Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Hidformer: Hierarchical dual-tower transformer using multi-scale mergence for long-term time series forecasting

Zhaoran Liu, Yizhi Cao, Hu Xu, Yuxin Huang, Qunshan He, Xinjie Chen, Xiaoyu Tang, Xinggao Liu [*]

State Key Laboratory of Industry Control Technology, College of Control Science & Engineering, Zhejiang University, Hangzhou, PR China

ARTICLE INFO

Keywords:
Time-series forecasting
Transformer
Token mixer
Domain features

ABSTRACT

Long-term time series forecasting has received a lot of popularity because of its great practicality. It is also an extremely challenging task since it requires using limited observations to predict values in the long future accurately. Recent works have demonstrated that Transformer has strong potential for this task. However, the permutation-invariant property of the Transformer and some other prominent shortcomings in the current Transformer-based models, such as missing multi-scale local features and information from the frequency domain, significantly limit their performance. To improve the accuracy of the long-term time series forecasting, we propose a Transformer-based model called Hidformer. This model can either learn temporal dynamics from the time domain or discover particular patterns from the frequency domain. We also design a segment-and-merge architecture to provide semantic meanings for the inputs and help the model capture multi-scale local features. Besides, we replace Transformer's multi-head attention with highly-efficient recurrence and linear attention, which gives our model an advantage over other Transformer-based models in terms of computational efficiency. Extensive experiments are conducted on seven real-world benchmarks to verify the effectiveness of Hidformer. The experimental results show that Hidformer achieves 72 top-1 and 69 top-2 scores out of 88 configurations. It dramatically improves the prediction accuracy and outperforms the previous state-of-the-art, proving the superiority of our proposed method.

1. Introduction

Time series data is everywhere, and forecasting is one of the most popular subtasks for time series analysis. An accurate time-series forecasting expert system can provide reliable predictive information for the future, thus benefiting downstream tasks such as decision-making and early warning. Time-series forecasting is widely used in many fields, including finance (C. Wang et al., 2022), weather (Venkatachalam et al., 2023), security (Zheng et al., 2022), traffic (Reza et al., 2022), etc. Long-term time series forecasting (LTSF) has recently attracted significant research interest because people always hope to look further into the future. Compared with short-term forecasting, the demand for LTSF is greater in reality, and it is also more challenging, as it needs to extract more historical information to make accurate predictions for the long future. With the development of deep learning, the deep neural network has become the mainstream method of time series modeling due to its powerful ability to learn features automatically (Huang et al., 2023; H. Wang et al., 2023; Chen et al., 2023).

The Transformer proposed by Vaswani et al. (2017) has attracted a great deal of interest in recent years. Transformer is the first sequence transduction model based entirely on attention mechanism. It includes an encoder and a decoder. The encoder is responsible for encoding the input sequence into a context-aware representation, and the decoder generates the target sequence based on the encoder's output. Both the encoder and decoder consist of multiple self-attention layers and feed-forward layers. The self-attention mechanism is the core of Transformer. It allows the model to determine the importance of each position during encoding and decoding by calculating the correlations between different positions in the input sequence and applying these importance levels as weights to the feature vectors at the corresponding positions. Transformer has shined in many deep learning research areas, including natural language processing (NLP), computer vision (CV), and speech recognition (H. Li et al., 2022; T. Wu et al., 2022). For example, Liu et al. (2023) used Transformer to learn contextual semantic representation

* Corresponding author.
E-mail addresses: laujiuyin@zju.edu.cn (Z. Liu), liuxinggaozju@163.com (X. Liu).

https://doi.org/10.1016/j.eswa.2023.122412
Received 10 July 2023; Received in revised form 29 October 2023; Accepted 29 October 2023
Available online 1 November 2023

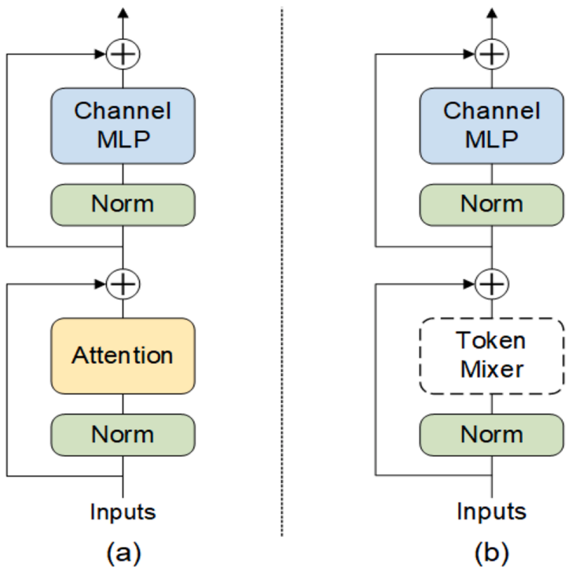

Fig. 1. The architecture of (a) Transformer block, (b) Metaformer block.

and extract relations between entities from the input sentences. Barbalau et al. (2023) introduced convolutional vision transformer blocks and pre-trained the Transformer for video anomaly detection. W. Chen et al. (2023) incorporated the properties of speech into their modified Transformer for paralinguistic speech processing. Benefiting from its self-attention mechanism, Transformer can learn the semantic correlations among the elements in a long sequence. Due to this advantage, Transformer shows great potential in the time series prediction, and there has been a surge of Transformer-based solutions these years, such as Reformer (Kitaev et al., 2020), Informer (H. Zhou et al., 2021), Autoformer (H. Wu et al., 2021), FEDformer (T. Zhou et al., 2022), and so on. These Transformer-based models perform well in short-term time series forecasting. However, in the LTSF situation, the Transformer-based models have not yet achieved satisfactory results. There is still an apparent error between the predictions and the observations. Worse even, the recent paper (Zeng et al., 2022) proves that a simple linear layer can perform better than previous Transformer-based models on the LTSF benchmarks, which undoubtedly questions the validity of this line of research.

We believe that Transformer has excellent potential and has yet to demonstrate its capabilities on LTSF tasks for the following three reasons. The first is the natural permutation-invariant property of the self-attention mechanism. Self-attention is computed in parallel for each point in the sequence without considering the points' orders. But in LTSF, we are mainly interested in the temporal relations in an ordered set of continuous points. Temporal dynamics is the most important information in time series analysis, which is self-attention not good at grasping. While using positional embedding or subseries embedding can introduce some location tips, the natural permutation-invariant property will still make Transformer inevitably lose lots of temporal information. Second, existing models have been unable to simultaneously capture the multi-scale local features and global features of time series. LTSF often requires the most extended lookback window to extract as much historical information as possible. Such a long sequence includes rich local and global features, which are crucial for predicting future points. Simply using the self-attention mechanism cannot fully extract features hidden in the time sequence, especially multi-scale local features. Third, most of the Transformer-based models ignore the information from the frequency domain. It is well-known that time series can be represented from the time and frequency domains. Still, people usually concentrate only on the information in the time domain and ignore the frequency domain. Information from the frequency domain is significant in time series analysis. For example, high-frequency signals in the frequency domain often correspond to trend changes in time series (T. Zhou et al., 2022), and frequency domain mapping can also help the models extract global features. Though frameworks like FEDformer dive into the frequency domain to make predictions, they ignore features from the time domain this time. More efforts are still needed to discover solutions that allow Transformer to well utilize both time domain and frequency domain features for LTSF.

The above challenges and issues limit Transformer-based models' performance in long-term forecasting scenarios. Given the excellent performance in other areas, we remain confident in Transformer's future for the LTSF task. Yu et al. (2022) discussed Transformer's architecture in depth. They pointed out that the general structure of the Transformer, instead of the attention module, was essential to the model's performance. Based on their hypothesis, they abstracted the self-attention module into a token mixer and proposed the Metaformer structure shown in Fig. 1. The token mixer can be designed depending on specific targets to achieve expected results. Inspired by Metaformer, a more suitable token mixer can better leverage the capabilities of Transformer in the LTSF task. To address the existing challenges in LSLF, we think a good Transformer-based framework should contain the following characteristics. First, the input tokens should have semantic meanings. Unlike NLP or CV, an isolated time step is meaningless in time series. For example, a token in NLP is usually a word with a specific semantic meaning, while a single time step represents only the values at that moment. Most of the previous Transformer-based models adopt point-wise tokens, thus losing the correlation between adjacent time steps. In contrast, it is more reasonable to consider a subsequence of the time series as a token, since such a token contains temporal correlations of several points, rather than only the values of an isolated point. Second, the model should be able to extract multi-scale features from the time series. After constructing tokens with semantic meanings, how to extract features from them becomes the main obstacle. Time series may represent diverse temporal dynamics at different resolutions (e.g., minutes, hours, days). Local features at various scales are all critical to forecasting the trend of future changes, while a brutal attention calculation on input tokens will cause a significant locality loss. It is desirable to find a hierarchical approach that computes tokens at different resolutions step by step to extract multi-scale local features, referring to the feature pyramid in the CV domain. Third, the model should be capable of utilizing information from both the time and frequency domains. Information from the time domain can help the model summarize historical trends and seasonality. Information from the frequency domain facilitates the model to capture the global properties and variable correlations of the input sequences. At the same time, frequency signals, especially those of high frequency, may assist the model in detecting key points that are hard to notice in the time domain. The time domain information should cooperate with the frequency domain information but not interfere with each other, so the model ought to recognize these two signals separately and integrate them. Fourth, the design of the token mixers should consider time series' temporal dynamics. For example, the positional correlation of each token is very important, so the token mixer should be able to use the positional information adequately. At last, the computational complexity of the model is as low as possible with guaranteed performance. Otherwise, it will incur substantial computational costs when dealing with very long sequences.

Based on our theories, we propose a **Hi**erarchical **d**ual-tower Trans**former**-based model using multi-scale mergence (**Hidformer**) to solve the LTSF task. Our contributions are summarized as follows:

1) We design a segment-and-merge approach to extract multi-scale temporal features from long time series. A long sequence is first segmented into several overlapped subseries, and each subseries is seen as an input token. Then, adjacent tokens will be merged at each model block, and the upper blocks handle the fewer tokens. After calculation, different blocks obtain local features corresponding to diverse scales.

2) We design a Transformer-based architecture that consists of two

towers. One is for processing the input sequences from the time domain perspective, and the other is for processing the input sequences from the frequency domain perspective. These two channels are independent, and the decoder will integrate their learned features. The dual-tower backbone ensures that our model can simultaneously utilize information from multiple domains to assist prediction.

3) To take full advantage of the Transformer structure, token mixers in the two towers are disparate. The time domain tower employs an efficient recurrent neural network as its token mixer, which can learn temporal dynamics from the time series while preserving its positional information. The frequency domain tower employs a sparse attention module as its token mixer. Real-world multivariate times series often exhibit low-rank properties under the Fourier basis. Thus, sparse attention is sufficient to represent features in the frequency domain.

4) Compared with previous state-of-the-art (SOTA), extensive experimental results on 7 real-world time series forecasting benchmarks show the effectiveness of our proposed model. Specifically, Hidformer achieves 72 top-1 scores and 69 top-2 scores out of 88 configurations. It reduces the prediction error under the multivariate LTSF setting and the univariate LTSF setting by an average of 7 % and 8 %, respectively, on the basis of the previous SOTA. It can be believed that Hidformer provides a good baseline for long-term time series forecasting research.

2. Related work

Time series forecasting is a widely studied problem. The existing solutions for this problem are mainly divided into three types: (1) statistical methods, (2) recurrent neural networks (RNNs), and (3) Transformer-based methods.

Traditionally, statistical methods were the main force before the rise of deep learning. Many statistical models exhibit regression properties and can uncover correlations between continuous time steps. These models include linear regression (Ilic et al., 2021), support vector regression (Karmy & Maldonado, 2019), partial least squares regression (Merino et al., 2020), etc. Autoregressive integrated moving average (Kaytez, 2020) and Prophet (Ning et al., 2022) are also representative statistical methods, but they are only appropriate for handling univariate time series. Statistical methods are often highly interpretable and fast to train, making them widely used in the early days. However, their performance pales when facing long sequences. These models apply only to simple linear or nonlinear relationships and often perform poorly in complex situations. They are prone to the long-term dependence problem, i.e., the error in the prediction results becomes more prominent as the time interval increases. This problem is usually due to the inability of these methods to capture the deeper pattern inherent in the time series. Moreover, in LTSF, seasonality is typical, and traditional statistical methods are hard to deal with such seasonal factors. These obvious shortcomings limit the application of conventional statistical methods in LTSF scenarios.

More recently, the development of deep learning has spawned a tremendous increase in deep neural network-based time series analysis methods, including multilayer perceptron (MLP) (Dudek, 2020), convolutional neural networks (CNNs) (Ensafi et al., 2022), and RNNs (Afrin & Yodo, 2022). Neural network-based solutions gradually dominate in time series analysis because of their ability to automatically learn complex patterns and relationships from large datasets. Among the various neural networks, RNNs have become the most preferred method in time series prediction, attributed to their capabilities of learning temporal dynamics via cycles in the network of nodes. Afrin & Yodo (2022) proposed a long short-term memory (LSTM)-based traffic data prediction framework, in which the LSTM was used to discover temporal and spatial trends. Teng et al. (2022) coupled LSTM with the empirical mode decomposition method and sample entropy index to form a model for PM2.5 concentration prediction. Their model accurately predicted short-term (within 6 h) concentration. The interval prediction framework for wind power systems proposed by F. Liu et al. (2022) took a bidirectional gated recurrent unit (GRU) as its core predictor. It learned the dependence of wind power time series in chronological and reversed chronological order. Despite the impressive results gained by RNN-based methods, they are often stuck in the problem of gradient vanishing or exploding. More importantly, RNNs are difficult to capture long-term dependencies. Their iterated multi-step (IMS) forecasting manner suffers from error accumulation effects. Therefore, RNNs' predictive ability decreases with increasing time steps, making them perform poorly in long-term forecasting tasks.

The huge success in other areas pushes Transformer-based methods into the spotlight of LTSF research. H. Zhou et al. (2021) is a pioneer in this line. Informer used self-attention distilling operation to reduce the total space complexity, which helps to receive long-term sequence. It also proposed a generative decoder that performed direct multi-step (DMS) forecasting, successfully avoiding error accumulation during the inference phase. Inspired by Informer, subsequent studies adopted DMS strategies, like Autoformer (H. Wu et al., 2021) and Pyraformer (Park et al., 2023). Autoformer decomposed the input series into a trend and a seasonal part and employed a series-wise attention mechanism to discover correlations hidden in the seasonal part. The refined representations from the two parts are summed up to get the final prediction. Pyraformer developed the pyramidal attention mechanism to learn temporal correlations of different ranges. It used a fully connected layer coupling spatial–temporal axes as the decoder. These models improved Transformer's performance on LTSF benchmarks, but the prediction results were still inaccurate. They still used point-wise tokens containing meaningless semantics and neglected vital information from the frequency domain. FEDformer (T. Zhou et al., 2022) applied Transformer to the frequency domain by Fourier-enhanced blocks. Through frequency domain mapping, FEDformer can capture global characteristics of time series. FEDformer also used a seasonal-trend decomposition strategy and matched up with its frequency attention decoder to get final results. FEDformer noticed the fact that the global properties of time series lie in the frequency domain but ignored multi-scale local features that exist in the time domain. Transformer is also well known for its memory bottleneck when handling long sequences and quadratic computation complexity of self-attention. Many Transformer-based works focused on improving computational efficiency. For example, Reformer (Kitaev et al., 2020) replaced dot-product attention with locality-sensitive hashing, while LogTrans (S. Li et al., 2019) adopted convolutional self-attention. These models are computation-friendly but inevitably bring performance decline. A. Zeng et al. (2022) questioned the line of exploring Transformers for time series forecasting, but this questioning was soon broken by the recent work PatchTST (Nie et al., 2023). PatchTST contained two key designs: one is constructing subseries-level tokens by aggregating time steps, and the other is the channel-independence strategy ensuring each token only had values from a single channel. PatchTST offers a bright path for building Transformer-based LTSF models, which, however, still require multi-resolution features and frequency domain features to improve performance further.

3. Methodology

3.1. Problem definition

Here we first give the problem definition of LTSF. Let $X_t = \{x_{t-L}, \cdots, x_t | x_i \in \mathbb{R}^{1\times d_x}\}$ be the historical time sequence with a size L lookback window, where x_i is the observed data at time step i, and d_x is the input's variable dimension. Then define $Y_t = \{y_{t+1}, \cdots, y_{t+H} | y_i \in \mathbb{R}^{1\times d_y}\}$ as the target series with a size H horizon, where y_i is the data at the i-th time step and d_y is the output's variable dimension. In multivariate forecasting situations $d_y = d_x > 1$, and in univariate forecasting situations $d_y = 1$. Time series forecasting aims at utilizing historical observation X_t to predict the future time sequence Y_t. In the LTSF problem, with a fixed

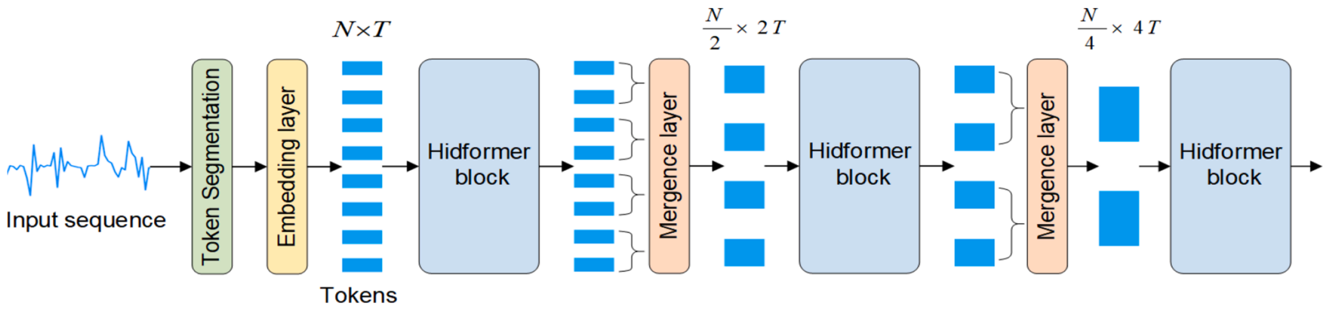

Fig. 2. The segment-and-merge architecture.

lookback window, we encourage a longer horizon, usually longer than 96 time steps, i.e., $H \geq 96$.

3.2. Token segmentation

This subsection will describe the construction of input tokens and then introduce our segment-and-merge approach. A historical time series X_t with length L is first segmented into different subseries. Each subseries contains a continuous point of time step and is seen as an input token. These tokens can be either overlapped or non-overlapped, controlled by the stride S, i.e., the non-overlapping region between two nearby tokens. Denote the token length as T, the input time series will generate N tokens after segmentation, where N can be calculated by the following equation:

$$N = \left\lfloor \frac{(L-T)}{S} \right\rfloor + 2 \tag{1}$$

Segment-wise tokens have two main advantages compared to point-wise tokens. First, a subseries naturally contains temporal dynamics, so a segment-wise token involves temporal correlations among continuous time steps, while a point-wise token lacks semantic meanings. Second, memory usage and computational complexity are largely reduced. By segmentation, the number of input tokens decreases from L to approximately L/S. It means that under the constrained memory and training time, segmentation operation allows models to see a longer lookback window, which can significantly improve models' performance.

3.3. Multi-scale mergence

Multi-scale mergence is a vital technique in our model, which is described in Fig. 2. Each Hidformer block generates the same number of output vectors as the input vectors. By introducing mergence layers between Hidformer blocks, every two adjacent vectors output by the previous block are merged into one vector before being fed into the next block. The calculation process is as follows:

$$\widehat{Z}_i^l = \begin{cases} N_i, l = 1 \\ \left[Z_{2i-1}^{l-1} : Z_{2i}^{l-1}\right], l > 1, 1 \leq i \leq \frac{L_{l-1}}{2} \end{cases} \tag{2}$$

where N_i denotes the i-th token segmented form the input time series; $\widehat{Z}_i^l$ denotes the i-th input vector of the l-th layer, Z_i^l denotes the i-th output vector of the l-th layer, $[:]$ represents the mergence operation; L_{l-1} denotes the number of segments in the layer $l-1$, and Z^{l-1} will be padded to appropriate length if L_{l-1} is not divisible by 2.

The output feature vectors of each block will be concatenated to obtain the final representation for prediction. Because of merging, the upper layer capture features at a coarser resolution. Accordingly, the final representation contains local features from different resolutions, ensuring that the model takes full advantage of the multi-scale information in the time series.

3.4. Preliminaries of our token mixers

This subsection describes two neural-based structures so that we can better articulate the reasons for choosing them as our model's token mixers later on.

1) SRU++

Modified simple recurrent unit (SRU++) (Lei, 2021) is a highly-efficient RNN that combines parallelizable recurrence and attention mechanisms, which is an upgrade from the SRU (Lei et al., 2018). A single layer of SRU takes the following computation:

$$f_t = \sigma\left(W_f z_t + v_f \odot c_{t-1} + b_f\right) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + (1-f_t) \odot (W z_t) \tag{4}$$

$$r_t = \sigma(W_r z_t + v_r \odot c_{t-1} + b_r) \tag{5}$$

$$h_t = r_t \odot c_t + (1-r_t) \odot z_t \tag{6}$$

where W, W_f, and W_r are weight matrices; v_f, v_r, b_f, and b_r are parameter vectors to be learnt; $\odot$ denotes element-wise multiplication. The SRU first adopts a light recurrence component (Equation (3) and (4) which successively computes the hidden states c_t by reading the input vector z_t for each step t. The hidden state c_t is a weighted average between the previous hidden state c_{t-1} and a linear transformation of the input vector z_t, and the weights is defined by a forget gate f_t. The SRU then uses a highway network (Equation (5) and (6) to compute on c_t. The final output state is a weighted average between the hidden state c_t and the input vector z_t, whose weights are controlled by a reset gate r_t.

Distinguishing from previous RNNs, SRU employs two significant code-level configurations to achieve parallelizable recurrence and fast running speed. First, for the input sequence $Z = \{z_1, \cdots, z_L\}$, SRU incorporates the three matrix multiplications across all time steps as a single multiplication:

$$U^{\mathrm{T}} = \begin{pmatrix} W \\ W_f \\ W_r \end{pmatrix} Z^{\mathrm{T}} \tag{7}$$

where U is the output tensor and the superscript T stands for transpose. Such batch operation essentially improves the computation efficiency, such as GPU utilization.

The second configuration is to do element-wise recurrent computations in an efficient way:

$$f_t = \sigma\left(U_{t,0} + v_f \odot c_{t-1} + b_f\right) \tag{8}$$

$$r_t = \sigma\left(U_{t,1} + v_r \odot c_{t-1} + b_r\right) \tag{9}$$

$$c_t = f_t \odot c_{t-1} + (1-f_t) \odot U_{t,2} \tag{10}$$

$$h_t = r_t \odot c_t + (1-r_t) \odot z_t \tag{11}$$

Once the vector U is obtained, each dimension of the hidden vectors is

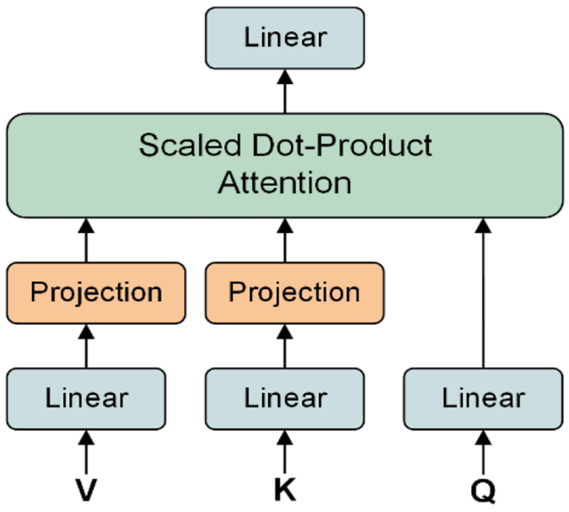

Fig. 3. Linear self-attention.

Table 1
Complexity of multi-head attention and linear self-attention.

Methods	Time	Memory
Multi-head attention	$O(n^2)$	$O(n^2)$
Linear self-attention	$O(n)$	$O(n)$

independent. Thus, the computation process can run in parallel across each dimension.

SRU++ modifies SRU by introducing more non-linear operations. Specifically, SRU++ replace the linear transformation of getting U (Equation (7) with self-attention mechanism. Firstly, query, key, value vectors are computed by the following equations:

$$Q = W_q Z^{\mathrm{T}} \tag{12}$$

$$K = W_k Q \tag{13}$$

$$V = W_v Q \tag{14}$$

where Z is the input sequence, $W_q \in \mathbb{R}^{d' \times d}$, $W_k, W_v \in \mathbb{R}^{d' \times d'}$ are weight matrices, and d' is much smaller than d to reduce memory usage.

Next, the scaled dot-product attention (Vaswani et al., 2017) calculation combines Q, K and V into a feature vector F:

$$F^{\mathrm{T}} = \mathrm{Softmax}\left(\frac{Q^{\mathrm{T}}K}{\sqrt{d}}\right)V^{\mathrm{T}} \tag{15}$$

Finally, a linear transformation turns F into the vector U for element-wise recurrent computations:

$$U^{\mathrm{T}} = W_u(Q + \alpha \bullet F) \tag{16}$$

where W_u is a weight matrix and α is a learnable scalar. This linear transformation involves the residual connection, which stabilizes the training process. And the self-attention mechanism enables the model to capture long-term dependencies from the input sequence.

2) Linear self-attention

The key components of linear self-attention (S. Wang et al., 2020) are two $(n \times k)$-dimensional linear transformation matrices A and B. Like Equation 12–14, the input sequence $\overline{Z}$ is first mapped to query, key and value by weight matrices W_q, W_k, and W_v:

$$Q = W_q \overline{Z} \tag{17}$$

$$K = W_k \overline{Z} \tag{18}$$

$$V = W_v \overline{Z} \tag{19}$$

where Q, K, and V are $(n \times d)$-dimensional matrices. Then the two linear transformation matrices A and B project K and V into $(k \times d)$-dimensional matrices $\widehat{K}$ and $\widehat{V}$ respectively:

$$\widehat{K} = A^{\mathrm{T}}K \tag{20}$$

$$\widehat{V} = B^{\mathrm{T}}V \tag{21}$$

Finally, scaled dot-product attention computation is introduced to get the feature vector F:

$$\begin{aligned} F &= \mathrm{Softmax}\left(\frac{W_q Z (A^{\mathrm{T}} W_k Z)^{\mathrm{T}}}{\sqrt{d}}\right) B^{\mathrm{T}} W_v Z \\ &= \mathrm{Softmax}\left(\frac{Q(\widehat{K})^{\mathrm{T}}}{\sqrt{d}}\right)\widehat{V} \end{aligned} \tag{22}$$

The above computational process is shown in Fig. 3. It only requires O(nk) time and space complexity to obtain F. So, if $k \ll n$, the computational complexity can be significantly reduced.Table 1

3.5. A dual-tower backbone

Hidformer takes a framework consisting of two towers: Time Tower and Frequency Tower. The Time Tower learns temporal dynamics hidden in the time series, and the Frequency Tower studies patterns corresponding to the time series in the frequency domain. Blocks in both towers adopt the Metaformer (Yu et al., 2022) structure, which has been proven a powerful architecture. As illustrated in Fig. 1, Metaformer has four key components: a token mixer, a channel MLP, layer normalization (Rodrawangpai & Daungjaiboon, 2022), and residual connection. In our model, channel MLP is a two linear layers feedforward neural network (FFN) with GELU (Guo et al., 2022) activation function. Therefore, the component that needs to be carefully designed is the token mixer. The token mixer should be designed according to different requirements to acquire desired features. This subsection will describe our proposed model in detail, whose architecture is illustrated in Fig. 4.

1) Time tower

Time blocks in Time Tower employ SRU++ as the token mixer. A time block is preceded by a mergence layer (except the first block). The mergence layer merges the vectors output from the previous block into the vectors with coarser resolutions. Then the SRU++ takes the merged vectors as its input and performs recurrent computations. Hidden states of SRU++ are passed to the next block after the transformation of the FFN. And the last hidden state of SRU++ is concatenated with the ones from the other blocks to construct the final temporal feature vector. Time Tower organizes every time block hierarchically and outputs the feature vector containing multi-scale local features from the time domain.

The main difference between our time block and vanilla transformer encoder is the abandonment of multi-head attention. Transformer is known for its highly parallelized computations. However, the multi-head attention does not consider the time dimension's characteristics, and it treats all points in the input sequence as occurring at the same time step. Temporal dynamics is an essential characteristic of time sequences, so the natural permutation-invariant property of multi-head attention is a fatal flaw when handling time series forecasting tasks. Positional embeddings can alleviate this problem to a certain extent, but they still cannot provide adequate time information for the model. At the same time, the computational cost of the Transformer increases obviously when the sequence is very long. It needs to process every position in the input sequence synchronously and perform attention computation for all positions. On the contrary, RNNs are good at learning temporal correlations due to their recurrence mode and can accommodate

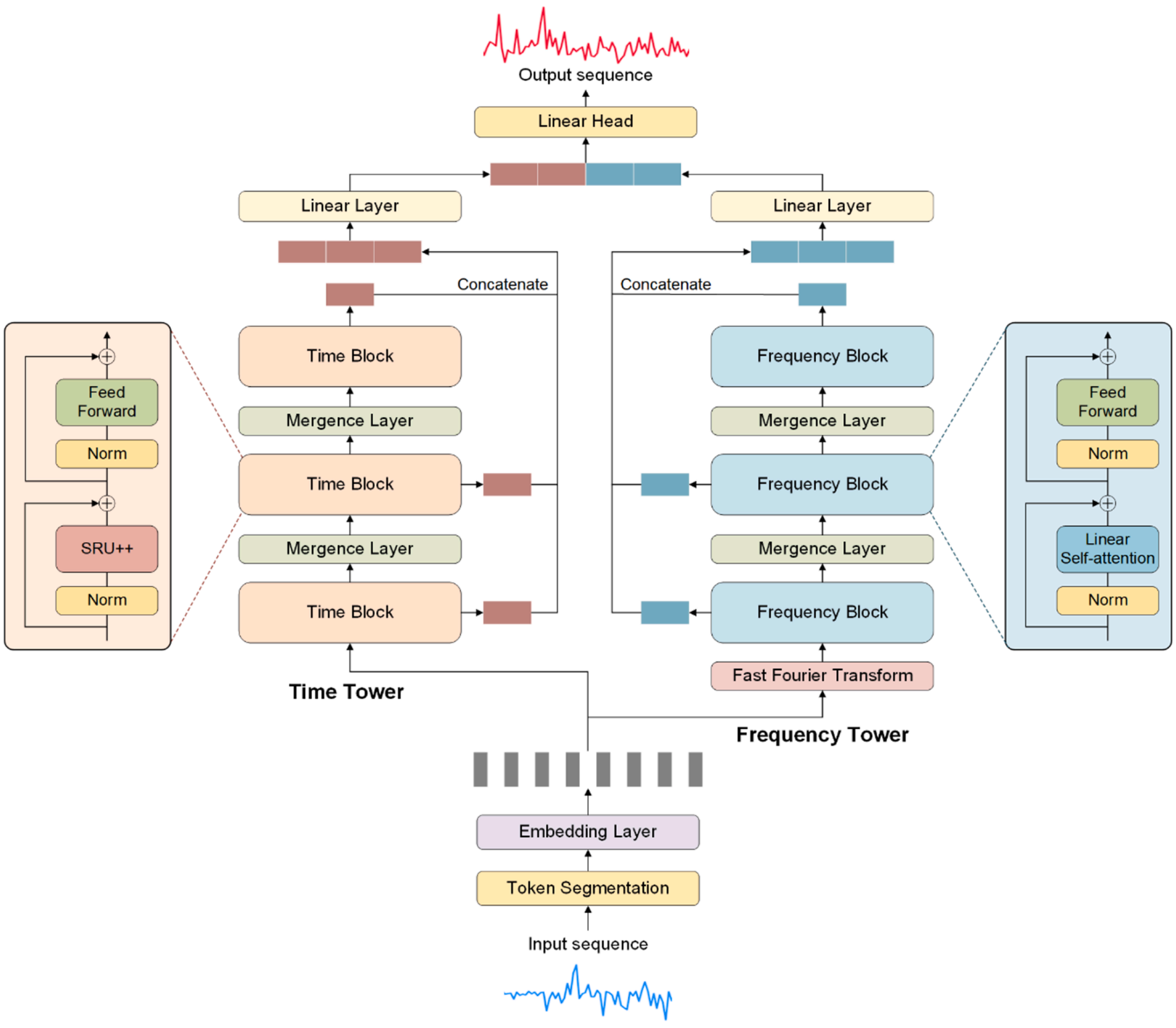

Fig. 4. The architecture of Hidformer.

Table 2
Statistics of the used datasets.

Datasets	Weather	Traffic	Electricity	ETTm1	ETTm2	ETTh1	ETTh2
Variables	21	862	321	7	7	7	7
Time steps	52,696	17,544	26,304	69,680	69,680	17,420	17,420

sequences of different lengths. But previous RNNs are criticized for their low computational efficiency and need to improve at capturing long-term dependencies in long series.

Compared to the multi-head attention mechanism and previous RNNs, the advantages of using SRU++ in LTSF are apparent. First, SRU++ well captures temporal dynamics by its recurrent computations. Second, it breaks the bottleneck of RNNs' parallelized computing and significantly improves computational efficiency. Third, it employs the self-attention mechanism to learn long-term dependencies hidden in the long sequences. Additionally, the number of parameters reduces explicitly compared to Transformer. SRU++ fits well with our target, and it learns multi-scale temporal features with the help of Time Tower's hierarchical architecture.

2) Frequency Tower

Frequency Tower is another pillar of our model. It is responsible for processing time series in the frequency domain, independent of the Time Tower. We first use a fast Fourier transform (FFT) layer to convert tokens N in the time domain into tokens $\overline{N}$ in the frequency domain:

$$\overline{N} = \mathrm{FFT}(N) \tag{23}$$

FFT is an efficient method for computing the Fourier transform, it compresses the computation complexity of Fourier transform from N^2 to $N\log N$.

Low-rankness is a crucial property of time series data in the frequency domain. In the frequency domain, the Fourier transform transforms a discrete-time series with n points into a complex vector of length n. Since the fundamental oscillation frequency is one of the periods of the original signal, these complex vectors usually have complex conjugate symmetry, which means that the entire frequency-domain signals can be described with only a few complex vectors. This property exhibits the characteristics of low-rank matrices, i.e., only a few vectors are needed to approximate the entire vector space. T. Zhou et al. (2022) also discovered that time series tend to have sparse representations on a basis like Fourier basis. Therefore, a low-rank Fourier matrix is enough to

Table 3
Multivariate LTSF results. The best result is highlighted in **bold**, and the second best is underlined.

Models		Hidformer-64		Hidformer-42		PatchTST/64		PatchTST/42		DLinear		FEDformer		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.134**	**0.173**	0.139	0.181	0.149	0.198	0.152	0.199	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405
	192	**0.172**	**0.218**	0.179	0.221	0.194	0.241	0.197	0.243	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434
	336	**0.212**	**0.246**	0.218	0.250	0.245	0.282	0.249	0.283	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543
	720	**0.263**	**0.275**	0.271	0.283	0.314	0.334	0.320	0.335	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705
Traffic	96	0.372	0.257	0.376	0.258	**0.360**	**0.249**	0.367	0.251	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410
	192	0.389	0.261	0.387	0.260	**0.379**	**0.256**	0.385	0.259	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435
	336	**0.386**	**0.258**	0.392	0.261	0.392	0.264	0.398	0.265	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434
	720	**0.410**	**0.272**	0.417	0.275	0.432	0.286	0.434	0.287	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466
Electricity	96	0.119	0.216	**0.118**	**0.214**	0.129	0.222	0.130	0.222	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393
	192	**0.134**	0.231	0.138	**0.230**	0.147	0.240	0.148	0.240	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417
	336	**0.138**	**0.239**	0.143	0.245	0.163	0.259	0.167	0.261	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422
	720	**0.162**	**0.261**	0.168	0.266	0.197	0.290	0.202	0.291	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427
ETTh1	96	**0.361**	**0.388**	0.364	0.390	0.370	0.400	0.375	0.399	0.375	0.399	0.376	0.415	0.435	0.446	0.941	0.769
	192	**0.394**	**0.415**	0.398	0.422	0.413	0.429	0.414	0.421	0.405	0.416	0.423	0.446	0.456	0.457	1.007	0.786
	336	**0.401**	**0.410**	0.408	0.416	0.422	0.440	0.431	0.436	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784
	720	**0.412**	**0.434**	0.420	0.439	0.447	0.468	0.449	0.466	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857
ETTh2	96	0.280	**0.336**	0.282	0.340	**0.274**	0.337	**0.274**	0.336	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952
	192	0.325	0.358	**0.321**	**0.356**	0.341	0.382	0.339	0.379	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542
	336	**0.312**	0.368	0.316	**0.366**	0.329	0.384	0.331	0.380	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642
	720	0.356	0.401	**0.353**	**0.395**	0.379	0.422	0.379	0.422	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619
ETTm1	96	**0.275**	**0.332**	0.280	0.328	0.293	0.346	0.290	0.342	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560
	192	**0.309**	0.349	0.314	**0.348**	0.333	0.370	0.332	0.369	0.335	0.365	0.365	0.415	0.514	0.495	0.725	0.619
	336	**0.339**	**0.362**	0.345	0.364	0.369	0.392	0.366	0.392	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741
	720	**0.373**	**0.385**	0.382	0.391	0.416	0.420	0.420	0.424	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845
ETTm2	96	0.167	0.256	**0.160**	**0.251**	0.166	0.256	0.165	0.255	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462
	192	0.211	0.285	**0.208**	**0.278**	0.223	0.296	0.220	0.292	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586
	336	**0.248**	0.306	0.254	**0.301**	0.274	0.329	0.278	0.329	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871
	720	**0.330**	**0.357**	0.336	0.362	0.362	0.385	0.367	0.385	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267
Count	**37**		**13**		**5**		**1**		**0**		**0**		**0**		**0**		

Table 4
Average error reduction of Hidformer compared to SOTA methods under the multivariate setting.

Horizon	96		192		336		720		Total	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Reduction	5.0 %	3.4 %	6.1 %	4.6 %	7.2 %	6.6 %	9.6 %	8.6 %	7.4 %	6.1 %

appropriately represent the input sequence's information in the frequency domain and helps reduce the computational complexity. FEDformer randomly selected a subset of Fourier components to construct such a low-rank matrix, which introduced indeterminacy. Instead, our frequency blocks employ the linear self-attention as the token mixer to reduce complexity.

As illustrated in Section 3.4, the linear self-attention calculation only requires O (nk) time and space complexity to obtain the output feature vector, and if $k \ll n$, the computational complexity can be significantly reduced. The aforementioned low-rank property of time series appears in the frequency domain allows us to theoretically choose matrices A and B with a small k in Equation (20) and (21) to project K and V. In fact, we can share the parameters of A and B to further reduce the number of parameters, i.e., $A = B$.

A frequency block uses the linear self-attention module as its token mixer. The rest of its components are the same as the time block. Frequency Tower hierarchically organizes every frequency block and concatenates output vectors from the blocks. Frequency Tower can help the model discover specific patterns to which Time Tower is hard to attend. Frequency domain analysis conduces to discover global features of time series (T. Zhou et al., 2022), which is an essential addition to the local features. More importantly, high-frequency signals may relate to some noteworthy events like trend changes, and such characteristics are hard to notice in the time domain. Besides, frequency domain analysis helps the model discover dependencies between different variables since variables in a multivariate time series have correlations and share similar frequency components. Variable correlation as an auxiliary of temporal correlation can effectively improve model performance.

3.6. *Decoder*

Directly concatenating output vectors from Time Tower and Frequency Tower may cause feature conflict problems since they are generated from different domains. Thus, an adaptor with a single linear layer is placed atop the two towers. The adaptors play the role of feature mapping and dimensionality reduction. We concatenate the two feature vectors from the two adaptors and use a linear head as the decoder to get the prediction result. Like other Transformer-based methods, the decoder adopts the DMS strategy to avoid error accumulation.

3.7. *Reversible instance normalization*

Reversible instance normalization (RevIN) (Shen et al., 2023) has proven an efficient plug-and-play strategy in Transformer-based time series forecasting models. RevIN is a normalization technique that can alleviate the distribution shift problem between the training and test time series data. We use RevIN to normalize the input sequences before segmenting them into tokens and denormalize the model output sequences.

4. Experiments

4.1. *Experimental settings*

1) Datasets

We conduct experiments on seven widely used real-world time series forecasting datasets, including Weather, Traffic, Electricity, and ETT (Electricity Transformer Temperature). ETT includes four datasets: ETTh1, ETTh2, ETTm1, and ETTm2. These widely-used benchmarks are open to the public and are available at (H. Wu et al., 2021). Table 2 summarizes statistics about the seven datasets, and here is a description of them:

Weather contains 21 meteorological indicators, such as air temperature, humidity, etc. The data is collected every 10 min for the whole 2020 year in Germany.

Traffic collects the road occupancy rates measured by different sensors on San Francisco Bay area freeways. Its data is recorded hourly from 2015 to 2016 by the California Department of Transportation.

Electricity describes 370 clients' hourly electricity consumption from 2012 to 2014.

ETT comprises two electricity transformers' data collected from two stations (labeled with 1 and 2) from July 2016 to July 2018. Datasets collected from the two transformers have two resolutions (15 min & 1 h) denoted with m and h. So ETT includes a total of four datasets: ETTm1, ETTm2, ETTh1, and ETTh2. Each dataset records the oil temperature and six power load variables.

2) Compared baselines

We choose five state-of-the-art LTSF models as the baselines for comparison: Informer (H. Zhou et al., 2021), Autoformer (H. Wu et al., 2021), Fedformer (T. Zhou et al., 2022), PatchTST (Nie et al., 2023), and DLinear (Zeng et al., 2022). These baselines are all Transformer-based LTSF models except for DLiner. DLiner challenges the effectiveness of the Transformer-based methods in the LTSF task, so we include it to break the doubt.

3) Implementation details

We train our model using ADAM (Kingma & Ba, 2017) optimizer with an initial learning rate of 10^{-4}. The batch size is set to 32. The training process adopts an early stopping strategy to stop after 5 epochs if there is no loss degradation on the validation set. SRU++ in the time blocks has two hidden layers. The Time Tower has 4 time blocks, and the Frequency Tower has 2 frequency blocks. Dimension of each linear layer is set to 128 and dropout rate of each block is set to 0.2.

For fair comparisons, we follow the setup of the baselines' original papers and set the prediction horizon $H \in \{96, 192, 336, 720\}$. The baseline results are collected from Nie et al. (2023) with the default lookback window $L = 96$ for Transformer-based models, and $L = 336$ for DLinear.

In consistent with PatchTST, our model has two variants in the experiments: Hidformer-42 and Hidformer-64. Hidformer-42 represents segmenting the input sequence into 42 tokens, which corresponds to the lookback window $L = 336$. Hidformer-64 represents segmenting the input sequence into 64 tokens, which corresponds to the lookback window $L = 512$. Token length is set to 16 ($T = 16$) and stride is set to 8 ($S = 8$). We run each configuration with five different random seeds and take the average of these five scores as the final performance. All the models are trained on NVIDIA TITAN RTX 24 GB GPUs.

4) Evaluation metrics

Following previous works, we choose mean square error (MSE) and mean absolute error (MAE) as the metrics to evaluate the models' performance. MSE is also the loss function during the training process of our model. MSE and MAE are calculated as follows:

$$\text{MSE} = \frac{1}{h}\sum_{i=1}^{h}(y_i - \widehat{y}_i)^2 \tag{24}$$

Table 5
Univariate LTSF results. The best result is highlighted in **bold**, and the second best is <u>underlined</u>.

Models		Hidformer-64		Hidformer-42		PatchTST/64		PatchTST/42		DLinear		FEDformer		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.048**	**0.170**	<u>0.049</u>	**0.170**	0.059	0.189	0.055	0.179	0.056	0.180	0.079	0.215	0.071	0.206	0.193	0.377
	192	<u>0.064</u>	<u>0.203</u>	**0.061**	**0.195**	0.074	0.215	0.071	0.205	0.071	0.204	0.104	0.245	0.114	0.262	0.217	0.395
	336	**0.071**	**0.213**	<u>0.073</u>	<u>0.215</u>	0.076	0.220	0.081	0.225	0.098	0.244	0.119	0.270	0.107	0.258	0.202	0.381
	720	**0.078**	<u>0.224</u>	<u>0.081</u>	**0.220**	0.087	0.236	0.087	0.232	0.189	0.359	0.142	0.299	0.126	0.283	0.183	0.355
ETTh2	96	0.132	0.282	0.134	0.282	0.131	0.284	<u>0.129</u>	0.282	0.131	<u>0.279</u>	**0.128**	**0.271**	0.153	0.306	0.213	0.373
	192	0.173	0.331	0.175	0.334	<u>0.171</u>	<u>0.329</u>	**0.168**	**0.328**	0.176	<u>0.329</u>	0.185	0.330	0.204	0.351	0.227	0.387
	336	<u>0.180</u>	<u>0.345</u>	0.186	0.350	**0.171**	**0.336**	0.185	0.351	0.209	0.367	0.231	0.378	0.246	0.389	0.242	0.401
	720	**0.214**	**0.374**	<u>0.216</u>	0.381	0.223	<u>0.380</u>	0.224	0.383	0.276	0.426	0.278	0.420	0.268	0.409	0.291	0.439
ETTm1	96	<u>0.020</u>	<u>0.112</u>	**0.019**	**0.110**	0.026	0.123	0.026	0.121	0.028	0.123	0.033	0.140	0.056	0.183	0.109	0.277
	192	<u>0.033</u>	<u>0.143</u>	**0.031**	**0.139**	0.040	0.151	0.039	0.150	0.045	0.156	0.058	0.186	0.081	0.216	0.151	0.310
	336	<u>0.046</u>	**0.153**	**0.044**	<u>0.156</u>	0.053	0.174	0.053	0.173	0.061	0.182	0.084	0.231	0.076	0.218	0.427	0.591
	720	**0.062**	**0.183**	<u>0.067</u>	<u>0.187</u>	0.073	0.206	0.074	0.207	0.080	0.210	0.102	0.250	0.110	0.267	0.438	0.586
ETTm2	96	0.066	0.187	0.067	0.189	<u>0.065</u>	0.187	<u>0.065</u>	<u>0.186</u>	**0.063**	**0.183**	0.067	0.198	0.065	0.189	0.088	0.225
	192	0.095	0.230	0.094	<u>0.229</u>	<u>0.093</u>	0.231	0.094	0.231	**0.092**	**0.227**	0.102	0.245	0.118	0.256	0.132	0.283
	336	**0.114**	**0.252**	<u>0.115</u>	<u>0.258</u>	0.121	0.266	0.120	0.265	0.119	0.261	0.130	0.279	0.154	0.305	0.180	0.336
	720	**0.162**	**0.306**	<u>0.165</u>	<u>0.310</u>	0.172	0.322	0.171	0.322	0.175	0.320	0.178	0.325	0.182	0.335	0.300	0.435
Count	**14**		**8**		**2**		**2**		**4**		**2**		**0**		**0**		

$$\mathrm{MAE} = \frac{1}{h}\sum_{i=1}^{h}|y_i - \widehat{y}_i| \tag{25}$$

where y represents the ground truth and $\widehat{y}$ represents models' prediction results.

4.2. Main results

1) Multivariate prediction

Table 3 summarizes the experimental results of all models under the multivariate time series forecasting setting. It can be seen that Hidformer shows top performance on most configurations. Specifically, Hidformer achieves 50 best scores and 46 s-best scores out of 56 configurations. Compared to the previous SOTA models, Hidformer reduces MSE by 7.4 % and MAE by 6.1 % on the seven datasets on average. Especially, as the predicted length becomes longer, the performance improvement of Hidformer compared to SOTA models shows an upward trend. We record this observation in Table 4. When the horizon takes 96, 192, and 336, Hidformer reduces MSE by 5.0 %, 6.1 %, and 7.2 % and reduces MAE by 3.4 %, 4.6 %, and 6.6 %, respectively. When the horizon takes 720, the average reduction ratios for MSE and MAE come to a staggering 9.6 % and 8.6 %. Statistically, under the $H = 720$ situation, Hidformer dramatically decreases [MSE, MAE] by: [16 %, 18 %] (Weather,) [5 %, 5 %] (Traffic), [19 %, 10 %] (Electricity), [8 %, 7 %] (ETTh1), [7 %, 6 %] (ETTh2), [10 %, 8 %] (ETTm1), and [9 %, 7 %] (ETTm2). We always hope to predict longer time series as accurately as possible to guide our future decisions. With this premise, the LTSF task emphasizes the prediction accuracy of the long future time steps. Therefore, Hidformer's impressive performance in scenarios requiring a very long horizon is more evidence of its superiority over baselines.

2) Univariate prediction

Table 5 compares the performance of Hidformer with baselines under the univariate time series forecasting setting. Hidformer ranks top-1 on 22 out of 32 configurations and ranks top-2 on 23 configurations. The top-1 results obtained by Hidformer reduce MSE and MAE by an average of 8.3 % and 5.2 % respectively on top of the previous best benchmarks. In univariate LTSF, Hidformer still exhibits robustness against long horizons. Under the $H = 720$ situation, Hidformer decreases [MSE, MAE] by: [10 %, 5 %] (ETTh1), [4 %, 2 %] (ETTh2), [15 %, 11 %] (ETTm1), and [5 %, 4 %] (ETTm2). The results also show that Hidformer has good predictive ability at different granularities (minute and hour) and has good generalization. However, Hidformer doesn't present absolute dominance in univariate LTSF as it performs in multivariate LTSF. We guess this is because fewer variable correlations exist in univariate forecasting tasks, and the Frequency Tower fails to show its power and sometimes may even cause the overfitting problem. In sum, the excellent performance in both multivariate and univariate settings demonstrates the effectiveness of Hidformer.

4.3. Ablation study

Hidformer has five key components: the segmentation layer, mergence layers, time blocks, SRU++, frequency blocks, and one normalization trick: RevIN. To discover their contributions to our model, ablation experiments that remove these parts are carried out. Here are the explanations of six different experimental settings:

1) w/o Segm

This setting abandons segmentation operation and takes each time step as an input token, just like the point-wise method in common LTSF models like Autoformer, Informer, etc.

2) w/o merge

This setting removes the mergence layers in both time blocks and frequency blocks. That is, every block in the hierarchical architecture has the same number of input tokens.

3) w/o time

Table 6
Ablation study results.

Models		Hidformer-64		w/o Segm		w/o Merge		w/o Time		w/o SRU		w/o Freq		w/o RevIN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.134**	**0.173**	0.146	0.194	0.144	0.189	0.649	0.743	0.172	0.240	0.150	0.194	0.139	0.178
	192	**0.172**	**0.218**	0.201	0.239	0.192	0.229	1.278	1.394	0.218	0.285	0.189	0.236	0.179	0.224
	336	**0.212**	**0.246**	0.236	0.265	0.226	0.257	1.780	1.573	0.254	0.309	0.247	0.291	0.222	0.254
	720	**0.263**	**0.275**	0.303	0.320	0.291	0.301	2.136	1.960	0.317	0.333	0.315	0.322	0.272	0.287
Electricity	96	**0.119**	**0.216**	0.129	0.223	0.123	0.218	1.009	0.912	0.141	0.243	0.135	0.230	0.127	0.220
	192	**0.134**	**0.231**	0.144	0.242	0.139	0.137	1.677	1.550	0.152	0.257	0.151	0.246	0.140	0.237
	336	**0.138**	**0.239**	0.158	0.257	0.160	0.262	2.456	2.381	0.171	0.279	0.170	0.265	0.151	0.248
	720	**0.162**	**0.261**	0.188	0.289	0.197	0.294	2.154	2.180	0.193	0.290	0.187	0.277	0.174	0.270
ETTh1	96	**0.048**	**0.170**	0.060	0.183	0.055	0.175	0.203	0.325	0.066	0.192	0.056	0.176	0.053	0.172
	192	**0.064**	**0.203**	0.077	0.220	0.073	0.213	0.257	0.441	0.085	0.216	0.074	0.216	0.075	0.216
	336	**0.071**	**0.213**	0.083	0.228	0.090	0.229	0.292	0.475	0.105	0.238	0.088	0.233	0.085	0.232
	720	**0.078**	**0.224**	0.092	0.239	0.107	0.250	0.453	0.615	0.136	0.285	0.097	0.245	0.088	0.237
ETTm1	96	**0.020**	**0.112**	0.024	0.120	0.026	0.121	0.139	0.242	0.032	0.163	0.030	0.140	0.027	0.128
	192	**0.033**	**0.143**	0.039	0.148	0.038	0.150	0.230	0.305	0.050	0.179	0.043	0.153	0.042	0.155
	336	**0.046**	**0.153**	0.055	0.177	0.062	0.182	0.425	0.556	0.069	0.208	0.058	0.181	0.058	0.177
	720	**0.062**	**0.183**	0.077	0.210	0.088	0.219	0.568	0.850	0.092	0.235	0.082	0.223	0.071	0.200

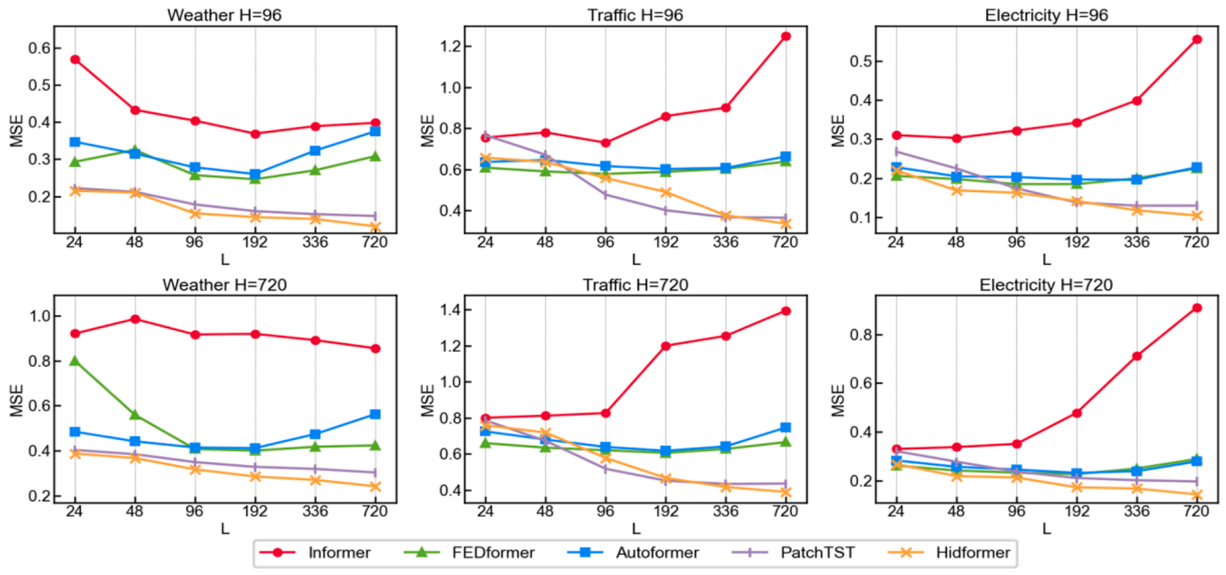

Fig. 5. MSE scores of models with varying lookback windows on Weather, Traffic, and Electricity datasets.

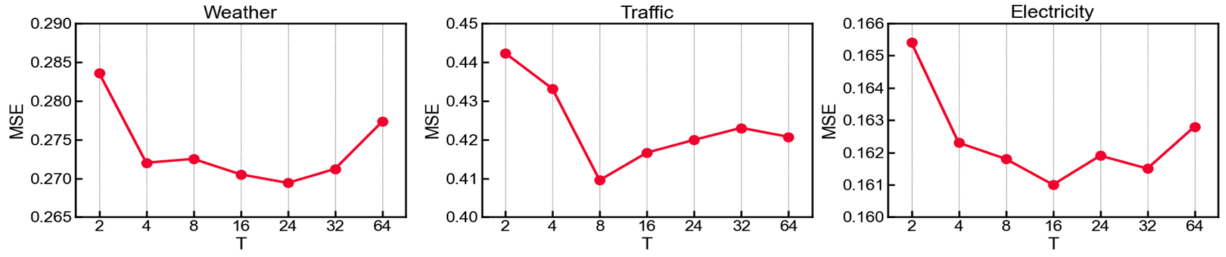

Fig. 6. MSE scores of Hidformer with varying token lengths on Weather, Traffic, and Electricity datasets.

This setting removes Time Tower and only keeps Frequency Tower. We increase the number of frequency blocks properly to ensure that the number of model parameters remains the same.

4) w/o SRU

This setting replaces the token mixer in time blocks from SRU++ to the multi-head attention mechanism. A time block becomes a vanilla Transformer encoder block with a mergence layer.

5) w/o Freq

This setting takes away Frequency Tower and only keeps Time Tower. We increase the number of time blocks properly to ensure that the number of model parameters remains the same.

6) w/o RevIN

This setting doesn't use RevIN to normalize the input sequences.

We test these variants under **1) multivariate forecasting:** Weather, Electricity; **2) univariate forecasting:** ETTh1, ETTm1. We report the results of our ablation experiments in Table 6. Among the variants, the original one remains the best performance on all datasets, illustrating that all the components are essential. **1) w/o Segm:** Segment-wise

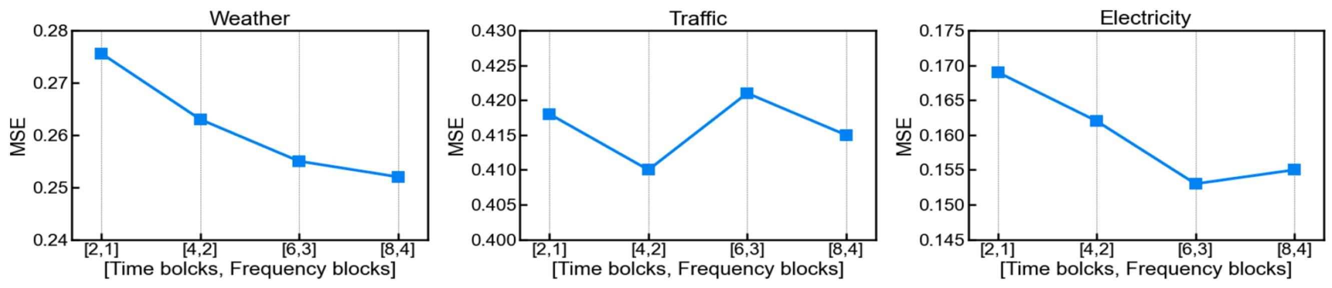

Fig. 7. MSE scores of Hidformer with varying encoder blocks on Weather, Traffic, and Electricity datasets.

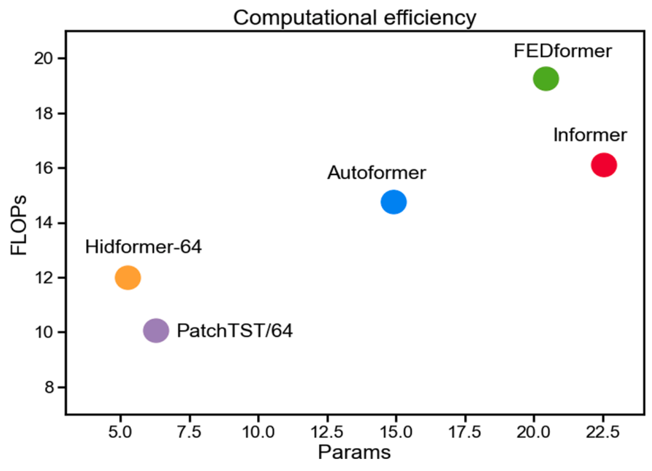

Fig. 8. Comparison of Params and FLOPs for different models.

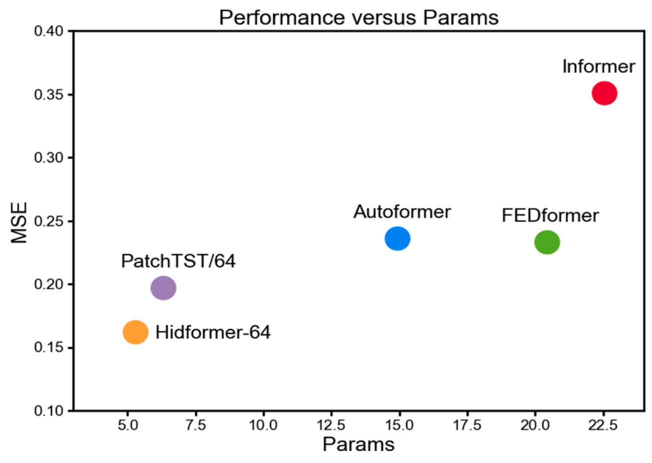

Fig. 9. The performance versus Params for different models.

Table 7
Running time of each model.

Models	Informer	FEDformer	Autoformer	PatchTST/64	Hidformer-64
Running time (s)	0.0153	0.0208	0.0164	0.0112	0.0103

tokens naturally contain temporal correlations of time steps, which offer them semantic meanings. Meanwhile, the segment-wise setting has a longer lookback window for the same number of tokens than the point-wise setting. Thus, the segment-wise setting shows superior results than the point-wise one. **2) w/o Merge:** The mergence layers let Hidformer "see" the time series at different resolutions. Without the mergence layers, the model is hard to capture multi-scale local features. The learned features are monotonous and redundant, so the error increases. **3) w/o Time:** This setting witnesses a dramatic performance degradation, which is expected. The model is designed for time series forecasting, and casting away temporal features will undoubtedly make it lose its predictive power. **4) w/o SRU:** The natural permutation-invariant property of the multi-head attention mechanism is theorized to be a drawback in Transformer-based LTSF models. This experiment illustrates in practice that this property is detrimental to the LTSF task. When replacing SRU++ with multi-head attention, MSE increases by about 28.2 %, and MAE increases by about 21.3 %. Therefore, SRU++ is a better choice as the token mixer. **5) w/o Freq:** The prediction accuracy drops by about 18.2 % or more after missing frequency domain signals. This result proves that information from the frequency domain is also significant for time series forecasting. Thus, it is necessary to consider frequency domain analysis when handling LTSF tasks. **6) w/o RevIN:** RevIN again demonstrates its ability to improve model performance by suppressing the distribution shift. We recommend this effective normalization technique for the other relevant works.

4.4. Varying lookback window

In the ablation study, we argue that the segment-wise setting easily has a longer lookback window than the point-wise setting. Intuitively, a longer lookback window can bring models a larger receptive field so as to improve models' performance. However, literature (Zeng et al., 2022) and (Nie et al., 2023) both observed that most of the Transformer-based LTSF models could not benefit from a longer lookback window. To explore whether Hidformer would benefit from a longer lookback window, we use various lookback window $L \in \{24, 48, 96, 192, 336, 720\}$, and set the prediction horizon $H \in \{96, 720\}$. Experiments are conducted on three datasets: Weather, Traffic, and Electricity. The other baselines share the same settings. Fig. 5 shows the results with varying lookback windows. It can be seen that except for Hidformer and PatchTST, other Transformer-based baselines do not benefit from a longer lookback window. And as L becomes longer, the prediction error of these models even tends to grow up. Our model, Hidformer, can continue to benefit from a longer lookback window and outperforms PatchTST. Since both PatchTST and our model adopt segment-wise tokens, we reasonably suspect that the segment-wise setting is more effective in learning temporal information from a larger receptive field. Further, our segment-and-merge approach more efficiently aggregates the learned temporal features. In contrast, the point-wise setting is ineffective in utilizing large receptive fields. Under the point-wise setting, a longer lookback window may instead cause problems such as feature redundancy, overfitting, and high computational complexity.

4.5. Varying token lengths

In our segment-and-merge architecture, the initial token length is an essential hyperparameter. To discover the effect of this value, we fix the lookback window to be 512 and set the different token lengths $T \in \{2, 4,$

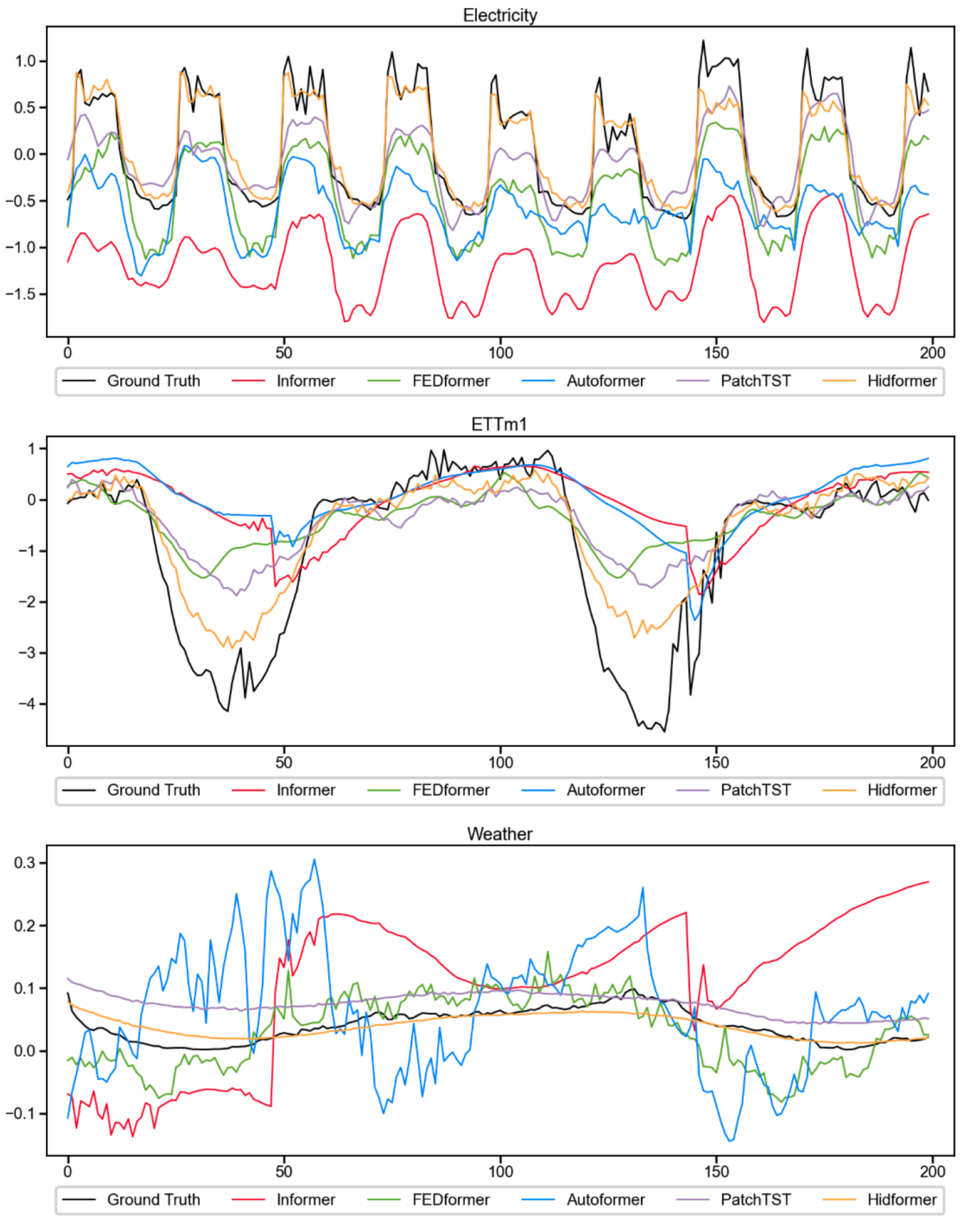

Fig. 10. Comparison of Params and FLOPs for different models.

$8, 16, 24, 32, 64\}$. The stride S is set equal to the token length, which means tokens are not overlapping. Each configuration gives a 720 steps prediction on three datasets: Weather, Traffic, and Electricity. As shown in Fig. 6, the performance of Hidformer does not show drastic fluctuations due to changes in token length. MSE score remain at a stable level as long as T is not too large or too small. This is because the mergence layers can internally integrate information from the sequences at different scales, so Hidformer is not sensitive to the initial token length. This sensitivity analysis exhibits the robustness of Hidformer against the hyperparameter T, and demonstrates it is appropriate to take the median value $T = 16$ as the default.

4.6. Varying encoder blocks

Hidformer employs a dual-tower backbone containing a Time Tower and a Frequency Tower. The number of blocks per tower will directly affect the model's performance. Usually, an increment in the parametric number will assign more representative capability to the model, but it will also introduce the risk of overfitting. In this experiment, we try different numbers of encoder blocks to investigate the model's maximum. Specifically, we apply the following combinations of **[Time blocks, Frequency blocks]** during the experiment: {[2,1], [4,2], [6,3], [8,4]}. The lookback window is set to 512, and the horizon is set to 720. Fig. 7 investigates the MSE scores on the Weather, Traffic, and Electricity datasets with varying encoder settings. As the number of blocks increases, the MSE of the predicted results gradually decreases. But when increasing from [6,3] to [8,4], the performance improves remarkably small, implying that the upper limit of the model is being approached. Overly large models impose a heavy computational burden with only slight accuracy gains. After trading off computation efficiency and model performance, we choose a compromise and adopt the [4:2] option in other experiments. This configuration also allows a fair comparison between Hidformer and the baselines, but note that Hidformer can perform better with more blocks.

4.7. Computational efficiency

The SRU++ and linear self-attention mechanism in Hidformer can effectively reduce runtime and memory usage. Here we further compare the computational efficiency of Hidformer and other Transformer-based LTSF models. Trainable parameters (Params) and floating point operations (FLOPs) are popular metrics for measuring networks' computational efficiency. Params represents the number of computed parameters in the model, and FLOPs measures the amount of calculation of a network forward pass. The preferable solution is to have the smallest possible Params and FLOPs with guaranteed prediction accuracy. Fig. 8 reports the computational efficiency of each Transformer-based LTSF model, with the horizontal axis indicating Params and the vertical axis indicating FLOPs. Fig. 9 shows the performance versus Params for each model, with the horizontal axis indicating Params and the vertical axis indicating models' MSE on the Electricity dataset. Our model is in the lower-left corner of the two figures, which means our approach has high computational efficiency. Table 7 records the running time (seconds) of each model for each batch on the Electricity dataset. It can be seen that our model runs fast and maintains the minimum runtime per batch. Note that the main experiment has certified that our prediction results are superior to those of other methods, so this experiment further demonstrates that Hidformer excels in both prediction performance and computational efficiency.

4.8. *Visualization of the prediction*

We visualize the prediction results of each model in Fig. 10 to intuitively show their performance differences. We let each model forecast 200 time steps on Electricity, ETTm1, and Weather datasets. Obviously, Hidformer best fits the true value curve among all the LTSF models. Compared to the predictions of other models, Hidformer's results have lower bias and variance. In the figure, there are some sharp peaks in the true value curves for both Electricity and ETTm1 data. These peaks may represent some important situations, so it's necessary to get as close to them as possible. It can be seen that our model predicts the peaks most accurately, the predicted and true curves still fit well at the peaks, whereas the other models basically just predicted the trend of the changes, or did not reflect the peaks at all. At the same time, the prediction results of Weather data in the figure show that even in situations where the true value curve is smooth, some models may still experience unexpected fluctuations, but Hidformer predicts this smooth transition well. In short, our model is able to provide excellent coverage of the actual time series under a variety of conditions.

5. Conclusion and future work

This article studies the popular long-term time series forecasting problem. We argue the defects that limit the performance of previous Transformer-based models and propose Hidformer. In Specifics, Hidformer is a Transformer-based model with a dual-tower backbone. The Time Tower adopts SRU++ as the token mixer to avoid the permutation-invariant problem and better learn temporal dynamics hidden in the time series through recurrent computations. While the Frequency Tower employs the linear self-attention module as the token mixer to capture global features and variable correlations from the frequency domain. Features learned from the time and frequency domains are integrated to obtain prediction results. We design the segment-and-merge architecture for both towers. The input sequences are segmented into several overlapping subseries (tokens) before being fed into the towers. The natural temporal correlations between contiguous time steps offer these segment-wise tokens semantic meanings. Then the two towers hierarchically organize their blocks and merge the tokens between neighboring blocks. The segment-and-merge design allows Hidformer to learn multi-scale local features from different resolutions. Moreover, the exploration of highly-efficient recurrence and the low-rankness of time series in the frequency domain effectively improves Hidformer's computational efficiency. We verify the effectiveness of Hidformer through experiments on 7 real-world datasets. The MSE and MAE scores are decreased by 9.6 % and 8.6 % under the multivariate forecasting setting and 8.5 % and 5.5 % under the univariate forecasting setting when the acquired prediction length is 720. The superiority over the previous SOTA baselines proves that our proposed model is promising in solving the LTSF task.

Although our Frequency Tower implicitly studies the variable correlations, we aspire to find a more explicit method to model the connection between variables. We believe explicitly representing variable dependencies is an effective way to improve predictive performance. At the same time, the successful application of SRU++ and linear self-attention encourages us to design more suitable token mixers. For example, a token mixer that can attend to more complex patterns. At last, the fact that good results can be achieved on some datasets using only simple linear models indicates that the features of these datasets can be learned easily. It should build some LTSF datasets with complex patterns to enhance the usefulness of the developed models. We are looking forward to constructing such datasets.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Afrin, T., & Yodo, N. (2022). A Long Short-Term Memory-based correlated traffic data prediction framework. *Knowledge-Based Systems, 237*, Article 107755. https://doi.org/10.1016/j.knosys.2021.107755

Barbalau, A., Ionescu, R. T., Georgescu, M.-I., Dueholm, J., Ramachandra, B., Nasrollahi, K., … Shah, M. (2023). SSMTL++: Revisiting self-supervised multi-task learning for video anomaly detection. *Computer Vision and Image Understanding, 229*, Article 103656. https://doi.org/10.1016/j.cviu.2023.103656

Chen, W., Xing, X., Xu, X., Pang, J., & Du, L. (2023). SpeechFormer++: A hierarchical efficient framework for paralinguistic speech processing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 31*, 775–788. https://doi.org/10.1109/TASLP.2023.3235194

Chen, Z., Ding, L., Chu, Z., Qi, Y., Huang, J., & Wang, H. (2023). Monotonic Neural Ordinary Differential Equation: Time-series Forecasting for Cumulative Data. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 4523–4529. 10.1145/3583780.3615487.

Dudek, G. (2020). Multilayer perceptron for short-term load forecasting: From global to local approach. *Neural Computing and Applications, 32*(8), 3695–3707. https://doi.org/10.1007/s00521-019-04130-y

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022). Time-series forecasting of seasonal items sales using machine learning – A comparative analysis. *International Journal of Information Management Data Insights, 2*(1), Article 100058. https://doi.org/10.1016/j.jjimei.2022.100058

Guo, Y., Zhou, D., Li, W., & Cao, J. (2022). Deep multi-scale Gaussian residual networks for contextual-aware translation initiation site recognition. *Expert Systems with Applications, 207*, Article 118004. https://doi.org/10.1016/j.eswa.2022.118004

Huang, Y., Wang, E. H., Liu, Z., Pan, L., Li, H., & Liu, X. (2023). Modeling Task Relationships in Multivariate Soft Sensor With Balanced Mixture-of-Experts. *IEEE Transactions on Industrial Informatics, 19*(5), 6556–6564. https://doi.org/10.1109/TII.2022.3202909

Ilic, I., Görgülü, B., Cevik, M., & Baydoğan, M. G. (2021). Explainable boosted linear regression for time series forecasting. *Pattern Recognition, 120*, Article 108144. https://doi.org/10.1016/j.patcog.2021.108144

Karmy, J. P., & Maldonado, S. (2019). Hierarchical time series forecasting via Support Vector Regression in the European Travel Retail Industry. *Expert Systems with Applications, 137*, 59–73. https://doi.org/10.1016/j.eswa.2019.06.060

Kaytez, F. (2020). A hybrid approach based on autoregressive integrated moving average and least-square support vector machine for long-term forecasting of net electricity consumption. *Energy, 197*, Article 117200. https://doi.org/10.1016/j.energy.2020.117200

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv: 1412.6980). arXiv. 10.48550/arXiv.1412.6980.

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). *Reformer: The Efficient Transformer* (arXiv: 2001.04451). arXiv. 10.48550/arXiv.2001.04451.

Lei, T. (2021). *When Attention Meets Fast Recurrence: Training Language Models with Reduced Compute* (arXiv:2102.12459; Version 3). arXiv. http://arxiv.org/abs/2102.12459.

Lei, T., Zhang, Y., Wang, S. I., Dai, H., & Artzi, Y. (2018). *Simple Recurrent Units for Highly Parallelizable Recurrence* (arXiv:1709.02755). arXiv. 10.48550/arXiv.1709.02755.

Li, H., Wang, W., Liu, Z., Niu, Y., Wang, H., Zhao, S., … Liu, X. (2022). A novel locality-sensitive hashing relational graph matching network for semantic textual similarity measurement. *Expert Systems with Applications, 207*, Article 117832. https://doi.org/10.1016/j.eswa.2022.117832

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., & Yan, X. (2019). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Advances in Neural Information Processing Systems, 32*. https://proceedings.neurips.cc/paper/2019/hash/6775a0635c302542da2c32aa19d86be0-Abstract.html.

Liu, F., Tao, Q., Yang, D., & Sidorov, D. (2022). Bidirectional Gated Recurrent Unit-Based Lower Upper Bound Estimation Method for Wind Power Interval Prediction. *IEEE Transactions on Artificial Intelligence, 3*(3), 461–469. https://doi.org/10.1109/TAI.2021.3123928

Liu, Z., Li, H., Wang, H., Liao, Y., Liu, X., & Wu, G. (2023). A novel pipelined end-to-end relation extraction framework with entity mentions and contextual semantic representation. *Expert Systems with Applications, 228*, Article 120435. https://doi.org/10.1016/j.eswa.2023.120435

Merino, A., Garcia-Alvarez, D., Sainz-Palmero, G. I., Acebes, L. F., & Fuente, M. J. (2020). Knowledge based recursive non-linear partial least squares (RNPLS). *ISA Transactions, 100*, 481–494. https://doi.org/10.1016/j.isatra.2020.01.006

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers* (arXiv:2211.14730). arXiv. http://arxiv.org/abs/2211.14730.

Ning, Y., Kazemi, H., & Tahmasebi, P. (2022). A comparative machine learning study for time series oil production forecasting: ARIMA, LSTM, and Prophet. *Computers & Geosciences, 164*, Article 105126. https://doi.org/10.1016/j.cageo.2022.105126

Park, J., Kim, H., Choi, Y., Lee, W., & Lee, J. (2023). Fast sharpness-aware training for periodic time series classification and forecasting. *Applied Soft Computing, 144*, Article 110467. https://doi.org/10.1016/j.asoc.2023.110467

Reza, S., Ferreira, M. C., Machado, J. J. M., & Tavares, J. M. R. S. (2022). A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. *Expert Systems with Applications, 202*, Article 117275. https://doi.org/10.1016/j.eswa.2022.117275

Rodrawangpai, B., & Daungjaiboon, W. (2022). Improving text classification with transformers and layer normalization. *Machine Learning with Applications, 10*, Article 100403. https://doi.org/10.1016/j.mlwa.2022.100403

Shen, L., Wei, Y., & Wang, Y. (2023). GBT: Two-stage transformer framework for non-stationary time series forecasting. *Neural Networks, 165*, 953–970. https://doi.org/10.1016/j.neunet.2023.06.044

Teng, M., Li, S., Xing, J., Song, G., Yang, J., Dong, J., … Qin, Y. (2022). 24-Hour prediction of PM2.5 concentrations by combining empirical mode decomposition and bidirectional long short-term memory neural network. *Science of The Total Environment, 821*, Article 153276. https://doi.org/10.1016/j.scitotenv.2022.153276

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems, 30*. https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Venkatachalam, K., Trojovský, P., Pamucar, D., Bacanin, N., & Simic, V. (2023). DWFH: An improved data-driven deep weather forecasting hybrid model using Transductive Long Short Term Memory (T-LSTM). *Expert Systems with Applications, 213*, Article 119270. https://doi.org/10.1016/j.eswa.2022.119270

Wang, C., Chen, Y., Zhang, S., & Zhang, Q. (2022). Stock market index prediction using deep Transformer model. *Expert Systems with Applications, 208*, Article 118128. https://doi.org/10.1016/j.eswa.2022.118128

Wang, H., Wang, Z., Niu, Y., Liu, Z., Li, H., Liao, Y., … Liu, X. (2023). An Accurate and Interpretable Framework for Trustworthy Process Monitoring. *IEEE Transactions on Artificial Intelligence, 1–12*. https://doi.org/10.1109/TAI.2023.3319606

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). *Linformer: Self-Attention with Linear Complexity* (arXiv:2006.04768). arXiv. 10.48550/arXiv.2006.04768.

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems, 34*, 22419–22430. https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Wu, T., Wang, G., Zhao, J., Liu, Z., Qi, G., Li, Y.-F., & Haffari, G. (2022). *Towards Relation Extraction From Speech* (arXiv:2210.08759). arXiv. http://arxiv.org/abs/2210.08759.

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., & Yan, S. (2022). *MetaFormer Is Actually What You Need for Vision* (arXiv:2111.11418). arXiv. 10.48550/arXiv.2111.11418.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2022). *Are Transformers Effective for Time Series Forecasting?* (arXiv:2205.13504; Version 3). arXiv. http://arxiv.org/abs/2205.13504.

Zheng, W., Cheng, J., Wu, X., Sun, R., Wang, X., & Sun, X. (2022). Domain knowledge-based security bug reports prediction. *Knowledge-Based Systems, 241*, Article 108293. https://doi.org/10.1016/j.knosys.2022.108293

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence, 35*(12), Article 12. 10.1609/aaai.v35i12.17325.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. *Proceedings of the 39th International Conference on Machine Learning*, 27268–27286. https://proceedings.mlr.press/v162/zhou22g.html.