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

Anonymous Author(s)
Affiliation
Address
email

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 reduce 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 over-stationarization, De-stationary Attention is devised to recover the intrinsic non-stationary information into temporal dependencies by approximating distinguishable attentions learned from unstationarized series. Our Non-stationary Transformers framework consistently boosts mainstream Transformers by a large margin, which reduces 49.43% MSE on Transformer, 47.34% on Informer, and 46.89% on Reformer, making them the state-of-the-art in time series forecasting.

1 Introduction

Time series forecasting has become increasingly ubiquitous in real-world applications, such as weather forecasting, energy consumption planning, estimation of future illness cases, and financial risk assessment. Recently, Transformers [32] have achieved progressive breakthrough on extensive areas [11, 12, 10, 22]. Especially in time series forecasting, credited to their stacked structure and the capability of attention mechanisms, Transformers can naturally capture the temporal dependencies among time points [36, 17, 20, 34], 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 are characterized by the continuous change of statistical properties and joint distribution over time [6, 14]. From the aspect of data, non-stationarity makes the time series less predictable [6, 14]. From the view of deep learning, it is a fundamental but challenging problem to make deep models generalize well on a varying distribution [26, 19, 5]. Therefore, it is generally acknowledged to pre-process time series by elimination of non-stationarity, and the related research has become a hot topic [24, 27, 15]. These studies attenuate the non-stationarity of real-world time series in order to obtain more predictable series and provide relatively stable data distribution for deep models.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

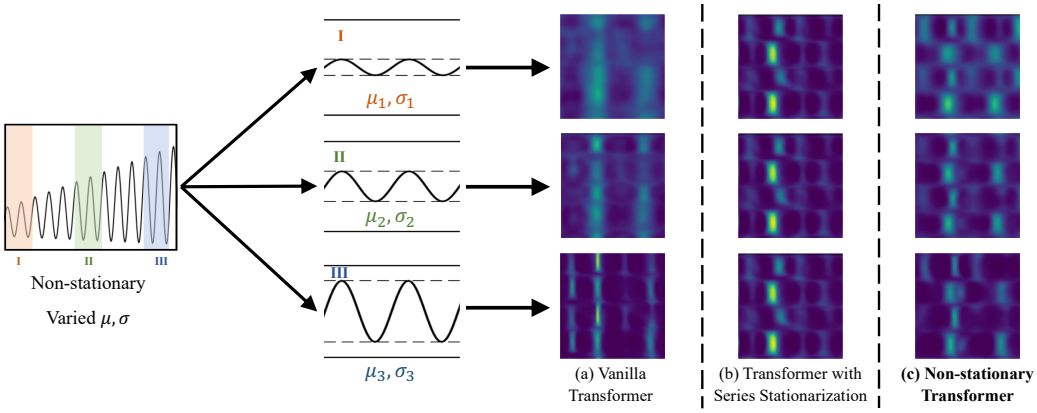

Figure 1: Visualization of learned attentions for different series with varied mean μ and standard deviation σ. (a) is from the vanilla Transformer [32] trained on raw series. (b) is from the Transformer trained on stationarized series, which presents similar attentions. (c) is from Non-stationary Transformers, which involves De-stationary Attention to avoid over-stationarization.

However, non-stationarity is the constitutional property of real-world time series that can be entangled with essential temporal dependencies for forecasting. Experimentally, we observe that training on the stationarized series will undermine the distinction of attentions learned by Transformers. As shown in Figure 1(b), even though input series are clearly distinct, Transformers [32] trained on the stationarized series still tends to generate quite similar attentions, and the reason for that is what we call *over-stationarization*. The unexpected side-effect will make 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 increase the series 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 [32] and its efficient variants [17, 36, 34] 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 statistics of each series without extra parameters. And De-stationary Attention approximates the attention of unstationarized data and restitutes 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 on base models. 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 [33, 35, 23, 29, 30] 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 [32] 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 [36] extends Self-Attention with KL-divergence criterion to select dominant queries. Reformer [17] 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 [34] fuses the decomposition blocks into a canonical structure and develops Auto-Correlation to discover series-wise connections. Pyraformer [21] designs pyramid attention module (PAM) to capture temporal dependencies with different hierarchies. Other deep but Transformer-free models also achieve remarkable performance. N-BEATS [25] 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 the essential property of time series [6, 14]. 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 the key to the predictability of time series [6, 14], 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 [24] applies z-score normalization for each series fragment by global statistics of a sampled set. DAIN [27] employs a nonlinear neural network to adaptively stationarize time series with observed training distribution. RevIN [15] introduces a two-stage instance normalization [31] 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 the key to time series predictability. However, previous "wiping off non-stationarity" designs 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. Non-stationary Transformers involves two complementary parts: Series Stationarization to enhance the stationarity of time series for better predictability, and De-stationary Attention to re-incorporate non-stationary information of raw series to prevent over-stationarization. Empowered by these designs, Non-stationary Transformers can improve data predictability and 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 [15] applies instance normalization with learnable affine parameters to each input and restores the statistics to the corresponding output, which makes each series follow a unified 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.

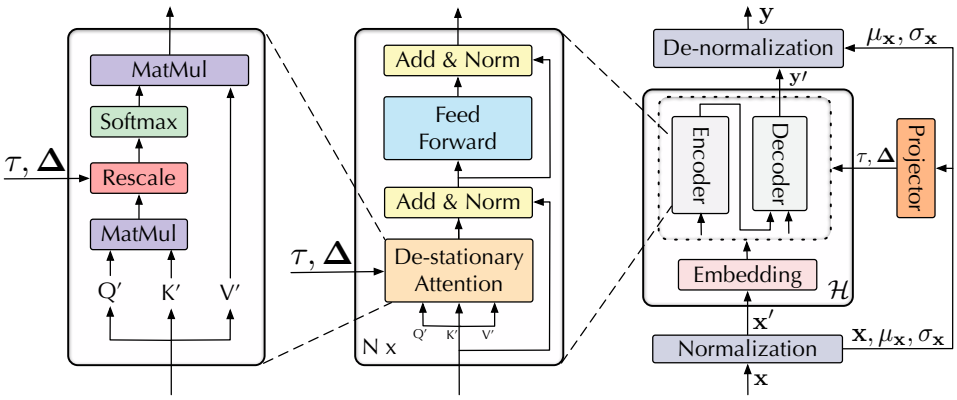

Figure 2: Non-stationary Transformers. Series Stationarization is adopted as a wrapper on the base model to normalize each incoming series and de-normalize the output. De-stationary Attention replaces the original Attention mechanism to approximate attention learned from unstationarized series, which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

Normalization module To guarantee the stationarity of inputs, we conduct normalization on the temporal dimension by a sliding window over time. For each input series $\mathbf{x} = [x_1, x_2, ..., x_S]^{\mathrm{T}} \in \mathbb{R}^{S \times C}$, we transform it into standard Gaussian distribution by translation and scaling operations and obtain $\mathbf{x}' = [x'_1, x'_2, ..., x'_S]^{\mathrm{T}} \in \mathbb{R}^{S \times C}$, 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 Hadamard Product. Note that Normalization module eliminates the statistical discrepancy among each input time series, making the model insusceptible to varied statistics.

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 $\mathbf{y}' = [y'_1, y'_2, ..., y'_O]^{\mathrm{T}} \in \mathbb{R}^{O \times C}$ with $\sigma_{\mathbf{x}}$ and $\mu_{\mathbf{x}}$ and obtain $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_O]^{\mathrm{T}}$ 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 that follow a relatively stable distribution and become 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 can not be fully recovered only by De-normalization. For an instance, Series Stationarization can cause 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 fail 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, since non-stationary time series are fragmented and normalized into each input series following the same distribution, 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 trivial attention 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 constitutional 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 [32]:

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

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[1] and f is conducted separately on each time point, that is, each query token in $\mathbf{Q} = [q_1, q_2, ..., q_S]^{\mathrm{T}}$ can be calculated as $q_i = f(x_i)$ with respect to the input series $\mathbf{x} = [x_1, x_2, \cdots, x_S]^{\mathrm{T}}$. 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 $\mathbf{x}' = (\mathbf{x} - \mathbf{1}\mu_{\mathbf{x}}^{\mathrm{T}})/\sigma_{\mathbf{x}}$, 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 $\mathbf{Q}' = [f(x_1'), ..., f(x_S')]^{\mathrm{T}} = (\mathbf{Q} - \mathbf{1}\mu_{\mathbf{Q}}^{\mathrm{T}})/\sigma_{\mathbf{x}}$, where $\mu_{\mathbf{Q}} \in \mathbb{R}^{d_k \times 1}$ is the mean of $\mathbf{Q}$ along the temporal dimension (See supplementary materials 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 $\mathbf{Q}\mathbf{K}^{\mathrm{T}}/\sqrt{d_k}$, while now the attention is calculated based on $\mathbf{Q}', \mathbf{K}'$:

$$\begin{aligned} \mathbf{Q}'\mathbf{K}'^{\mathrm{T}} &= \frac{1}{\sigma_{\mathbf{x}}^2}\left(\mathbf{Q}\mathbf{K}^{\mathrm{T}} - \mathbf{1}(\mu_{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}) - (\mathbf{Q}\mu_{\mathbf{K}})\mathbf{1}^{\mathrm{T}} + \mathbf{1}(\mu_{\mathbf{Q}}^{\mathrm{T}}\mu_{\mathbf{K}})\mathbf{1}^{\mathrm{T}}\right), \\ \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right) &= \mathrm{Softmax}\left(\frac{\sigma_{\mathbf{x}}^2\,\mathbf{Q}'\mathbf{K}'^{\mathrm{T}} + \mathbf{1}(\mu_{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}) + (\mathbf{Q}\mu_{\mathbf{K}})\mathbf{1}^{\mathrm{T}} - \mathbf{1}(\mu_{\mathbf{Q}}^{\mathrm{T}}\mu_{\mathbf{K}})\mathbf{1}^{\mathrm{T}}}{\sqrt{d_k}}\right). \end{aligned} \tag{4}$$

We find that $\mathbf{Q}\mu_{\mathbf{K}} \in \mathbb{R}^{S \times 1}$ and $\mu_{\mathbf{Q}}^{\mathrm{T}}\mu_{\mathbf{K}} \in \mathbb{R}$, and they are repeatedly operated on each column and element of $\sigma_{\mathbf{x}}^2\mathbf{Q}'\mathbf{K}'^{\mathrm{T}} \in \mathbb{R}^{S \times S}$ respectively. Since $\mathrm{Softmax}(\cdot)$ is invariant to the same translation on the row dimension of input, we have the following equation:

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right) = \mathrm{Softmax}\left(\frac{\sigma_{\mathbf{x}}^2\,\mathbf{Q}'\mathbf{K}'^{\mathrm{T}} + \mathbf{1}\mu_{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right). \tag{5}$$

The Equation 5 deduces a direct expression of the attention $\mathrm{Softmax}\left(\mathbf{Q}\mathbf{K}^{\mathrm{T}}/\sqrt{d_k}\right)$ 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 the derivation in 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*. Considering the deep model scenario under the real case, de-stationary factors should be disentangled from the statistics of unstationarized $\mathbf{x}, \mathbf{Q}$ and $\mathbf{K}$. As we can only discover limited non-stationary information from current $\mathbf{Q}', \mathbf{K}'$, the unique and reasonable source to restitute 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:

$$\begin{aligned} \log\tau &= \mathrm{MLP}(\sigma_{\mathbf{x}}, \mathbf{x}), \mathbf{\Delta} = \mathrm{MLP}(\mu_{\mathbf{x}}, \mathbf{x}), \\ \mathrm{Attn}(\mathbf{Q}', \mathbf{K}', \mathbf{V}', \tau, \mathbf{\Delta}) &= \mathrm{Softmax}\left(\frac{\tau\,\mathbf{Q}'\mathbf{K}'^{\mathrm{T}} + \mathbf{1}\mathbf{\Delta}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V}', \end{aligned} \tag{6}$$

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 $\mathbf{Q}', \mathbf{K}'$ 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.

[1]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.

Overall architecture Following the prior use of Transformers [36, 34] 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 [32], 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 [17, 36, 34], we transform the terms inside $\mathrm{Softmax}(\cdot)$ with the de-stationary factors τ, $\mathbf{\Delta}$ to re-integrate the non-stationary information (See supplementary materials 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. See supplementary materials for the code.

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** [36] 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** [18] 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. Table 1 summarizes overall statistics of the datasets. We also record the Augmented Dick-Fuller test statistic [13] of each dataset and list them in ascending order 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 3:1:1 for the ETT dataset and 7:2:2 for others.

Table 1: Summary of dataset statistics. We organize datasets in ascending order of stationarity.

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

* A smaller ADF test statistic indicates a more stationary dataset.

Baselines We evaluate the vanilla Transformer [32] equipped by the Non-stationary Tranformers 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 [34], Pyraformer [21], Informer [36], LogTrans [20], Reformer [17] and LSTNet [18]. For univariate forecasting, we include seven competitive baselines: N-HiTS [9], N-BEATS [25], Autoformer [34], Pyraformer [21], Informer [36], Reformer [17] and ARIMA [7]. In addition, we adopt the proposed framework on both the canonical and efficient variants of Transformers: Transformer [32], Informer [36], Reformer [17] and Autoformer [34] to validate the generality of our framework.

Implementation details All the experiments are implemented in PyTorch [28] and conducted for three runs on a single NVIDIA TITAN V 12GB GPU. Each model is trained by ADAM [16] 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 report the test MSE/MAE under different prediction lengths as the performance metric. A lower MSE/MAE indicates better performance of time series forecasting.

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. With 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 the supplementary materials.

Models		**Ours**		Autoformer [34]		Pyraformer [21]		Informer [36]		LogTrans [20]		Reformer [17]		LSTNet [18]	
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

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

Models		**Ours**		N-HiTS [9]		N-BEATS [25]		Autoformer [34]		Pyraformer [21]		Informer [36]		Reformer [17]		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 our proposed framework to Transformers. We report the averaged MSE/MAE of all prediction lengths (stated in Table 2) and the relative MSE reduction ratios (Promotion) by our framework. Complete results can be found in the supplementary materials.

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%	

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 supplementary materials for details), and thereby their computational complexities can be preserved. It validates that Non-stationary Transformers 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.

4.2 Model Analysis

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 solely Series Stationarization, and 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 stationary but uneventful series and neglect the nature of non-stationary real-world data. With the aid of De-stationary Attention, the model gives concern back to the constitutional 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.

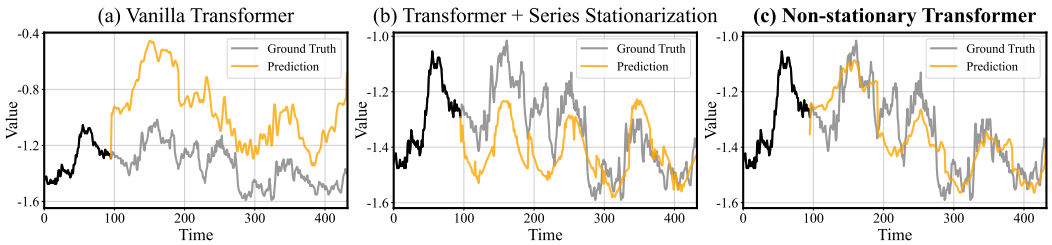

Figure 3: Visualization of ETTm2 predictions given by different models.

Quantitative performance In addition to the above case study, we also provide quantitative forecasting performance comparison with stationarization methods: a deep method RevIN [15] 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 solely 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.

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 the supplementary materials.

Base Models	Transformer						Reformer					
Methods	+ RevIN [15]		+ Series Stationarization		+ **Ours**		+ RevIN [15]		+ 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**

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 stationarity with the ground truth (Figure 4). While models solely equipped with stationarization methods tend to output series with high stationarity, the results assisted by De-stationary Attention are close to the actual value (relative stationarity $\in [97\%, 103\%]$). Besides, as the stationarity of time series increases, the over-stationarization problem becomes more significant. The huge stationarity discrepancy can account for the inferior performance of Transformer with stationarization only. And it also demonstrates that De-stationary Attention as an internal renovation of Transformers can alleviate over-stationarization.

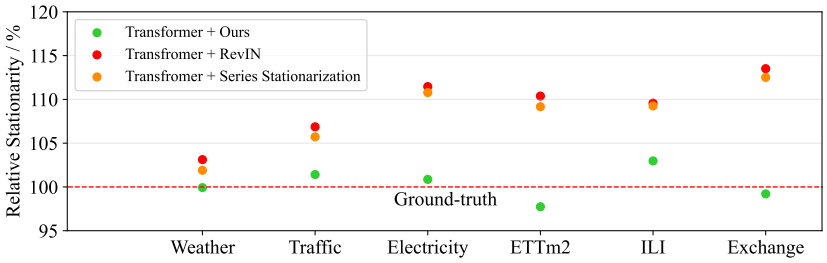

Figure 4: Relative stationarity of predictions. For all six datasets, we calculate the relative stationarity, which is the ratio of the predictions stationarity (measured by ADF test statistic) to the ground truth.

5 Conclusion

This paper addresses time series forecasting from the perspective of stationarity. Unlike previous studies that solely reduce the non-stationarity of original series, we propose Non-stationary Transformers to increase the predictability of time series and fully exploit the predictive capability of the model simultaneously. Our proposed framework consists of two complementary mechanisms, where Series Stationarization is an efficient implementation to stationarize time series and De-stationary Attention renovates the internal mechanism to recover the constitutional non-stationarity of real-world series. The introduction of our framework boosts the forecasting performance of mainstream Transformers to the state-of-the-art on six real-world benchmarks. The impressive generality and performance of the proposed framework can be interesting to future research for time series forecasting.

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

[12] 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.

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

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

[15] 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.

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

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

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

[28] 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.

[29] 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.

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

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

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

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

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

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

[36] 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.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

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 of main text.
   (b) Did you describe the limitations of your work? [Yes] See Section 7 of supplementary materials.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6 of supplementary materials.
   (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 Section 3.2 of main text and Section 1 of supplementary materials.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3.2 of main text and Section 1 of supplementary materials.

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 have provided the code in supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 of supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 3 of supplementary materials.
   (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 of main text.
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] See Section 4 of main text.
   (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] See Section 4 of main text.
   (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]