applied sciences

MDPI

Article

Addressing the Non-Stationarity and Complexity of Time Series Data for Long-Term Forecasts

Ranjai Baidya and Sang-Woong Lee *

Pattern Recognition and Machine Learning Lab, Department of AI·Software, Gachon University, Seongnam 13120, Republic of Korea; ranjai@gachon.ac.kr
* Correspondence: slee@gachon.ac.kr

Abstract: Real-life time series datasets exhibit complications that hinder the study of time series forecasting (TSF). These datasets inherently exhibit non-stationarity as their distributions vary over time. Furthermore, the intricate inter- and intra-series relationships among data points pose challenges for modeling. Many existing TSF models overlook one or both of these issues, resulting in inaccurate forecasts. This study proposes a novel TSF model designed to address the challenges posed by real-life data, delivering accurate forecasts in both multivariate and univariate settings. First, we propose methods termed "weak-stationarizing" and "non-stationarity restoring" to mitigate distributional shift. These methods enable the removal and restoration of non-stationary components from individual data points as needed. Second, we utilize the spectral decomposition of weak-stationary time series to extract informative features for forecasting. To learn features from the spectral decomposition of weak-stationary time series, we exploit a mixer architecture to find inter- and intra-series dependencies from the unraveled representation of the overall time series. To ensure the efficacy of our model, we conduct comparative evaluations against state-of-the-art models using six real-world datasets spanning diverse fields. Across each dataset, our model consistently outperforms or yields comparable results to existing models.

Keywords: time series forecasting; non-stationary; weak-stationary; multivariate; univariate; spectral decomposition; ConvMixer

Citation: Baidya, R.; Lee, S.-W. Addressing the Non-Stationarity and Complexity of Time Series Data for Long-Term Forecasts. *Appl. Sci.* **2024**, *14*, 4436. https://doi.org/10.3390/app14114436

Academic Editors: Hyeonjoon Moon and Lien Minh Dang

Received: 22 April 2024
Revised: 20 May 2024
Accepted: 21 May 2024
Published: 23 May 2024

1. Introduction

Time series forecasting (TSF) plays a crucial role in various real-life applications, including transportation [1], health care [2,3], and energy management [4]. In such TSF-dependent fields, accurate forecasts over extended horizons contribute to long-term planning and resilience to future challenges. For instance, in sales forecasting, historical sales data can inform inventory management to anticipate future demands.

Interest in accurate TSF methods has been established. Traditional approaches such as Kalman filters [5], hidden Markov models [6], and statistical models such as auto regressive integrated moving average (ARIMA) [7] have been commonly used. However, these models often require external input and struggle with long-sequence time series forecasting (LSTF). Similarly, a variety of deep learning architectures have emerged for LSTF. Temporal convolutional networks (TCNs) [8,9] and their variants have demonstrated success in LSTF [10–13], achieving a wide receptive field with reduced computational complexity through dilated convolutions. However, TCNs are limited in their ability to capture temporal relationships. Recently, transformer architectures have garnered attention for their capacity to model one-to-one relationships in sequential data [14], proving effective in LSTF [4,15]. Nonetheless, despite their ability to model one-to-one relationships, transformers require massive computational resources and they also appear to struggle in capturing the inter- and intra-series patterns in time series data.

Despite the abundance of architectures designed to address TSF and LSTF, many of these models overlook key challenges inherent in real-life time series data. First, a significant portion of time series data are non-stationary, with statistical properties such as mean and variance changing over time [16]. Second, complex patterns exist within inter- and intra-series relationships of these time series data, posing difficulties for modeling [17]. Overlooking these issues can lead to inaccurate forecasts [17,18]. Recent efforts using modern deep learning techniques have begun to address these challenges. For instance, an adaptive recurrent neural network model was suggested to handle time series non-stationarity [19], while another work suggested a simple normalization technique to mitigate distribution shifts in time series. This approach involved normalization followed by the restoration of the distribution to recover withdrawn information [18]. Additionally, techniques such as the disentanglement of time series into trends and seasonality have been beneficial for improving forecasts [20]. Moreover, learning features of time series across multiple temporal scales has shown promise [21].

In our approach, we draw inspiration from traditional methods to tackle non-stationary time series and unravel complex patterns. Initially, we consider the widely used statistical ARIMA model, which uses weighted averages of past data point differences to predict future values [7]. We suggest using a separate block to compute these differences, ensuring only past data points with similar phase positions corresponding to each input data point are used. Additionally, prior to forecastings, we restore non-stationarity information to enhance forecast accuracy. Next, we perform spectral decomposition of the time series data to extract suitable features before learning the features for forecasts. Spectral decomposition, commonly employed to decompose input into its frequency components for detailed analysis, is leveraged to simplify intricate time series patterns into spectral components, enabling the learning of hidden features for long-term forecasts. Finally, inspired by the success of mixer architectures such as multilayer perceptron mixer [22] and ConvMixer [23] for mixing spatial and channel-wise data points, we deploy ConvMixer to capture the intra- and inter-series dependencies from the spectral representation of the multivariate time series signals. The choice of ConvMixer is motivated by its effectiveness in challenging transformers in computer vision tasks, as well as its unambiguous and consistent architecture.

This paper introduces a deep learning architecture tailored for LSTF tasks, specifically addressing the challenges posed by non-stationary time series and aiming to unravel the inter-series and intra-series dependencies. The proposed architecture comprises two novel components, a "weak-stationarizing" block and "non-stationarity restoring" block, designed to handle non-stationary input time series. The "weak-stationarizing" block operates by duplicating the original time series and aligning each data point to similar phase positions as the original series. Subsequently, the resulting time series undergoes differencing to render it weak-stationary. This block utilizes the power spectral density (PSD) of the time series at individual frequencies to determine the appropriate number of roll backs needed before differencing, aligning the time series based on the dominant frequency. The weak-stationary time series is then transformed using fast Fourier transform (FFT) [24] to obtain its spectral decomposition, which serves as the basis for feature learning using the ConvMixer architecture for forecasting. To account for the loss of information due to the "weak-stationarizing" block, the "non-stationarity restoring" block is employed to restore the non-stationary properties of the time series. The resulting architecture, formed integrating these methods, surpasses previous state-of-the-art (SOTA) results across six benchmark datasets. Our contributions are summarized as follows:

- We propose a generalized deep learning model capable of addressing both univariate and multivariate forecasting problems.
- We present a novel "weak-stationarizing" block, which utilizes PSD values at different frequency levels to determine the appropriate number of rollbacks before differencing, effectively rendering the time series weak-stationary. The "non-stationarity restoring" black is employed to restore non-stationarity, ensuring information preservation for

final predictions. Ablation studies demonstrate the significant performance improvement achieved with these blocks.
- We modify the ConvMixer architecture for use in TSF, which operates on the spectral decompositions of the time series to produce high-quality forecasts.
- The proposed overall architecture obtains an average of 21% and up to 64.6% of relative improvements compared to the previous state-of-the-art methods on six real-world datasets, ETT, electricity, traffic, weather, ILI, and exchange, in various settings.

1.1. Related Works

TSF has been extensively studied, with traditional methods such as hidden Markov models [6], Kalman filters [5], and statistical models such as ARIMA [25], exponentially weighted moving averages [26,27], and vector auto regressors [28] demonstrating notable performance. In the field of deep learning, RNNs were initially prominent for TSF due to their effectiveness in modeling sequential data [29–32]. Subsequently, TCN architectures gained a decent amount of popularity [8–13], with TCNs and RNNs often used in conjunction with graph neural networks (GNNs) to capture both spatial patterns and temporal patterns [13,33–36]. Transformer architectures have emerged as dominant players in sequential modeling tasks, largely replacing RNNs [14]. The success of transformers is attributed to their self-attention mechanism; however, the quadratic computation and memory complexity of self-attention pose challenges for handling long sequences. Consequently, recent efforts in transformer-based LSTF models have focused on developing more efficient architectures, often by proposing sparser query matrices for computing self-attention [4,15]. Incorporating classical concepts alongside modern deep learning concepts has performed reasonably well [20,21]. For instance, Autoformer [20] decomposes the original time series into seasonality and trend components, extracting trend information through multiple decomposition steps using average pooling and treating the difference between the original signal and trend as seasonality. Additionally, Autoformer uses an autocorrelation block to extract the dependencies [20]. Whereas SCINet [21] leverages multiple resolution analysis in deep learning, employing a unique downsampling technique alongside convolutions and an interaction block [21].

1.1.1. Distribution Shift and Non-Stationary Time Series

Domain adaptation (DA) [37–41] and domain generalization (DG) [42–45] address distribution shifts in machine learning when predefining the domain is feasible. DA pertains to scenarios where training and test sets' distributions differ, while DG involves training with multiple domain sources. However, in non-stationary time series data, domain shift occurs gradually over time, making predefining domain specification impractical. Recently, the adaptive RNN architecture was proposed to handle distribution shifts in time series [19]. It splits the training data into periods to adapt the model for distribution shifts. In contrast reversible instance normalization [18], compatible with various forecasting models, employs normalization along with additional learnable parameters to obtain forecasts and restore the prior distribution.

1.1.2. Spectral Decomposition

Spectral decomposition of time series data finds applications in speech and music recognition [46,47], machine health and mechanical vibration monitoring [48], river and oceanographic tide modeling [49], and power demand prediction [50]. In TSF, spectral decomposition is gaining traction. A new decomposition method based on Koopman theory [51] and comparable to Fourier transform [24,52,53] has been suggested for long-term forecasts [54]. Furthermore, methods such as StemGNN utilize graph Fourier transform and discrete Fourier transform to exploit spectral dependencies for intra- and inter-series correlations in time series [17]. However, StemGNN's complexity, combining gated RNN, self-attention, and GNN, may hinder its application in real-world datasets. Moreover, StemGNN has not been evaluated in LSTF settings.

2. Methodology

The proposed framework addresses both univariate and multivariate TSF problems, focusing on the non-stationarity of time series and simplifying time series data. This section outlines the concepts of "spectral decomposition", the blocks for addressing non-stationarity, ("weak-stationarizing" block and "non-stationarity restoring" block), and the ConvMixer architecture, and provides an overview of the architecture.

2.1. Problem Formulation

LSTF can be categorized into two main problem settings: multivariate TSF and univariate TSF. In the multivariate setting, we are given N different time series $X^{1:N} = [X^1, X^2, X^3, \ldots\ldots, X^N]$, which are interdependent. If the current time is denoted as t, the values of the N time series at time t are represented as $X_t^{1:N} = [X_t^1, X_t^2, X_t^3, \ldots\ldots, X_t^N]$. Given a look-back window of length T aiming to forecast future data points up to a horizon length of T', the input time series can be expressed as $X_{t-T+1:t}^{1:N} = [X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t]^{1:N}$, and the forecasts and actual values can be expressed as $X_{t+1:t+T'}^{1:N\ \prime} = [X'_{t+1}, X'_{t+2}, X'_{t+3}, \ldots, X'_{t+T'}]^{1:N}$ and $X_{t+1:t+T'}^{N} = [X_{t+1}, X_{t+2}, X_{t+3}, \ldots, X_{t+T'}]^{1:N}$, respectively.

The univariate setting is a case where the number of dependent time series, N, is 1. In this case, the input time series can be denoted as $X_{t-T+1:t} = [X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t]$ and the forecasts and actual future values can be represented as $X'_{t+1:t+T'} = [X'_{t+1}, X'_{t+2}, X'_{t+3}, \ldots, X'_{t+T'}]$ and $X_{t+1:t+T'} = [X_{t+1}, X_{t+2}, X_{t+3}, \ldots, X_{t+T'}]$, respectively.

2.2. Spectral Decomposition

Spectral analysis is widely used for time series analysis, enabling the identification of frequency components present in the original signal [16,55]. Spectral decomposition involves breaking down the original signal into constituent components for further analysis. According to spectral decomposition principles, a time series X of length T starting at the time t can be decomposed into a linear combination of sines and cosines with different frequencies f.

$$X_{t:t+T} = \sum_f A(f)cos(2\pi f(t:t+T)) + B(f)sin(2\pi f(t:t+T)) \quad (1)$$

Here, $A(f)$ and $B(f)$ are the amplitudes of the sine and cosine components at frequency f. Traditionally, while performing forecasting using spectral decomposition, the extracted spectral components are extensively repeated to the desired horizon length, and then, merged. We adapt the concept of spectral decomposition to clarify the frequency components upon which the input signal is dependent. We use DFFT [24] to obtain the spectral representation of the signal, which is used to learn relevant features for LSTF. Since, the output of DFFT is imaginary, its real and imaginary portions are treated as separate channels.

2.3. Weak-Stationarizing Block and Non-Stationarity Restoring Block

Spectral analysis requires the input time series to be weakly stationary [16]. meaning that the joint moments of the first- and second-order across equal-length segments are consistent. While achieving perfect stationarity is difficult, our objective is to reduce the non-stationary properties of the time series.

ARIMA models use differencing to obtain the weak-stationary signals [7]. Building on this concept, the "weak-stationarizing" block is designed to yield a weak-stationary signal with a single differencing step. This involves aligning two copies of the input, where each data point in the copies lies in similar phase positions but varies by a single period of the dominant frequency component in the time series. One copy remains unchanged, while the other is rolled back by a value we term as the "optimum roll back value" (ORV), representing a single period of the dominant frequency.

The ORV is determined using the PSD of the time series [56]. The PSD is calculated by multiplying the FFT of the input with its complex conjugate, with the dominant frequency components having the highest PSD values. Since there are multiple dependent time series, the ORV is chosen as the most frequent position with the highest PSD values across the different dependent time series. Then, one copy of the input signal is rolled back by the ORV and subtracted from the original input signal. However, since the ORV may not be optimum for all the dependent time series, two additional learnable parameters are used as weights and biases to adjust the effect of the differencing for each input series. The learnable weights and biases are single dimensional arrays of length N, where, as in previous sections, this represents the number of dependent time series. The output of the "weak-stationarizing" block is a weak-stationary output (WSO).

The PSD can be obtained as

$$PSD = FFT([X]^{1:N}) * Conj(FFT([X]^{1:N})) \tag{2}$$

where *Conj*() gives the complex conjugate value of the input. Using *PSD*, the *ORV* is then obtained as

$$ORV = Mode(Position(Max(PSD))) \tag{3}$$

Finally, the *WSO* is given by

$$WSO = \omega^{1:N} * ([X]^{1:N} - RollBack([X]^{1:N}, ORV)) + \beta^{1:N} \tag{4}$$

where $\omega^{1:N} = [\omega^1, \omega^2, \omega^3, \ldots, \omega^N]$ and $\beta^{1:N} = [\beta^1, \beta^2, \beta^3, \ldots, \beta^N]$ are the weights and biases for 'N' dependent time series.

Transforming the time series using the "weak-stationarizing" block causes the loss of important statistical information. To compensate for this loss of information the "non-stationarity restoring" block is introduced before the projection layer to restore the necessary details. The output of this block is a non-stationary output (NSO), where the rolled back portion is denoted as non-stationary information representation (NSIR), i.e., NSIR = $RollBack([X]^{1:N}, ORV)$, for simplicity. This output then goes through the ConvMixer layers. If F represents the output of the ConvMixer architecture (discussed in Section 2.4), then, the process in the "non-stationarity restoring" block can be represented as

$$NSO = \frac{F(WSO) - \beta^{1:N}}{\omega^{1:N}} + NSIR \tag{5}$$

The detailed architectures of the "weak-stationarizing" block and the "non-stationarity restoring" block can be seen in the lower half of Figure 1.

2.4. ConvMixer

Extraction of prominent features is critical in deep learning tasks. Leveraging the success of the ConvMixer architecture in computer vision applications, we adopt it with some modifications to suit our purpose. The fundamental concept of the mixer architecture is to shuffle the data both spatially and channel-wise using depthwise and pointwise convolutions, respectively. Additionally, ConvMixer maintains the input structure throughout the mixer layers.

Unlike the original ConvMixer, which operates on image data and includes an embedding layer to deal with image patches, we omit the embedding layer as our data can be directly processed in its existing form. However, as mentioned in Section 2.1, for obtaining spectral features the real and imaginary parts of the output of DFFT are treated as separate channels. Hence, the number of input channels becomes twice the number of dependent time series N. Additionally, we use 1D convolutions due to the nature of our data. Furthermore, we use layer normalization, contrary to the original ConvMixer architecture. For input X, the pointwise convolution (PW) is represented as

$$PW = LayerNorm(GELU(1D\_PointWiseConv(X)) \tag{6}$$

and the depthwise convolution (DW) is represented as

$$DW = LayerNorm(GELU(1D\_DepthWiseConv(X)) \tag{7}$$

Then, the function F representing the working of ConvMixer architecture can be represented as

$$F = [PW(Residual(DW(\mathfrak{F}(WSO))))] * L \tag{8}$$

where *Residual* signifies the presence of a residual connection in the block, $\mathfrak{F}$ represents the DFFT operation, and L is the number of repetitions of the mixer layer.

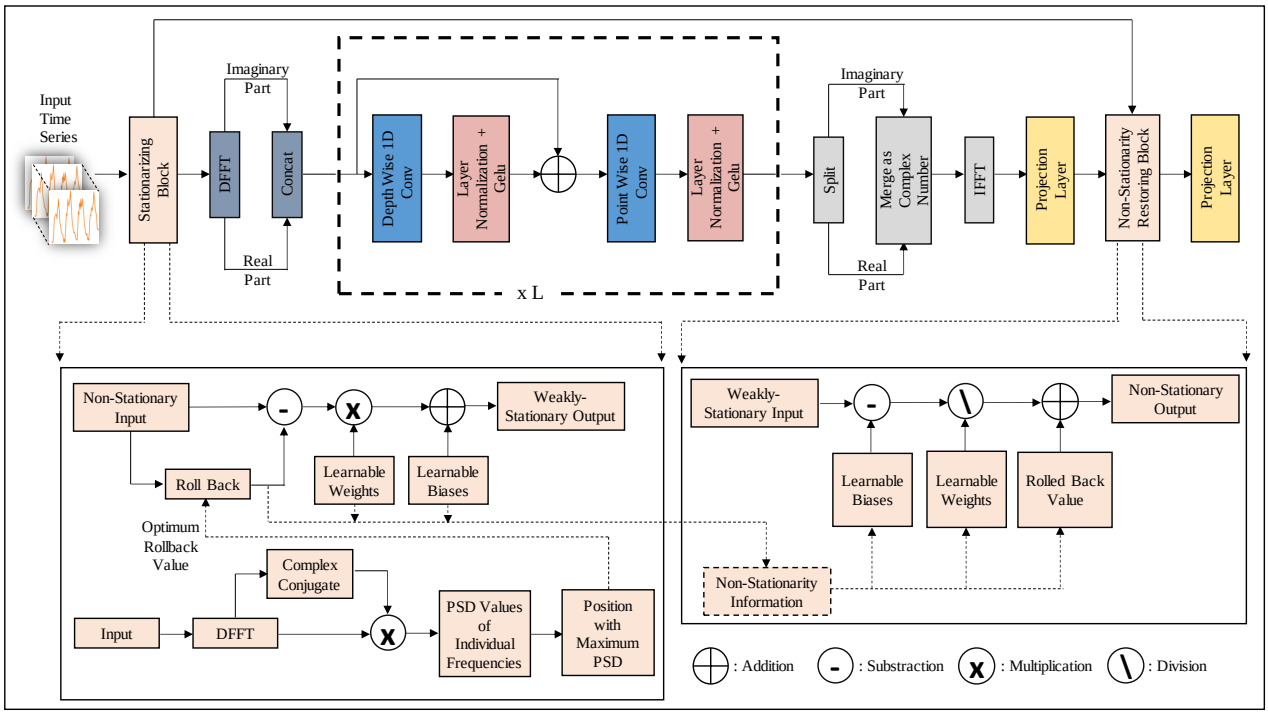


Figure 1. Overview of the proposed architecture. Input time series could be either of univariate or multivariate. The details of the "weak-stationarizing" block and "non-stationarity restoring" block are shown in the boxes at the bottom. The structure of the ConvMixer is shown in the dashed box.

2.5. Architecture Overview

The suggested framework, as illustrated in Figure 1, begins by transforming the non-stationary input into a weakly stationary form using the "weak-stationarizing" block. Subsequently, the resulting weak-stationary output undergoes a DFFT operation to obtain its spectral representation. Since the DFFT output comprises imaginary numbers, the real and imaginary parts are concatenated as separate channels for further processing. To acquire suitable features from the input containing spectral information, we employ a ConvMixer architecture. The learned features are then transformed back into the time domain. To address information loss due to the "weak-stationarizing" block, we revert the processes via the "non-stationarity restoring" block. The resulting combination of learned features with additional non-stationarity information is fed to a projection layer to generate the forecasts.

3. Experiments

We conducted comprehensive qualitative and quantitative evaluations of the proposed architecture on six real-world datasets. We also performed ablation studies to evaluate the different components of our suggested model.

3.1. Datasets

- ETT [4]: It consists of oil temperature readings of electrical transformers and six other factors affecting the temperature; collected from July 2016 and July 2018.
- Exchange [57]: It includes daily exchange rates of eight different currencies collected from 1990 to 2016.
- Electricity (https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014, accessed on 22 September 2021): This dataset contains electricity consumption recordings of 321 customers from 2012 to 2014.
- Weather (https://www.bgc-jena.mpg.de/wetter/, accessed on 22 September 2021): A collection of measurements of 21 different meteorological indicators, such as air temperature and humidity, collected every 10 min throughout 2020.
- Traffic (https://pems.dot.ca.gov/, accessed on 22 September 2021): Records of readings collected hourly from sensors on San Francisco Bay area freeways, indicating the occupancy rate of roads; provided by the California Department of Transportation.
- ILI (https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html, accessed on 22 September 2021): This dataset was collected by the Center for Disease Control and Prevention of the United States; it consists of the weekly counts of patients displaying influenza-like illness symptoms between 2002 and 2021.

3.2. Implementation Details

We trained our model using L2 loss and the Adam [58] optimizer. Hyperparameters such as the initial learning rate, number of ConvMixer layers, and batch sizes were determined via grid search on the held-out validation datasets. Early stopping was deployed to prevent overfitting. The code was implemented in PyTorch (version 1.11, https://pytorch.org, accessed on 18 April 2022), and the experiments were performed on a single NVIDIA TITAN RTX 24 GB GPU. For ensuring fair comparisons with other baselines, we set the look-back window lengths (LWLs) similar to that of Autoformer [20].

To gain an insight on the performance of our model, we compared our model's performance with eight other baseline methods: SCINet [21], Autoformer [20], Informer [4], Reformer [59], LogTrans [15], LSTNet [57], N-BEATS [60], DeepAR [32], and ARIMA [7].

3.3. Results

To evaluate the suggested model's performance along with other existing solutions, we establish a constant LWL while varying the horizon lengths. For datasets other than ILI, the LWL was set to '96', with horizon lengths varying in the range of 96, 192, 332, and 720. Similarly, for the ILI dataset in the multivariate setting, the LWL was set to 36, with horizon lengths of 24, 36, 48, and 60.

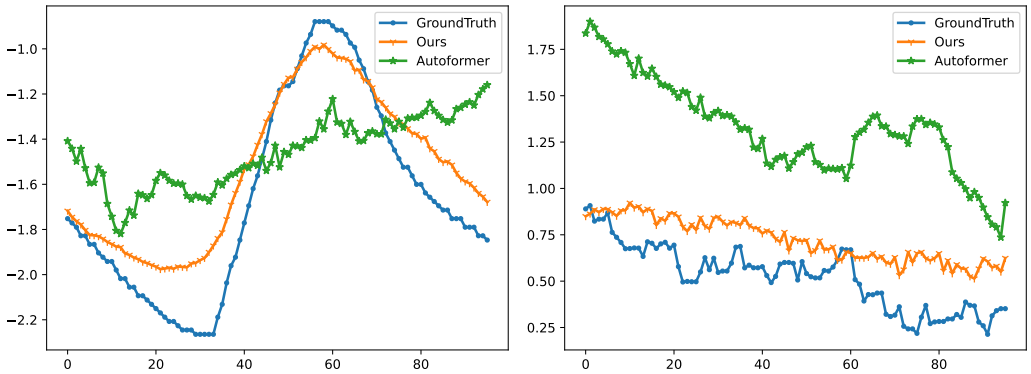
3.3.1. Results for Multivariate Setting

As presented in Table 1, our method outperforms existing baselines across all of the settings except for the "weather" dataset at horizon lengths of 96 and 192. Our proposed model is second only to SCINet in terms of mean squared error (MSE) for the 'weather' dataset, and only for forecast horizons of 96 and 192 time steps. However, our model outperforms all other models in terms of mean absolute error (MAE) in these settings. This observation indicates that for the 'weather' dataset and shorter forecast horizons (96 and 192 time steps) our model exhibits a more consistent error distribution but is more susceptible to outliers than SCINet under these specific conditions. The average relative improvements in mean squared error (MSE) compared to the previous SOTA are 13.7% on ETTm2, 20.26% on electricity, 56.45% on exchange, 16.5% on traffic, 0.3% on weather, and 21.7% on the ILI dataset. The overall average improvement in MSE is 21.5%, and the most significant improvement seems to be in the exchange dataset. The best result obtained in any particular setting also seems to be on the exchange dataset with a horizon length of 720, with a 64.82% relative improvement. In contrast to other methods, there does not seem to be any significant drop-off in performance as the length of the

prediction horizon increases. The plots of the forecasts for these datasets in the multivariate setting are shown in Figure 2. In Figure 2, we can visualize the superiority of the results of our model as compared to those of the SOTA Autoformer. Our model seems to forecast seasonal data such as traffic and electricity very well. In the case of data that do not exhibit seasonality, the plots are not groundbreaking; however, our model still outperforms the existing SOTA Autoformer. We attribute the success of our model to its ability to make careful consideration of the non-stationary property of the time series and better generalization of the complexities in inter- and intra-series relationships in the multivariate time series.

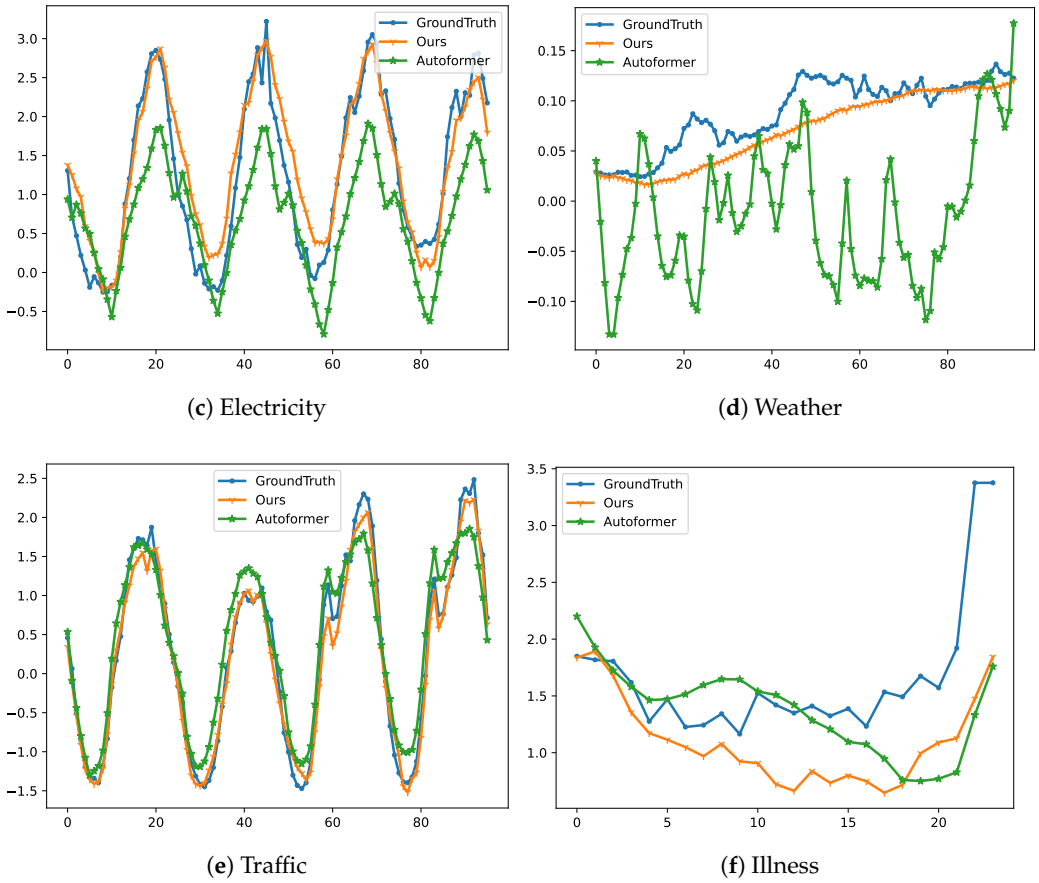
Table 1. Results in the multivariate setting. The look-back length is constant for all settings for each dataset (36 for ILI and 96 for the remaining datasets). The horizon lengths are 24, 36, 48, and 60 for ILI; and 96, 192, 336, and 720 for the rest. The best results are shown in bold and the second best results are underlined.

Model's		Ours		Autoformer [20]		SCINet [21]		Informer [4]		LogTrans [15]		Reformer [59]		LSTNet [57]	
Dataset	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	**0.183**	**0.259**	0.255	0.339	0.413	0.470	0.365	0.453	0.768	0.642	0.658	0.619	3.142	1.365
	192	**0.245**	**0.303**	0.281	0.340	0.433	0.481	0.533	0.563	0.989	0.757	1.078	0.827	3.154	1.369
	336	**0.307**	**0.348**	0.339	0.372	0.633	0.580	1.363	0.887	1.334	0.872	1.549	0.972	3.160	1.369
	720	**0.405**	**0.404**	0.422	0.419	0.864	0.680	3.379	1.388	3.048	1.328	2.631	1.242	3.171	1.368
Electricity	96	**0.154**	**0.249**	0.201	0.317	0.212	0.321	0.274	0.368	0.258	0.357	0.312	0.402	0.680	0.645
	192	**0.166**	**0.261**	0.222	0.334	0.242	0.345	0.296	0.386	0.266	0.368	0.348	0.433	0.725	0.676
	336	**0.177**	**0.275**	0.231	0.338	0.248	0.354	0.300	0.394	0.280	0.380	0.350	0.433	0.828	0.727
	720	**0.231**	**0.326**	0.254	0.361	0.270	0.368	0.373	0.439	0.283	0.376	0.340	0.420	0.957	0.811
Exchange	96	**0.082**	**0.203**	0.197	0.323	0.309	0.412	0.847	0.752	0.968	0.812	1.065	0.829	1.551	1.058
	192	**0.149**	**0.283**	0.300	0.369	1.354	0.783	1.204	0.895	1.040	0.851	1.188	0.906	1.477	1.028
	336	**0.243**	**0.368**	0.509	0.524	1.656	0.888	1.678	1.036	1.659	1.081	1.357	0.976	1.507	1.031
	720	**0.509**	**0.559**	1.447	0.941	1.272	0.855	2.478	1.310	1.941	1.127	1.510	1.016	2.285	1.243
Traffic	96	**0.516**	**0.316**	0.613	0.388	0.690	0.440	0.719	0.391	0.684	0.384	0.732	0.423	1.107	0.685
	192	**0.499**	**0.307**	0.616	0.382	0.708	0.453	0.696	0.379	0.685	0.390	0.733	0.420	1.157	0.685
	336	**0.525**	**0.327**	0.622	0.337	0.752	0.474	0.777	0.420	0.733	0.408	0.742	0.420	1.216	0.730
	720	**0.557**	**0.337**	0.660	0.408	0.812	0.494	0.864	0.472	0.717	0.396	0.755	0.423	1.481	0.805
Weather	96	0.206	**0.230**	0.266	0.336	**0.190**	0.258	0.300	0.384	0.458	0.490	0.689	0.596	0.594	0.587
	192	0.242	**0.264**	0.307	0.367	**0.235**	0.298	0.598	0.544	0.658	0.586	0.752	0.638	0.560	0.587
	336	**0.283**	**0.299**	0.359	0.395	0.292	0.343	0.578	0.523	0.797	0.652	0.639	0.596	0.597	0.587
	720	**0.341**	**0.342**	0.419	0.428	0.377	0.401	1.059	0.741	0.869	0.675	1.130	0.792	0.618	0.599
ILI	24	**2.564**	**1.034**	3.483	1.287	11.293	2.576	5.764	1.677	4.480	1.444	4.400	1.382	6.026	1.770
	36	**2.165**	**0.945**	3.103	1.148	10.817	2.468	4.755	1.467	4.799	1.467	4.783	1.448	5.340	1.668
	48	**2.323**	**0.994**	2.669	1.085	10.982	2.467	4.763	1.469	4.800	1.468	4.832	1.465	6.080	1.787
	60	**2.293**	**0.998**	2.770	1.125	10.967	2.479	5.264	1.564	5.278	1.560	4.882	1.483	5.548	1.720



(**a**) ETTm2 (**b**) Exchange

Figure 2. *Cont.*



(**c**) Electricity (**d**) Weather

(**e**) Traffic (**f**) Illness

Figure 2. Plots of forecasts by our model compared with the ground truth and forecasts of the Autoformer for six different datasets. Three datasets exhibiting seasonality (ETTm2, electricity, and traffic) and three datasets not showing seasonality (exchange, weather, and ILI) are considered.

3.3.2. Results for Univariate Setting

The results of the experiments in the univariate setting are presented in Table 2. Except for two instances, i.e., ETTm2 with a forecast horizon of 96 and exchange with a forecast horizon of 720, our method surpasses existing baselines in all other settings. The average improvement in the overall MSE value in the ETTm2 dataset is 4%, and in the exchange dataset is 12.5%. While considering the individual settings, the best improvement in performance is achieved in the exchange dataset at a forecast horizon length (FHL) of 96, with a 61.8% relative improvement compared to the previous best. Also, in the univariate setting there does not seem to be any striking degradation in performance as the length of the forecasting horizon increases.

Table 2. Results in the univariate setting for a constant look-back length of 96 and varying horizon lengths of 96, 192, 336, and 720. The best results are shown in bold and the second best results are underlined.

	Model's	**Our Method**		**Autoformer [20]**		**SCINet [21]**		**Informer [4]**		**LogTrans [15]**		**N-BEATS [60]**		**DeepAR [32]**		**ARIMA [7]**	
Dataset	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm2	96	0.071	0.190	**0.065**	**0.189**	0.082	0.217	0.088	0.225	0.082	0.217	0.082	0.219	0.099	0.237	0.211	0.362
	192	**0.104**	**0.237**	0.118	0.256	0.187	0.341	0.132	0.283	0.133	0.284	0.120	0.268	0.154	0.310	0.261	0.406
	336	**0.134**	**0.277**	0.154	0.305	0.171	0.324	0.180	0.336	0.201	0.361	0.226	0.370	0.277	0.428	0.317	0.448
	720	**0.180**	**0.326**	0.182	0.335	0.198	0.346	0.300	0.435	0.268	0.407	0.188	0.338	0.332	0.468	0.366	0.487
Exchange	96	**0.092**	**0.228**	0.241	0.387	0.207	0.362	0.591	0.615	0.279	0.441	0.156	0.299	0.417	0.515	0.112	0.245
	192	**0.184**	**0.348**	0.273	0.403	0.395	0.497	1.183	0.912	1.950	1.048	0.669	0.665	0.813	0.735	0.304	0.404
	336	**0.326**	**0.451**	0.508	0.539	0.659	0.640	1.367	0.984	2.438	1.262	0.611	0.605	1.331	0.962	0.736	0.598
	720	1.036	0.791	**0.991**	**0.768**	1.223	0.875	1.872	1.072	2.010	1.247	1.111	0.860	1.894	1.181	1.871	0.935

3.4. Ablation Study

To interpret the effects of the individual elements of our model, we observe the performance of our model following the application of several modifications to it. The datasets used in this experiment are ETTm1, ECL, and exchange. For convenient comparison, the batch size and initial learning rates were set to fixed values of 32 and 0.003, respectively, throughout all the experiments. This section can be divided into two sections: the study of the impacts of learning the features in the spectral domain and the usage of "weak-stationarizing" and "non-stationarity restoring" blocks, and the study of the impact of varying the number of mixer layers.

3.4.1. Impact of Processing the Time Series in the Spectral Domain and the Usage of "Weak-Stationarizing" and "Non-Stationarity Restoring" Blocks

The observation results for this section can be seen in Table 3. We set up four different variations of our model for this study. The first one is the model suggested in the methodology section without any alteration (results represented by the "spectral domain" sub-column under "with WS and NSR blocks" column in Table 3). For the second variation, we omit the step of transforming the input into its spectral-domain and instead the output of the "weak-stationarizing" block is directly fed to the ConvMixer block for further processing (results represented by the 'time domain' sub-column under 'with WS and NSR blocks' column in Table 3). The third and fourth variations are to study the effects of the "weak-stationarizing" and "non-stationarity restoring" blocks, both of which are variations without these blocks. The third variation has a simple skip connection at the same position as the connection between the "weak-stationarizing" block and the "non-stationarity restoring" block, and the fourth variation does not have this additional skip connection. This third variation is to check whether the effects of the "weak-stationarizing" and "non-stationarity restoring" blocks are due to the removal and addition of the non-stationary properties or just skip-connection-like properties of the connection between the two blocks.

Table 3. Impact of processing the time series in the spectral domain and usage of "weak-stationarizing" and "non-stationarity restoring" blocks observed in ETTm1, ECL, and exchange datasets. The best results are shown in bold.

	Model	**With WS and NSR Blocks**				**Without WS and NSR Blocks**			
	Variation's	**Spectral Domain**		**Time Domain**		**With Skip Connection**		**Without Skip Connection**	
Dataset	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	24	0.253	0.313	0.256	0.314	**0.232**	**0.300**	0.260	0.330
	48	**0.308**	**0.340**	0.321	0.349	0.327	0.360	0.368	0.401
	96	**0.338**	**0.354**	0.341	0.360	0.342	0.373	0.462	0.474
	288	**0.402**	**0.397**	0.404	0.369	0.406	0.404	0.541	0.530
	672	**0.474**	**0.439**	0.477	0.441	0.489	0.460	0.717	0.635
ECL	96	**0.163**	**0.260**	0.183	0.274	0.183	0.282	0.310	0.398
	192	**0.177**	**0.276**	0.188	0.280	0.196	0.294	0.332	0.413
	336	**0.194**	**0.295**	0.202	0.295	0.215	0.320	0.310	0.388
	720	**0.238**	**0.330**	0.248	0.339	0.242	0.338	0.325	0.399
Exchange	96	**0.086**	**0.207**	0.092	0.215	0.298	0.417	1.048	0.830
	192	**0.153**	**0.283**	0.154	0.285	0.364	0.482	1.591	1.031
	336	**0.243**	**0.368**	0.252	0.378	0.397	0.475	1.984	1.114
	720	0.920	0.715	**0.827**	**0.684**	0.947	0.730	2.579	1.220

The results presented in Table 3 suggest that, except for the instances of ETTm1 with an FHL of 24 and exchange with an FHL of 720, the results of the suggested model are better as compared to the other variations in each instance. While the results of the second and third variations are comparable to that of our model, the fourth variation seems to perform significantly worse. This shows that while the transformation of the input into the spectral

domain and the transformation of non-stationary input into the weak-stationary form and back via the "weak-stationarizing" and "non-stationarity restoring" blocks are important, the role of the additional connection between these two blocks is much more noteworthy.

3.4.2. Impact of Varying the Number of ConvMixer Layers

The results of varying the number of mixer layers for the ETTm1, ECL, and exchange datasets can be observed in Table 4. As per Table 4, a single best choice for the number of mixer layers to be used cannot be suggested for all of the datasets. However, it seems logical to increase the number of mixer layers in our architecture for more complex datasets and longer forecast horizons. In particular, the benefit of using a higher number of mixer layers can be observed in the ECL dataset. The best performances can be observed for all settings with the number of mixer layers set to either 4 or 5 in all cases in the ECL dataset. The best results are obtained in the ETTm1 dataset with the number of layers set to a different number for different forecast horizons. Setting the number of mixer layers to 1 or 2 seems to be best for the exchange dataset.

Table 4. Effect of varying the number of ConvMixer layers observed in the ETTm1, ECL, and exchange datasets. The best results are shown in bold.

Number of Layers		1		2		3		4		5	
Dataset	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	24	0.253	0.313	0.322	0.347	0.252	0.314	0.257	0.316	**0.251**	**0.311**
	48	0.308	0.340	0.316	0.345	**0.305**	**0.338**	0.316	0.346	0.310	0.342
	96	**0.338**	**0.354**	0.344	0.361	0.347	0.363	0.339	0.358	0.353	0.365
	288	0.402	0.397	**0.402**	**0.395**	0.406	0.397	0.406	0.398	0.415	0.404
	672	0.474	0.439	0.471	0.435	0.478	0.442	**0.468**	**0.434**	0.479	0.443
ECL	96	0.163	0.260	0.158	0.256	0.156	0.254	0.156	0.253	**0.154**	**0.251**
	192	0.176	0.276	0.171	0.271	0.167	0.267	0.168	0.267	**0.166**	**0.265**
	336	0.194	0.295	0.185	0.286	0.194	0.295	0.184	0.287	**0.183**	**0.285**
	720	0.238	0.330	0.220	0.319	0.218	0.318	**0.213**	**0.313**	0.222	0.321
Exchange	96	0.086	0.207	**0.085**	**0.204**	0.086	0.207	0.085	0.205	0.087	0.209
	192	**0.153**	**0.283**	0.154	0.285	0.163	0.289	0.156	0.287	0.154	0.283
	336	**0.243**	**0.368**	0.244	0.374	0.249	0.374	0.252	0.379	0.243	0.373
	720	0.921	0.715	**0.887**	**0.698**	0.932	0.719	1.008	0.209	0.903	0.707

3.4.3. Analysis of the Generalization Capabilities

To analyze the generalization capabilities of the model, we performed five-fold cross validation on the ETTm1, ECL, and exchange datasets. The results of using five-fold cross validation are shown in Table 5. Based on Table 5, we can see that the performance of the model is consistent over the different test sets.

Table 5. Five-fold cross validation results on the ETTm1, ECL, and exchange datasets.

Test Set		1		2		3		4		5		Average	
Dataset	**Metric**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	24	0.244	0.306	0.246	0.3072	0.253	0.313	0.281	0.332	0.267	0.323	0.258	0.316
	48	0.308	0.341	0.305	0.338	0.308	0.340	0.317	0.348	0.328	0.355	0.313	0.344
	96	0.336	0.353	0.343	0.3672	0.338	0.354	0.342	0.359	0.348	0.364	0.343	0.360
	288	0.402	0.397	0.403	0.400	0.402	0.397	0.404	0.398	0.417	0.407	0.406	0.400
	672	0.473	0.441	0.481	0.445	0.474	0.440	0.482	0.444	0.485	0.447	0.479	0.443
ECL	96	0.165	0.258	0.166	0.261	0.165	0.260	0.163	0.261	0.167	0.261	0.165	0.260
	192	0.176	0.277	0.173	0.270	0.175	0.272	0.177	0.276	0.175	0.277	0.175	0.274
	336	0.191	0.293	0.192	0.294	0.193	0.293	0.194	0.295	0.206	0.311	0.195	0.297
	720	0.233	0.328	0.235	0.328	0.234	0.326	0.238	0.330	0.235	0.331	0.235	0.329
Exchange	96	0.082	0.204	0.093	0.212	0.086	0.207	0.084	0.205	0.083	0.204	0.086	0.206
	192	0.169	0.292	0.158	0.285	0.153	0.283	0.149	0.283	0.155	0.288	0.157	0.286
	336	0.275	0.385	0.248	0.372	0.243	0.368	0.252	0.375	0.250	0.378	0.254	0.376
	720	0.837	0.684	0.914	0.712	0.921	0.715	0.880	0.696	0.867	0.715	0.884	0.704

3.5. Efficiency Analysis

We compare the memory and run-time requirements of recent transformer architectures, which present SOTA results, with our suggested method with a varying number of mixer layers. We use the ETTm2 dataset in a univariate setting with a batch size set to 8. The memory and run-time data were collected during the training phase for all compared architectures. The results are shown in Figure 3.

Figure 3a,b show the comparison of the memory requirements with varying values for horizon length and look-back window length. Similarly, Figure 3c,d show the comparison of run-time with varying values for horizon length and LWL, respectively. From Figure 3 we can conclude that the suggested architecture is much more efficient as compared to the recently famous transformer architecture. The usage of resource-intensive self-attention, prob-sparse self-attention, and autocorrelation results in the inefficiency of the transformer, Informer, and Autoformer architectures.

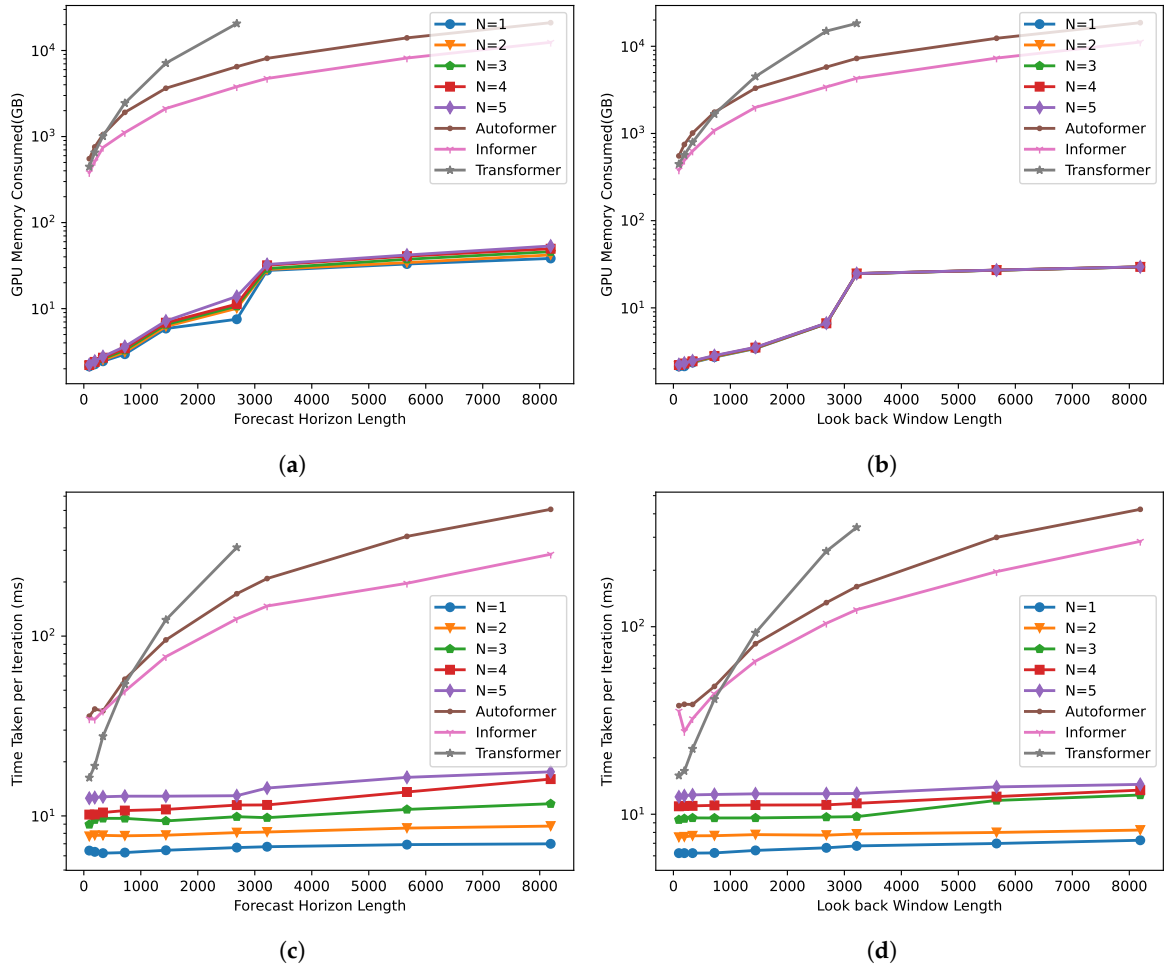


Figure 3. Run-time requirement and memory consumption analysis while varying FHL and LWL. A comparison was made among Autoformer, Informer, transformer, and our method (with the number of layers ranging from one to five) on a log-scale. (**a**) Comparison of GPU memory consumption while varying the FHL; (**b**) comparison of GPU memory consumption while varying the LWL; (**c**) comparison of time taken for execution while varying the FHL; (**d**) comparison of time taken for execution while varying the LWL.

4. Conclusions

This study proposed a deep learning architecture designed to address the challenges of TSF in both multivariate and univariate settings. It focuses on the non-stationary nature of real-world time series data as well as the complexities of intra- and inter-series relationships. The suggested architecture comprises novel components such as the "weak-

stationarizing" block and "non-stationarity restoring" block to handle non-stationarity, while also leveraging spectral decomposition and a ConvMixer architecture to capture complex relations within the data. The experimental results demonstrate the effectiveness of the proposed model across six real-world datasets, achieving superior or comparable performance to SOTA methods in most cases. Additionally, the proposed model requires significantly less memory and execution time compared to other transformer-based models. This makes the model suitable for usage in scenarios where the computation resources are limited. Although the focus is on long-sequence time series, the model is adaptable to short-sequence scenarios as well. Moving forward, it would be interesting to explore how the concepts introduced in the proposed model can complement existing forecasting models, such as transformers, TCNs, and RNNs. Additionally, there is room for improvement in handling datasets without seasonality, suggesting avenues for further research to enhance the model's performance in such scenarios.

Author Contributions: Conceptualization, R.B.; methodology, R.B.; software, R.B.; validation, R.B. and S.-W.L.; formal analysis, R.B.; investigation, R.B.; resources, S.-W.L.; data curation, R.B.; writing—original draft preparation, R.B.; writing—review and editing, R.B. and S.-W.L.; visualization, S.-W.L.; supervision, S.-W.L.; project administration, S.-W.L.; funding acquisition, S.-W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Gachon University research fund of 2020(202008450003) and the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. RS-2023-00250978).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the datasets used in this paper are publicly available datasets and all of them have been cited above.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

TSF	Time series forecasting
ARIMA	Auto regressive integrated moving average
LSTF	Long-sequence time series forecasting
TCN	Temporal convolutional network
PSD	Power spectral density
FFT	Fast Fourier transform
DFFT	Discrete fast Fourier transform
SOTA	State of the art
RNN	Recurrent neural network
DA	Domain adaptation
DG	Domain generalization
GNN	Graph neural network
ORV	Optimum roll back value
WSO	Weak-stationary output
NSO	Non-stationary output
WS	Weak stationarizing
NSR	Non-stationarity restoring
DW	Depthwise convolution
PW	Pointwise convolution
LWL	Look-back window length
MSE	Mean squared error
MAE	Mean average error

References

1. Laptev, N.; Yosinski, J.; Li, L.E.; Smyl, S. Time-series extreme event forecasting with neural networks at uber. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 34, pp. 1–5.
2. Song, H.; Rajan, D.; Thiagarajan, J.J.; Spanias, A. Attend and diagnose: Clinical time series analysis using attention models. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
3. Churpek, M.M.; Adhikari, R.; Edelson, D.P. The value of vital sign trends for detecting clinical deterioration on the wards. *Resuscitation* **2016**, *102*, 1–5. [CrossRef] [PubMed]
4. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021.
5. de Bézenac, E.; Rangapuram, S.S.; Benidis, K.; Bohlke-Schneider, M.; Kurle, R.; Stella, L.; Hasson, H.; Gallinari, P.; Januschowski, T. Normalizing kalman filters for multivariate time series analysis. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 2995–3007.
6. Tuncel, K.S.; Baydogan, M.G. Autoregressive forests for multivariate time series modeling. *Pattern Recognit.* **2018**, *73*, 202–215. [CrossRef]
7. Taylor, S.J.; Letham, B. Forecasting at scale. *Am. Stat.* **2018**, *72*, 37–45. [CrossRef]
8. Van Den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.W.; Kavukcuoglu, K. WaveNet: A generative model for raw audio. *SSW* **2016**, *125*, 2.
9. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional time series forecasting with convolutional neural networks. *arXiv* **2017**, arXiv:1703.04691.
10. Sen, R.; Yu, H.F.; Dhillon, I.S. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
11. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv* **2017**, arXiv:1707.01926.
12. Yu, B.; Yin, H.; Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv* **2017**, arXiv:1709.04875.
13. Song, C.; Lin, Y.; Guo, S.; Wan, H. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 914–921.
14. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
15. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
16. Koopmans, L.H. *The Spectral Analysis of Time Series*; Elsevier: Amsterdam, The Netherlands, 1995.
17. Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17766–17778.
18. Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.H.; Choo, J. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.
19. Du, Y.; Wang, J.; Feng, W.; Pan, S.; Qin, T.; Xu, R.; Wang, C. Adarnn: Adaptive learning and forecasting of time series. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual, 1–5 November 2021; pp. 402–411.
20. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
21. Liu, M.; Zeng, A.; Xu, Z.; Lai, Q.; Xu, Q. Time Series is a Special Sequence: Forecasting with Sample Convolution and Interaction. *arXiv* **2021**, arXiv:2106.09305.
22. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
23. Trockman, A.; Kolter, J.Z. Patches Are All You Need? *arXiv* **2022**, arXiv:2201.09792.
24. Cooley, J.W.; Tukey, J.W. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* **1965**, *19*, 297–301. [CrossRef]
25. Box, G.E.; Jenkins, G.M.; MacGregor, J.F. Some recent advances in forecasting and control. *J. R. Stat. Soc. Ser. C (Appl. Stat.)* **1974**, *23*, 158–179. [CrossRef]
26. Holt, C.C. Forecasting seasonals and trends by exponentially weighted moving averages. *Int. J. Forecast.* **2004**, *20*, 5–10. [CrossRef]
27. Winters, P.R. Forecasting sales by exponentially weighted moving averages. *Manag. Sci.* **1960**, *6*, 324–342. [CrossRef]
28. Lütkepohl, H. *New Introduction to Multiple Time Series Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2005.
29. Wen, R.; Torkkola, K.; Narayanaswamy, B.; Madeka, D. A multi-horizon quantile recurrent forecaster. *arXiv* **2017**, arXiv:1711.11053.
30. Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; Cottrell, G. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv* **2017**, arXiv:1704.02971.

31. Maddix, D.C.; Wang, Y.; Smola, A. Deep factors with gaussian processes for forecasting. *arXiv* **2018**, arXiv:1812.00098.
32. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **2020**, *36*, 1181–1191. [CrossRef]
33. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. *arXiv* **2019**, arXiv:1906.00121.
34. Bai, L.; Yao, L.; Li, C.; Wang, X.; Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17804–17815.
35. Huang, R.; Huang, C.; Liu, Y.; Dai, G.; Kong, W. LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks. In Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 7–15 January 2021; pp. 2355–2361.
36. Li, M.; Zhu, Z. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35; pp. 4189–4196.
37. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 2030–2096.
38. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7167–7176.
39. Wang, J.; Chen, Y.; Feng, W.; Yu, H.; Huang, M.; Yang, Q. Transfer learning with dynamic distribution adaptation. *ACM Trans. Intell. Syst. Technol. (TIST)* **2020**, *11*, 1–25. [CrossRef]
40. Wang, J.; Feng, W.; Chen, Y.; Yu, H.; Huang, M.; Yu, P.S. Visual domain adaptation with manifold embedded distribution alignment. In Proceedings of the 26th ACM International Conference on MULTIMEDIA, Seoul, Republic of Korea, 22–26 October 2018; pp. 402–410.
41. Zhu, Y.; Zhuang, F.; Wang, J.; Ke, G.; Chen, J.; Bian, J.; Xiong, H.; He, Q. Deep subdomain adaptation network for image classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1713–1722. [CrossRef] [PubMed]
42. Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Zeng, W.; Qin, T. Generalizing to unseen domains: A survey on domain generalization. *arXiv* **2021**, arXiv:2103.03097.
43. Balaji, Y.; Sankaranarayanan, S.; Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montreal, QC, Canada, 3–8 December 2018; Volume 31.
44. Li, H.; Pan, S.J.; Wang, S.; Kot, A.C. Domain generalization with adversarial feature learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5400–5409.
45. Muandet, K.; Balduzzi, D.; Schölkopf, B. Domain generalization via invariant feature representation. In Proceedings of the International Conference on Machine Learning. PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 10–18.
46. Hadjidimitriou, S.K.; Hadjileontiadis, L.J. Toward an EEG-based recognition of music liking using time-frequency analysis. *IEEE Trans. Biomed. Eng.* **2012**, *59*, 3498–3510. [CrossRef] [PubMed]
47. Kepesi, M.; Weruaga, L. Adaptive chirp-based time–frequency analysis of speech signals. *Speech Commun.* **2006**, *48*, 474–492. [CrossRef]
48. Tu, G.; Dong, X.; Chen, S.; Zhao, B.; Hu, L.; Peng, Z. Iterative nonlinear chirp mode decomposition: A Hilbert-Huang transform-like method in capturing intra-wave modulations of nonlinear responses. *J. Sound Vib.* **2020**, *485*, 115571. [CrossRef]
49. Guo, L.; van der Wegen, M.; Jay, D.A.; Matte, P.; Wang, Z.B.; Roelvink, D.; He, Q. River-tide dynamics: Exploration of nonstationary and nonlinear tidal behavior in the Yangtze River estuary. *J. Geophys. Res. Ocean.* **2015**, *120*, 3499–3521. [CrossRef]
50. Lin, Y.H.; Tsai, M.S. Development of an improved time–frequency analysis-based nonintrusive load monitor for load demand identification. *IEEE Trans. Instrum. Meas.* **2013**, *63*, 1470–1483. [CrossRef]
51. Mezić, I. On applications of the spectral theory of the Koopman operator in dynamical systems and control theory. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 7034–7041.
52. Cooley, J.W.; Lewis, P.A.; Welch, P.D. The fast Fourier transform and its applications. *IEEE Trans. Educ.* **1969**, *12*, 27–34. [CrossRef]
53. Brigham, E.O. *The Fast Fourier Transform and its Applications*; Prentice-Hall, Inc.: Englewood Cliffs, NJ, USA, 1988.
54. Lange, H.; Brunton, S.L.; Kutz, J.N. From Fourier to Koopman: Spectral Methods for Long-term Time Series Prediction. *J. Mach. Learn. Res.* **2021**, *22*, 1–38.
55. Percival, D.B.; Walden, A.T. *Spectral Analysis for Physical Applications*; Cambridge University Press: Cambridge, UK, 1993.
56. Chatfield, C. *Time-Series Forecasting*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2000.
57. Lai, G.; Chang, W.C.; Yang, Y.; Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 95–104.
58. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
59. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The efficient transformer. *arXiv* **2020**, arXiv:2001.04451.
60. Oreshkin, B.N.; Carpov, D.; Chapados, N.; Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv* **2019**, arXiv:1905.10437.