Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting

Anonymous Author(s)
Affiliation
Address
email

Abstract

Extending the forecasting time is a critical demand for real applications, such as extreme weather early warning and long-term energy consumption planning. This paper studies the *long-term forecasting* problem of time series. Prior Transformer-based models adopt various self-attention mechanisms to discover the long-range dependencies. However, intricate temporal patterns of the long-term future prohibit the model from finding reliable dependencies. Also, Transformers have to adopt the sparse versions of point-wise self-attentions for long series efficiency, resulting in the information utilization bottleneck. Towards these challenges, we propose *Autoformer* as a novel decomposition architecture with an *Auto-Correlation* mechanism. We go beyond the pre-processing convention of series decomposition and renovate it as a basic inner block of deep models. This design empowers Autoformer with progressive decomposition capacities for complex time series. Further, inspired by the stochastic process theory, we design the Auto-Correlation mechanism based on the series periodicity, which conducts the dependencies discovery and representation aggregation at the sub-series level. Auto-Correlation outperforms self-attention in both efficiency and accuracy. In long-term forecasting, Autoformer yields state-of-the-art accuracy, with a 38% relative improvement on six benchmarks, covering five practical applications: energy, traffic, economics, weather and disease.

1 Introduction

Time series forecasting has been widely used in energy consumption, traffic and economics planning, weather and disease propagation forecasting. In these real-world applications, one pressing demand is to extend the forecast time into the far future. Thus, in this paper, we study the *long-term forecasting* problem of time series, characterizing itself by the large length of predicted series, which can be severalfold of the length of the input series. Recent deep forecasting models [40, 16, 19, 34, 26, 18] have achieved great progress, especially the Transformer-based models. Benefiting from the self-attention mechanism, Transformers obtain great advantage in modeling long-term dependencies for sequential data, which enables more powerful big models [7, 10].

However, the forecasting task is extremely challenging under the long-term setting. First, it is unreliable to discover the temporal dependencies directly from the long-term time series because the dependencies can be obscured by entangled temporal patterns. Second, canonical Transformers with self-attention mechanisms are computationally prohibitive for long-term forecasting because of the quadratic complexity of sequence length. Previous Transformer-based forecasting models [40, 16, 19] mainly focus on improving self-attention to a *sparse* version. While performance is significantly improved, these models still utilize the point-wise representation aggregation. Thus, in the process of efficiency improvement, they will sacrifice the information utilization because of the sparse point-wise connections, resulting in a bottleneck for long-term forecasting of time series.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

To reason about the intricate temporal patterns, we try to take the idea of decomposition, which is a standard method in time series analysis [1, 25]. It can be used to process the complex time series and extract more predictable components. However, under the forecasting context, it can only be used as the *pre-processing* of past series because the future is unknown [14]. This common usage limits the capabilities of decomposition and overlooks the potential future interactions among decomposed components. Thus, we attempt to go beyond pre-processing usage of decomposition and propose a generic architecture to empower the deep forecasting models with immanent capacity of progressive decomposition. Further, decomposition can ravel out the entangled temporal patterns and highlight the inherent properties of time series [14]. Benefiting from this, we try to take advantage of the series periodicity to renovate the point-wise connection in self-attention. We observe that the sub-series at the same phase position among periods often present similar temporal processes. Thus, we try to construct a series-level connection based on the process similarity derived by series periodicity.

Based on the above motivations, we go beyond Transformer and propose a novel **Autoformer** for long-term time series forecasting. Autoformer still follows residual and encoder-decoder structure but renovates Transformer into a decomposition forecasting architecture. By embedding our proposed decomposition blocks as the inner operators, Autoformer can progressively separate the long-term trend information from predicted hidden variables. This design allows our model to alternately decompose and refine the intermediate results during the forecasting procedure. Inspired by the stochastic process theory [8, 22], Autoformer introduces an **Auto-Correlation** mechanism in place of self-attention, which discovers the sub-series similarity based on the series periodicity and aggregates similar sub-series from underlying periods. This series-wise mechanism achieves $\mathcal{O}(L \log L)$ complexity for length-L series and breaks the information utilization bottleneck by expanding the point-wise representation aggregation to sub-series level. Autoformer achieves the state-of-the-art accuracy on six benchmarks. The contributions are summarized as follows:

- To tackle the intricate temporal patterns of the long-term future, we present *Autoformer* as a decomposition architecture and design the inner decomposition block to empower the deep forecasting model with immanent progressive decomposition capacity.
- We propose an *Auto-Correlation* mechanism with dependencies discovery and information aggregation at the series level. Our mechanism is beyond previous self-attention family and can simultaneously benefit the computation efficiency and information utilization.
- Autoformer achieves a 38% relative improvement under the long-term setting on six benchmarks, covering five real-world applications: energy, traffic, economics, weather and disease.

2 Related Work

2.1 Models for Time Series Forecasting

Due to the immense importance of time series forecasting, various models have been well developed. Many time series forecasting methods start from the classic tools [31]. ARIMA [6, 4] tackles the forecasting problem by transforming the non-stationary process to stationary through differencing. The filtering method is also introduced for series forecasting [17, 9]. Besides, recurrent neural networks (RNNs) models are used to model the temporal dependencies for time series [35, 24, 39, 21]. DeepAR [26] combines autoregressive methods and RNNs to model the probabilistic distribution of future series. LSTNet [18] introduces convolutional neural networks (CNNs) with recurrent-skip connections to capture the short-term and long-term temporal patterns. Attention-based RNNs [38, 29, 30] introduce the temporal attention to explore the long-range dependencies for prediction. Also, many works based on temporal convolution networks (TCN) [33, 5, 3, 27, 28] attempt to model the temporal causality with the causal convolution. These deep forecasting models mainly focus on the temporal relation modeling by recurrent connections, temporal attention or causal convolution.

Recently, Transformers [34, 37] based on the self-attention mechanism shows great power in sequential data, such as natural language processing [10, 7], audio processing [13] and even computer vision [11, 20]. However, applying self-attention to long-term time series forecasting is computationally prohibitive because of the quadratic complexity of sequence length L in both memory and time. LogTrans [19] introduces the local convolution to Transformer and proposes the LogSparse attention to select time steps following the exponentially increasing intervals, which reduces the complexity to $\mathcal{O}(L(\log L)^2)$. Reformer [16] presents the local-sensitive hashing (LSH) attention and reduces the complexity to $\mathcal{O}(L \log L)$. Informer [40] extends Transformer with KL-divergence based ProbSparse

attention and also achieves $\mathcal{O}(L \log L)$ complexity. Note that these methods are based on the vanilla Transformer and try to improve the self-attention mechanism to a *sparse* version, which still follows the point-wise dependency and aggregation. In this paper, our proposed Auto-Correlation mechanism is based on the inherent periodicity of time series and can provide series-wise connections.

2.2 Decomposition of Time Series

As a standard method in time series analysis, time series decomposition [1, 25] deconstructs a time series into several components, each representing one of the underlying categories of patterns that are more predictable. It is primarily useful for exploring historical changes over time. For the forecasting tasks, decomposition is always used as the *pre-processing* of historical series before predicting future series [14], such as Prophet [32] and others [2]. However, such pre-processing is limited by the plain decomposition effect of historical series and overlooks the hierarchical interaction between the underlying patterns of series in the long-term future. This paper takes the decomposition idea from a new progressive dimension. Our Autoformer harnesses the decomposition as an inner block of deep models, which progressively decomposes the hidden series throughout the whole forecasting process.

3 Autoformer

The time series forecasting problem is to predict the most probable length-O series in the future given the past length-I series, denoting as *input-I-predict-O*. The *long-term forecasting* setting is to predict the long-term future given the short-term history, i.e. $O \gg I$. As aforementioned, we have highlighted the difficulties of long-term series forecasting: handling intricate temporal patterns and breaking the bottleneck of computation efficiency and information utilization. To tackle these two challenges, we introduce the decomposition as a builtin block to the deep forecasting model and propose *Autoformer* as a decomposition architecture. Besides, we design the *Auto-Correlation* mechanism to discover the period-based dependencies and aggregate similar sub-series from underlying periods.

3.1 Decomposition Architecture

We renovate Transformer [34] to a deep decomposition architecture (Figure 1), including the inner series decomposition block, Auto-Correlation mechanism, and corresponding Encoder and Decoder.

Series decomposition block To learn with the complex temporal patterns in long-term forecasting context, we take the idea of decomposition [1, 25], which can separate the series into trend-cyclical and seasonal parts. These two parts reflect the long-term progression and the seasonality of the series respectively. However, directly decomposing is unrealizable for future series because the future is just unknown. To tackle this dilemma, we present a *series decomposition block* as an inner operation of Autoformer (Figure 1), which can extract the long-term stationary trend from predicted intermediate hidden variables progressively. Concretely, we adapt the moving average to smooth out periodic fluctuations and highlight the long-term trends. For length-L input series $\mathcal{X} \in \mathbb{R}^{L \times d}$, the process is:

$$\begin{aligned} \mathcal{X}_{\mathrm{t}} &= \mathrm{AvgPool}(\mathrm{Padding}(\mathcal{X})) \\ \mathcal{X}_{\mathrm{s}} &= \mathcal{X} - \mathcal{X}_{\mathrm{t}}, \end{aligned} \tag{1}$$

where $\mathcal{X}_{\mathrm{s}}, \mathcal{X}_{\mathrm{t}} \in \mathbb{R}^{L \times d}$ denote the seasonal and the extracted trend-cyclical part respectively. We use $\mathcal{X}_{\mathrm{s}}, \mathcal{X}_{\mathrm{t}} = \mathrm{SeriesDecomp}(\mathcal{X})$ to summarize above equations, which is the inner block of Autoformer.

Model inputs The inputs of encoder part are the past I time steps $\mathcal{X}_{\mathrm{en}} \in \mathbb{R}^{I \times d}$. As a decomposition architecture (Figure 1), the input of Autoformer decoder contains both the seasonal part $\mathcal{X}_{\mathrm{des}} \in \mathbb{R}^{(\frac{I}{2}+O) \times d}$ and trend-cyclical part $\mathcal{X}_{\mathrm{det}} \in \mathbb{R}^{(\frac{I}{2}+O) \times d}$ to be refined. Each initialization consists of two parts: the component decomposed from the latter half of encoder's input $\mathcal{X}_{\mathrm{en}}$ with length $\frac{I}{2}$ to provide recent information, placeholders with length O filled by scalars. It's formulized as follows:

$$\begin{aligned} \mathcal{X}_{\mathrm{ens}}, \mathcal{X}_{\mathrm{ent}} &= \mathrm{SeriesDecomp}(\mathcal{X}_{\mathrm{en}\,\frac{I}{2}:I}) \\ \mathcal{X}_{\mathrm{des}} &= \mathrm{Concat}(\mathcal{X}_{\mathrm{ens}}, \mathcal{X}_{0}) \\ \mathcal{X}_{\mathrm{det}} &= \mathrm{Concat}(\mathcal{X}_{\mathrm{ent}}, \mathcal{X}_{\mathrm{Mean}}), \end{aligned} \tag{2}$$

where $\mathcal{X}_{\mathrm{ens}}, \mathcal{X}_{\mathrm{ent}} \in \mathbb{R}^{\frac{I}{2} \times d}$ denote the seasonal and trend-cyclical parts of $\mathcal{X}_{\mathrm{en}}$ respectively, and $\mathcal{X}_{0}, \mathcal{X}_{\mathrm{Mean}} \in \mathbb{R}^{O \times d}$ denote the placeholders filled with zero and mean of $\mathcal{X}_{\mathrm{en}}$ respectively.

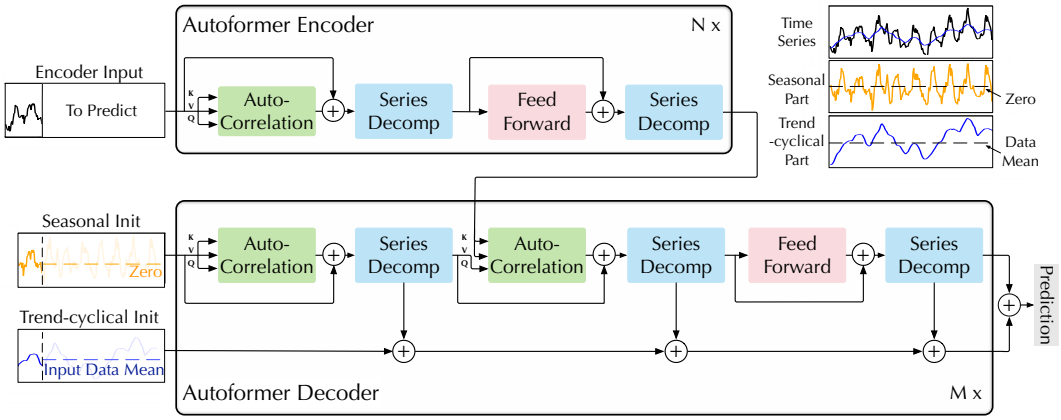


Figure 1: Autoformer architecture. The encoder eliminates the long-term trend-cyclical part by series decomposition blocks (blue blocks) and focuses on seasonal patterns modeling. The decoder accumulates the trend part extracted from hidden variables progressively. The past seasonal information from encoder is utilized by the encoder-decoder Auto-Correlation (center green block in decoder).

Encoder As shown in Figure 1, the encoder focuses on the seasonal part modeling. The output of the encoder contains the past seasonal information and will be used as the cross information to help the decoder refine prediction results. Suppose we have N encoder layers. The overall equations for l-th encoder layer are summarized as $\mathcal{X}_{\text{en}}^{l} = \text{Encoder}(\mathcal{X}_{\text{en}}^{l-1})$. Details are shown as follows:

$$\begin{aligned} \mathcal{S}_{\text{en}}^{l,1}, _ &= \text{SeriesDecomp}\Big(\text{AutoCorrelation}(\mathcal{X}_{\text{en}}^{l-1}) + \mathcal{X}_{\text{en}}^{l-1}\Big) \\ \mathcal{S}_{\text{en}}^{l,2}, _ &= \text{SeriesDecomp}\Big(\text{FeedForward}(\mathcal{S}_{\text{en}}^{l,1}) + \mathcal{S}_{\text{en}}^{l,1}\Big), \end{aligned} \tag{3}$$

where "_" is the eliminated trend part. $\mathcal{X}_{\text{en}}^{l} = \mathcal{S}_{\text{en}}^{l,2}, l \in \{1, \cdots, N\}$ denotes the output of l-th encoder layer and $\mathcal{X}_{\text{en}}^{0}$ is the embedded $\mathcal{X}_{\text{en}}$. $\mathcal{S}_{\text{en}}^{l,i}, i \in \{1, 2\}$ represents the seasonal component after the i-th series decomposition block in the l-th layer respectively. We will give detailed description of $\text{AutoCorrelation}(\cdot)$ in the next section, which can seamlessly replace the self-attention.

Decoder The decoder contains two parts: the accumulation structure for trend-cyclical components and the stacked Auto-Correlation mechanism for seasonal components (Figure 1). Each decoder layer contains the inner Auto-Correlation and encoder-decoder Auto-Correlation, which can refine the prediction and utilize the past seasonal information respectively. Note that the model extracts the potential trend from the intermediate hidden variables during the decoder, allowing Autoformer to progressively refine the trend prediction and eliminate interference information for period-based dependencies discovery in Auto-Correlation. Suppose there are M decoder layers. With the latent variable $\mathcal{X}_{\text{en}}^{N}$ from the encoder, the equations of l-th decoder layer can be summarized as $\mathcal{X}_{\text{de}}^{l} = \text{Decoder}(\mathcal{X}_{\text{de}}^{l-1}, \mathcal{X}_{\text{en}}^{N})$. The decoder can be formalized as follows:

$$\begin{aligned} \mathcal{S}_{\text{de}}^{l,1}, \mathcal{T}_{\text{de}}^{l,1} &= \text{SeriesDecomp}\Big(\text{AutoCorrelation}(\mathcal{X}_{\text{de}}^{l-1}) + \mathcal{X}_{\text{de}}^{l-1}\Big) \\ \mathcal{S}_{\text{de}}^{l,2}, \mathcal{T}_{\text{de}}^{l,2} &= \text{SeriesDecomp}\Big(\text{AutoCorrelation}(\mathcal{S}_{\text{de}}^{l,1}, \mathcal{X}_{\text{en}}^{N}) + \mathcal{S}_{\text{de}}^{l,1}\Big) \\ \mathcal{S}_{\text{de}}^{l,3}, \mathcal{T}_{\text{de}}^{l,3} &= \text{SeriesDecomp}\Big(\text{FeedForward}(\mathcal{S}_{\text{de}}^{l,2}) + \mathcal{S}_{\text{de}}^{l,2}\Big) \\ \mathcal{T}_{\text{de}}^{l} &= \mathcal{T}_{\text{de}}^{l-1} + \mathcal{W}_{l,1} * \mathcal{T}_{\text{de}}^{l,1} + \mathcal{W}_{l,2} * \mathcal{T}_{\text{de}}^{l,2} + \mathcal{W}_{l,3} * \mathcal{T}_{\text{de}}^{l,3}, \end{aligned} \tag{4}$$

where $\mathcal{X}_{\text{de}}^{l} = \mathcal{S}_{\text{de}}^{l,3}, l \in \{1, \cdots, M\}$ denotes the output of l-th decoder layer. $\mathcal{X}_{\text{de}}^{0}, \mathcal{T}_{\text{de}}^{0}$ denote the embedded $\mathcal{X}_{\text{des}}, \mathcal{X}_{\text{det}}$ respectively. And $\mathcal{S}_{\text{de}}^{l,i}, \mathcal{T}_{\text{de}}^{l,i}, i \in \{1, 2, 3\}$ represent the seasonal component and trend-cyclical component after the i-th series decomposition block in the l-th layer respectively. $\mathcal{W}_{l,i}, i \in \{1, 2, 3\}$ represents the linear projector for the i-th extracted trend $\mathcal{T}_{\text{de}}^{l,i}$ as the adaptor.

The final prediction is the sum of the refined decomposed components: $\mathcal{W}_{\mathcal{S}} * \mathcal{X}_{\text{de}}^{M} + \mathcal{W}_{\mathcal{T}} * \mathcal{T}_{\text{de}}^{M}$, where $\mathcal{W}_{\mathcal{S}}$ and $\mathcal{W}_{\mathcal{T}}$ denote the projector for seasonal and trend-cyclical components respectively.

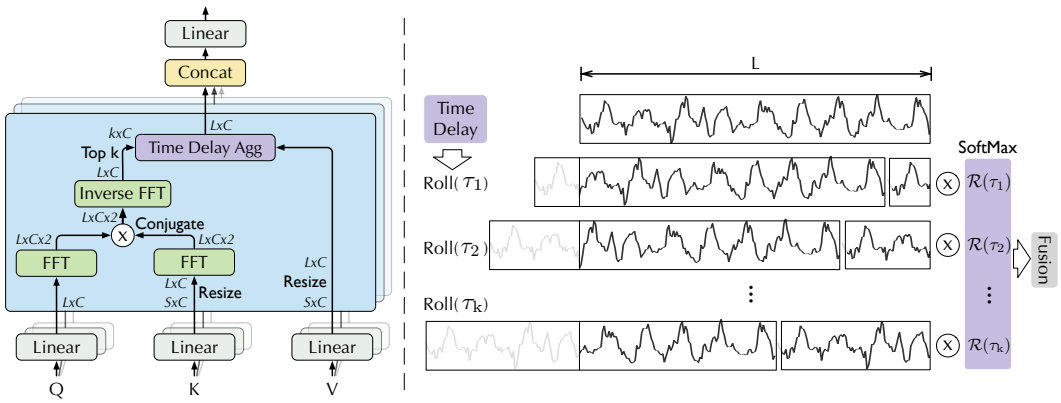


Figure 2: Auto-Correlation (left) and Time Delay Aggregation (right). We utilize the Fast Fourier Transform to calculate the autocorrelation $\mathcal{R}(\tau)$, which reflects the time-delay similarities. Then the similar sub-processes are rolled to the same index based on selected delay τ and aggregated by $\mathcal{R}(\tau)$.

3.2 Auto-Correlation Mechanism

As shown in Figure 2, we propose the Auto-Correlation mechanism with series-wise connections to expand the information utilization. Auto-Correlation discovers the period-based dependencies by calculating the series autocorrelation and aggregates similar sub-series by time delay aggregation.

Period-based dependencies It is observed that the same phase position among periods naturally provides similar sub-processes. Inspired by the stochastic process theory [8, 22], for a real discrete-time process $\{\mathcal{X}_t\}$, we can obtain the autocorrelation $\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$ by the following equations:

$$\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau) = \lim_{L\to\infty} \frac{1}{L} \sum_{t=0}^{L-1} \mathcal{X}_t \mathcal{X}_{t-\tau}. \tag{5}$$

$\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$ reflects the time-delay similarity between $\{\mathcal{X}_t\}$ and its τ lag series $\{\mathcal{X}_{t-\tau}\}$. As shown in Figure 2, we use the autocorrelation $\mathcal{R}(\tau)$ as the unnormalized confidence of estimated period length τ. Then, we choose the most possible k period lengths $\tau_1, \cdots, \tau_k$. The period-based dependencies are derived by the above estimated periods and can be weighted by the corresponding autocorrelation.

Time delay aggregation The period-based dependencies connect the sub-series among estimated periods. Thus, we present the *time delay aggregation* block (Figure 2), which can roll the series based on selected time delay $\tau_1, \cdots, \tau_k$. This operation can align similar sub-series that are at the same phase position of estimated periods, which is different from the point-wise dot-product aggregation in self-attention family. Finally, we aggregate the sub-series by softmax normalized confidences.

For the single head situation and time series $\mathcal{X}$ with length-L, after the projector, we get query $\mathcal{Q}$, key $\mathcal{K}$ and value $\mathcal{V}$. Thus, it can replace self-attention seamlessly. The Auto-Correlation mechanism is:

$$\begin{aligned} \tau_1, \cdots, \tau_k &= \underset{\tau \in \{1,\cdots,L\}}{\arg\operatorname{Topk}} \left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}(\tau)\right) \\ \widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}(\tau_1), \cdots, \widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}(\tau_k) &= \operatorname{SoftMax}\left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}(\tau_1), \cdots, \mathcal{R}_{\mathcal{Q},\mathcal{K}}(\tau_k)\right) \\ \operatorname{AutoCorrelation}(\mathcal{Q},\mathcal{K},\mathcal{V}) &= \sum_{i=1}^{k} \operatorname{Roll}(\mathcal{V},\tau_k)\widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}(\tau_k), \end{aligned} \tag{6}$$

where $\arg\operatorname{Topk}(\cdot)$ is to get the arguments of the Topk autocorrelations and let $k = \lfloor c \times \log L \rfloor$, c is a hyper-parameter. $\mathcal{R}_{\mathcal{Q},\mathcal{K}}$ is autocorrelation between series $\mathcal{Q}$ and $\mathcal{K}$. $\operatorname{Roll}(\mathcal{X},\tau)$ represents the operation to $\mathcal{X}$ with time delay τ, during which elements that are shifted beyond the first position are re-introduced at the last position. For the encoder-decoder Auto-Correlation (Figure 1), $\mathcal{K}, \mathcal{V}$ are from the encoder $\mathcal{X}_{\text{en}}^{N}$ and will be resized to length-O, $\mathcal{Q}$ is from the previous block of the decoder.

For the multi-head version used in Autoformer, with hidden variables of d_{model} channels and h heads, the query, key and value for i-th head are $\mathcal{Q}_i, \mathcal{K}_i, \mathcal{V}_i \in \mathbb{R}^{L \times \frac{d_{\text{model}}}{h}}, i \in \{1, \cdots, h\}$. The process is:

$$\begin{aligned} \operatorname{MultiHead}(\mathcal{Q},\mathcal{K},\mathcal{V}) &= \operatorname{Concat}(\operatorname{head}_1, \cdots, \operatorname{head}_h) \\ \text{where } \operatorname{head}_i &= \operatorname{AutoCorrelation}(\mathcal{Q}_i,\mathcal{K}_i,\mathcal{V}_i). \end{aligned} \tag{7}$$

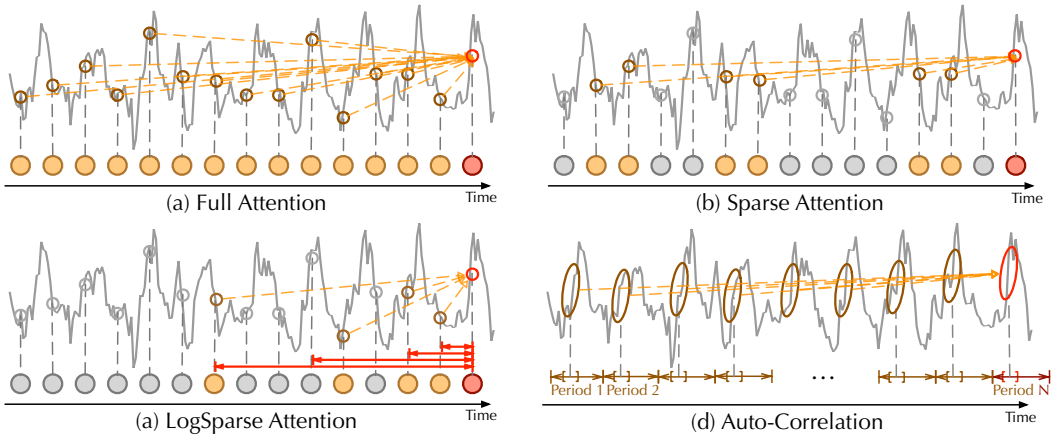


Figure 3: Auto-Correlation vs. self-attention family. Full Attention [34] (a) adapts the fully connection among all time points. Sparse Attention [16, 40] (b) selects points based on the proposed similarity metrics. LogSparse Attention [19] (c) chooses points following the exponentially increasing intervals. Auto-Correlation (d) focuses on the connections of sub-series among underlying periods.

Efficient computation For period-based dependencies, these dependencies point to processes at the same phase position of underlying periods and are inherently sparse. Here, we select the most possible τ to avoid picking the opposite phases. Thus, the complexity of Equation 6,7 is $\mathcal{O}(L \log L)$. For the autocorrelation computation (Equation 5), given time series $\{\mathcal{X}_t\}$, $\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$ can be calculated by Fast Fourier Transforms (FFT) based on the Wiener–Khinchin theorem [36]:

$$\begin{aligned} \mathcal{S}_{\mathcal{X}\mathcal{X}}(f) &= \mathcal{F}\left(\mathcal{X}_t\right) \mathcal{F}^{*}\left(\mathcal{X}_t\right) = \int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi t f} \mathrm{d}t \overline{\int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi t f} \mathrm{d}t} \\ \mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau) &= \mathcal{F}^{-1}\left(\mathcal{S}_{\mathcal{X}\mathcal{X}}(f)\right) = \int_{-\infty}^{\infty} \mathcal{S}_{\mathcal{X}\mathcal{X}}(f) e^{i2\pi f \tau} \mathrm{d}f, \end{aligned} \tag{8}$$

where $\tau \in \{1, \cdots, L\}$, $\mathcal{F}$ denotes the FFT and $\mathcal{F}^{-1}$ is its inverse. $*$ denotes the conjugate operation and $\mathcal{S}_{\mathcal{X}\mathcal{X}}(f)$ is in the frequency domain. Note that the series autocorrelation of all lags in $\{1, \cdots, L\}$ can be calculated at once by FFT. Thus, Auto-Correlation achieves the $\mathcal{O}(L \log L)$ complexity.

Auto-Correlation vs. self-attention family Different from the point-wise self-attention family, Auto-Correlation presents the series-wise connections (Figure 3). Concretely, for the temporal dependencies, we find the dependencies among sub-series based on the periodicity. In contrast, the self-attention family only calculates the relation between scattered points. Though some self-attentions [19, 40] consider the local information, they only utilize this to help point-wise dependencies discovery. For the information aggregation, we adopt the time delay block to aggregate the similar sub-series from underlying periods. In contrast, self-attentions aggregate the selected points by dot-product. Benefiting from the inherent sparsity and sub-series-level representation aggregation, Auto-Correlation can simultaneously benefit the computation efficiency and information utilization.

4 Experiments

We extensively evaluate the proposed Autoformer on six real-world benchmarks, covering five mainstream time series forecasting applications: energy, traffic, economics, weather and disease.

Datasets Here is a description of the six experiment datasets: (1) *ETT* [40] dataset contains the data collected from electricity transformers, including load and oil temperature that are recorded every 15 minutes between July 2016 and July 2018. (2) *Electricity*[1] dataset contains the hourly electricity consumption of 321 customers from 2012 to 2014. (3) *Exchange* [18] records the daily exchange rates of eight different countries, including Australia, British, Canada, Switzerland, China, Japan, New Zealand and Singapore, ranging from 1990 to 2016. (4) *Traffic*[2] is a collection of hourly data

[1] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[2] http://pems.dot.ca.gov

Table 1: Quantitative results with different prediction lengths $O \in \{96, 192, 336, 720\}$. We set the input length I as 24 for ILI and 96 for the others. A lower MSE or MAE indicates a better prediction.

Models		**Autoformer**		Informer[40]		LogTrans[19]		Reformer[16]		LSTNet[18]		LSTM[12]		TCN[3]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	**0.194**	**0.284**	0.343	0.444	0.768	0.642	0.658	0.619	3.142	1.365	2.041	1.073	3.041	1.330
	192	**0.261**	**0.323**	0.533	0.563	0.989	0.757	1.078	0.827	3.154	1.369	2.249	1.112	3.072	1.339
	336	**0.351**	**0.384**	1.363	0.887	1.334	0.872	1.549	0.972	3.160	1.369	2.568	1.238	3.105	1.348
	720	**0.491**	**0.470**	3.379	1.388	3.048	1.328	2.631	1.242	3.171	1.368	2.720	1.287	3.135	1.354
Electricity	96	**0.197**	**0.312**	0.274	0.368	0.258	0.357	0.312	0.402	0.680	0.645	0.375	0.437	0.985	0.813
	192	**0.208**	**0.321**	0.296	0.386	0.266	0.368	0.348	0.433	0.725	0.676	0.442	0.473	0.996	0.821
	336	**0.213**	**0.328**	0.300	0.394	0.280	0.380	0.350	0.433	0.828	0.727	0.439	0.473	1.000	0.824
	720	**0.245**	**0.352**	0.373	0.439	0.283	0.376	0.340	0.420	0.957	0.811	0.980	0.814	1.438	0.784
Exchange	96	**0.134**	**0.270**	0.847	0.752	0.968	0.812	1.065	0.829	1.551	1.058	1.453	1.049	3.004	1.432
	192	**0.272**	**0.374**	1.204	0.895	1.040	0.851	1.188	0.906	1.477	1.028	1.846	1.179	3.048	1.444
	336	**0.488**	**0.510**	1.672	1.036	1.659	1.081	1.357	0.976	1.507	1.031	2.136	1.231	3.113	1.459
	720	**1.367**	**0.901**	2.478	1.310	1.941	1.127	1.510	1.016	2.285	1.243	2.984	1.427	3.150	1.458
Traffic	96	**0.609**	**0.376**	0.719	0.391	0.684	0.384	0.732	0.423	1.107	0.685	0.843	0.453	1.438	0.784
	192	**0.637**	**0.377**	0.696	0.379	0.685	0.390	0.733	0.420	1.157	0.706	0.847	0.453	1.463	0.794
	336	**0.634**	**0.394**	0.777	0.420	0.733	0.408	0.742	0.420	1.216	0.730	0.853	0.455	1.479	0.799
	720	**0.646**	**0.390**	0.864	0.472	0.717	0.396	0.755	0.423	1.481	0.805	1.500	0.805	1.499	0.804
Weather	96	**0.231**	**0.312**	0.300	0.384	0.458	0.490	0.689	0.596	0.594	0.587	0.369	0.406	0.615	0.589
	192	**0.278**	**0.343**	0.598	0.544	0.658	0.589	0.752	0.638	0.560	0.565	0.416	0.435	0.629	0.600
	336	**0.335**	**0.378**	0.578	0.523	0.797	0.652	0.639	0.596	0.597	0.587	0.455	0.454	0.639	0.608
	720	**0.429**	**0.436**	1.059	0.741	0.869	0.675	1.130	0.792	0.618	0.599	0.535	0.520	0.639	0.610
ILI	24	**3.825**	**1.345**	4.388	1.360	4.322	1.381	4.366	1.382	6.026	1.770	5.914	1.734	6.624	1.830
	36	**3.319**	**1.216**	4.651	1.391	4.186	1.332	4.446	1.389	5.340	1.668	6.631	1.845	6.858	1.879
	48	**2.854**	**1.122**	4.581	1.419	4.476	1.411	4.572	1.436	6.080	1.787	6.736	1.857	6.968	1.892
	60	**3.227**	**1.232**	4.583	1.432	4.766	1.477	4.743	1.478	5.548	1.720	6.870	1.879	7.127	1.918

from California Department of Transportation, which describes the road occupancy rates measured by different sensors on San Francisco Bay area freeways. (5) *Weather*[3] is recorded every 10 minutes for 2020 whole year, which contains 21 meteorological indicators, such as air temperature, humidity, etc. (6) *ILI*[4] includes the weekly recorded influenza-like illness (ILI) patients data from Centers for Disease Control and Prevention of the United States between 2002 and 2021, which describes the ratio of patients seen with ILI and the total number of the patients. We follow standard protocol and split all datasets into training, validation and test set in chronological order by the ratio of 7:1:2.

Implementation details Our method is trained with the L2 loss, using the ADAM [15] optimizer with an initial learning rate of 10^{-4}. And batch size is set to 32. The training process is stopped after 5 epochs. All experiments are repeated three times, implemented in PyTorch [23] and conducted on a single NVIDIA TITAN RTX 24GB GPUs. The hyper-parameter c of time delay aggregation is set to 3 for all experiments (supplementary materials). Autoformer contains 2 encoder layers and 1 decoder layer. We select three latest state-of-the-art transformer-based models: Informer [40], Reformer [16], LogTrans [19], two RNN-based models: LSTNet [18], LSTM [12] and CNN-based TCN [3] as our baselines. All the experiments are multivariate forecasting. The univariate experiments are in supplementary materials with more baselines, such as DeepAR [26], ARMIA [1] and Prophet [32].

4.1 Main Results

Quantitative results We evaluate our Autoformer with different prediction lengths: 96, 192, 336 and 720 and the corresponding ratio of future and past $\frac{O}{I}$ are 1, 2, 3.5 and 7.5. It precisely meets the definition of long-term forecasting. As shown in Table 1, Autoformer achieves the consistent state-of-the-art performance in all benchmarks and all prediction length settings. Especially, under the input-96-predict-336 setting, compared to previous state-of-the-art results, Autoformer gives **74%** (1.334→0.351) MSE improvement in ETT, **24%** (0.280→0.213) in Electricity, **64%** (1.357→0.488)

[3] https://www.bgc-jena.mpg.de/wetter/

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

in Exchange, **14%** (0.733→0.634) in Traffic and **26%** (0.455→0.335) in Weather. For the input-24-predict-60 setting of ILI, Autoformer makes **30%** (4.583→3.227) MSE improvement. Overall, Autoformer yields a **38%** averaged MSE improvement among above settings. Note that Autoformer still provides remarkable improvements in the *Exchange* dataset that is **without obvious periodicity**. See supplementary materials for detailed showcases. Besides, we can also find that the performance of Autoformer changes quite steadily as the prediction length O increases. It means that Autoformer retains better **long-term robustness**, which is meaningful for real-world applications, such as weather early warning and long-term energy consumption planning.

Decomposition architecture With our proposed progressive decomposition architecture, other models can gain consistent promotion, especially as the prediction length O increases (Table 2). This verifies that our method can generalize to other models and release the capacity of other dependencies learning mechanisms, alleviate the distraction caused by intricate patterns. Besides, our architecture outperforms the pre-processing, although the latter employs a bigger model and more parameters. Especially, pre-decomposing may even bring negative effect because it neglects the interaction of components during long-term future, such as Transformer [34] predict-720, Informer [40] predict-336.

Table 2: Ablation of decomposition in ETT dataset with MSE metric. **Ours** adopts our progressive architecture into other models. **Sep** employs two models to forecast pre-decomposed seasonal and trend-cyclical components separately. Promotion is the MSE improvement compared to **Origin**.

Input-96	Transformer[34]			Informer[40]			LogTrans[16]			Reformer[19]			Promotion	
Predict-O	Origin	Sep	Ours	Origin	Sep	Ours	Origin	Sep	Ours	Origin	Sep	Ours	Sep	Ours
96	0.604	0.311	**0.204**	0.343	0.490	**0.354**	0.768	0.862	**0.231**	0.658	0.445	**0.218**	0.066	0.342
192	1.060	0.760	**0.266**	0.533	0.658	**0.432**	0.989	0.533	**0.378**	1.078	0.510	**0.336**	0.300	0.562
336	1.413	0.665	**0.375**	1.363	1.469	**0.847**	1.334	0.762	**0.362**	1.549	1.028	**0.366**	0.434	0.927
720	2.672	3.200	**0.537**	3.379	2.766	**2.482**	3.048	2.601	**0.539**	2.631	2.845	**0.502**	0.079	1.918

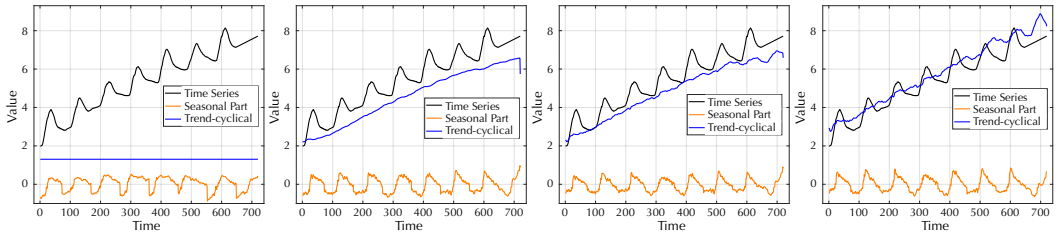
Auto-Correlation vs. self-attention family As shown in Table 3, our proposed Auto-Correlation achieves the best performance under various input-I-predict-O settings, which verifies the effectiveness of series-wise connections comparing to point-wise self-attentions (Figure 3). Furthermore, we can also observe that Auto-Correlation is memory efficiency from the last column of Table 3, which can be used in long sequence forecasting, such as input-336-predict-1440.

Table 3: Comparison of Auto-Correlation and self-attention in ETT dataset. We **replace** the Auto-Correlation in Autoformer with different self-attention modules. The "-" indicates the out-of-memory.

Input Length I		96			192			336		
Prediction Length O		336	720	1440	336	720	1440	336	720	1440
Auto-Correlation	MSE	**0.351**	**0.491**	**0.575**	**0.387**	**0.462**	**0.747**	**0.418**	**0.544**	**0.890**
	MAE	**0.384**	**0.470**	**0.527**	**0.414**	**0.461**	**0.603**	**0.453**	**0.522**	**0.691**
Full Attention[34]	MSE	0.375	0.537	0.667	0.450	0.702	0.951	0.462	0.647	-
	MAE	0.425	0.502	0.589	0.470	0.570	0.704	0.485	0.578	-
LogSparse Attention[19]	MSE	0.362	0.539	0.582	0.420	0.552	0.958	0.474	0.601	-
	MAE	0.413	0.522	0.529	0.450	0.513	0.736	0.474	0.524	-
LSH Attention[16]	MSE	0.366	0.502	0.663	0.407	0.636	1.069	0.442	0.615	-
	MAE	0.404	0.475	0.567	0.421	0.571	0.756	0.476	0.532	-
ProbSparse Attention[40]	MSE	0.847	2.482	1.501	0.482	0.996	1.756	0.479	0.684	1.168
	MAE	0.654	1.065	0.894	0.482	0.716	0.938	0.495	0.597	0.773

4.2 Model Analysis

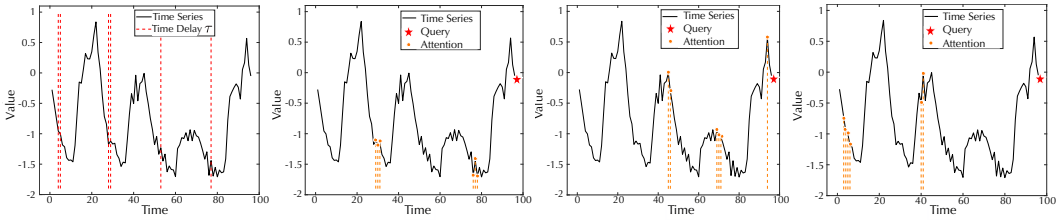
Time series decomposition As shown in Figure 4, without our series decomposition block, the forecasting model cannot capture the increasing trend and peaks of the seasonal part. By adding the series decomposition blocks, Autoformer can aggregate and refine the trend-cyclical part from series progressively. This design also facilitates the learning of the seasonal part, especially the peaks and troughs. This verifies the necessity of our proposed progressive decomposition architecture.



(a) Without decomposition block (b) One decomposition block (c) Two decomposition blocks (d) Three decomposition blocks

Figure 4: Visualization of learned seasonal $\mathcal{X}_{\text{de}}^{M}$ and trend-cyclical $\mathcal{T}_{\text{de}}^{M}$ of the last decoder layer. We gradually add the decomposition blocks in decoder from left to right. This case is from ETT dataset under input-96-predict-720 setting. For clearness, we add the linear growth to raw data additionally.

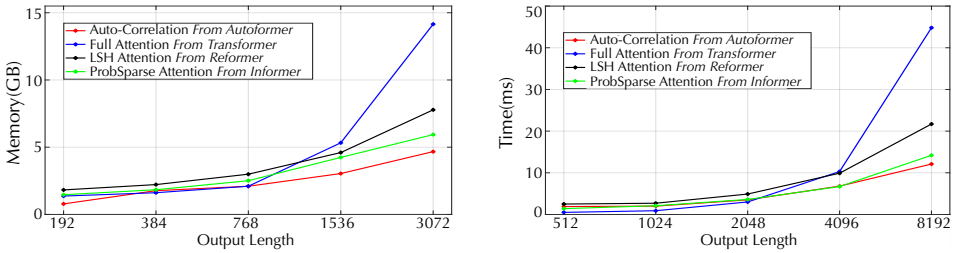
Dependencies learning The marked time delay sizes in Figure 5(a) indicate the most likely periods. Our learned periodicity can guide the model to aggregate the sub-series from the same or neighbor phase of periods by $\text{Roll}(\mathcal{X}, \tau_i),\ i \in \{1, \cdots, 6\}$. For the last time step (declining stage), Auto-Correlation fully utilizes all similar sub-series without omissions or errors compared to self-attentions.



(a) Auto-Correlation (b) Full Attention (c) LSH Attention (d) ProbSparse Attention

Figure 5: Visualization of learned dependencies. For clearness, we select the top-6 time delay sizes $\tau_1, \cdots, \tau_6$ of Auto-Correlation and mark them in raw series (red lines). For self-attentions, top-6 similar points with respect to the last time step (red stars) are also marked by orange points.

Efficiency analysis We compare the running memory and time among Auto-Correlation-based and self-attention-based models (Figure 6). The proposed Autoformer shows $\mathcal{O}(L \log L)$ complexity in both memory and time and achieves better long-term sequences efficiency.



(a) Memory Efficiency Analysis (b) Running Time Efficiency Analysis

Figure 6: Efficiency Analysis. For memory, we replace Auto-Correlation with self-attention family in Autoformer and record the memory with input 96. For running time, we run the Auto-Correlation or self-attentions 10^3 times to get the execution time per step. The output length increases exponentially.

5 Conclusions

This paper studies the long-term forecasting problem of time series, which is a pressing demand for real-world applications. However, the intricate temporal patterns prevent the model from learning reliable dependencies. We propose the Autoformer as a decomposition architecture by embedding the series decomposition block as an inner operator, which can progressively aggregate the long-term trend part from intermediate prediction. Besides, we design an efficient Auto-Correlation mechanism to conduct dependencies discovery and information aggregation at the series level, which contrasts clearly from the previous self-attention family. Autoformer can naturally achieve $\mathcal{O}(L \log L)$ complexity and yield consistent state-of-the-art performance in extensive real-world datasets.

References

[1] O. Anderson and M. Kendall. Time-series. 2nd edn. *The Statistician*, 25:308, 1976.

[2] Reza Asadi and Amelia C Regan. A spatio-temporal decomposition based deep neural network for time series forecasting. *Applied Soft Computing*, 87:105963, 2020.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[4] David J Bartholomew. Time series analysis forecasting and control., 1971.

[5] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[6] George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 17(2):91–109, 1968.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[8] Chris Chatfield. *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2003.

[9] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems*, 2020.

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

[11] 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 *International Conference on Learning Representations*, 2021.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[13] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations*, 2019.

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

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[16] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[17] Richard Kurle, Syama Sundar Rangapuram, Emmanuel de Bézenac, Stephan Günnemann, and Jan Gasthaus. Deep rao-blackwellised particle filters for time series forecasting. In *Advances in Neural Information Processing Systems*, 2020.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[19] 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 *Advances in Neural Information Processing Systems*, 2019.

[20] 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. *arXiv preprint arXiv:2103.14030*, 2021.

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

[22] Athanasios Papoulis and H Saunders. Probability, random variables and stochastic processes. 1989.

[23] 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 *Advances in Neural Information Processing Systems*, 2019.

[24] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, 2018.

[25] Cleveland Robert, C William, and Terpenning Irma. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics*, 6(1):3–73, 1990.

[26] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[27] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*, 2019.

[28] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[29] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.

[30] Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[31] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, 2007.

[32] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[33] Aäron van den Oord, S. Dieleman, H. Zen, K. Simonyan, Oriol Vinyals, A. Graves, Nal Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016.

[34] 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 *Advances in Neural Information Processing Systems*, 2017.

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

[36] Norbert Wiener. Generalized harmonic analysis. *Acta Math.*, 55(1):117–258, 1930.

[37] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. In *Advances in Neural Information Processing Systems*, 2020.

[38] Q. Yao, D. Song, H. Chen, C. Wei, and G. W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

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

[40] 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 *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 9 in supplementary materials.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 9 in 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]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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)? [No] Code will be made available. Datasets are already public.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Section 4 in the main text and Section 8 in supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Section 1, 2, and 4.3 in 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] Figure 6 and Section 4 in the 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]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include a new COVID-19 dataset in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We anonymous the country names of the additional COVID-19 dataset in supplementary materials.
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]