# STLformer: Exploit STL decomposition and Rank Correlation for Time Series Forecasting

Zuokun Ouyang, Meryem Jabloun, Philippe Ravier
*PRISME Laboratory, University of Orléans, France*
{zuokun.ouyang,meryem.jabloun,philippe.ravier}@univ-orleans.fr

***Abstract*—The challenge of time series forecasting has been the focus of research in recent years, with Transformer-based models using various self-attention mechanisms to uncover long-range dependencies. However, complex trends and nonlinear serial dependencies presented in some specific datasets may not always be captured properly. To address these issues, we present STLformer, a novel Transformer-based model that utilizes an STL decomposition architecture and the rank correlation function to improve long-term time series forecasting. STLformer outperforms four state-of-the-art Transformers and two RNN models across multiple datasets and forecasting horizons.**

***Index Terms*—Time Series, Forecasting, Transformer, Rank Correlation, Nonlinear Dependencies, STL Decomposition**

## I. Introduction

Time series forecasting is a widely used technique that involves predicting the future values of a given time series based on its historical behavior. This technique has found extensive applications in diverse fields such as GDP prediction [1], weather forecasting [2], energy consumption forecasting [3], and traffic forecasting [4]. While traditional statistical methods like AutoRegression Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), and Theta method [5]–[8] have long dominated the field, deep learning has emerged as a promising alternative for time series forecasting in recent years, achieving remarkable success [9]–[12]. In this context, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers have become the most popular deep learning models for time series forecasting. Notably, CNNs and RNNs are particularly suited for forecasting tasks that require capturing sequential dependencies in the time series [13], which is demonstrated by the success of several representative works such as TCN [14], LSTNet [15], DeepAR [16], and N-BEATS [17].

Since its inception in 2017, Transformer models have gained increasing popularity and have been successfully applied in various fields, including machine translation, computer vision, and text generation [18]–[21]. Informer [22] was the first work introducing the Transformer model for long-term time series forecasting, employing a ProbSparse self-attention calculation and a self-attention distilling mechanism to handle the quadratic computation complexity. Autoformer [23] replaced the self-attention block with an AutoCorrelation mechanism to identify period-based dependencies and employed a decomposition structure to separate the long-term stationary trend and seasonal patterns. Our previous work, Rankformer, utilized the Spearman Rank Correlation to better capture the autoregressive conditional heteroskedasticity (ARCH) effect [24]. Other Transformer models like LogTrans [25] and Reformer [26] have also been applied to time series forecasting tasks.

Although the Transformer models discussed previously have attempted to capture long-range dependencies in time series, they have not always been successful. For example, while Informer [22] utilized the ProbSparse self-attention mechanism to reduce computation complexity, it did not extract hidden long-range dependencies effectively. Similarly, Autoformer [23] used the AutoCorrelation mechanism to discover period-based dependencies. However, its use of the Pearson correlation function only supports linear correlation, which may not be sufficient for certain time series. Rankformer [24] exploited the Rank Correlation to handle nonlinear dependencies. However, it relies solely on simple moving averages for decomposition, which may not accurately extract seasonal patterns, resulting in suboptimal modeling for seasonal and trend parts and final results.

This paper presents STLformer, a novel Transformer-based model for long-term time series forecasting. STLformer is the first of its kind to incorporate the Seasonal-Trend decomposition using LOESS (LOcall Estimated Scatterplot Smoothing) architecture into the Transformer framework. By leveraging the STL decomposition, our model is able to capture and model the trend and seasonal patterns present in certain time series, resulting in improved forecasting performance. STLformer builds upon our prior work, Rankformer, which uses rank correlation to detect nonlinear serial dependencies. Extensive experiments on four benchmark datasets demonstrate the superiority of STLformer over other Transformer models for four different forecasting horizons. Overall, our contribution lies in combining the Transformer architecture with STL decomposition and demonstrating its effectiveness for long-term time series forecasting.

The rest of the paper is organized as follows. Sec. II introduces the proposed STLformer model. We then present the experimental setups and configurations in Sec. III. The comparison and discussions based on the results are given in Sec. IV. Finally, Sec. V concludes the paper.

## II. Method

This section presents an overview of the STLformer architecture and its two essential components, i.e., the STL

This research was partly supported by Association Nationale de la Recherche et de la Technologie (CIFRE 2019/0551).

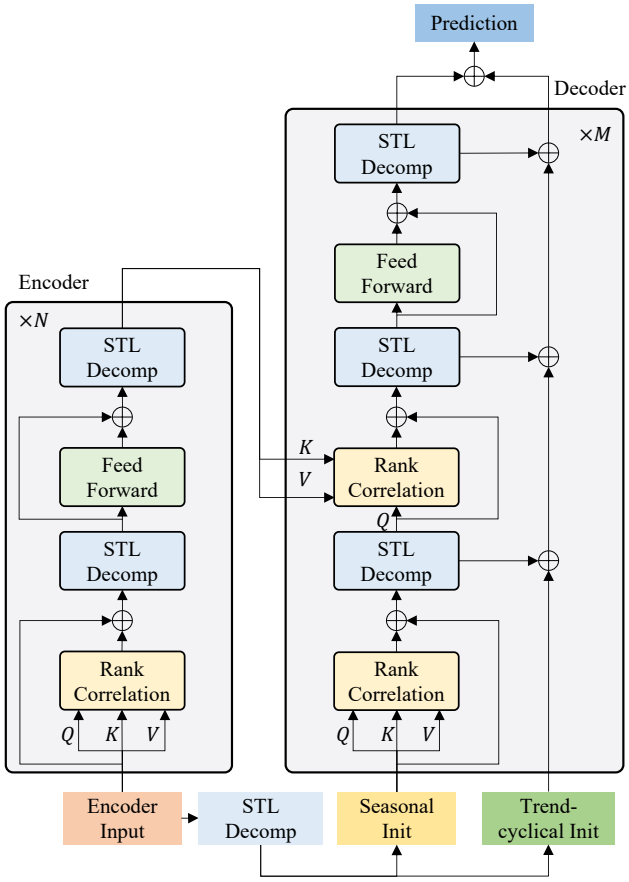



Fig. 1. The architecture of STLformer.

Decomposition block and the Rank Correlation block.

### A. STLformer Architecture

STLformer employs an identical encoder-decoder architecture as Rankformer and Autoformer, as depicted in Fig. 1. To ensure the comprehensiveness of this article, we reiterate the architecture.

The encoder consists of $N$ identical layers, each comprising a multi-head Rank Correlation (RankCorr) block, two STL Decomposition (STLDecomp) blocks, and a Feed-Forward (FF) block. Similarly, the decoder contains $M$ identical layers, with each layer having two multi-head RankCorr blocks, three STLDecomp blocks, and one FF block. The final prediction is formed by combining the outputs of the last STLDecomp block and the refined trend-cyclical part in the decoder. Further details of the STLformer architecture are detailed in the subsequent sections.

*1) Encoder:* Using RankCorr and STLDecomp blocks, the encoder decomposes the time series into seasonal and trend-cyclical parts, with the latter being ignored during modeling. As a result, the encoder primarily focuses on modeling the seasonal part. The output of the $l$-th encoder layer is represented as $\mathcal{X}_{\text{en}}^{l} = \text{Encoder}(\mathcal{X}_{\text{en}}^{l-1})$. The process in a single encoder layer is formulated as follows:

$$\begin{aligned}\mathcal{S}_{\text{en}}^{l,1}, _ &= \text{STLDecomp}\big(\text{RankCorr}(\mathcal{X}_{\text{en}}^{l-1}) + \mathcal{X}_{\text{en}}^{l-1}\big),\\ \mathcal{S}_{\text{en}}^{l,2}, _ &= \text{STLDecomp}\big(\text{FF}(\mathcal{S}_{\text{en}}^{l,1}) + \mathcal{S}_{\text{en}}^{l,1}\big),\end{aligned} \tag{1}$$

where $\mathcal{S}_{\text{en}}^{l,i}$ represents the seasonal component after the $i$-th STLDecomp block. $\mathcal{X}_{\text{en}}^{l} = \mathcal{S}_{\text{en}}^{l,2},\ l \in \{1, 2, ..., N\}$.

*2) Decoder:* In STLformer, the decoder has two streams: the trend-cyclical stream and the seasonal stream. The seasonal stream continuously refines the seasonal part of the time series, while the trend-cyclical stream focuses on modeling the trend-cyclical component. Using a notation similar to the encoder, we can summarize the process in a single decoder layer as $\mathcal{X}_{\text{de}}^{l}, \mathcal{T}_{\text{de}}^{l} = \text{Decoder}(\mathcal{X}_{\text{de}}^{l-1}, \mathcal{T}_{\text{de}}^{l-1})$, and formalize it as follows:

$$\begin{aligned}\mathcal{S}_{\text{de}}^{l,1}, \mathcal{T}_{\text{de}}^{l,1} &= \text{STLDecomp}\Big(\text{RankCorr}(\mathcal{X}_{\text{de}}^{l-1}) + \mathcal{X}_{\text{de}}^{l-1}\Big),\\ \mathcal{S}_{\text{de}}^{l,2}, \mathcal{T}_{\text{de}}^{l,2} &= \text{STLDecomp}\Big(\text{RankCorr}(\mathcal{S}_{\text{de}}^{l,1}, \mathcal{X}_{\text{en}}^{N}) + \mathcal{S}_{\text{de}}^{l,1}\Big),\\ \mathcal{S}_{\text{de}}^{l,3}, \mathcal{T}_{\text{de}}^{l,3} &= \text{STLDecomp}\Big(\text{FF}(\mathcal{S}_{\text{de}}^{l,2}) + \mathcal{S}_{\text{de}}^{l,2}\Big),\\ \mathcal{T}_{\text{de}}^{l} &= \mathcal{T}_{\text{de}}^{l-1} + W_{l,1}\mathcal{T}_{\text{de}}^{l,1} + W_{l,2}\mathcal{T}_{\text{de}}^{l,2} + W_{l,3}\mathcal{T}_{\text{de}}^{l,3},\end{aligned} \tag{2}$$

$\mathcal{S}_{\text{en}}^{l,i}$ and $\mathcal{T}_{\text{de}}^{l,i}$ denote the seasonal and trend-cyclical components, respectively, and $W_{l,1}$, $W_{l,2}$, $W_{l,3}$ are trainable weights. The output of the $l$-th decoder layer consists of two parts: the refined seasonal patterns $\mathcal{X}_{\text{de}}^{l} = \mathcal{S}_{\text{de}}^{l,3}$, and the multiple level trend-cyclical patterns $\mathcal{T}_{\text{de}}^{l}$, where $l \in \{1, 2, ..., M\}$.

*3) Model Inputs and Outputs:* With $I$ denoting the input length, $O$ representing the output length, and $d$ indicating the model dimension, the inputs of STLformer's encoder and decoder are then defined as follows:

- The encoder input, which consists of the last $I$ time steps in the time series: $\mathcal{X}_{\text{en}} \in \mathbb{R}^{I \times d}$.
- The seasonal init input, which concatenates the latter half of the encoder's decomposed input and $O$ zero-placeholders: $\mathcal{X}_{\text{de,S}} = \text{concat}(\mathcal{X}_{\text{en,S}}, \mathcal{X}_0) \in \mathbb{R}^{(\frac{I}{2}+O) \times d}$.
- The trend-cyclical init input, which also consists of the latter half of the decomposed $\mathcal{X}_{\text{en}}$ and a placeholder filled with the average of $\mathcal{X}_{\text{en}}$: $\mathcal{X}_{\text{de,T}} = \text{concat}(\mathcal{X}_{\text{en,T}}, \mathcal{X}_{\text{avg}}) \in \mathbb{R}^{(\frac{I}{2}+O) \times d}$.

The relationship between the STLformer inputs can be formalized as follows:

$$\begin{aligned}\mathcal{X}_{\text{en,S}}, \mathcal{X}_{\text{en,T}} &= \text{STLDecomp}\left(\mathcal{X}_{\text{en}}\left[\frac{I}{2} : I\right]\right),\\ \mathcal{X}_{\text{de,S}} &= \text{concat}(\mathcal{X}_{\text{en,S}}, \mathcal{X}_0),\\ \mathcal{X}_{\text{de,T}} &= \text{concat}(\mathcal{X}_{\text{en,T}}, \mathcal{X}_{\text{avg}}).\end{aligned} \tag{3}$$

The final output of the STLformer model is a combination of the seasonal and trend-cyclical parts formed in the decoder. It can be formalized as $W_{\text{S}}\mathcal{X}_{\text{de}}^{M} + \mathcal{T}_{\text{de}}^{M}$, where $W_{\text{S}}$ is a trainable weight to project $\mathcal{X}_{\text{de}}^{M}$ into the target dimension.

### B. STL Decomposition Block

The key difference between STLformer and Rankformer, and the primary contribution of our work, is the substitution of the Multi-Level Decomposition block in Rankformer with the STL decomposition block, which utilizes LOESS regression.

LOESS was first proposed by Cleveland [27]. It is a nonparametric robust locally weighted regression method for smoothing a scatterplot, $(x_i, y_i), i = 1, ..., n$, in which the fitted value at $x_k$ is the value of a polynomial fit to the data using weighted least squares, where the weight for $(x_i, y_i)$ is large if $x_i$ is close to $x_k$ and small if it is not. It splits the data into several small sections, performs weighted linear regressions on different sections, and connects the center of these curves to form the complete regression curve. Specifically, LOESS is defined by the following sequence of operations for a given time series:

1) For one data point, often called the *focal point*, select $k$ nearest points around it to form a local window. Every focal point has a corresponding local window.
2) Calculate the weights of every point in the window through a weight function $W$, which is conventionally a Tricube function as shown in (4).

$$W(x) = \begin{cases} (1 - |x|^3)^3, & \text{for } |x| < 1, \\ 0, & \text{for } |x| \geq 1. \end{cases} \tag{4}$$

3) Fit a weighted linear regression in the window. For $n$ focal points, we have $n$ weighted linear regressions.
4) Connect the center points of the $n$ weighted regressions to form the final fitted curve.

The time complexity of LOESS mainly involves traversing the entire dataset to select the $k$ nearest points for each point to form the local window, which leads to an $\mathcal{O}(N^2)$ complexity. This issue can be resolved by using a $k$-d tree for acceleration, which can rapidly find the nearest neighbors of a data point. Implementing a $k$-d tree can reduce the time complexity of the LOESS algorithm to $\mathcal{O}(N \log N)$ [27].

In 1990, Cleveland et al. proposed the famous STL decomposition method [28], which leverages LOESS to estimate the trend and seasonal components, contributing to a versatile and robust method for decomposing time series.

Our model adopted the idea of STL decomposition and implemented the $k$-d tree LOESS to decompose the input time series into seasonal and trend-cyclical components. The STLDecomp block employs the LOESS regression to fit a locally smoothed trend-cyclical component. We formalize the STLDecomp block as follows:

$$\begin{aligned} \mathcal{X}_{\text{seasonal}}, \mathcal{X}_{\text{trend-cyclical}} &= \text{STLDecomp}\left(\mathcal{X}\right), \\ \mathcal{X}_{\text{trend-cyclical}} &= \text{LOESS}\left(\mathcal{X}\right), \\ \mathcal{X}_{\text{seasonal}} &= \mathcal{X} - \mathcal{X}_{\text{trend-cyclical}}, \end{aligned} \tag{5}$$

where LOESS is the LOESS regression function, and $\mathcal{X}_{\text{trend-cyclical}}$ and $\mathcal{X}_{\text{seasonal}}$ denote the trend-cyclical and seasonal components, respectively.

### C. RankCorrelation Block

STLformer leverages the RankCorrelation block, which is also used in our previous work Rankformer [24]. We revisit the concept behind it to provide a comprehensive understanding.

The AutoCorrelation Function (ACF) uses the well-known Pearson correlation function, i.e., Pearson's $\rho$, to measure the correlation between two instants in a stationary time series $y_t$:

$$\text{ACF}(k) = \rho_p(y_{t-k}, y_t) = \frac{\text{cov}(y_{t-k}, y_t)}{\sigma(y_{t-k})\sigma(y_t)}, \ k = 0, 1, 2, ..., \forall t. \tag{6}$$

However, in some time series, the long-term dependencies are not linear, and this nonlinearity can result in a low Pearson's $\rho$, leading to the erroneous measurement of dependencies. To address this issue, we use the Rank Correlation Function (RCF), more commonly known as Spearman's $\rho$, to measure nonlinear correlations. Spearman's $\rho$ is defined as follows:

$$\rho_s(X, Y) = \frac{\text{cov}(R(X), R(Y))}{\sigma(R(X))\sigma(R(Y))}, \tag{7}$$

where $R(X)$ and $R(Y)$ denote the ranks of two random variables $X$ and $Y$, respectively. The usual Pearson correlation coefficient is applied to the ranks to compute the Ranked AutoCorrelation Function (RACF). $\rho_s$ is defined in the range $[-1, 1]$, where $-1$ indicates a perfect negative monotonic relationship, $0$ indicates no monotonic relationship, and $1$ indicates a perfect positive monotonic relationship. Our RACF is defined as follows:

$$\text{RACF}(k) = \rho_s(y_{t-k}, y_t), \ k = 0, 1, 2, ..., \forall t. \tag{8}$$

We compute the RACF exploiting the Wiener-Khinchin theorem, which states that the ACF of a stationary time series can be computed by the Fourier transform of its power spectrum. The latter is accelerated to $\mathcal{O}(N \log N)$ with Fast Fourier Transform. In our implementation, we rank the time series using the efficient $\mathcal{O}(N \log N)$ sorting operator [29] provided by the `torchsort`[1] library and then compute the ACF of the ranked time series to obtain the RACF. Therefore, the overall computation complexity of calculating the RACF is $\mathcal{O}(N \log N)$.

## III. Experiments

This section presents our experimental settings and results.

### A. Datasets

We evaluated STLformer against other state-of-the-art methods on four popular datasets, as per Rankformer:

- Electricity Transformer Temperature (ETT) [22]: 15-minute level oil temperature and six power load features data recorded in two Chinese counties from July 2016 to July 2018. Seasonal data.
- Exchange-Rate[2]: Daily exchange rates from 1990 to 2016 of eight countries, i.e., Australia, British, Canada, China, Japan, New Zealand, Singapore, and Switzerland. Nonseasonal.
- Weather[3]: 10-minute level local climate data of 21 meteorological features from Max-Planck-Institut für Biogeochemie, Jena for 2020. Complex seasonal data.

[1]https://github.com/teddykoker/torchsort
[2]https://github.com/laiguokun/multivariate-time-series-data
[3]https://www.bgc-jena.mpg.de/wetter/

TABLE I
DATASET DESCRIPTION

| Dataset | ETT | Exchange | Weather | ILI |
|---|---|---|---|---|
| **Length** | 69680 | 7588 | 52696 | 966 |
| **Dimension** | 7 | 8 | 21 | 7 |
| **Frequency** | 15 min | 1 day | 10 min | 1 week |
| **Seasonality** | Seasonal | Nonseasonal | Complex | Seasonal |
| **Engle's test p-value** | $<$ 2.2e-16 | $<$ 2.2e-16 | $<$ 2.2e-16 | 0.8126 |
| **ARCH effect** | Significant | Significant | Significant | Insignificant |

- Influenza-Like Illness (ILI)[4]: Weekly ILI patients data containing the ratio of ILI patients and the total number of patients collected by the U.S. Centers for Disease Control and Prevention from 2002 to 2021. Seasonal data.

We performed Engle's Lagrange Multiplier Test [30] on the four datasets to evaluate the significance of nonlinearity in the serial dependencies. This test assesses the significance of ARCH effects in a time series. A significant result reveals the presence of nonlinear serial dependencies in the series. The statistics of the datasets are listed in Tab. I, along with Engle's test results [24].

We split all datasets into train/validation/test sets in chronological order, using a 7/1/2 split for all datasets except for ETT, which was split into 6/2/2, as per Rankformer [24] and Autoformer [23].

### *B. Experimental Settings*

We used the same number of encoder-decoder layers as Rankformer, i.e., two encoder layers and one decoder layer. STLformer was trained using the Mean Square Error loss and the Adam optimizer [31], with an initial learning rate of $10^{-4}$. We set the batch size to 32 and trained the model for ten epochs with a learning rate scheduler that reduces the learning rate by a factor of 0.5 when the validation loss plateaus. The model was implemented in PyTorch [32] and trained on a single NVIDIA Tesla V100 GPU.

## IV. RESULTS AND DISCUSSION

STLformer was compared with the following Transformer methods: Rankformer [24], Autoformer [23], Informer [22], LogTrans [25], Reformer [26]. We also compared STLformer with baseline deep learning models, i.e., LSTNet [15] and LSTM [33]. Mean Square Error (MSE) and Mean Absolute Error (MAE) were used to evaluate the performance of the models. The results are shown in Tab. II with the best results highlighted in bold, while the second-best ones are underlined.

In general, STLformer achieved superior performance compared to the other methods on the ETT, Exchange-Rate, and Weather datasets while being slightly less accurate than Autoformer and Rankformer on the ILI dataset. STLformer also performed slightly better than Autoformer on the Weather dataset but also marginally poorer than Rankformer.

[4]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

On the Exchange-Rate dataset, despite the absence of any significant periodicity in the dataset, STLformer still achieved the best performance boost on average for both Rankformer (4.84%) and Autoformer (21.45%). We attribute this to STLformer's STL Decomposition Block being better at extracting the trend and seasonal patterns, thus resulting in better handling of the dataset's nonlinear serial dependencies. STLformer also achieved decent improvements on the ETT dataset, with an average boost of 4.80% for Rankformer and 9.74% for Autoformer, compared to their respective performances with their original decomposition blocks. This is less than the performance boost on the Exchange-Rate dataset. We observed that the ETT dataset has a significant seasonality, where a standard moving average can effectively separate the trend and seasonal patterns. At the same time, Exchange-Rate is nonseasonal, which requires a more sophisticated decomposition method to help the model focus on modeling different patterns. This explains why STLformer achieved a better performance boost on the Exchange-Rate dataset.

For the Weather dataset, STLformer outperformed Autoformer but not Rankformer. We think as the Weather dataset exhibits a complex seasonality, Rankformer's Multi-Level Decomposition block is more suitable for the dataset than STLformer's STL-based decomposition block. For ILI, both STLformer and Rankformer filed to outperform Autoformer, and there is no performance boost from STL decomposition. The p-value of Engle's Lagrange Multiplier test on the ILI dataset was 0.8126, indicating statistically significant linear serial dependencies that can be handled more appropriately by the AutoCorrelation Block adopted in Autoformer.

Regarding the computation complexity, STLformer adopted a $k$-d tree implementation for the LOESS regression and an FFT and Wiener-Khinchin theorem-based solution for RACF, both of which can be done in $O(N \log N)$ time. This makes STLformer more efficient than the original Transformer when dealing with long sequences.

## V. CONCLUSION

This paper proposes a new Transformer-based model, STLformer, for time series forecasting with a better decomposition strategy. The model leverages the well-known STL decomposition to extract trend and seasonal patterns and utilizes the Rank Correlation for nonlinear serial dependencies. Experiments on two real-world datasets demonstrate that STLformer marginally outperforms the previous Rankformer, which solely corporates moving averages for decomposition, and Autoformer, which relies on linear serial dependencies, proving the effectiveness of the STL decomposition on certain datasets. We believe that our STL decomposition block can be integrated into other Transformer-based models to enhance their performance by performing a more sophisticated series decomposition. In future work, we plan to explore the robust LOESS for the presence of perturbations and investigate the possibility of integrating other nonlinear serial dependencies measurement methods, such as Generalized ARCH (GARCH), to further enhance the performance of STLformer.

TABLE II
FORECASTING RESULTS FOR DIFFERENT MODELS ON DIFFERENT FORECAST HORIZONS

| Models | | STLformer | | Rankformer [24] | | Autoformer [23] | | Informer [22] | | LogTrans [25] | | Reformer [26] | | LSTNet [15] | | LSTM [33] | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETT | 24 | **0.142** | **0.255** | 0.149 | 0.258 | 0.153 | 0.261 | 0.173 | 0.301 | 0.211 | 0.332 | 0.333 | 0.429 | 1.101 | 0.831 | 0.580 | 0.572 |
| | 48 | **0.178** | **0.281** | 0.183 | 0.281 | 0.178 | 0.280 | 0.303 | 0.409 | 0.427 | 0.487 | 0.558 | 0.571 | 2.619 | 1.393 | 0.747 | 0.630 |
| | 96 | **0.209** | **0.298** | 0.221 | 0.302 | 0.255 | 0.339 | 0.365 | 0.453 | 0.768 | 0.642 | 0.658 | 0.619 | 3.142 | 1.365 | 2.041 | 1.073 |
| | 288 | **0.295** | **0.356** | 0.314 | 0.357 | 0.342 | 0.378 | 1.047 | 0.804 | 1.090 | 0.806 | 2.441 | 1.190 | 2.856 | 1.329 | 0.969 | 0.742 |
| Exchange | 96 | **0.151** | **0.279** | 0.162 | 0.290 | 0.197 | 0.323 | 0.847 | 0.752 | 0.968 | 0.812 | 1.065 | 0.829 | 1.551 | 1.058 | 1.453 | 1.049 |
| | 192 | **0.238** | **0.356** | 0.251 | 0.365 | 0.300 | 0.369 | 1.204 | 0.895 | 1.040 | 0.851 | 1.188 | 0.906 | 1.477 | 1.028 | 1.846 | 1.179 |
| | 336 | **0.419** | **0.478** | 0.428 | 0.486 | 0.509 | 0.524 | 1.672 | 1.036 | 1.659 | 1.081 | 1.357 | 0.976 | 1.507 | 1.031 | 2.136 | 1.231 |
| | 720 | **1.098** | **0.813** | 1.157 | 0.837 | 1.447 | 0.941 | 2.478 | 1.310 | 1.941 | 1.127 | 1.510 | 1.016 | 2.285 | 1.243 | 2.984 | 1.427 |
| Weather | 96 | 0.264 | 0.333 | **0.263** | **0.332** | 0.266 | 0.336 | 0.300 | 0.384 | 0.458 | 0.490 | 0.689 | 0.596 | 0.594 | 0.587 | 0.369 | 0.406 |
| | 192 | 0.310 | 0.365 | **0.298** | **0.356** | 0.307 | 0.367 | 0.598 | 0.544 | 0.658 | 0.589 | 0.752 | 0.638 | 0.560 | 0.565 | 0.416 | 0.435 |
| | 336 | 0.350 | 0.394 | **0.350** | **0.390** | 0.359 | 0.395 | 0.578 | 0.523 | 0.797 | 0.652 | 0.639 | 0.596 | 0.597 | 0.587 | 0.455 | 0.454 |
| | 720 | 0.433 | 0.440 | 0.430 | 0.435 | **0.419** | **0.428** | 1.059 | 0.741 | 0.869 | 0.675 | 1.130 | 0.792 | 0.618 | 0.599 | 0.535 | 0.520 |
| ILI | 24 | 3.690 | 1.353 | 3.556 | 1.319 | **3.483** | **1.287** | 5.764 | 1.677 | 4.480 | 1.444 | 4.400 | 1.382 | 6.026 | 1.770 | 5.914 | 1.734 |
| | 36 | 3.012 | 1.133 | **2.821** | **1.112** | 3.103 | 1.148 | 4.755 | 1.467 | 4.799 | 1.467 | 4.783 | 1.448 | 5.340 | 1.668 | 6.631 | 1.845 |
| | 48 | 3.134 | 1.198 | 2.907 | 1.144 | **2.669** | **1.085** | 4.763 | 1.469 | 4.800 | 1.468 | 4.832 | 1.465 | 6.080 | 1.787 | 6.736 | 1.857 |
| | 60 | 3.531 | 1.308 | 3.232 | 1.239 | **2.770** | **1.125** | 5.264 | 1.564 | 5.278 | 1.560 | 4.882 | 1.483 | 5.548 | 1.720 | 6.870 | 1.879 |

## REFERENCES


[1] L. Longo, M. Riccaboni, and A. Rungi, "A neural network ensemble approach for GDP forecasting," *J. Econ. Dyn. Control*, vol. 134, p. 104 278, 2022.

[2] X. Shi *et al.*, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Proc. NeurIPS*, 2015.

[3] Y. Yaslan and B. Bican, "Empirical mode decomposition based denoising method with support vector regression for time series prediction: A case study for electricity load forecasting," *Measurement*, vol. 103, pp. 52–61, 2017.

[4] J. Zuo *et al.*, "Graph convolutional networks for traffic forecasting with missing values," *Data. Min. Knowl. Disc.*, vol. 37, no. 2, pp. 913–947, 2023.

[5] V. Assimakopoulos and K. Nikolopoulos, "The theta model: A decomposition approach to forecasting," *Int. J. Forecast.*, vol. 16, no. 4, pp. 521–530, 2000.

[6] G. E. P. Box *et al.*, *Time Series Analysis: Forecasting and Control*, 5th ed. John Wiley & Sons, Inc., 2015.

[7] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed. OTexts, 2021.

[8] Z. Ouyang, P. Ravier, and M. Jabloun, "STL Decomposition of Time Series Can Benefit Forecasting Done by Statistical Methods but Not by Machine Learning Ones," *Eng. Proc.*, vol. 5, no. 1, p. 42, 2021.

[9] A. L. Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *Proc. ECML/PKDD*, 2016.

[10] J. F. Torres *et al.*, "Deep Learning for Time Series Forecasting: A Survey," *Big Data*, vol. 9, no. 1, pp. 3–21, 2021.

[11] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," *Int. J. Forecast.*, vol. 37, no. 1, pp. 388–427, 2021.

[12] K. Benidis *et al.*, "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey," *ACM Comput. Surv.*, vol. 55, no. 6, 121:1–121:36, 2022.

[13] Z. Ouyang, P. Ravier, and M. Jabloun, "Are Deep Learning Models Practically Good as Promised? A Strategic Comparison of Deep Learning Models for Time Series Forecasting," in *Proc. EUSIPCO*, 2022.

[14] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," in *Proc. ICLR*, 2018.

[15] G. Lai *et al.*, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *Proc. ACM SIGIR*, 2018.

[16] D. Salinas *et al.*, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecast.*, vol. 36, no. 3, pp. 1181–1191, 2020.

[17] B. N. Oreshkin *et al.*, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proc. ICLR*, 2020.

[18] A. Vaswani *et al.*, "Attention is All you Need," in *Proc. NeurIPS*, 2017.

[19] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL*, 2019.

[20] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," in *Proc. NeurIPS*, 2020.

[21] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Proc. ICLR*, 2021.

[22] H. Zhou *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *Proc. AAAI*, 2021.

[23] H. Wu *et al.*, "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting," in *Proc. NeurIPS*, 2021.

[24] Z. Ouyang, M. Jabloun, and P. Ravier, "Rankformer: Leveraging Rank Correlation for Transformer-based Time Series Forecasting," in *Proc. IEEE SSP*, 2023.

[25] S. Li *et al.*, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting," in *Proc. NeurIPS*, 2019.

[26] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," *Proc. ICLR*, 2020.

[27] W. S. Cleveland, "Robust Locally Weighted Regression and Smoothing Scatterplots," *J. Am. Stat. Assoc.*, vol. 74, no. 368, pp. 829–836, 1979.

[28] R. B. Cleveland *et al.*, "A Seasonal-Trend Decomposition Procedure Based on Loess," *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.

[29] M. Blondel *et al.*, "Fast Differentiable Sorting and Ranking," in *Proc. ICML*, 2020.

[30] R. F. Engle, "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation," *Econometrica*, vol. 50, no. 4, pp. 987–1007, 1982.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2014.

[32] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proc. NeurIPS*, 2019.

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