationarized circumstance for training makes the deep model more likely to output uneventful series with significant high stationarity and neglect the nature of non-stationary real-world data. With the aid of De-stationary Attention, the model gives concern back to the inherent non-stationarity of real-world time series. It is beneficial for an accurate prediction of the detailed series variation, which is vital in real-world time series forecasting.

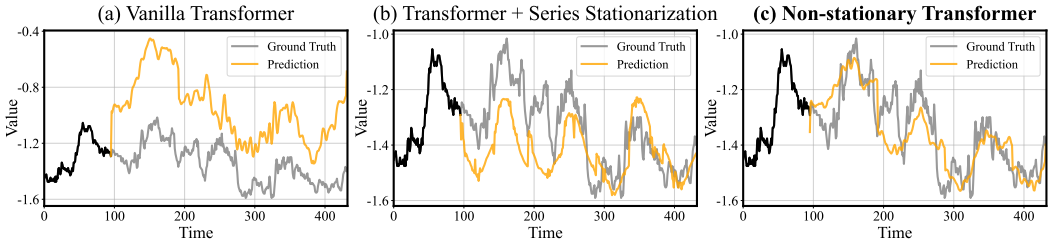

Figure 3: Visualization of ETTm2 predictions given by different models.

Table 5: Forecasting results obtained by applying different methods to Transformer and Reformer. We report the averaged MSE/MAE of all prediction lengths (stated in Table 2) for comparison. Complete results can be found in Appendix C.3.

Base Models	Transformer						Reformer					
Methods	+ RevIN [17]		+ Series Stationarization		**+ Ours**		+ RevIN [17]		+ Series Stationarization		**+ Ours**	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	0.567	0.487	0.569	0.488	**0.461**	**0.454**	0.469	0.472	0.470	0.473	**0.462**	**0.468**
ILI	2.205	0.934	2.206	0.934	**2.077**	**0.914**	3.024	1.096	3.023	1.096	**2.865**	**1.065**
ETTm2	0.460	0.416	0.461	0.416	**0.306**	**0.347**	0.542	0.459	0.537	0.459	**0.493**	**0.441**
Electricity	0.197	0.298	0.197	0.298	**0.193**	**0.296**	0.208	0.309	0.207	0.309	**0.206**	**0.308**
Traffic	0.643	0.352	0.641	0.352	**0.628**	**0.345**	0.687	0.378	0.691	0.380	**0.682**	**0.372**
Weather	0.301	0.316	0.304	0.317	**0.288**	**0.314**	0.291	0.309	0.292	0.309	**0.286**	**0.308**

Quantitative performance In addition to the above case study, we also provide quantitative forecasting performance comparison with stationarization methods: a deep method RevIN [17] and Series Stationarization (Section 3.1). As is shown in Table 5, the forecasting results assisted by RevIN and Series Stationarization are basically the same, which indicates that the parameter-free version of normalization in our framework performs sufficiently to stationarize time series. Besides, the proposed De-stationary Attention in Non-stationary Transformers further boosts the performance and achieves the best in all six benchmarks. The MSE reduction brought by De-stationary Attention becomes significant, especially when the dataset is highly non-stationary (Exchange: $0.569 \rightarrow 0.461$, ETTm2: $0.461 \rightarrow 0.306$). The comparison reveals that simply stationarizing time series still limits the predictive capability of Transformers, and the complementary mechanisms in Non-stationary Transformers can properly release the models' potential for non-stationary series forecasting.

4.3 Model Analysis

Over-stationarization problem To verify the over-stationarization problem from a statistical view, we train Transformers with the aforementioned methods respectively, arrange all predicted time series in chronological order and compare the degree of stationarity with the ground truth (Figure 4). While models solely equipped with stationarization methods tend to output series with unexpected high degree of stationarity, the results assisted by De-stationary Attention are close to the actual value (relative stationarity $\in [97\%, 103\%]$). Besides, as the degree of series stationarity increases, the over-stationarization problem becomes more significant. The huge discrepancy of the degree of stationarity can account for the inferior performance of Transformer with only stationarization. And it also demonstrates that De-stationary Attention as an internal renovation alleviates over-stationarization.

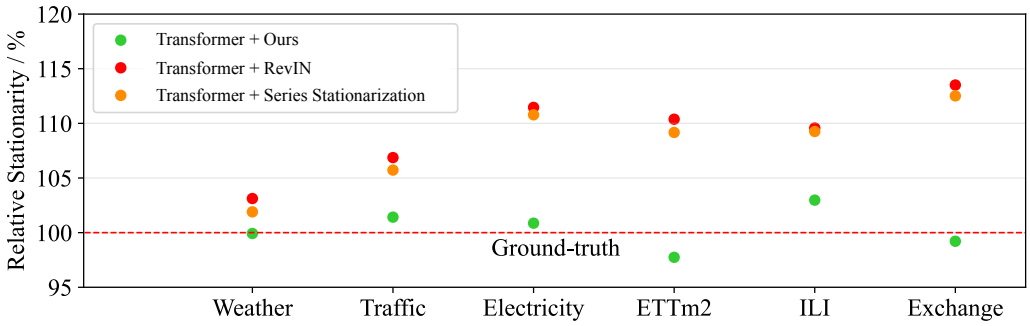

Figure 4: Relative stationarity is calculated as the ratio of ADF test statistics between the model predictions and ground truth. From left to right, the dataset is increasingly non-stationary. While models equipped with only stationarization tend to output highly stationary series, our method gives predictions with stationarity closer to ground truth.

Exploring of Non-stationary Information Re-incorporation It is notable that by specifying over-stationarization as less distinguishable attention, we narrow down our design space into the attention calculation mechanism. To explore other approaches to retrieve non-stationary information, we conduct experiments by re-incorporating the μ and σ into feed-forward layers (DeFF), which is the left part of the Transformer architecture. In detail, we feed learned μ and σ into each feed-forward layer iteratively. As is shown in Table 6, re-incorporating non-stationarity is necessary only when the inputs are stationarized (Stationary), which is beneficial for forecasting but will lead to stationarity discrepancy of model outputs. And our proposed design (Stat + DeAttn) makes further promotion and achieves the best in most cases (77%). In addition to the theoretical analysis, experimental results further validate the effectiveness of our design in re-incorporating non-stationarity on attention.

Table 6: Ablation of framework design. *Baseline* means vanilla Transformer, *Stationary* means adding Series Stationarization, *DeFF* means re-incorporating non-stationarity on feed-forward layers, *DeAttn* means re-incorporating by De-stationary Attention, *Stat + DeFF* means adding Series Stationarization and re-incorporating on feed-forward layers. *Stat + DeAttn* means our proposed framework.

Models		Baseline		Stationary		DeFF		DeAttn		Stat + DeFF		Stat + DeAttn	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	0.567	0.591	0.136	0.258	0.784	0.696	0.611	0.613	0.116	0.243	**0.111**	**0.237**
	192	1.150	0.825	0.239	0.348	1.162	0.866	1.202	0.840	0.280	0.383	**0.219**	**0.335**
	336	1.792	1.084	0.425	0.479	1.346	0.963	1.516	0.981	**0.371**	**0.452**	0.421	0.476
	720	2.191	1.159	1.475	0.865	2.042	1.163	2.894	1.377	**0.934**	**0.704**	1.092	0.769
ILI	24	4.748	1.430	2.573	0.980	4.850	1.445	4.734	1.424	2.404	0.985	**2.294**	**0.945**
	36	4.671	1.430	1.955	0.870	4.848	1.452	4.927	1.482	2.585	0.983	**1.825**	**0.848**
	48	4.994	1.482	2.057	0.902	4.903	1.466	4.996	1.483	2.496	0.991	**2.010**	**0.900**
	60	5.041	1.499	2.238	0.982	5.196	1.524	5.184	1.519	2.667	1.059	**2.178**	**0.963**
ETTm2	96	0.572	0.552	0.253	0.311	0.767	0.635	0.304	0.406	0.275	0.329	**0.192**	**0.274**
	192	1.161	0.793	0.453	0.404	0.960	0.717	0.820	0.652	0.406	0.403	**0.280**	**0.339**
	336	1.209	0.842	0.546	0.461	1.159	0.811	1.406	0.883	0.502	0.465	**0.334**	**0.361**
	720	3.061	1.289	0.593	0.489	3.187	1.308	2.858	1.108	0.694	0.575	**0.417**	**0.413**
Electricity	96	0.260	0.358	0.171	0.275	0.260	0.356	0.253	0.351	0.170	0.274	**0.169**	**0.273**
	192	0.266	0.367	0.192	0.296	0.264	0.365	0.257	0.358	0.188	0.293	**0.182**	**0.286**
	336	0.280	0.375	0.208	0.306	0.277	0.374	0.270	0.365	0.206	0.309	**0.200**	**0.304**
	720	0.302	0.386	**0.216**	**0.315**	0.299	0.384	0.295	0.380	0.223	0.323	0.222	0.321
Traffic	96	0.647	0.357	0.614	0.337	0.646	0.353	0.650	0.358	**0.605**	**0.333**	0.612	0.338
	192	0.649	0.356	0.637	0.351	0.645	0.352	0.655	0.358	0.617	0.342	**0.613**	**0.340**
	336	0.667	0.364	0.653	0.359	0.672	0.360	0.656	0.355	0.635	0.349	**0.618**	**0.328**
	720	0.697	0.376	0.661	0.360	0.695	0.376	0.681	0.366	**0.649**	**0.351**	0.653	0.355
Weather	96	0.395	0.427	0.175	0.225	0.417	0.445	0.296	0.364	0.178	0.226	**0.173**	**0.223**
	192	0.619	0.560	0.273	0.297	0.699	0.604	0.480	0.464	0.256	0.295	**0.245**	**0.285**
	336	0.689	0.594	0.333	**0.325**	0.773	0.620	0.581	0.519	0.338	0.351	**0.321**	0.338
	720	0.926	0.710	0.436	0.420	1.008	0.718	0.795	0.642	0.417	0.412	**0.414**	**0.410**

5 Conclusion

This paper addresses time series forecasting from the view of stationarity. Unlike previous studies that simply attenuate non-stationarity leading to over-stationarization, we propose an efficient way to increase series stationarity and renovate the internal mechanism to re-incorporate non-stationary information, thus boosting data predictability and model predictive capability simultaneously. Experimentally, our method shows great generality and performance on six real-world benchmarks. And detailed derivations and ablations are provided to testify the effectiveness of each component in our proposed Non-stationary Transformers framework. In the future, we will explore a more model-agnostic solution for the over-stationarization problem.

Acknowledgments

This work was supported by the National Key Research and Development Plan (2021YFC3000905), National Natural Science Foundation of China (62022050 and 62021002), Beijing Nova Program (Z201100006820041), and BNRist Innovation Fund (BNR2021RC01002).

References

[1] ECL load. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.

[2] Illness Dataset. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

[3] Traffic Dataset. http://pems.dot.ca.gov/.

[4] Weather Dataset. https://www.bgc-jena.mpg.de/wetter/.

[5] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. *NeurIPS*, 2021.

[6] O. Anderson and M. Kendall. Time-series. 2nd edn. *J. R. Stat. Soc. (Series D)*, 1976.

[7] G. E. P. Box and Gwilym M. Jenkins. Time series analysis, forecasting and control. 1970.

[8] George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *J. R. Stat. Soc. (Series-C)*, 1968.

[9] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886*, 2022.

[10] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 2021.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[12] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[14] Graham Elliott, Thomas J. Rothenberg, and James H. Stock. Efficient tests for an autoregressive unit root. *Econometrica*, 1996.

[15] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers. *NeurIPS*, 2021.

[16] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[17] 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 *ICLR*, 2022.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[20] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, 2018.

[21] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[22] 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. In *NeurIPS*, 2019.

[23] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *ICLR*, 2021.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[25] Danielle C Maddix, Yuyang Wang, and Alex Smola. Deep factors with gaussian processes for forecasting. *arXiv preprint arXiv:1812.00098*, 2018.

[26] Eduardo Ogasawara, Leonardo C. Martinez, Daniel de Oliveira, Geraldo Zimbrão, Gisele L. Pappa, and Marta Mattoso. Adaptive normalization: A novel data normalization approach for non-stationary time series. In *IJCNN*, 2010.

[27] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR*, 2019.

[28] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *TKDE*, 2009.

[29] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Deep adaptive input normalization for time series forecasting. *TNNLS*, 2019.

[30] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[31] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *NeurIPS*, 2018.

[32] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.*, 2020.

[33] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[35] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *NeurIPS*, 2017.

[36] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:1406.1078*, 2022.

[37] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

[38] Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term forecasting using tensor-train rnns. *arXiv preprint arXiv:1711.00073*, 2017.

[39] 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 *AAAI*, 2021.

[40] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix F.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the data link and code in `https://github.com/thuml/Nonstationary_Transformers`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix C.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] The data source is described in Section 4.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide the code in `https://github.com/thuml/Nonstationary_Transformers`.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of De-stationary Attention

Definition. Self-Attention [34] is defined as:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{7}$$

where $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V} \in \mathbb{R}^{S \times d_k}$ are length-S query, key and value, where S is the length of input sequence and d_k is the feature dimension, and $\mathrm{Softmax}(\cdot)$ is conducted on each row.

Assumption 1. *The embedding layer and feed forward layer are functions conducted separately at each time point of the input and hold the linear property.*

For example, the query $\mathbf{Q}$ as the input of the first $\mathrm{Attn}(\cdot)$ layer is obtained by feeding the input $\mathbf{x} = [x_1, x_2, \cdots, x_S]^\top \in \mathbb{R}^{S \times C}$ into the embedding layer $f : \mathbb{R}^{C \times 1} \to \mathbb{R}^{d_k \times 1}$, where C is the number of series variables. And each of the query token in $\mathbf{Q} = [q_1, q_2, ..., q_S]^\top$ can be calculated as $q_i = f(x_i)$ w.r.t. each time point in $\mathbf{x} = [x_1, x_2, \cdots, x_S]^\top$. Function f holds the linear property means that $f(ax + by) = af(x) + bf(y)$, where a, b are scalars and x, y are vectors.

Assumption 2. *Each variable of the input series has the same variance.*

For each input time series $\mathbf{x}$, we calculate its mean and variance as follows:

$$\mu_{\mathbf{x}} = \frac{1}{S}\sum_{i=1}^{S} x_i,\ \sigma_{\mathbf{x}}^2 = \frac{1}{S}\sum_{i=1}^{S}(x_i - \mu_{\mathbf{x}})^2,$$

where $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}} \in \mathbb{R}^{C \times 1}$ is the mean and standard deviation of all x_is. Since it is a convention to conduct normalization on each series variable to avoid certain variable that dominates the scale, we can assume that each variable shares the same variance, and thus $\sigma_{\mathbf{x}}$ is reduced to a scalar.

Theorem.

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) = \mathrm{Softmax}\left(\frac{\sigma_{\mathbf{x}}^2\, \mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}\mu_{\mathbf{Q}}^\top\mathbf{K}^\top}{\sqrt{d_k}}\right). \tag{8}$$

Equation 8 means the $\mathrm{Softmax}\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}\right)$ learned from raw series $\mathbf{x}$ can be calculated by current $\mathbf{Q}', \mathbf{K}'$ learned from stationarized series $\mathbf{x}'$, and the calculation also requires the non-stationary information $\sigma_{\mathbf{x}}, \mu_{\mathbf{Q}}, \mathbf{K}$ that are eliminated during stationarization.

Proof 1. (First layer analysis) After our stationarization, the model receives the normalized input $\mathbf{x}' = [x'_1, x'_2, ..., x'_S]^\top$ and each $x'_i = (1/\sigma_{\mathbf{x}}) \odot (x_i - \mu_{\mathbf{x}})$. Based on Assumption 2, $\sigma_{\mathbf{x}}$ is reduced to a scalar and we can simplify the normalized input of each time point to $x'_i = (x_i - \mu_{\mathbf{x}})/\sigma_{\mathbf{x}}$. Then $\mathbf{x}'$ is fed into the embedding layer f. Based on Assumption 1, we get current query $\mathbf{Q}' = [q'_1, ..., q'_S]^\top$ of the first $\mathrm{Attn}(\cdot)$ layer:

$$q'_i = f(\frac{x_i - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}) = \frac{f(x_i) - f(\mu_{\mathbf{x}})}{\sigma_{\mathbf{x}}} = \frac{q_i - f(\frac{1}{S}\sum_{i=1}^{S} x_i)}{\sigma_{\mathbf{x}}} = \frac{q_i - \frac{1}{S}\sum_{i=1}^{S} f(x_i)}{\sigma_{\mathbf{x}}} = \frac{q_i - \mu_{\mathbf{Q}}}{\sigma_{\mathbf{x}}},$$

where $\mu_{\mathbf{Q}} = \frac{1}{S}\sum_{i=1}^{S} q_i \in \mathbb{R}^{d_k \times 1}$. Then $\mathbf{Q}' = [q'_1, ..., q'_S]^\top$ can be written as $(\mathbf{Q} - \mathbf{1}\mu_{\mathbf{Q}}^\top)/\sigma_{\mathbf{x}}$ and $\mathbf{1} \in \mathbb{R}^{S \times 1}$ is an all-ones vector. And so is the corresponding transformed $\mathbf{K}'$. Without the stationarization, the input of $\mathrm{Softmax}(\cdot)$ in Self-Attention should be $(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k})$, while now the attention is calculated based on $\mathbf{Q}', \mathbf{K}'$. And we have the following equations:

$$\mathbf{Q}'\mathbf{K}'^\top = \frac{1}{\sigma_{\mathbf{x}}^2}\left(\mathbf{Q}\mathbf{K}^\top - \mathbf{1}(\mu_{\mathbf{Q}}^\top\mathbf{K}^\top) - (\mathbf{Q}\mu_{\mathbf{K}})\mathbf{1}^\top + \mathbf{1}(\mu_{\mathbf{Q}}^\top\mu_{\mathbf{K}})\mathbf{1}^\top\right),$$

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) = \mathrm{Softmax}\left(\frac{\sigma_{\mathbf{x}}^2\, \mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}(\mu_{\mathbf{Q}}^\top\mathbf{K}^\top) + (\mathbf{Q}\mu_{\mathbf{K}})\mathbf{1}^\top - \mathbf{1}(\mu_{\mathbf{Q}}^\top\mu_{\mathbf{K}})\mathbf{1}^\top}{\sqrt{d_k}}\right).$$

We find that $\mathbf{Q}\mu_{\mathbf{K}} \in \mathbb{R}^{S \times 1}$ and $\mu_{\mathbf{Q}}^\top\mu_{\mathbf{K}} \in \mathbb{R}$, and they are repeatedly operated on each column and element of $\sigma_{\mathbf{x}}^2\mathbf{Q}'\mathbf{K}'^\top \in \mathbb{R}^{S \times S}$. Since $\mathrm{Softmax}(\cdot)$ is invariant to the same translation on the row dimension of input, we have the following equation:

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) = \mathrm{Softmax}\left(\frac{\sigma_{\mathbf{x}}^2\, \mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}\mu_{\mathbf{Q}}^\top\mathbf{K}^\top}{\sqrt{d_k}}\right).$$

Proof 2. (Multiple layers analysis) We have deduced an equivalent expression of the output of the first $\mathrm{Softmax}(\cdot)$. If we can successfully approximate the attention map that related to $\mathbf{Q}$ and $\mathbf{K}$, we only need to consider $\mathrm{Attn}(\cdot)$ with respect to the change of $\mathbf{V}$. Fortunately, $\mathrm{Attn}(\cdot)$ as the function of $\mathbf{V}$ gives each time point of the output $\mathbf{E} = [e_1, ..., e_S]^\top \in \mathbb{R}^{S \times d_k}$ as a simplex:

$$e_j = \left\{ \sum_{i=1}^{S} w_i v_i | \mathbf{V} = [v_1, v_2, ..., v_S]^\top, \sum_{i=1}^{S} w_i = 1, w_i \geq 0 \right\},$$

which also holds the linear property $f(a\mathbf{V_1} + b\mathbf{V_2}) = af(\mathbf{V_1}) + bf(\mathbf{V_2})$. Therefore, the $\mathrm{Attn}(\cdot)$ layer is also a function that satisfies our Assumption 1. We will have each time point of the output $\mathbf{E}$ varies linearly with each time point of the input $\mathbf{x}$, and then $\mathbf{E}$ will become the next block's input. As the feed forward layer, residual adding and $\mathrm{Attn}(\cdot)$ layer are the repeating building blocks of Transformer, they also compose a function with linear property as stated in Assumption 1. By the first layer analysis and induction on each layer, Equation 8 will holds for $\mathrm{Softmax}(\cdot)$ of all layers under our assumptions.

Attention design Based on the analysis, we develop De-stationary Attention as:

$$\begin{gathered} \log \tau = \mathrm{MLP}(\sigma_{\mathbf{x}}, \mathbf{x}), \mathbf{\Delta} = \mathrm{MLP}(\mu_{\mathbf{x}}, \mathbf{x}), \\ \mathrm{Attn}(\mathbf{Q}', \mathbf{K}', \mathbf{V}', \tau, \mathbf{\Delta}) = \mathrm{Softmax}\left(\frac{\tau\, \mathbf{Q}'\mathbf{K}'^\top + \mathbf{1}\mathbf{\Delta}^\top}{\sqrt{d_k}} \right) \mathbf{V}', \end{gathered} \tag{9}$$

where $\tau \in \mathbb{R}^+$ and $\mathbf{\Delta} \in \mathbb{R}^{S \times 1}$ is defined as the scaling and shifting de-stationary factors respectively to approximate $\sigma_{\mathbf{x}}^2$ and $\mathbf{K}\mu_{\mathbf{Q}}$ under the real scenario. Since the key to making Equation 8 established is to approximate the attention map successfully, we apply a direct deep learning implementation. To be concisely, we use a multi-layer perceptron as the projector to learn de-stationary factors $\tau, \mathbf{\Delta}$ from the statistics $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}$ and unstationarized $\mathbf{x}$. De-stationary Attention learns the temporal dependencies from both stationarized series $\mathbf{Q}'$, $\mathbf{K}'$ and non-stationary series $\mathbf{x}$, $\mu_{\mathbf{x}}$, $\sigma_{\mathbf{x}}$, and multiplies by the stationarized values $\mathbf{V}'$ to keep the linear property. It can benefit from the predictability of stationarized series and re-incorporate the inherent non-stationarity of raw series simultaneously.

B Hyperparameter Sensitivity

We verify the robustness of the proposed Non-stationary Transformers framework with respect to hyper-parameter dim, which is the hidden layer dimension of the MLP projector that learns de-stationary factors. Considering the efficiency of hyperparameters search, we fix the number of hidden layers, and the hidden layer dimension varies in $\{64, 128, 256\}$. The results are shown in Table 7. For datasets with relatively high non-stationarity (Exchange and ILI), large dim would be a better choice, which indicates that non-stationary information entangled with unstationarized input should be learned by a projector with big capacity. Besides, as the dataset presents higher non-stationarity, the influence of de-stationary project design becomes more significant.

Table 7: The performance of Non-stationary Transformers under different choices of the hidden layer dimension (dim) in the projector. We adopt the forecasting setting as input-36-predict-48 for the ILI dataset and input-96-predict-336 for the other datasets.

Dataset	Exchange		ILI		ETTm2		Electricity		Traffic		Weather	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$dim = 64$	0.448	0.493	2.067	0.908	**0.334**	**0.361**	**0.200**	**0.304**	0.629	0.345	**0.321**	**0.338**
$dim = 128$	0.432	0.477	**2.010**	**0.900**	0.370	0.388	0.201	0.301	**0.618**	**0.328**	0.340	0.354
$dim = 256$	**0.421**	**0.476**	2.223	0.928	0.367	0.381	0.201	0.304	0.631	0.351	0.333	0.347

C Supplementary of Main Results

C.1 Multivariable Forecasting Results

As shown in Table 8, we list additional benchmark on the ETT datasets [39], which includes the hourly recorded ETTh1/ETTh2 and 15-minutely recorded ETTm1. Non-stationary Transformer also achieves remarkable improvement over the state-of-the-art on various forecasting horizons. For the input-96-predict-336 long-term setting, Non-stationary Transformer surpasses previous best results by **4.4%**$(0.615 \rightarrow 0.588)$ in ETTh1, **3.5%**$(0.572 \rightarrow 0.552)$ in ETTh2 and **26.7%**$(0.675 \rightarrow 0.495)$ MSE reduction in ETTm1. The overall results show averaged **11.5%** MSE reduction over previous state-of-the-art deep forecasting models.

We also list additional model comparison in Table 9, including the concurrent work FEDformer [40], and non-Transformer models LSSL [15] and GRU [11]. Our method still outperforms these models in most cases (83%). Notably, LSSL [15] achieves good performance on Weather [4] dataset with the highest stationarity but poorly performs on others, especially non-stationary datasets.

Table 8: Forecasting results comparison under different prediction lengths $O \in \{96, 192, 336, 720\}$ on ETT dataset. The input sequence length is set to 96.

Models		**Ours**		Autoformer[37]		Pyraformer[23]		Informer[39]		LogTrans[22]		Reformer[19]		LSTNet[20]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.513**	**0.491**	0.536	0.548	0.783	0.657	0.984	0.786	0.767	0.758	0.773	0.640	1.457	0.961
	192	**0.534**	**0.504**	0.543	0.551	0.863	0.709	1.027	0.791	1.003	0.849	0.910	0.704	1.998	1.215
	336	**0.588**	**0.535**	0.615	0.592	0.941	0.753	1.032	0.774	1.362	0.952	1.000	0.760	2.655	1.369
	720	0.643	0.616	**0.599**	**0.600**	1.042	0.819	1.169	0.858	1.397	1.291	1.242	0.860	2.143	1.380
ETTh2	96	**0.476**	**0.458**	0.492	0.517	1.380	0.943	2.826	1.330	0.829	0.751	1.595	1.031	3.568	1.688
	192	**0.512**	**0.493**	0.556	0.551	3.809	1.634	6.186	2.070	1.807	1.036	2.671	1.300	3.243	2.514
	336	**0.552**	**0.551**	0.572	0.578	4.282	1.792	5.268	1.942	3.875	1.763	2.596	1.297	2.544	2.591
	720	**0.562**	**0.560**	0.580	0.588	4.252	1.790	3.667	1.616	3.913	1.552	2.647	1.304	4.625	3.709
ETTm1	96	**0.386**	**0.398**	0.523	0.488	0.536	0.506	0.615	0.556	0.588	0.593	0.778	0.623	2.003	1.218
	192	**0.459**	**0.444**	0.543	0.498	0.539	0.520	0.723	0.620	0.769	0.793	0.929	0.707	2.764	1.544
	336	**0.495**	**0.464**	0.675	0.551	0.720	0.635	1.300	0.908	1.462	1.320	1.016	0.733	1.257	2.076
	720	**0.585**	**0.516**	0.720	0.573	0.940	0.740	0.972	0.744	1.669	1.461	1.122	0.793	1.917	2.941

Table 9: Forecasting results comparison with additional baseline forecasting models.

Models		Ours		FEDformer [40]		LSSL [15]		GRU [11]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	**0.111**	**0.237**	0.148	0.271	0.395	0.474	1.453	1.049
	192	**0.219**	**0.335**	0.271	0.380	0.776	0.698	1.846	1.179
	336	**0.421**	**0.476**	0.460	0.500	1.029	0.797	2.136	1.231
	720	**1.092**	**0.769**	1.195	0.841	2.283	1.222	2.984	1.427
ILI	24	**2.294**	**0.945**	3.228	1.260	4.381	1.425	5.914	1.734
	36	**1.825**	**0.848**	2.679	1.080	4.442	1.416	6.631	1.845
	48	**2.010**	**0.900**	2.622	1.078	4.559	1.443	6.736	1.857
	60	**2.178**	**0.963**	2.857	1.157	4.651	1.474	6.870	1.879
ETTm2	96	**0.192**	**0.274**	0.203	0.287	0.243	0.342	2.041	1.073
	192	0.280	0.339	**0.269**	**0.328**	0.392	0.448	2.249	1.112
	336	0.334	**0.361**	**0.325**	0.366	0.932	0.724	2.568	1.238
	720	**0.417**	**0.413**	0.421	0.415	1.372	0.879	2.720	1.287
Electricity	96	**0.169**	**0.273**	0.193	0.308	0.300	0.392	0.375	0.437
	192	**0.182**	**0.286**	0.201	0.315	0.297	0.390	0.442	0.473
	336	**0.200**	**0.304**	0.214	0.329	0.317	0.403	0.439	0.473
	720	**0.222**	**0.321**	0.246	0.355	0.338	0.417	0.980	0.814
Traffic	96	0.612	**0.338**	**0.587**	0.366	0.798	0.436	0.843	0.453
	192	0.613	**0.340**	**0.604**	0.373	0.849	0.481	0.847	0.453
	336	**0.618**	**0.328**	0.621	0.383	0.828	0.476	0.853	0.455
	720	0.653	**0.355**	**0.626**	0.382	0.854	0.489	1.500	0.805
Weather	96	**0.173**	**0.223**	0.217	0.296	0.174	0.252	0.369	0.406
	192	0.245	**0.285**	0.276	0.336	**0.238**	0.313	0.416	0.435
	336	0.321	**0.338**	0.339	0.380	**0.287**	0.355	0.455	0.454
	720	0.414	**0.410**	0.403	0.428	**0.384**	0.415	0.535	0.520

C.2 Performance of Non-stationary Transformer and Variants

We apply our proposed Non-stationary Transformers framework to six Transformer variants: Transformer [34], Informer [39], Reformer [19], Autoformer [37], ETSformer [36] and FEDformer [40]. The averaged results are shown in Table 4 due to the limited pages. We provide supplementary forecasting results in Table 10 and

Table 11. The experimental results demonstrate that our Non-stationary Transformers framework can consistently promotes these Transformer variants, even on the concurrent work ETSformer and FEDformer.

Table 10: Detailed forecasting performance of Non-stationary Transformers. We report the MSE/MAE of different prediction lengths $O \in \{96, 192, 336, 720\}$ and $\{24, 36, 48, 60\}$ for comparison. The input sequence length is set to 36 for ILI and 96 for the others.

Models		Transformer + Ours		Informer + Ours		Reformer + Ours		Autoformer + Ours	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	0.111±0.015	0.237±0.010	0.129±0.018	0.258±0.012	0.128±0.019	0.258±0.012	0.171±0.005	0.276±0.006
	192	0.219±0.031	0.335±0.020	0.251±0.042	0.354±0.035	0.246±0.045	0.356±0.037	0.273±0.005	0.365±0.007
	336	0.421±0.032	0.476±0.022	0.373±0.047	0.434±0.032	0.422±0.039	0.478±0.030	0.481±0.010	0.573±0.009
	720	1.092±0.027	0.769±0.024	1.229±0.035	0.795±0.049	1.050±0.050	0.781±0.047	1.024±0.012	0.751±0.012
ILI	24	2.294±0.152	0.945±0.041	2.856±0.177	1.071±0.067	3.206±0.277	1.131±0.079	3.029±0.116	1.166±0.028
	36	1.825±0.128	0.848±0.033	1.805±0.143	0.860±0.051	2.750±0.161	1.018±0.074	2.648±0.134	1.023±0.032
	48	2.010±0.134	0.900±0.035	1.780±0.194	0.849±0.054	2.710±0.184	1.017±0.050	2.202±0.161	0.965±0.038
	60	2.178±0.146	0.963±0.037	2.058±0.173	0.933±0.058	2.792±0.153	1.095±0.047	2.302±0.088	1.003±0.024
ETTm2	96	0.192±0.023	0.274±0.016	0.241±0.035	0.312±0.025	0.209±0.040	0.287±0.028	0.236±0.022	0.319±0.019
	192	0.280±0.021	0.339±0.013	0.433±0.036	0.420±0.025	0.435±0.037	0.421±0.026	0.263±0.026	0.316±0.025
	336	0.334±0.011	0.361±0.017	0.507±0.032	0.464±0.023	0.559±0.033	0.475±0.024	0.320±0.019	0.349±0.014
	720	0.417±0.009	0.413±0.011	0.659±0.019	0.539±0.028	0.769±0.021	0.582±0.021	0.402±0.015	0.396±0.010
Electricity	96	0.169±0.008	0.273±0.002	0.195±0.008	0.302±0.003	0.190±0.007	0.293±0.004	0.193±0.009	0.295±0.003
	192	0.182±0.007	0.286±0.003	0.215±0.007	0.321±0.006	0.199±0.009	0.301±0.008	0.211±0.006	0.310±0.007
	336	0.200±0.005	0.304±0.005	0.235±0.006	0.339±0.006	0.208±0.005	0.310±0.005	0.220±0.005	0.316±0.004
	720	0.222±0.016	0.321±0.013	0.260±0.014	0.358±0.014	0.226±0.015	0.326±0.018	0.241±0.019	0.337±0.017
Traffic	96	0.612±0.019	0.338±0.014	0.649±0.028	0.370±0.016	0.669±0.037	0.364±0.020	0.604±0.027	0.342±0.012
	192	0.613±0.028	0.340±0.018	0.689±0.035	0.393±0.019	0.680±0.036	0.369±0.022	0.607±0.034	0.383±0.020
	336	0.618±0.018	0.328±0.012	0.755±0.055	0.431±0.054	0.688±0.038	0.371±0.033	0.611±0.019	0.353±0.010
	720	0.653±0.014	0.355±0.003	0.783±0.026	0.440±0.004	0.692±0.019	0.385±0.014	0.653±0.014	0.376±0.013
Weather	96	0.173±0.006	0.223±0.004	0.186±0.017	0.235±0.014	0.195±0.020	0.242±0.013	0.215±0.024	0.263±0.019
	192	0.245±0.014	0.285±0.015	0.259±0.024	0.292±0.019	0.255±0.027	0.289±0.023	0.257±0.027	0.296±0.018
	336	0.321±0.016	0.338±0.023	0.295±0.026	0.317±0.018	0.306±0.030	0.323±0.025	0.307±0.009	0.321±0.011
	720	0.414±0.032	0.410±0.031	0.361±0.020	0.362±0.022	0.388±0.024	0.376±0.026	0.364±0.006	0.357±0.007

C.3 Comparison with Stationarization Methods

We provide full comparison among Non-stationary Transformers and two stationarization methods: Revin[17] and Series Stationarization. The averaged results are shown in Table 5 due to the limited pages. As is listed in Table 12, our framework achieves the state-of-the-art performance especially on datasets with high non-stationarity. For Transformer, the proposed method achieves **25.6%**$(1.467 \rightarrow 1.092)$ MSE reduction on Exchange under the predict-720 settings, **10.8%**$(2.572 \rightarrow 2.294)$ on ILI under the predict-24 settings, and **30.3%**$(0.598 \rightarrow 0.417)$ on ETTm2 under the predict-720 settings. As for Reformer, since De-stationary Attention is not directly deduced from the LSH attention [19], current approximation as stated in Equation 8 may not be the optimal solution, but the introducing of De-stationary Attention still achieves relative **11.6%**$(0.632 \rightarrow$

Table 11: Performance promotion by applying the proposed framework to concurrent ETSformer and FEDformer. We report the averaged MSE/MAE of all prediction lengths (stated in Table 2) and the relative MSE reduction ratios (Promotion) by our framework.

Dataset	Exchange		ILI		ETTm2		Electricity		Traffic		Weather	
Model	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETSformer [36]	0.410	0.427	2.619	1.034	0.293	0.342	0.208	0.323	0.629	0.403	0.271	0.334
+ Ours	0.369	0.407	2.353	1.017	0.290	0.334	0.203	0.314	0.618	0.380	0.254	0.293
Promotion	**10.00%**		**10.16%**		0.77%		**2.17%**		1.75%		**6.18%**	
FEDformer [40]	0.519	0.500	2.847	1.144	0.305	0.349	0.214	0.327	0.610	0.376	0.309	0.360
+ Ours	0.500	0.487	2.728	1.046	0.312	0.346	0.198	0.300	0.604	0.362	0.268	0.292
Promotion	**3.66%**		**4.18%**		-2.38%		**7.38%**		0.86%		**13.36%**	

0.559) promotion on ETTm2 and **4.4%**(2.834 → 2.770) on ILI under the predict-336 setting. The comparison demonstrates De-stationary Attention mechanism can further benefit the predictive ability of Transformers.

Table 12: Detailed forecasting results obtained by applying different methods to Transformer and Reformer. We report the MSE/MAE of different prediction lengths for comparison.

Base Models		Transformer						Reformer					
Methods		+ RevIN [17]		+ Series Stationarization		+ **Ours**		+ RevIN [17]		+ Series Stationarization		+ **Ours**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	0.136	0.258	0.136	0.258	**0.111**	**0.237**	0.133	0.263	0.139	0.265	**0.128**	**0.258**
	192	0.239	0.348	0.239	0.348	**0.219**	**0.335**	0.256	0.363	0.257	0.364	**0.246**	**0.356**
	336	0.425	0.479	0.425	0.479	**0.421**	**0.476**	0.426	0.477	0.426	0.477	**0.422**	**0.478**
	720	1.467	0.862	1.475	0.865	**1.092**	**0.769**	1.059	0.786	1.059	0.786	**1.050**	**0.781**
ILI	24	2.572	0.980	2.573	0.980	**2.294**	**0.945**	3.399	1.170	3.399	1.170	**3.206**	**1.131**
	36	1.955	0.870	1.955	0.870	**1.825**	**0.848**	2.909	1.049	2.909	1.048	**2.750**	**1.018**
	48	2.056	0.902	2.057	0.902	**2.010**	**0.900**	2.834	1.067	2.832	1.067	**2.710**	**1.017**
	60	2.238	0.982	2.238	0.982	**2.178**	**0.963**	2.954	1.099	2.952	1.098	**2.792**	**1.095**
ETTm2	96	0.267	0.317	0.253	0.311	**0.192**	**0.274**	0.211	0.295	0.212	0.297	**0.209**	**0.287**
	192	0.456	0.405	0.453	0.404	**0.280**	**0.339**	0.478	0.426	0.477	0.426	**0.435**	**0.421**
	336	0.528	0.455	0.546	0.461	**0.334**	**0.361**	0.632	0.485	0.613	0.483	**0.559**	**0.475**
	720	0.589	0.487	0.593	0.489	**0.417**	**0.413**	0.845	0.631	0.846	0.630	**0.769**	**0.582**
Electricity	96	0.172	0.275	0.171	0.275	**0.169**	**0.273**	0.188	0.291	**0.184**	**0.289**	0.190	0.293
	192	0.192	0.296	0.192	0.296	**0.182**	**0.286**	0.198	0.301	0.199	0.302	**0.198**	**0.301**
	336	0.207	0.306	0.208	0.306	**0.200**	**0.304**	0.212	0.314	0.212	0.314	**0.208**	**0.310**
	720	0.217	0.316	**0.216**	**0.315**	0.222	0.321	0.232	0.331	0.231	0.330	**0.226**	**0.326**
Traffic	96	0.620	0.341	0.614	0.337	**0.612**	**0.338**	**0.650**	**0.364**	0.655	0.366	0.669	0.364
	192	0.630	0.348	0.637	0.351	**0.613**	**0.340**	0.688	0.374	0.683	0.377	**0.680**	**0.369**
	336	0.656	0.360	0.653	0.359	**0.634**	**0.348**	0.708	0.383	0.704	0.383	**0.688**	**0.371**
	720	0.666	0.360	0.661	0.360	**0.653**	**0.355**	0.700	0.392	0.722	0.395	**0.692**	**0.385**
Weather	96	0.175	0.225	0.175	0.225	**0.173**	**0.223**	**0.189**	**0.236**	0.190	0.237	0.195	0.242
	192	0.273	0.298	0.273	0.297	**0.245**	**0.285**	0.269	0.294	0.269	0.294	**0.255**	**0.289**
	336	0.333	0.326	0.333	0.325	**0.321**	**0.338**	0.312	0.328	0.313	0.329	**0.306**	**0.323**
	720	0.424	0.415	0.436	0.420	**0.414**	**0.410**	0.395	0.376	0.395	0.376	**0.388**	**0.376**

C.4 Prediction Showcases

We provide supplementary showcases of predictions given by three models: vanilla Transformer, Transformer with Series Stationarization, and Non-stationary Transformer. We plot the last dimension of forecasting results that comes from the *test set* of ETTm1 for qualitative comparison.

As is shown in Figures 5, 6, 7, and 8, we find that vanilla Transformer is inclined to output predictions with scale and level far from the ground truth, but its ability to capture local series variation remains strong. While Series Stationarization benefits Transformer by aligning the statistics among each series, the base model neglects the intrinsic non-stationarity of time series and becomes more likely to output stationary but uneventful series. With the help of our framework, the equipped model will be free from the disturbance caused by data non-stationarity and fulfill the potential to capture local variations.

Table 13: Parameters increment and performance promotion of Non-stationary Transformers.

Models	Transformer	Informer	Reformer	Autoformer	FEDformer	ETSformer
Param increment	0.10%	0.09%	0.21%	0.10%	0.06%	0.19%
Performance gain	49.43%	47.34%	46.89%	10.57%	4.51%	5.17%

C.5 Efficiency of Non-stationary Transformers

As is shown in Table 13, we list the parameters increment and the performance gain (see Promotion in Table 4) of our proposed method. It is obvious that Non-stationary Transformers significantly boosts the forecasting performance by a large margin with hardly any additional parameters.

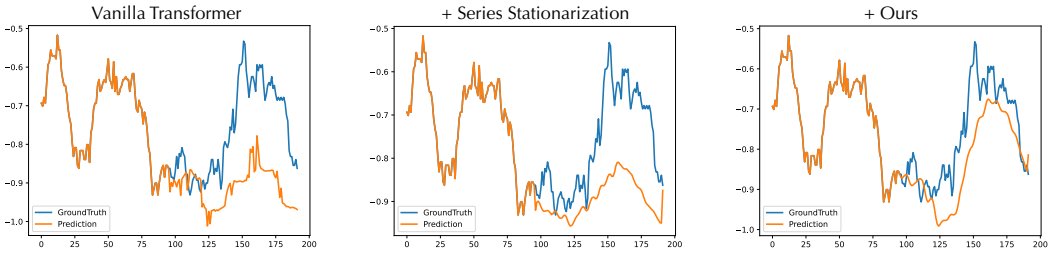

Figure 5: Visualization of ETTm1 predictions given by different models under the input-96-predict-96 setting. Blue lines stand for the ground truth and orange lines stand for predictions of the model. The first shared part is the time series input with length 96.

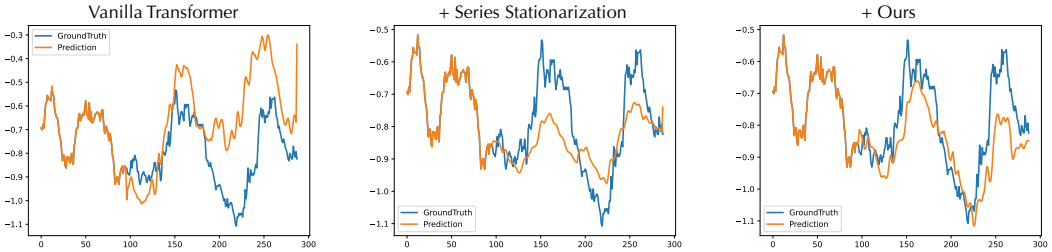

Figure 6: Visualization of predictions given by models under the input-96-predict-192 setting.

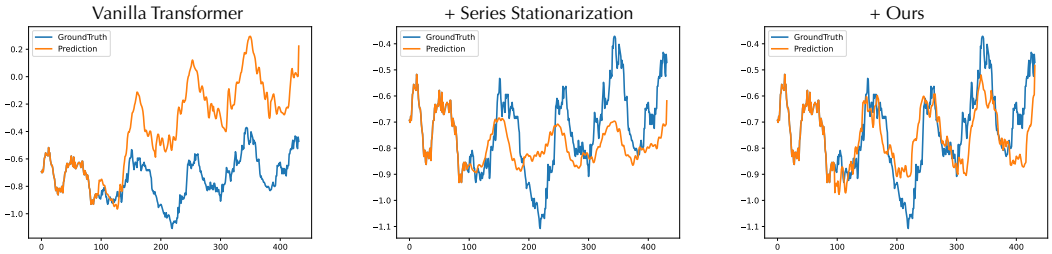

Figure 7: Visualization of predictions given by models under the input-96-predict-336 setting.

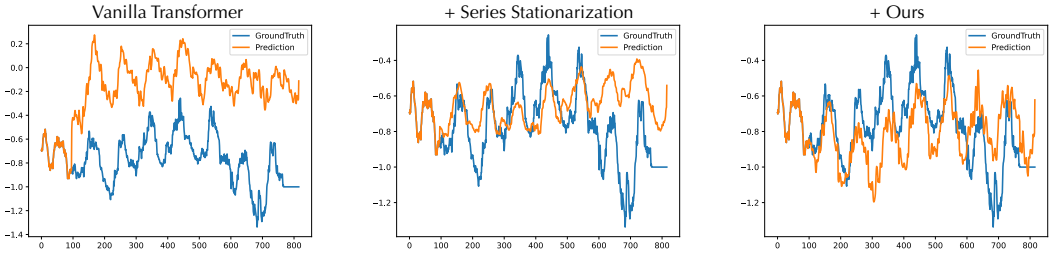

Figure 8: Visualization of predictions given by models under the input-96-predict-720 setting.

Table 14: ADF test statistic of raw series and series processed by our normalization.

Dataset	Exchange	ILI	ETTm2	Electricity	Traffic	Weather
Raw series	-1.889	-5.406	-6.225	-8.483	-15.046	-26.661
After Normalization	-9.937	-10.313	-33.485	-20.888	-18.946	-35.010

D Ablations

D.1 Effects of Series Stationarization

We propose Series Stationarization, which has no additional learnable parameters, to increase the degree of stationarity and make the time series distribution more stable. As is shown in Table 14, after our normalization module processing, the ADF test statistic of the time series gets obviously smaller, which verifies normalization as an effective design to attenuate the non-stationarity of real world time series.

D.2 Ablation of De-stationary Factors

To explore the influence of de-stationary factors, we compare the forecasting results obtained by three variants: only using τ, only using $\mathbf{\Delta}$, and using both. We conduct experiments on two typical datasets: Exchange (8 variables) and Electricity (321 variables). As is shown in Table 15, the forecasting performance will degrade in all cases if we only employ single one of τ and $\mathbf{\Delta}$, especially without τ ($0.196 \rightarrow 0.212, 0.441 \rightarrow 0.550$ under the predict-336 setting), which validates the complete form as stated in Equation 9 is a better choice.

Table 15: Ablation on de-stationary factors: Column (only τ) means only use the scaling de-stationary factor in Equation 9, Column (only $\mathbf{\Delta}$) means only use the shifting de-stationary factors, and Column (τ and $\mathbf{\Delta}$) means use both.

Models		Only τ		Only $\mathbf{\Delta}$		τ and $\mathbf{\Delta}$	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	0.177	0.279	0.186	0.287	**0.169**	**0.273**
	192	0.191	0.297	0.196	0.299	**0.185**	**0.289**
	336	0.197	0.300	0.212	0.310	**0.196**	**0.297**
	720	0.221	0.320	0.227	0.326	**0.217**	**0.317**
Exchange	96	0.128	0.253	0.128	0.253	**0.120**	**0.247**
	192	0.263	0.369	0.263	0.370	**0.250**	**0.353**
	336	0.446	0.491	0.550	0.553	**0.441**	**0.488**
	720	1.348	0.847	1.621	0.911	**1.338**	**0.847**

E Non-stationary Transformers: Experimental Details

E.1 Detailed Experiment Configurations

We compare each Transformers with and without our framework using the same training strategy. The only hyperparameters for our framework come from the projector design which learns de-stationary factors. We search the hyperparameters as stated in Appendix B. The best hyperparameter is selected on the *validation set*.

As for other forecast models for the baseline comparison, most of the results are from Autoformer [37]. By contacting the authors of Autoformer, we obtain the hyper-parameter selection strategy as follows: for N-BEATS [27], we conduct a grid search for hidden channel in $\{256, 512, 768\}$, number of layers in $\{2, 3, 4, 5\}$, learning rate in $\{5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$. For LSTNet [20], since the paper also experiments on the Traffic [3], Electricity [1] and Exchange [20] datasets, the hyper-parameter setting is following the experimental details of the original paper. For N-HiTs [9], ETSformer [36], and FEDformer [40], as these methods share the same benchmark, we use their official code with three random seeds.

E.2 Implementation Details of Non-stationary Transformer and Variants

We provide the pseudo-code of Series Stationarization, De-stationary Attention and Non-stationary Transformers in Algorithms 1, 2, 3 and 4. All Transformers have two-layer encoder and one-layer decoder with the feature dimension d_k=512, including Transformer [34], Informer [39], Reformer [19], Autoformer [37], ETSformer [36] and FEDformer [40]. Besides, we adopt embedding method and one-step generation strategy of Informer [39]. It is worth noting that for the row length of attention map differs from $S \times S$, where S is the initial input sequence length, we omit the shifting de-stationary factor $\mathbf{\Delta}$ in Equation 9 (i.e., the Self-Attention layer of Transformer decoder, and the Self-Attention layer of the Informer encoder where the shape of attention map is changed over layers), since the performance of only use τ will not degenerate a lot as shown in Table 15. For the cross attention, we first conduct the rescaling operation with de-stationary factors and then multiply by the corresponding mask. For Transformer variants, we conduct the rescaling operation on the pre-$\mathrm{Softmax}$ scores.

Algorithm 1 Series Stationarization - Normalization.

Require: Input past time series $\mathbf{x} \in \mathbb{R}^{S \times C}$; Input Length S; Variables number C.

$\mu_{\mathbf{x}} = \texttt{Mean}(\mathbf{x}, \texttt{dim=0})$ ▷ $\mu_{\mathbf{x}} \in \mathbb{R}^{1 \times C}$

$\sigma_{\mathbf{x}} = \texttt{Std}(\mathbf{x}, \texttt{dim=0})$ ▷ $\sigma_{\mathbf{x}} \in \mathbb{R}^{1 \times C}$

$\mathbf{x}' = \texttt{Repeat}\big((1/\sigma_{\mathbf{x}}), \texttt{dim=0}\big) \odot \big(\mathbf{x} - \texttt{Repeat}(\mu_{\mathbf{x}}, \texttt{dim=0})\big)$ ▷ Normalize to $\mathbf{x}' \in \mathbb{R}^{S \times C}$

Return $\mathbf{x}', \mu_{\mathbf{x}}, \sigma_{\mathbf{x}}$ ▷ Return stationarized input and original statistics

Algorithm 2 Series Stationarization - De-normalization.

Require: Predicted time series $\mathbf{y}' \in \mathbb{R}^{O \times C}$ by the base model; original statistics of input $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}} \in \mathbb{R}^{1 \times C}$; Output Length O; Variables number C.

$\mathbf{y} = \texttt{Repeat}\big(\sigma_{\mathbf{x}}, \texttt{dim=0}\big) \odot \mathbf{y}' + \texttt{Repeat}(\mu_{\mathbf{x}}, \texttt{dim=0})$ ▷ De-normalize to $\mathbf{y} \in \mathbb{R}^{O \times C}$

Return $\mathbf{y}$ ▷ Return de-normalized output

Algorithm 3 De-stationary Attention.

Require: Queries $\mathbf{Q}' \in \mathbb{R}^{S \times d_k}$; Keys $\mathbf{K}' \in \mathbb{R}^{S \times d_k}$; Values $\mathbf{V}' \in \mathbb{R}^{S \times d_k}$; De-stationary factors $\tau \in \mathbb{R}^{+}, \mathbf{\Delta} \in \mathbb{R}^{S \times 1}$; Input Length S; Feature dimension d_k.

$\texttt{Output} = \texttt{Softmax}\left(\left(\tau\, \mathbf{Q}'\mathbf{K}'^{\top} + \texttt{Repeat}(\mathbf{\Delta}, \texttt{dim=1})\right)/\sqrt{d_k}\right)\mathbf{V}'$ ▷ rescaling by τ and $\mathbf{\Delta}$

Return $\texttt{Output}$ ▷ Return de-stationary attention output

Algorithm 4 Non-stationary Transformers - Overall Architecture.

Require: Input past time series $\mathbf{x} \in \mathbb{R}^{S \times C}$; Input Length S; Predict length O; Variables number C; Feature dimension d_k; Encoder layers number N; Decoder layers number M. Technically, we set d_k as 512, N as 2, M as 1.

$\mathbf{x}', \mu_{\mathbf{x}}, \sigma_{\mathbf{x}} = \texttt{Normaliztion}(\mathbf{x})$ ▷ $\mathbf{x}' \in \mathbb{R}^{S \times C}, \mu_{\mathbf{x}} \in \mathbb{R}^{1 \times C}, \sigma_{\mathbf{x}} \in \mathbb{R}^{1 \times C}$

$\log \tau, \mathbf{\Delta} = \texttt{MLP}(\mathbf{x}, \mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$ ▷ $\tau \in \mathbb{R}^{+}, \mathbf{\Delta} \in \mathbb{R}^{S \times 1}$

$\mathbf{x}'_{\text{enc}}, \mathbf{x}'_{\text{dec}} = \mathbf{x}', \texttt{ConCat}\big(\mathbf{x}'_{\frac{S}{2}:S}, \texttt{Zeros}(O, C)\big)$ ▷ $\mathbf{x}'_{\text{enc}} \in \mathbb{R}^{S \times C}, \mathbf{x}'_{\text{dec}} \in \mathbb{R}^{(\frac{S}{2}+O) \times C}$

$\mathbf{x}^{0\prime}_{\text{enc}} = \texttt{Embed}(\mathbf{x}'_{\text{enc}})$ ▷ $\mathbf{x}^{0\prime}_{\text{enc}} \in \mathbb{R}^{S \times d_k}$

for l **in** $\{1, \cdots, N\}$**:** ▷ Non-stationary Encoder

  $\mathbf{x}^{l-1\prime}_{\text{enc}} = \texttt{LayerNorm}\big(\mathbf{x}^{l-1\prime}_{\text{enc}} + \texttt{Attn}(\mathbf{x}^{l-1\prime}_{\text{enc}}, \tau, \mathbf{\Delta})\big)$ ▷ $\mathbf{x}^{l-1\prime}_{\text{enc}} \in \mathbb{R}^{S \times d_k}$

  $\mathbf{x}^{l\prime}_{\text{enc}} = \texttt{LayerNorm}\big(\mathbf{x}^{l-1\prime}_{\text{enc}} + \texttt{FFN}(\mathbf{x}^{l-1\prime}_{\text{enc}})\big)$ ▷ $\mathbf{x}^{l\prime}_{\text{enc}} \in \mathbb{R}^{S \times d_k}$

End for

$\mathbf{x}^{0\prime}_{\text{dec}} = \texttt{Embed}(\mathbf{x}'_{\text{dec}})$ ▷ $\mathbf{x}^{0\prime}_{\text{dec}} \in \mathbb{R}^{(\frac{S}{2}+O) \times d_k}$

for l **in** $\{1, \cdots, M\}$**:** ▷ Non-stationary Decoder

  $\mathbf{x}^{l-1\prime}_{\text{dec}} = \texttt{LayerNorm}\big(\mathbf{x}^{l-1\prime}_{\text{dec}} + \texttt{Attn}(\mathbf{x}^{l-1\prime}_{\text{dec}}, \tau, \mathbf{\Delta} = 0)\big)$ ▷ $\mathbf{x}^{l-1\prime}_{\text{dec}} \in \mathbb{R}^{(\frac{S}{2}+O) \times d_k}$

  $\mathbf{x}^{l-1\prime}_{\text{dec}} = \texttt{LayerNorm}\big(\mathbf{x}^{l-1\prime}_{\text{dec}} + \texttt{Attn}(\mathbf{x}^{l-1\prime}_{\text{dec}}, \mathbf{x}^{N\prime}_{\text{enc}}, \tau, \mathbf{\Delta})\big)$ ▷ $\mathbf{x}^{l-1\prime}_{\text{dec}} \in \mathbb{R}^{(\frac{S}{2}+O) \times d_k}$

  $\mathbf{x}^{l\prime}_{\text{dec}} = \texttt{LayerNorm}\big(\mathbf{x}^{l-1\prime}_{\text{dec}} + \texttt{FFN}(\mathbf{x}^{l-1\prime}_{\text{dec}})\big)$ ▷ $\mathbf{x}^{l\prime}_{\text{dec}} \in \mathbb{R}^{(\frac{S}{2}+O) \times d_k}$

End for

$\mathbf{y}' = \texttt{MLP}(\mathbf{x}^{M\prime}_{\text{dec}})_{\frac{S}{2}:\frac{S}{2}+O}$ ▷ $\mathbf{y}' \in \mathbb{R}^{O \times d_k}$

$\mathbf{y} = \texttt{De-normaliztion}(\mathbf{y}', \mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$ ▷ $\mathbf{y} \in \mathbb{R}^{O \times d_k}$

Return $\mathbf{y}$ ▷ Return the prediction results

F Broader Impact

F.1 Impact on Real-world Applications

We focus on real-world time series forecasting, which is challenging for Transformers because of data non-stationarity. Our method goes beyond previous studies that only stationarize the time series. We fully utilize the predictive capability of attention mechanism that captures essential temporal dependencies associated with inherent non-stationarity. Our proposed method achieves state-of-the-art performance in five real-world applications, which makes it more promising for Transformers to tackle real-world forecasting applications, and helps our society make better decisions and prevent risks in advance for various fields. And our paper mainly focuses on scientific research and has no obvious negative social impact.

F.2 Impact on Future Research

In this paper, we analyze the generalization difficulty of Transformers in distribution-varying time series forecasting. We propose a general framework to fulfill the potential of Transforms constrained by data non-stationary. Our work introduces an essential and promising direction to improve forecasting performance: to increase the stationarity of time series towards better predictability and mitigate the over-stationarization problem for the predictive capability of deep models simultaneously. The remarkable generality and effectiveness of the proposed framework can be instructive for future research.

G Limitation

Our De-stationary Attention is deduced by analyzing the vanilla Self-Attention, which may not be the optimal solution for advanced attention mechanisms. There also remains room for re-incorporating non-stationarity on other classical stationarization methods, like differencing and quantile. Besides, the proposed framework is currently limited to the Transformer-based models, while the over-stationarization problem can appear on any deep time forecasting models if using stationarization methods inappropriately. Therefore, a more model-agnostic solution for the over-stationarization problem will be our exploring direction.