InParformer: Evolutionary Decomposition Transformers with Interactive Parallel Attention for Long-Term Time Series Forecasting

Haizhou Cao[1,2], **Zhenhao Huang**[3], **Tiechui Yao**[1,2], **Jue Wang**[1,2,*], **Hui He**[3], **Yangang Wang**[1,2]

[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]North China Electric Power University, Beijing, China
{caohaizhou, yaotiechui}@cnic.cn, {wangjue, wangyg}@sccas.cn, {huangzhenhao, huihe}@ncepu.edu.cn

Abstract

Long-term time series forecasting (LTSF) provides substantial benefits for numerous real-world applications, whereas places essential demands on the model capacity to capture long-range dependencies. Recent Transformer-based models have significantly improved LTSF performance. It is worth noting that Transformer with the self-attention mechanism was originally proposed to model language sequences whose tokens (i.e., words) are discrete and highly semantic. However, unlike language sequences, most time series are sequential and continuous numeric points. Time steps with temporal redundancy are weakly semantic, and only leveraging time-domain tokens is hard to depict the overall properties of time series (e.g., the overall trend and periodic variations). To address these problems, we propose a novel Transformer-based forecasting model named InParformer with an Interactive Parallel Attention (InPar Attention) mechanism. The InPar Attention is proposed to learn long-range dependencies comprehensively in both frequency and time domains. To improve its learning capacity and efficiency, we further design several mechanisms, including query selection, key-value pair compression, and recombination. Moreover, InParformer is constructed with evolutionary seasonal-trend decomposition modules to enhance intricate temporal pattern extraction. Extensive experiments on six real-world benchmarks show that InParformer outperforms the state-of-the-art forecasting Transformers.

Introduction

As time series is increasingly complex and pervasive in the era of big data, time series forecasting (TSF) has become an integral part of numerous real-world applications in energy, economics, traffic, weather, etc. Compared to ordinary TSF, long-term time series forecasting (LTSF) offers stronger assistance for long-term planning (e.g., in power systems (Lindberg et al. 2019)) and early warning, but brings more challenges. Typically, it requires a higher model capacity to discover longer temporal dependencies and model tougher nonlinear dynamics. Although RNN-based TSF models (Lai et al. 2018; Rangapuram et al. 2018; Salinas et al. 2020) have made notable strides, these iterated multi-step (IMS) (Shi and Yeung 2018) approaches suffer from error accumulation (Bengio et al. 2015), inefficient inference, and tricky parallelization (Vaswani et al. 2017). CNNs or temporal convolutional networks (TCNs) have shown effectiveness in sequence modeling (Bai, Kolter, and Koltun 2018; Sen, Yu, and Dhillon 2019), but are still limited by local receptive fields. Similar to breakthroughs in NLP (Vaswani et al. 2017; Devlin et al. 2018) and CV (Dosovitskiy et al. 2022; Liu et al. 2021) fields, LTSF has recently benefited from the Transformer (Vaswani et al. 2017) architecture. With the self-attention mechanism, Transformer-based models achieve superiority in capturing long-term dependencies, which is essential for LTSF. The canonical Transformer with self-attention has quadratic computational and memory costs. Recent forecasting Transformers (Li et al. 2019; Kitaev, Kaiser, and Levskaya 2020; Zhou et al. 2021) are mainly committed to using a sparse scheme to improve the efficiency of self-attention.

It is worth noting that the standard attention mechanism was originally proposed to model human-generated language sequences. Each step token (i.e., word) is discrete and highly semantic. Standard point-wise alignment schemes (including their sparse versions) are reasonable when capturing semantic dependencies. However, most time series are sequential and continuous numeric points, extremely dissimilar to sequences like language sentences. When the Transformer architecture is adopted for time series, there are three key challenges:

1. Time series contain several steps with information redundancy (e.g., missing values can be obtained by interpolation in some cases). This indicates that performing full-length queries is computationally redundant. Typically, Informer (Zhou et al. 2021) proposes the ProbSparse attention which selects top-u dominant queries based on query sparsity measurement. But this will limit the overall representation of time series. Moreover, compressing the length of key-value pairs is also worth considering. For example, a memory compressed attention (Liu et al. 2018) reduces the length by using a strided convolution to process long texts. On the other hand, periodic time series may have similar sub-processes. Autoformer (Wu et al. 2021) develops an Auto-Correlation mechanism that calculates series autocorrelation and selects top-k possible time delays to conduct a sub-series level aggre-

*Corresponding author.

gation.

2. The semantic density of time series is low. The inherent characteristics of time series are difficult to depict with local or a small number of time-domain steps. However, many attention mechanisms for TSF (Li et al. 2019; Kitaev, Kaiser, and Levskaya 2020; Zhou et al. 2021) are time-domain sparse. Performing a time-to-frequency domain transformation is widely used in time series analysis and can be a suitable solution. After transformation, each token in the frequency domain becomes highly "semantic" because it reflects a frequency feature of time series. FEDformer (Zhou et al. 2022) selects a random subset of frequency components and multiplies them with learnable complex-number parameters to learn a sparse representation of time series. It selects a small constant-length subset for any time series, so this sparse scheme easily leads to global information loss (e.g., patterns related to frequency).
3. Real-world time series are often composed of intricate temporal patterns and entangled with noise. Time series decomposition is a common strategy to tackle these knotted patterns (Hyndman and Athanasopoulos 2021). Besides pattern extraction, an effective decomposition method can reduce noise. Traditionally, decomposition is applied as a feature engineering technique during the preprocessing phase. Autoformer (Wu et al. 2021) introduces the idea of seasonal-trend decomposition (Robert, William, and Irma 1990) into Transformer by utilizing a fixed-window moving average. FEDformer (Zhou et al. 2022) replaces a fixed window with a set of ones. However, the detrended part (as the seasonal component) in these works relies on predefined average filters.

Motivated by the above, we propose InParformer, an evolutionary decomposition Transformer with interactive parallel attention for LTSF. We provide the following solutions to the listed challenges: (1) For redundant time steps, we employ query selection and key-value pair compression. Query selection provides two random subsets of queries to the parallel attention module, reducing the time/space cost significantly. It also acts as a generalization mechanism on time steps like a dropout. Key-value pair compression is implemented by an interactive partitioned convolution module that learns compact multi-resolution temporal partitions. (2) Considering the limitation of time-domain tokens and the advantages of frequency-domain ones, we create a parallel attention module with two sub-attention mechanisms that work in the frequency and time domains separately. Furthermore, we add global context information to the recombined outputs of two sub-attention mechanisms. (3) To ease intricate temporal pattern extraction, we follow the decomposition architecture designed by Autoformer (Wu et al. 2021). But we use the proposed evolutionary seasonal-trend decomposition module and deploy fewer ones. With the proposed binary decomposition algorithm, the module can obtain the seasonal component without predefined fixed average filters. The key contributions are summarized as follows:

- We propose a novel forecasting Transformer named InParformer. It introduces evolutionary seasonal-trend decomposition (EvoSTD) modules to enhance its ability to extract intricate temporal patterns.
- We propose an interactive parallel attention (InPar Attention) mechanism with frequency-aware attention and time-aware attention to learn long-range dependencies comprehensively.
- To improve the learning capacity and computation efficiency of InPar Attention, we design several mechanisms, including query selection, key-value pair compression (via an interactive partitioned convolution module), and recombination.
- Extensive experiments on six popular real-world benchmarks across multiple domains show that InParformer achieves better performance than the state-of-the-art methods under the long-term multivariate and univariate forecasting settings.

Related Work

Time Series Forecasting As a long-standing and valuable research topic, TSF has undergone substantial development. Traditional TSF methods mainly focus on statistical approaches such as ARIMA (Box and Jenkins 1968) and exponential smoothing (Gardner Jr 1985), which have difficulty in modeling non-linear temporal dynamics. To solve this defect, classical machine learning models are introduced for TSF, such as support vector regression (SVR) (Smola and Schölkopf 2004) and gradient boosted trees (Chen and Guestrin 2016). Nevertheless, their performance relies heavily on feature engineering.

Recently, with the thrilling success of deep learning in many fields, various deep neural networks with powerful learning capabilities have been developed for TSF. Due to the strength in sequence modeling, RNNs, including LSTM (Hochreiter and Schmidhuber 1997) and GRU (Cho et al. 2014), are widely used to capture temporal dependencies. DeepAR (Salinas et al. 2020) combines RNNs with autoregressive methods to predict a probabilistic distribution of time series. However, due to the recurrent structure, RNN-based models easily suffer from error accumulation (Bengio et al. 2015) and have trouble with parallelization (Vaswani et al. 2017). In addition, CNNs have also found their abilities in learning temporal representation, such as temporal convolution networks (TCNs) (Bai, Kolter, and Koltun 2018; Sen, Yu, and Dhillon 2019), but are still limited by the local receptive fields.

Transformer-Based Models Similar to remarkable progresses in NLP (Vaswani et al. 2017; Devlin et al. 2018) and CV (Dosovitskiy et al. 2022; Liu et al. 2021) fields, TSF, especially long-term TSF, has recently benefited from the Transformer (Vaswani et al. 2017) architecture. As the centerpiece of Transformer, the self-attention mechanism has $O(1)$ maximum path length of signals traveling, which is advantageous for long-range modeling. However, the canonical Transformer has quadratic computation complexity due to the self-attention mechanism. Numerous sparse strategies are proposed to improve the efficiency of self-attention (Lin et al. 2021). LogTrans (Li et al. 2019) proposes the

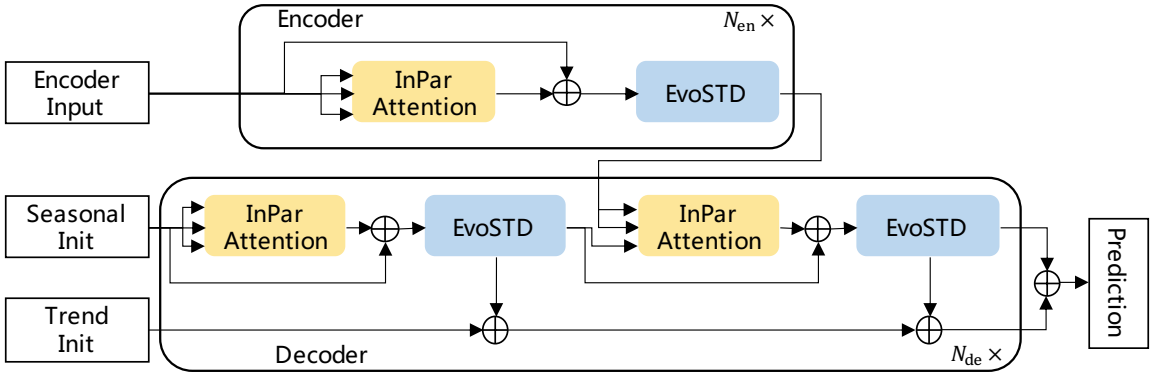

Figure 1: InParformer architecture. The interactive parallel attention (InPar Attention) is used to perform dependency discovery in both frequency and time domains. The evolutionary seasonal-trend decomposition (EvoSTD) is used to extract intricate temporal patterns.

LogSparse attention that each step only attends to previous steps with exponential intervals to break the memory bottleneck. Reformer (Kitaev, Kaiser, and Levskaya 2020) separates tokens into several buckets using locality-sensitive hashing (LSH) and conducts attention within each bucket. Informer (Zhou et al. 2021) proposes the ProbSparse attention which only calculates top-u dominant queries based on the measured query sparsity. Note that the sparse schemes developed by these works still follow the standard point-wise alignment. Autoformer (Wu et al. 2021) proposes an Auto-Correlation mechanism to conduct a sub-series level aggregation for the inherent periodicity of time series. FEDformer (Zhou et al. 2022) learns a sparse temporal representation using frequency enhanced block/attention based on a random subset of frequency components and learnable complex-number parameters. Besides, these two Transformer-based models incorporate seasonal-trend decomposition blocks to learn temporal patterns. Nevertheless, the decomposition blocks are based on single or multiple predefined fixed-window average filters.

Methodology

Preliminary

Given historical data with input length L_x, time series forecasting is to predict future horizon with output length L_y. For LTSF, the output length L_y is larger, i.e., long-term prediction. The data dimension in the model is denoted as D.

Model Architecture

The overall framework of InParformer is shown in Figure 1. It is a Transformer-based decomposition architecture consisting of interactive parallel attention (InPar Attention) modules and evolutionary seasonal-trend decomposition (EvoSTD) modules. The details of these modules will be described in subsequent sections.

Encoder The encoder is designed to learn the historical seasonal information and stacked with N_{en} encoder layers. As shown in Figure 1, each encoder layer has one InPar Attention module and one EvoSTD module. Given the embedded input of encoder $\mathbf{X}_{\text{en}}^0 \in \mathbb{R}^{L_x \times D}$, the details in l-th encoder layer are:

$$\mathbf{S}_{\text{en}}^l, \_ = \text{EvoSTD}(\text{InParAttn}(\mathbf{X}_{\text{en}}^{l-1}) + \mathbf{X}_{\text{en}}^{l-1}), \quad (1)$$

where $\mathbf{X}_{\text{en}}^l = \mathbf{S}_{\text{en}}^l \in \mathbb{R}^{L_x \times D}$ is the seasonal part and is used as the input of l-th encoder layer; $l \in \{1, \ldots, N_{\text{en}}\}$. The process is summarized as: $\mathbf{X}_{\text{en}}^l = \text{Encoder}(\mathbf{X}_{\text{en}}^{l-1})$.

Decoder The decoder is used to output the prediction and contains N_{de} decoder layers. As shown in Figure 1, each decoder layer has two InPar Attention modules (the second is used as cross-attention) and two EvoSTD modules. The initialization of the decoder's inputs is similar to that of Autoformer (Wu et al. 2021) except for the decomposition method (using binary decomposition). Given the decoder's embedded inputs $\mathbf{X}_{\text{de}}^0, \mathbf{T}_{\text{de}}^0 \in \mathbb{R}^{(\frac{L_x}{2}+L_y)\times D}$, the details in l-th decoder layer are:

$$\begin{aligned}
\mathbf{S}_{\text{de}}^{l,1}, \mathbf{T}_{\text{de}}^{l,1} &= \text{EvoSTD}(\text{InParAttn}(\mathbf{X}_{\text{de}}^{l-1}) + \mathbf{X}_{\text{de}}^{l-1}), \\
\mathbf{S}_{\text{de}}^{l,2}, \mathbf{T}_{\text{de}}^{l,2} &= \text{EvoSTD}(\text{InParAttn}(\mathbf{S}_{\text{de}}^{l,1}, \mathbf{X}_{\text{en}}^{N_{\text{en}}}) + \mathbf{S}_{\text{de}}^{l,1}), \\
\mathbf{T}_{\text{de}}^{l} &= \mathbf{T}_{\text{de}}^{l-1} + \mathcal{W}_{l,1} * \mathbf{T}_{\text{de}}^{l,1} + \mathcal{W}_{l,2} * \mathbf{T}_{\text{de}}^{l,2},
\end{aligned} \quad (2)$$

where $\mathbf{S}_{\text{de}}^{l,i}$ and $\mathbf{T}_{\text{de}}^{l,i}$ ($i \in \{1,2\}$) denote the seasonal and trend parts; $\mathbf{X}_{\text{de}}^l = \mathbf{S}_{\text{de}}^{l,2}$; $\mathcal{W}_{l,i}$ ($i \in \{1,2\}$) represents the projector for the trend part $\mathbf{T}_{\text{de}}^{l,i}$. The process is summarized as: $\mathbf{X}_{\text{de}}^l = \text{Decoder}(\mathbf{X}_{\text{de}}^{l-1}, \mathbf{X}_{\text{en}}^{N_{\text{en}}})$.

The prediction result is the sum of two parts: $\mathcal{W}_S * \mathbf{X}_{\text{de}}^{N_{\text{de}}} + \mathbf{T}_{\text{de}}^{N_{\text{de}}}$, where $\mathcal{W}_S$ is to project the seasonal part $\mathbf{X}_{\text{de}}^{N_{\text{de}}}$ to the target dimension.

Interactive Parallel Attention

As shown in Figure 2, the InPar Attention is proposed to learn long-range dependencies from two perspectives: frequency domain and time domain. To improve its learning capacity and computation efficiency, several mechanisms are employed, including query selection, key-value pair compression (by an interactive partitioned convolution), and recombination.

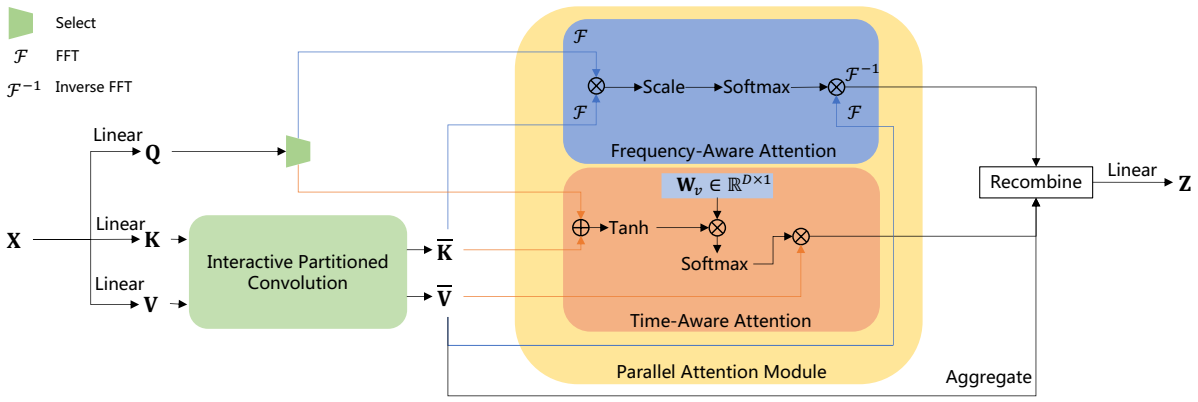

Figure 2: Interactive parallel attention (InPar Attention). Two subsets of queries are selected and fed into the parallel attention module. Benefiting from query selection, the parallel attention can be considered as a single attention. Furthermore, an interactive partitioned convolution (IPConv) is employed to compress the key-value pairs to a smaller length.

Query Selection for Parallel Attention Module For parallel attention, it is intuitive to reserve full queries for each, but this yields twice computational complexity and memory usage compared to single attention. Previous analyses show both theoretically and experimentally that the self-attention matrix is often low rank (Wang et al. 2020; Guo et al. 2019) and sparse (Child et al. 2019). Numerous sparse strategies are proposed to avoid computing full (single) attention (Lin et al. 2021). Considering that time series contain redundant time steps, we select two subsets from full queries $\mathbf{Q} \in \mathbb{R}^{L \times D}$ for two sub-attention mechanisms:

$$\begin{aligned} \mathbf{i}_{\mathrm{F}}, \mathbf{i}_{\mathrm{T}} &= \mathrm{SplitIndex}(L, L_{\mathrm{F}}, L_{\mathrm{T}}), \\ \mathbf{Q}_{\mathrm{F}} &= \mathbf{Q}[\mathbf{i}_{\mathrm{F}}], \\ \mathbf{Q}_{\mathrm{T}} &= \mathbf{Q}[\mathbf{i}_{\mathrm{T}}], \end{aligned} \tag{3}$$

where $\mathrm{SplitIndex}(\cdot)$ is to get two subsets of index randomly which may have overlaps (random selection is to avoid introducing bias of structural information) ; $\mathbf{Q}_{\mathrm{F}} \in \mathbb{R}^{L_{\mathrm{F}} \times D}$ and $\mathbf{Q}_{\mathrm{T}} \in \mathbb{R}^{L_{\mathrm{T}} \times D}$ are the subsets of queries selected by $\mathbf{i}_{\mathrm{F}}$ and $\mathbf{i}_{\mathrm{T}}$, respectively. For the multi-head version, $\mathrm{SplitIndex}(\cdot)$ gets different $\mathbf{i}_{\mathrm{F}}$ and $\mathbf{i}_{\mathrm{T}}$ for each head, which expands the sampling space of queries and reduces information loss. In this work, we select $L_{\mathrm{F}} = L - \lfloor c \times \log L \rfloor$ and $L_{\mathrm{T}} = \lfloor c \times \log L \rfloor$ queries randomly for each subspace (head) where c is a sampling factor. We will describe the processing for insufficient output length caused by incomplete queries in subsection *Recombination*.

Interactive Partitioned Convolution Unlike human-generated language sequences which are information-dense, most natural signals are not highly semantic (He et al. 2021). Inspired by the memory-compressed attention (Liu et al. 2018) and the pyramid structures in computer vision tasks (Chen et al. 2017; Zhao et al. 2017), we propose an interactive partitioned convolution (IPConv) to learn a compact temporal representation at multiple resolutions. As shown in Figure 3, multiple parallel convolution branches with different settings are applied to capture multi-resolution information. Additional padding, convolution, and pooling operations can be used in some branches to make all branches output the same length.

For parallel attention module, IPConv is used to provide a smaller length of key-value pairs and its parameters are shared between the keys and values processing:

$$\begin{aligned} \overline{\mathbf{K}} &= \mathrm{IPConv}(\mathbf{K}), \\ \overline{\mathbf{V}} &= \mathrm{IPConv}(\mathbf{V}), \end{aligned} \tag{4}$$

where $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times D}$; $\overline{\mathbf{K}}, \overline{\mathbf{V}} \in \mathbb{R}^{L' \times D}$. The output length are reduced to $L' = \frac{1}{4}L$. In practice, two convolution branches are employed. For branch 1, there is a 1-D convolution with $kernel\_size = 4$, $stride = 4$. For branch 2, there are two 1-D convolutions with $kernel\_size = (6, 2)$, $stride = (2, 2)$, and a padding operation before the convolutions. Due to the $out\_channels = D_1 = D_2 = D/2$ for each convolution, the total cost of computation and memory is similar to that of a single $out\_channels = D$ convolution. After concatenation, we do not perform fusion operations like $kernel\_size = 1$ convolution. Besides the consideration of computational cost, this makes multi-head key-value pairs with multiple resolutions, depending on the convolution branch they belong to.

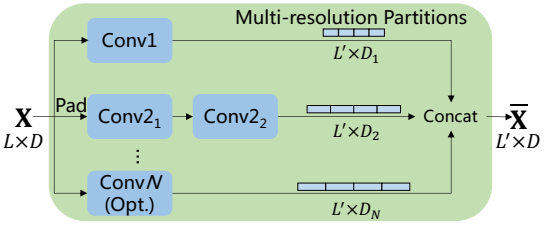

Figure 3: Interactive partitioned convolution (IPConv) as a context and compression module. Multiple convolution branches work with different settings and output reduced dimensions. The multi-resolution partitions from the convolution branches are concatenated.

Frequency-Aware Attention Given the projected queries $\mathbf{Q} \in \mathbb{R}^{L_Q \times D_K}$, keys $\mathbf{K} \in \mathbb{R}^{L_K \times D_K}$, and values $\mathbf{V} \in$

$\mathbb{R}^{L_K \times D_V}$, frequency-aware attention (FAA) performs a scaled dot-product attention in the frequency domain:

$$\mathrm{FAA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathcal{F}^{-1}(\mathrm{softmax}(\frac{\mathcal{F}(\mathbf{Q})\mathcal{F}(\mathbf{K})^\top}{\sqrt{D_K}})\mathcal{F}(\mathbf{V})), \tag{5}$$

where $\mathcal{F}, \mathcal{F}^{-1}$ denote the Fast Fourier Transform (FFT) and its inverse (IFFT). In practice, since the input is real, the output of FFT is Hermitian-symmetric (Oppenheim 1999) and the negative frequencies are redundant. Therefore, with the real FFT (RFFT) which only computes the positive frequencies, the dot-product operation in FAA is performed with the half-length queries, keys, and values in the frequency domain. Finally, the inverse RFFT transforms the result to the time domain with the original length. The formulation of multi-head version is similar to the canonical one and omitted.

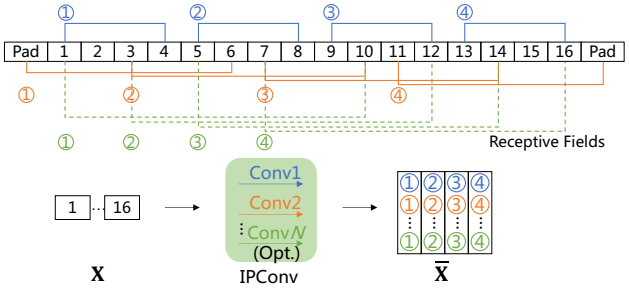

Figure 4: Illustration of multi-resolution partitioning by IPConv for a simple sequence. The result is the concatenation of partitions by multiple convolution branches with different receptive fields. The length is reduced from 16 to 4.

When as a sub-attention in InPar Attention, its input is the subset of queries $\mathbf{Q}_\mathrm{F} \in \mathbb{R}^{L_\mathrm{F} \times D}$ and the compressed key-value pairs, $\overline{\mathbf{K}}, \overline{\mathbf{V}} \in \mathbb{R}^{L' \times D}$. The process is summarized as follows:

$$\mathbf{Z}_\mathrm{F} = \mathrm{FAA}(\mathbf{Q}_\mathrm{F}, \overline{\mathbf{K}}, \overline{\mathbf{V}}), \tag{6}$$

where $\mathbf{Z}_\mathrm{F} \in \mathbb{R}^{L_\mathrm{F} \times D}$. Compared to canonical attention with full input, FAA for InPar Attention performs dependency discovery in the frequency domain and greatly reduces the computational cost of the dot product through query selection, key-value pair compression, and RFFT.

Time-Aware Attention Given the projected queries $\mathbf{Q} \in \mathbb{R}^{L_Q \times D_K}$, keys $\mathbf{K} \in \mathbb{R}^{L_K \times D_K}$, and values $\mathbf{V} \in \mathbb{R}^{L_K \times D_V}$, time-aware attention (TAA) performs additive attention (Bahdanau, Cho, and Bengio 2014) which contains learnable parameters for scoring (in the time domain):

$$\mathrm{TAA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\tanh(\mathbf{Q} +^* \mathbf{K})\mathbf{w}_v)\mathbf{V}, \tag{7}$$

where $+^*$ denotes the addition with broadcasting; $\mathbf{w}_v \in \mathbb{R}^{D_K}$ is a learnable weight vector. The formulation of multi-head version is similar to the canonical one and omitted.

When as a sub-attention in InPar Attention, its key-value pairs $\overline{\mathbf{K}}, \overline{\mathbf{V}} \in \mathbb{R}^{L' \times D}$ are shared with FAA while the queries are $\mathbf{Q}_\mathrm{T} \in \mathbb{R}^{L_\mathrm{T} \times D}$. It can be written as:

$$\mathbf{Z}_\mathrm{T} = \mathrm{TAA}(\mathbf{Q}_\mathrm{T}, \overline{\mathbf{K}}, \overline{\mathbf{V}}), \tag{8}$$

where $\mathbf{Z}_\mathrm{T} \in \mathbb{R}^{L_\mathrm{T} \times D}$. Besides the compressed key-value pairs, we set the query length $L_\mathrm{T} = \lfloor c \times \log L \rfloor$ (c is a small constant factor) in practice, reducing the cost and achieving $\mathcal{O}(L \log L)$ complexity.

Recombination From the parallel attention module, we get $\mathbf{Z}_\mathrm{F}$ and $\mathbf{Z}_\mathrm{T}$ for the queries with indices $\mathbf{i}_\mathrm{F}$ and $\mathbf{i}_\mathrm{T}$ (from Eq.(3)). $\mathbf{Z}_\mathrm{F}$ and $\mathbf{Z}_\mathrm{T}$ will be recombined together based on the indices $\mathbf{i}_\mathrm{F}$ and $\mathbf{i}_\mathrm{T}$. Since there are unselected indices, the total output length is insufficient. To address this problem, we obtain the global context information by value aggregation and incorporate it into the recombination result (as the values of unselected indices). Especially for the multi-head version (each head has different unselected indices), this processing can be considered as the concatenation between the feature maps of global context and those of the parallel attention module. The recombination process is as follows:

$$\begin{aligned} \mathbf{Z} &= \mathrm{Aggregate}(\overline{\mathbf{V}}), \\ \mathbf{Z}[\mathbf{i}_\mathrm{F}] &= \mathbf{Z}_\mathrm{F}, \\ \mathbf{Z}[\mathbf{i}_\mathrm{T}] &= \mathbf{Z}_\mathrm{T}, \end{aligned} \tag{9}$$

where $\mathbf{Z} \in \mathbb{R}^{L \times D}$ is the recombination result and initialized with the global context information by value aggregation. In practice, the aggregation is performed by $\mathrm{sum}(\cdot)/L$.

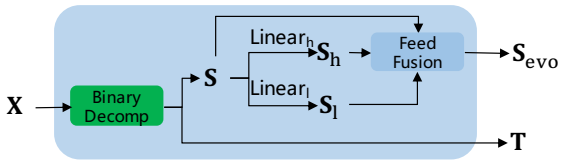

Figure 5: Evolutionary seasonal-trend decomposition (EvoSTD) module. The binary decomposition yields the seasonal and trend components. The high and low-frequency information $\mathbf{S}_\mathrm{h}, \mathbf{S}_\mathrm{l}$ of seasonal component are yielded by the specific linear layers. They are further combined with the seasonal component by a special Feed-Forward layer, i.e., FeedFusion.

Evolutionary Seasonal-Trend Decomposition

As shown in Figure 5, we propose an evolutionary seasonal-trend decomposition (EvoSTD) module to learn entangled temporal patterns. In EvoSTD, the seasonal and trend parts are first yielded by the binary decomposition. Since the seasonal part contains rich frequency information, two linear layers initialized with the wavelet coefficients are adopted to extract high and low-frequency information. Then the seasonal part is combined with the frequency information and transformed by FeedFusion for an evolution. The details can be formalized as:

$$\begin{aligned} \mathbf{S}, \mathbf{T} &= \mathrm{BinaryDecomp}(\mathbf{X}), \\ \mathbf{S}_\mathrm{evo} &= \mathrm{FeedFusion}(\mathrm{Concat}[\mathbf{S}, \mathbf{W}_\mathrm{h}\mathbf{S}, \mathbf{W}_\mathrm{l}\mathbf{S}]) \end{aligned} \tag{10}$$

where $\mathbf{X} \in \mathbb{R}^{L \times D}$ is the input series; $\mathbf{W}_\mathrm{h}, \mathbf{W}_\mathrm{l} \in \mathbb{R}^{L \times L}$ are the specific linear layers; FeedFusion is a special Feed-Forward layer with $in\_channels = 3D$; $\mathbf{S}_\mathrm{evo}, \mathbf{T} \in \mathbb{R}^{L \times D}$ are the evolutionary seasonal part and trend. The process can be summarized as: $\mathbf{S}_\mathrm{evo}, \mathbf{T} = \mathrm{EvoSTD}(\mathbf{X})$.

Models		InParformer		FEDformer		Autoformer		Informer		LogTrans		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	**0.198**	0.288	0.203	**0.287**	0.255	0.339	0.365	0.453	0.768	0.642	0.658	0.619
	192	**0.260**	**0.323**	0.269	0.328	0.281	0.340	0.533	0.563	0.989	0.757	1.078	0.827
	336	**0.319**	**0.365**	0.325	0.366	0.339	0.372	1.363	0.887	1.334	0.872	1.549	0.972
	720	**0.420**	0.417	0.421	**0.415**	0.422	0.419	3.379	1.338	3.048	1.328	2.631	1.242
Electr.	96	**0.184**	**0.296**	0.193	0.308	0.201	0.317	0.274	0.368	0.258	0.357	0.312	0.402
	192	**0.186**	**0.298**	0.201	0.315	0.222	0.334	0.296	0.386	0.266	0.368	0.348	0.433
	336	**0.202**	**0.314**	0.214	0.329	0.231	0.338	0.300	0.394	0.280	0.380	0.350	0.433
	720	**0.228**	**0.335**	0.246	0.355	0.254	0.361	0.373	0.439	0.283	0.376	0.340	0.420
Exchange	96	**0.120**	**0.252**	0.148	0.278	0.197	0.323	0.847	0.752	0.968	0.812	1.065	0.829
	192	**0.230**	0.354	0.271	**0.308**	0.300	0.369	1.204	0.895	1.040	0.851	1.188	0.906
	336	**0.427**	**0.482**	0.460	0.500	0.509	0.524	1.672	1.036	1.659	1.081	1.357	0.976
	720	**1.106**	**0.813**	1.195	0.841	1.447	0.941	2.478	1.310	1.941	1.127	1.510	1.016
Traffic	96	**0.557**	**0.340**	0.587	0.366	0.613	0.388	0.719	0.391	0.684	0.384	0.732	0.423
	192	**0.577**	**0.349**	0.604	0.373	0.616	0.382	0.696	0.379	0.685	0.390	0.733	0.420
	336	**0.597**	**0.359**	0.621	0.383	0.622	0.337	0.777	0.420	0.733	0.408	0.742	0.420
	720	**0.612**	**0.364**	0.626	0.382	0.660	0.408	0.864	0.472	0.717	0.396	0.755	0.423
Weather	96	**0.212**	**0.286**	0.217	0.296	0.266	0.336	0.300	0.384	0.458	0.490	0.689	0.596
	192	**0.259**	**0.322**	0.276	0.336	0.307	0.367	0.598	0.544	0.658	0.589	0.752	0.638
	336	**0.317**	**0.358**	0.339	0.380	0.359	0.395	0.578	0.523	0.797	0.652	0.639	0.596
	720	**0.395**	**0.409**	0.403	0.428	0.419	0.428	1.059	0.741	0.869	0.675	1.130	0.792
ILI	24	**2.934**	**1.107**	3.228	1.260	3.483	1.287	5.764	1.677	4.480	1.444	4.400	1.382
	36	3.049	**1.069**	**2.679**	1.080	3.103	1.148	4.755	1.467	4.799	1.467	4.783	1.448
	48	3.067	1.088	**2.622**	**1.078**	2.669	1.085	4.763	1.469	4.800	1.468	4.832	1.465
	60	3.043	**1.116**	**2.857**	1.157	2.770	1.125	5.264	1.564	5.278	1.560	4.882	1.483

Table 1: Multivariate results with different prediction lengths $L_y \in \{96, 192, 336, 720\}$ and fixed input length $L_x = 96$ (For ILI, $L_y \in \{24, 36, 48, 60\}, L_x = 36$). Electr. is short for the Electricity dataset.

Algorithm 1: Binary Decomposition

Input: Series $\mathbf{X}$ with length L.

Initialization: seasonal component $\mathbf{S} = \mathbf{X}$; trend component $\mathbf{T} = \mathbf{0}$; current segment is $\mathbf{X}$
while length of a current segment ≥ 2 **do**
  let $\mathcal{M}$: average of each current segment
  update $\mathbf{S}$: subtract $\mathcal{M}$
  update $\mathbf{T}$: add $\mathcal{M}$
  update current segments: split in half
end while
update $\mathbf{T}$: smooth $\mathbf{T}$ by moving average
return $\mathbf{S}$, $\mathbf{T}$

Binary Decomposition For the intricate seasonal patterns in real-world data, we design a binary decomposition algorithm based on the divide-and-conquer paradigm, which extracts seasonality without predefined fixed-window average filters. The pseudo-code for binary decomposition is presented in Algorithm 1.

Complexity Analysis

In this work, the theoretical complexities of FAA and TAA are $O(L'L)$ and $O(L' \log L)$, respectively (L' is the compressed length by IPConv). However, the query lengths of FAA and TAA mainly depend on query selection, so they are adjustable. Moreover, IPConv and RFFT further reduce the computational cost. Concretely, the attention matrix of TAA is $c \log L \times L'$, while that of FAA is $\frac{L - c \log L}{2} \times \frac{L'}{2}$, where '$\frac{1}{2}$' is caused by RFFT. Although InPar Attention contains parallel attention modules, it is efficient.

Experiment

To evaluate the proposed InParformer, we perform extensive experiments on six popular real-world benchmarks, including energy, traffic, economics, weather, and disease domains. More detailed experimental information is provided in the Appendices.

Datasets (1) **ETT** (Zhou et al. 2021) contains load and oil temperature collected from electricity transformers. It has four sub-datasets as ETTh1, ETTh2, ETTm1, and ETTm2 in two resolutions (1 hour and 15 minutes). (2) **Electricity**[1] includes the hourly electricity consumption of 321 clients. (3) **Exchange** (Lai et al. 2018) collects daily exchange rates for

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

Models		InParformer		FEDformer		Autoformer		Informer		LogTrans		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.0022**	**0.036**	0.0062	0.062	0.0110	0.081	0.004	0.044	0.0046	0.052	0.012	0.087
	192	**0.0038**	**0.048**	0.0060	0.062	0.0075	0.067	0.002	0.040	0.0060	0.060	0.010	0.044
	336	**0.0033**	**0.045**	0.0041	0.050	0.0063	0.062	0.004	0.049	0.0060	0.054	0.013	0.100
	720	**0.0030**	**0.042**	0.0055	0.059	0.0085	0.070	0.003	0.042	0.0070	0.059	0.011	0.083
ETTh2	96	**0.117**	**0.264**	0.128	0.271	0.153	0.306	0.213	0.373	0.217	0.379	1.411	0.838
	192	**0.169**	**0.319**	0.185	0.330	0.204	0.351	0.227	0.387	0.281	0.429	5.658	1.671
	336	**0.225**	**0.373**	0.231	0.378	0.246	0.389	0.242	0.401	0.293	0.437	4.777	1.582
	720	0.241	0.399	0.278	0.420	0.268	0.409	0.291	0.439	**0.218**	**0.387**	2.042	1.039
Exchange	96	**0.105**	**0.247**	0.154	0.304	0.241	0.387	1.327	0.944	0.237	0.377	0.298	0.444
	192	**0.207**	**0.360**	0.286	0.420	0.300	0.369	1.258	0.924	0.738	0.619	0.777	0.719
	336	**0.400**	**0.498**	0.511	0.555	0.509	0.524	2.179	1.296	2.018	1.070	1.833	1.128
	720	**1.172**	**0.836**	1.301	0.879	1.260	0.867	1.280	0.953	2.405	1.175	1.203	0.956
ILI	24	**0.598**	**0.564**	0.708	0.627	0.948	0.732	5.282	2.050	3.607	1.662	3.838	1.720
	36	**0.553**	**0.592**	0.584	0.617	0.634	0.650	4.554	1.916	2.407	1.363	2.934	1.520
	48	**0.653**	**0.658**	0.717	0.697	0.791	0.752	4.273	1.846	3.106	1.575	3.755	1.749
	60	**0.789**	**0.748**	0.855	0.774	0.874	0.797	5.214	2.057	3.698	1.733	4.162	1.847

Table 2: Univariate results with different prediction lengths $L_y \in \{96, 192, 336, 720\}$ and fixed input length $L_x = 96$ on typical datasets (For ILI, $L_y \in \{24, 36, 48, 60\}$, $L_x = 36$). Weather, ETTh2, Exchange, and ILI datasets are 10-minutely, hourly, daily, and weekly recorded, respectively.

8 different countries from 1990 to 2016. (4) **Traffic**[2] contains the hourly road occupancy rates from the California Department of Transportation. (5) **Weather**[3] records 21 meteorological indicators every 10 minutes for one year. (6) **ILI**[4] includes the weekly patients with influenza-like illness (ILI) from 2002 to 2021. For all datasets, train/valid/test sets are split as $0.7/0.1/0.2$ in chronological order.

Baselines Since classic models like ARIMA and CNN/RNN-based networks lack competitive performance (Zhou et al. 2021; Wu et al. 2021), we select five state-of-the-art Transformer-based models: FEDformer (Zhou et al. 2022), Autoformer (Wu et al. 2021), Informer (Zhou et al. 2021), Reformer (Kitaev, Kaiser, and Levskaya 2020), and LogTrans (Li et al. 2019). The FEDformer (FEDformer-f, also based on Fourier transform) is used as the main baseline model because of its best performance in these baselines.

Implementation Settings The mean square error (MSE) and mean absolute error (MAE) are used as metrics. The proposed model is trained using Adam (Kingma and Ba 2014) optimizer with an initial learning rate of 10^{-4} and contains 2 encoder layers and 1 decoder layer. The sampling factor c for query selection is set to 3. The batch size is set to 32, and the training epochs are set to 10 with early stopping. The common hyperparameters are consistent with FEDformer (Zhou et al. 2022) and Autoformer (Wu et al. 2021). All experiments are implemented with PyTorch (Paszke et al. 2019) and conducted on the VenusAI platform (Yao et al. 2022) with 4 NVIDIA GeForce RTX 2080Ti 11GB GPUs.

Main Results

To evaluate different future horizons, we fix the input length $L_x = 96$ ($L_x = 36$ for ILI) and set the prediction lengths $L_y \in \{96, 192, 336, 720\}$ ($L_y \in \{24, 36, 48, 60\}$ for ILI). The detailed results of ETT full benchmark are given in Appendix A.

Multivariate Results As shown in Table 1, InParformer achieves the best performance in most cases. For example, under the input-96-predict-192 setting, compared to FEDformer, InParformer yields **3.3%** (0.269 $\rightarrow$ 0.260) MSE reduction in ETTm2, **7.5%** (0.201 $\rightarrow$ 0.186) in Electricity, **15.1%** (0.271 $\rightarrow$ 0.230) in Exchange and **6.2%** (0.276 $\rightarrow$ 0.259) in Weather. Overall, InParformer achieves a relative MSE reduction of **5.9%** (excluding ILI) compared with FEDformer. Even for the Exchange dataset that lacks clear periodicity, InParformer still outperforms other models. Notably, the performance of InParformer varies consistently as the future horizon increases, which indicates InParformer performs a stable prediction.

Univariate Results As shown in Table 2, InParformer still achieves the best performance on the typical datasets with various temporal resolutions. Compared to FEDformer, InParformer significantly improves the forecasting performance and gives an overall MSE reduction of **20.5%**. In particular, for both datasets with obvious periodicity (ETTh2) and those without (Exchange), InParformer outperforms other models. Besides, for these typical datasets with various domains and resolutions, InParformer achieves signif-

[2]http://pems.dot.ca.gov

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

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

icant and stable improvements, which indicates its advantages in prediction capacity.

Decomp		EvoSTD		MOE		STD	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.379**	**0.419**	0.514	0.483	0.488	0.479
	192	**0.416**	**0.440**	0.563	0.513	0.546	0.503
	336	**0.455**	**0.458**	0.536	0.506	0.582	0.525
	720	**0.475**	**0.488**	0.602	0.558	0.574	0.549
Weather	96	**0.212**	**0.286**	0.277	0.350	0.240	0.316
	192	**0.259**	**0.322**	0.336	0.382	0.311	0.367
	336	**0.317**	**0.358**	0.379	0.413	0.344	0.388
	720	**0.395**	**0.409**	0.409	0.420	0.423	0.434

Table 3: Comparison of decomposition methods. STD and MOE denote the seasonal-trend decomposition blocks from Autoformer and FEDformer, respectively.

Attn		ProbAttn		AutoCorr		FEA		InParAttn	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.439	0.456	0.397	0.427	0.381	0.423	**0.379**	**0.419**
	192	0.562	0.524	0.439	0.456	0.417	0.443	**0.416**	**0.440**
	336	0.509	0.502	0.464	0.466	0.457	0.465	**0.455**	**0.458**
	720	0.603	0.565	0.473	0.490	0.476	**0.483**	**0.475**	0.488
Weather	96	0.242	0.330	**0.200**	**0.285**	0.210	0.288	0.212	0.286
	192	0.362	0.423	0.292	0.362	0.271	0.332	**0.259**	**0.322**
	336	0.381	0.427	0.363	0.416	0.451	0.451	**0.317**	**0.358**
	720	0.575	0.466	0.529	0.529	0.414	0.427	**0.395**	**0.409**

Table 4: Comparison of attention mechanisms. ProbAttn, AutoCorr, and FEA denote the attention mechanisms from Informer, Autoformer, and FEDformer, respectively.

Ablation Studies

To evaluate the effectiveness of our designs, we perform additional experiments on typical datasets.

Decomposition Methods We replace EvoSTD in InParformer with the series decomposition blocks of Autoformer and FEDformer, i.e., STD and MOE. For a fair comparison, STD and MOE in InParformer are under their default settings and followed by a Feed-Forward layer. As shown in Table 3, EvoSTD achieves better performance than other series decomposition blocks, which indicates its strength in extracting temporal patterns.

Attention Mechanisms We replace InPar Attention in InParformer with the attention mechanisms of Informer, Autoformer, and FEDformer. As shown in table 4, the proposed InPar Attention achieves the best performance on two typical datasets. For the ETTh1 dataset with clear periodicity, InPar Attention yields a slight improvement compared to FEA (the frequency enhanced attention from FEDformer) but significant improvements compared to other attention mechanisms. This implies that frequency information is vital for periodic time series. For the Weather dataset without clear periodicity, InPar Attention still brings significant improvements, which shows the success of parallel attention design.

IPConv and Time-Aware Attention Types For InPar Attention, we evaluate the effectiveness of IPConv and the impact of time-aware attention (TAA) types. As shown in the left part of Table 5, IPConv gives obvious performance improvement, which implies it can learn effective multi-resolution temporal partitions. As shown in the right part of Table 5, TAA based on additive attention (Bahdanau, Cho, and Bengio 2014), which contains learnable parameters for scoring, is more suitable for InPar Attention than that based on dot-product attention.

Option		IPConv		W/o IPConv		TAA-A		TAA-D	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.379	0.419	**0.378**	**0.415**	**0.379**	**0.419**	0.439	0.457
	192	**0.416**	**0.440**	0.418	0.441	**0.416**	**0.440**	0.568	0.537
	336	**0.455**	**0.458**	0.456	0.461	**0.455**	**0.458**	0.502	0.498
	720	**0.475**	**0.488**	0.546	0.536	**0.475**	**0.488**	0.604	0.560
Weather	96	0.212	0.286	**0.211**	**0.268**	0.212	0.286	**0.205**	**0.279**
	192	**0.259**	**0.322**	0.293	0.352	**0.259**	**0.322**	0.275	0.333
	336	**0.317**	**0.358**	0.350	0.386	**0.317**	**0.358**	0.379	0.420
	720	**0.395**	**0.409**	0.425	0.428	**0.395**	**0.409**	0.454	0.458

Table 5: Ablation studies of IPConv and time-aware attention (TAA) types. Left: The effectiveness of IPConv in InPar Attention. Right: The impact of TAA types (with IPConv). TAA-A denotes TAA based on additive attention. TAA-D denotes TAA based on dot-product attention.

Conclusions

In this paper, we propose a novel Transformer-based model named InParformer for long-term time series forecasting. To capture long-range dependencies comprehensively, we propose InPar Attention as an interactive parallel attention mechanism performing in both the time and frequency domains. Considering that time series with temporal redundancy are weakly semantic, we design query selection and key-value pair compression (via an interactive partitioned convolution module), which can also improve the computation efficiency. In the recombination stage, the global context information and the outputs of two sub-attention mechanisms are aggregated together. Moreover, the evolutionary seasonal-trend decomposition module is deployed in InParformer to enhance intricate pattern extraction. Extensive experiments on real-world benchmarks show that InParformer is capable of forecasting long-term time series effectively.

Acknowledgements

This work was supported by the National Key R&D Program of China (2021ZD0110403). We also gratefully acknowledge the support of MindSpore team.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Box, G. E.; and Jenkins, G. M. 1968. Some Recent Advances in Forecasting and Control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2): 91–109.

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2017. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected Crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848.

Chen, T.; and Guestrin, C. 2016. Xgboost: A Scalable Tree Boosting System. In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–794.

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating Long Sequences with Sparse Transformers. arXiv:1904.10509.

Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2022. An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Gardner Jr, E. S. 1985. Exponential Smoothing: The State of the Art. *Journal of forecasting*, 4(1): 1–28.

Guo, Q.; Qiu, X.; Xue, X.; and Zhang, Z. 2019. Low-Rank and Locality Constrained Self-Attention for Sequence Modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12): 2213–2222.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2021. Masked Autoencoders Are Scalable Vision Learners. arXiv:2111.06377.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural computation*, 9(8): 1735–1780.

Hyndman, R. J.; and Athanasopoulos, G. 2021. *Forecasting: Principles and Practice*. OTexts, 3rd edition.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *Eighth International Conference on Learning Representations*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling Long-and Short-Term Temporal Patterns with Deep Neural Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Lin, T.; Wang, Y.; Liu, X.; and Qiu, X. 2021. A Survey of Transformers. arXiv:2106.04554.

Lindberg, K.; Seljom, P.; Madsen, H.; Fischer, D.; and Korpås, M. 2019. Long-Term Electricity Load Forecasting: Current and Future Trends. *Utilities Policy*, 58: 102–119.

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018. Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Oppenheim, A. V. 1999. *Discrete-Time Signal Processing*. Pearson Education India.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems*, 32.

Rangapuram, S. S.; Seeger, M. W.; Gasthaus, J.; Stella, L.; Wang, Y.; and Januschowski, T. 2018. Deep State Space Models for Time Series Forecasting. *Advances in neural information processing systems*, 31.

Robert, C.; William, C.; and Irma, T. 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of official statistics*, 6(1): 3–73.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting*, 36(3): 1181–1191.

Sen, R.; Yu, H.-F.; and Dhillon, I. S. 2019. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. *Advances in neural information processing systems*, 32.

Shi, X.; and Yeung, D.-Y. 2018. Machine Learning for Spatiotemporal Sequence Forecasting: A Survey. arXiv:1808.06865.

Smola, A. J.; and Schölkopf, B. 2004. A Tutorial on Support Vector Regression. *Statistics and computing*, 14(3): 199–222.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. arXiv:1706.03762.

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-Attention with Linear Complexity. arXiv:2006.04768.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 34, 22419–22430. Curran Associates, Inc.

Yao, T.; Wang, J.; Wan, M.; Xin, Z.; Wang, Y.; Cao, R.; Li, S.; and Chi, X. 2022. VenusAI: An artificial intelligence platform for scientific discovery on supercomputers. *Journal of Systems Architecture*, 128: 102550.

Zhao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J. 2017. Pyramid Scene Parsing Network. arXiv:1612.01105.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, 27268–27286. PMLR.