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

Haizhou Cao\textsuperscript{1,2}, Zhenhao Huang\textsuperscript{3}, Tiechui Yao\textsuperscript{1,2}, Jue Wang\textsuperscript{1,2,7}, Hui He\textsuperscript{3}, Yangang Wang\textsuperscript{1,2}

\textsuperscript{1}Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China
\textsuperscript{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-
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
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 pointwise 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 $X_{en}^0 \in \mathbb{R}^{L_x \times D}$, the details in $l$-th encoder layer are:

$$S_{en}^l := \text{EvoSTD}(\text{InParAttn}(X_{en}^{l-1}) + X_{en}^{l-1}),$$

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

**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 $X_{de}^0, T_{de}^0 \in \mathbb{R}^{(L_x + L_y) \times D}$, the details in $l$-th decoder layer are:

$$S_{de}^{l,1}, T_{de}^{l,1} := \text{EvoSTD}(\text{InParAttn}(X_{de}^{l-1}) + X_{de}^{l-1}),$$

$$S_{de}^{l,2}, T_{de}^{l,2} := \text{EvoSTD}(\text{InParAttn}(S_{de}^{l,1} + X_{en}^{l-1}),$$

$$T_{de}^{l} := T_{de}^{l-1} + W_{l,1} \ast T_{de}^{l-1} + W_{l,2} \ast T_{de}^{l-2},$$

where $S_{de}^{l,i}$ and $T_{de}^{l,i}$ ($i \in \{1, 2\}$) denote the seasonal and trend parts; $X_{de}^{l-1} = S_{de}^{l-2}, W_{l,i}$ ($i \in \{1, 2\}$) represents the projector for the trend part $T_{de}^{l,1}$. The process is summarized as: $X_{de}^{l} = \text{Decoder}(X_{de}^{l-1}, X_{en}^{l-1})$.

The prediction result is the sum of two parts: $\mathcal{W}_S \ast X_{de}^{N_{de}} + T_{de}^{N_{de}}$, where $\mathcal{W}_S$ is to project the seasonal part $X_{de}^{N_{de}}$ 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.
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 insufficient output length caused by incomplete queries in a context and compression module. Multiple 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 output_channels = D_1 = D_2 = D/2 for each convolution, the total cost of computation and memory is similar to that of a single output_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.

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:

\[
\mathbf{K} = \text{IPConv} (\mathbf{K}), \\
\mathbf{V} = \text{IPConv} (\mathbf{V}),
\]

where \( \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times D}; \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L' \times D} \). The output length are reduced to \( L' = \frac{1}{2} L \). 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 output_channels = D_1 = D_2 = D/2 for each convolution, the total cost of computation and memory is similar to that of a single output_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.

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 \( \mathbf{V} \in \mathbb{R}^{L_V \times D_V} \), we propose an interactive partitioned convolution (IPConv) to learn a compact temporal representation at multiple resolutions.
frequency-aware attention (FAA) performs a scaled dot-product attention in the frequency domain:

\[
\text{FAA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathcal{F}^{-1}(\text{softmax}(\frac{\mathcal{F}(\mathbf{Q}) \mathcal{F}(\mathbf{K}) \mathcal{F}(\mathbf{V})}{\sqrt{D_K}}))
\]

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 input 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.

\[
\text{FAA}(\mathbf{Q}_1, \mathbf{K}_1, \mathbf{V}_1) = \mathcal{F}^{-1}(\text{softmax}(\frac{\mathcal{F}(\mathbf{Q}_1) \mathcal{F}(\mathbf{K}_1) \mathcal{F}(\mathbf{V}_1)}{\sqrt{D_K}}))
\]

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

**Recombination** From the parallel attention module, we get \(\mathbf{Z}_F\) and \(\mathbf{Z}_T\) for the queries with indices \(I_F\) and \(I_T\) (from Eq.(3)). \(\mathbf{Z}_F\) and \(\mathbf{Z}_T\) will be recombined together based on the indices \(I_F\) and \(I_T\). 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:

\[
\mathbf{Z} = \text{Aggregate}(\mathbf{V}),
\]

\[
\mathbf{Z}[I_F] = \mathbf{Z}_F,
\]

\[
\mathbf{Z}[I_T] = \mathbf{Z}_T,
\]

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 \(\text{sum}(\cdot)/L\).

**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:

\[
\mathbf{S}, \mathbf{T} = \text{BinaryDecomp}(\mathbf{X}),
\]

\[
\mathbf{S}_{\text{evo}}, \mathbf{T} = \text{FeedFusion}(\text{Concat}([\mathbf{S}, \mathbf{W}_b\mathbf{S}, \mathbf{W}_l\mathbf{S}]])
\]

where \(\mathbf{X} \in \mathbb{R}^{L \times D}\) is the input series; \(\mathbf{W}_b, \mathbf{W}_l \in \mathbb{R}^{L \times L}\) are the specific linear layers; FeedFusion is a special Feed-Forward layer with \(\text{in_channels} = 3D\); \(\mathbf{S}_{\text{evo}}, \mathbf{T} \in \mathbb{R}^{L \times D}\) are the evolutionary seasonal part and trend. The process can be summarized as: \(\mathbf{S}_{\text{evo}}, \mathbf{T} = \text{EvoSTD}(\mathbf{X})\).
<table>
<thead>
<tr>
<th>Models</th>
<th>MSE</th>
<th>MAE</th>
<th>MSE</th>
<th>MAE</th>
<th>MSE</th>
<th>MAE</th>
<th>MSE</th>
<th>MAE</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>InParformer</td>
<td>0.198</td>
<td>0.288</td>
<td>0.203</td>
<td>0.287</td>
<td>0.255</td>
<td>0.339</td>
<td>0.365</td>
<td>0.453</td>
<td>0.768</td>
<td>0.642</td>
</tr>
<tr>
<td>FEDformer</td>
<td>0.260</td>
<td>0.323</td>
<td>0.269</td>
<td>0.328</td>
<td>0.281</td>
<td>0.340</td>
<td>0.533</td>
<td>0.563</td>
<td>0.989</td>
<td>0.757</td>
</tr>
<tr>
<td>Autoformer</td>
<td>0.319</td>
<td>0.365</td>
<td>0.325</td>
<td>0.366</td>
<td>0.339</td>
<td>0.372</td>
<td>1.363</td>
<td>0.887</td>
<td>1.334</td>
<td>0.872</td>
</tr>
<tr>
<td>Informer</td>
<td>0.420</td>
<td>0.417</td>
<td>0.421</td>
<td>0.415</td>
<td>0.422</td>
<td>0.419</td>
<td>3.379</td>
<td>1.338</td>
<td>3.048</td>
<td>1.328</td>
</tr>
<tr>
<td>LogTrans</td>
<td>1.447</td>
<td>0.941</td>
<td>0.719</td>
<td>0.391</td>
<td>0.578</td>
<td>0.523</td>
<td>0.212</td>
<td>0.286</td>
<td>0.256</td>
<td>0.379</td>
</tr>
<tr>
<td>Reformer</td>
<td>1.836</td>
<td>1.248</td>
<td>1.319</td>
<td>0.872</td>
<td>1.106</td>
<td>0.813</td>
<td>1.195</td>
<td>0.841</td>
<td>1.447</td>
<td>0.941</td>
</tr>
</tbody>
</table>

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 \( X \) with length \( L \).

1. Initialization: seasonal component \( S = X \); trend component \( T = 0 \); current segment is \( X \).
2. **while** length of a current segment \( \geq 2 \) do
   3. let \( M \): average of each current segment
   4. update \( S \): subtract \( M \)
   5. update \( T \): add \( M \)
   6. update current segments: split in half
   7. **end while**
   8. update \( T \): smooth \( T \) by moving average
9. return \( S, T \)

**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 \( \frac{L-L/\log L}{2} \times \frac{L}{2} \), 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. (1) **ETT** (Zhou et al. 2021) contains load and oil temperature collected from electricity transformers. It has four sub-datasets as ETT1h, ETT12h, ETTm1, and ETTm2 in two resolutions (1 hour and 15 minutes).
2. (2) **Electricity** includes the hourly electricity consumption of 321 clients.
3. (3) **Exchange** (Lai et al. 2018) collects daily exchange rates for

---

\(^1\)https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
8 different countries from 1990 to 2016. (4) **Traffic** contains the hourly road occupancy rates from the California Department of Transportation. (5) **Weather** records 21 meteorological indicators every 10 minutes for one year. (6) **ILI** 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), Reformformer (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.

<table>
<thead>
<tr>
<th>Models</th>
<th>InParformer</th>
<th>FEDformer</th>
<th>Autoformer</th>
<th>Informer</th>
<th>LogTrans</th>
<th>Reformformer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>Weather</td>
<td>96</td>
<td><strong>0.0022</strong></td>
<td><strong>0.036</strong></td>
<td>0.0062</td>
<td>0.062</td>
<td>0.0110</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td><strong>0.0038</strong></td>
<td><strong>0.048</strong></td>
<td>0.0060</td>
<td>0.062</td>
<td>0.0075</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td><strong>0.0033</strong></td>
<td><strong>0.045</strong></td>
<td>0.0041</td>
<td>0.050</td>
<td>0.0063</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td><strong>0.0030</strong></td>
<td><strong>0.042</strong></td>
<td>0.0055</td>
<td>0.059</td>
<td>0.0085</td>
</tr>
<tr>
<td>ETTh2</td>
<td>96</td>
<td><strong>0.117</strong></td>
<td><strong>0.264</strong></td>
<td>0.128</td>
<td>0.271</td>
<td>0.153</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td><strong>0.169</strong></td>
<td><strong>0.319</strong></td>
<td>0.185</td>
<td>0.330</td>
<td>0.204</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td><strong>0.225</strong></td>
<td><strong>0.373</strong></td>
<td>0.231</td>
<td>0.378</td>
<td>0.246</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td><strong>0.241</strong></td>
<td><strong>0.399</strong></td>
<td>0.278</td>
<td>0.420</td>
<td>0.268</td>
</tr>
<tr>
<td>Exchange</td>
<td>96</td>
<td><strong>0.105</strong></td>
<td><strong>0.247</strong></td>
<td>0.154</td>
<td>0.304</td>
<td>0.241</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td><strong>0.207</strong></td>
<td><strong>0.360</strong></td>
<td>0.286</td>
<td>0.420</td>
<td>0.300</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td><strong>0.400</strong></td>
<td><strong>0.498</strong></td>
<td>0.511</td>
<td>0.555</td>
<td>0.509</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td><strong>1.172</strong></td>
<td><strong>0.836</strong></td>
<td>1.301</td>
<td>0.879</td>
<td>1.260</td>
</tr>
<tr>
<td>ILI</td>
<td>24</td>
<td><strong>0.598</strong></td>
<td><strong>0.564</strong></td>
<td>0.708</td>
<td>0.627</td>
<td>0.948</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td><strong>0.553</strong></td>
<td><strong>0.592</strong></td>
<td>0.584</td>
<td>0.617</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td><strong>0.653</strong></td>
<td><strong>0.658</strong></td>
<td>0.717</td>
<td>0.697</td>
<td>0.791</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td><strong>0.789</strong></td>
<td><strong>0.748</strong></td>
<td>0.855</td>
<td>0.774</td>
<td>0.874</td>
</tr>
</tbody>
</table>

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, \(L_y \in \{24, 36, 48, 60\}\). Weather, ETTh2, Exchange, and ILI datasets are 10-minutely, hourly, daily, and weekly recorded, respectively.

form (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 ET 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 → 0.260) MSE reduction in ETTh2, 7.5% (0.201 → 0.186) in Electricity, 15.1% (0.271 → 0.230) in Exchange and 6.2% (0.276 → 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-
significant and stable improvements, which indicates its advantages in prediction capacity.

<table>
<thead>
<tr>
<th>Decomp</th>
<th>EvoSTD</th>
<th>MOE</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>ETTh1</td>
<td>96</td>
<td><strong>0.379</strong></td>
<td><strong>0.419</strong></td>
</tr>
<tr>
<td>192</td>
<td><strong>0.416</strong></td>
<td><strong>0.440</strong></td>
<td>0.563</td>
</tr>
<tr>
<td>336</td>
<td><strong>0.455</strong></td>
<td><strong>0.458</strong></td>
<td>0.536</td>
</tr>
<tr>
<td>720</td>
<td><strong>0.475</strong></td>
<td><strong>0.488</strong></td>
<td>0.602</td>
</tr>
<tr>
<td>Weather</td>
<td>96</td>
<td><strong>0.212</strong></td>
<td><strong>0.286</strong></td>
</tr>
<tr>
<td>192</td>
<td><strong>0.259</strong></td>
<td><strong>0.322</strong></td>
<td>0.336</td>
</tr>
<tr>
<td>336</td>
<td><strong>0.317</strong></td>
<td><strong>0.358</strong></td>
<td>0.379</td>
</tr>
<tr>
<td>720</td>
<td><strong>0.395</strong></td>
<td><strong>0.409</strong></td>
<td>0.409</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attn</th>
<th>ProbAttn</th>
<th>AutoCorr</th>
<th>FEA</th>
<th>InParAttn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>ETTh1</td>
<td>96</td>
<td>0.439</td>
<td>0.456</td>
<td>0.397</td>
</tr>
<tr>
<td>192</td>
<td>0.562</td>
<td>0.524</td>
<td>0.439</td>
<td>0.516</td>
</tr>
<tr>
<td>336</td>
<td>0.509</td>
<td>0.502</td>
<td>0.464</td>
<td>0.466</td>
</tr>
<tr>
<td>720</td>
<td>0.603</td>
<td>0.565</td>
<td>0.473</td>
<td>0.490</td>
</tr>
<tr>
<td>Weather</td>
<td>96</td>
<td>0.242</td>
<td>0.330</td>
<td>0.200</td>
</tr>
<tr>
<td>192</td>
<td>0.362</td>
<td>0.423</td>
<td>0.292</td>
<td>0.362</td>
</tr>
<tr>
<td>336</td>
<td>0.381</td>
<td>0.427</td>
<td>0.363</td>
<td>0.416</td>
</tr>
<tr>
<td>720</td>
<td>0.575</td>
<td>0.466</td>
<td>0.529</td>
<td>0.529</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Option</th>
<th>IPConv</th>
<th>W/o IPConv</th>
<th>TAA-A</th>
<th>TAA-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>ETTh1</td>
<td>96</td>
<td>0.379</td>
<td>0.419</td>
<td>0.378</td>
</tr>
<tr>
<td>192</td>
<td><strong>0.416</strong></td>
<td><strong>0.440</strong></td>
<td>0.418</td>
<td>0.441</td>
</tr>
<tr>
<td>336</td>
<td><strong>0.455</strong></td>
<td><strong>0.458</strong></td>
<td>0.456</td>
<td>0.461</td>
</tr>
<tr>
<td>720</td>
<td><strong>0.475</strong></td>
<td><strong>0.488</strong></td>
<td>0.546</td>
<td>0.536</td>
</tr>
<tr>
<td>Weather</td>
<td>96</td>
<td>0.212</td>
<td>0.286</td>
<td>0.211</td>
</tr>
<tr>
<td>192</td>
<td>0.259</td>
<td>0.322</td>
<td>0.293</td>
<td>0.352</td>
</tr>
<tr>
<td>336</td>
<td><strong>0.317</strong></td>
<td><strong>0.358</strong></td>
<td>0.350</td>
<td>0.386</td>
</tr>
<tr>
<td>720</td>
<td><strong>0.395</strong></td>
<td><strong>0.409</strong></td>
<td>0.425</td>
<td>0.428</td>
</tr>
</tbody>
</table>

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


