# PWDformer: deformable transformer for long-term series forecasting.

WANG, Z., RAN, H., REN, J. and SUN, M.

2024

# PWDformer: Deformable transformer for long-term series forecasting

Zheng Wang [a,b,c,*], Haowei Ran [a,b,c], Jinchang Ren [d], Meijun Sun [a,b,c]

[a] Tianjin University, 135 Yaguan Road, Jinnan District, 300192, Tianjin, China
[b] Tianjin Key Laboratory of Machine Learning, Tianjin University, 300192, Tianjin, China
[c] Engineering Research Center of City intelligence and Digital Governance, Ministry of Education of the People's Republic of China, 300192, Tianjin, China
[d] National Subsea Centre & School of Computing, Robert Gordon University, 3 International Avenue, Aberdeen, AB21 0BH, UK

## ABSTRACT

*Long-term forecasting* is of paramount importance in numerous scenarios, including predicting future energy, water, and food consumption. For instance, extreme weather events and natural disasters can profoundly impact infrastructure operations and pose severe safety concerns. Traditional CNN-based models often struggle to capture long-distance dependencies effectively. In contrast, Transformers-based models have shown significant promise in *long-term forecasting*.

This paper investigates the *long-term forecasting* problem and identifies a common limitation in existing Transformer-based models: they tend to reduce computational complexity at the expense of time information aggregation capability. Moreover, the order of time series plays a crucial role in accurate predictions, but current Transformer-based models lack sensitivity to time series order, rendering them unreasonable.

To address these issues, we propose a novel Deformable-Local (DL) aggregation mechanism. This mechanism enhances the model's ability to aggregate time information and allows the model to adaptively adjust the size of the time aggregation window. Consequently, the model can discern more complex time patterns, leading to more accurate predictions. Additionally, our model incorporates a Frequency Selection module to reinforce effective features and reduce noise. Furthermore, we introduce Position Weights to mitigate the order-insensitivity problem present in existing methods.

In extensive evaluations of *long-term forecasting* tasks, we conducted benchmark tests on six datasets covering various practical applications, including energy, traffic, economics, weather, and disease. Our method achieved state-of-the-art (SOTA) results, demonstrating significant improvements. For instance, on the ETT dataset, our model achieved an average MSE improvement of approximately **19%** and an average MAE improvement of around **27%**. Remarkably, for predicted lengths of 96 and 192, we achieved outstanding MSE and MAE improvements of **32.1%** and **30.9%**, respectively.

*Keywords:* Long-term forecasting; Time series forecasting; Deep learning; Transformer

## 1. Introduction

In real-life scenarios, time series prediction plays a vital role in numerous fields, such as demand forecasting in supply chain management [1], stock market forecasting [2], energy consumption forecasting [3], and weather and climate forecasting [4]. With the development of deep learning technology, deep neural networks [5] have been introduced into the field of time series forecasting. This data-driven approach reduces the reliance on artificial features [6]. Particularly, models based on Convolutional Neural Network (CNN) [7] have achieved promising results. Recurrent Neural Networks (RNNs) [8] were the pioneering models in time series prediction, aiming to address the difficulty of CNN in capturing long-distance dependencies. Long Short-Term Memory (LSTM) [9] networks effectively manage memory and forgetting within neural networks through gating mechanisms. Additionally, the encoder and decoder architecture in DeepAR [10] is based on RNNs, leading to significant breakthroughs in performance.

Some examples combine CNN with traditional methods. For instance, Rangapuram et al. [11] used Deep State Space Models (DSSM) to achieve more accurate predictions, and Wang et al. [12] proposed Deep Factors.

However, in practical applications, models are required to predict future values for an extended period. The increasing length of time series data imposes challenges on the predictive capability of models. Nonetheless, most existing methods are designed for short-term prediction settings. Early traditional methods can only handle short prediction lengths, including State Space Models (SSMs) [13], Matrix Factorization methods [14], exponential smoothing [15], Auto-Regressive Integrated Moving Average (ARIMA) [16], Seasonal Autoregressive Integrated Moving Average (SARIMA) [17], and Gaussian process (GP) [18]. Even recent research [19] incorporates methods that combine EM with DSSM. In the early stages of *long-term forecasting*, researchers mainly focused on addressing super-long inputs (e.g., truncation, summarization, sampling, etc.), which resulted in substantial information loss. Other approaches, such as Truncated BPTT [20], only use the last time step to estimate the gradient. Auxiliary Losses [21] introduce auxiliary gradients for weight updating, and regularization attempts include Bootstrapping Regularizercitecao2020better. For graph neural networks (GNN)-based methods, MTGNN [22] proposes a general graph neural network framework, while SDGL [23] introduces a dynamic graph learning network for modeling unknown patterns.

However, these models usually set the prediction length as 48 data points or less. The *long-term forecasting* problem requires models with a stronger ability to capture long-distance dependencies compared to traditional CNN models.

Inspired by the tremendous success of Transformer [24] in the field of vision, many researchers have recently attempted to apply the transformer model architecture to the *long-term forecasting* problem. Transformers have a significant advantage in modeling long-term dependencies for sequential data, thanks to their self-attention mechanism.

For example, in Informer [25], the *long-term forecasting* problem is proposed to address real-world applications that require predictions for long time series, such as power consumption planning. Reformer [26] introduces the Local-sensitive hashing (LSH) attention, while Logtrans [27] proposes the LogSparse attention to select time steps following exponentially increasing intervals. ConvTrans [28] introduces an improved attention mechanism to reduce the computational cost of self-attention. Autoformer [29] leverages correlation to select the time nodes that should be prioritized and achieves state-of-the-art (SOTA) results. Moreover, many researchers try to combine expertise from other fields and design models based on transformers to address time series prediction challenges. Spectral filtering technology is used in [30], while [31] views the correlation between sequences from the perspective of the spectral domain.

However, through our research, we identified two fatal flaws in most existing Transformer-based models, preventing further improvements in *long-term forecasting*.

The first bottleneck arises from the high computing cost of self-attention [24]. To address this issue, many existing methods attempt to reduce computational complexity, but often at the expense of time information aggregation capability. For instance, Informer [25] highlights that the canonical self-attention (dot product of QK) forms a long-tailed distribution in most cases. As a solution, Informer [25] introduces query sparsity measurement to screen the $u$ most important queries using Kullback–Leibler divergences. However, this approach leads to time-point-wise information aggregation, resulting in significant information loss when selecting the top-k queries. Similarly, Autoformer [29] uses autocorrelation coefficients to calculate the correlation between QK under different time delays. Despite the effort to reduce complexity, the information aggregation remains time point-wise. Reformer [26] designs self-attention based on local-sensitive hash, and Longformer [32] employs a heuristic self-attention mechanism, both facing the issue of time point-wise aggregation and loss of information during top-k operations.

The second bottleneck stems from the existing self-attention based methods' inability to recognize the importance of location order in time series prediction tasks, particularly in *long-term forecasting*. The insensitivity to position hampers these models' ability to capture complex time patterns and extract crucial time features, making it a significant limitation.

To address these two challenges, we propose a mechanism called Position-Weight-Deformable Correlation (PWD correlation). PWD correlation consists of two components: Deformable-Local aggregation (DL aggregation) and Position Weights.

DL aggregation aims to overcome the issue of insufficient time information aggregation capability by introducing a deformable time aggregation window. Unlike conventional approaches, our model can dynamically adjust the size of the time aggregation window based on local information. This adaptability empowers our model to capture and understand more intricate time patterns, enabling more accurate predictions in complex *long-term forecasting* tasks.

Position Weights are designed to tackle the order-insensitivity problem. By allowing the model to learn the weight of each position in the input time series, we can adjust the final attention. This breakthrough frees our model from the constraints of position insensitivity and enables it to learn time information features closely related to the position order.

In summary, the main contribution of this paper are highlighted below:

- We propose a novel Deformable-Local aggregation (DL aggregation) mechanism that enables the model to adaptively adjust the size of the time aggregation window based on the information around the current time point. This mechanism addresses the problem of insufficient information aggregation capability in existing Transformer-based models.
- To tackle the order-insensitivity problem in Transformer-based models, we introduce Position Weights, allowing the model to be weighted differently at various locations of the input time series, even in different multi-channel scales. This breakthrough enables the discovery of more complex time series features associated with location order.
- We design the Frequency-Selection module to reduce noise interference in the encoder stage. This module operates from the perspective of the frequency domain and effectively reduces the model's attention to noise.
- We propose the Position Weight Deformable former (PWDformer) specifically for *long-term forecasting*, and extensively evaluate it on six widely used datasets covering various practical applications such as energy, traffic, economics, weather, and disease. In most cases, PWDformer outperforms state-of-the-art (SOTA) models.

## 2. Preliminaries

### 2.1. Problem definition

*Long-term forecasting* is a more difficult time series prediction problem. We define the input time series $X$ as having a length of $I$, and the time series to be predicted $Y$ with a length of $O$. As time series can be either univariate or multivariate, we define the dimension of the variable as $N$. Hence, we have $X \in \mathbb{R}^{I\times N}$ and $Y \in \mathbb{R}^{O\times N}$.

One fundamental assumption in this context is the conditional independence between different time instants.

For the input time series $\left\{ x^n_{1:t_p-1} \right\}^N_{n=1}$, where $x^n_{1:t_p-1} = (x^n_1, x^n_2, \cdots, x^n_{t_p-1})$, $t_p \in \mathbb{N}$ denotes the forecast h zon, and $\tau \in \mathbb{N}$ represents the forecast horizon, and $\tau \in \mathbb{N}$ represents the forecast length. Consequently, the total sequence length becomes $l_{total} = t_p + \tau$. The goal of this problem is to model the conditional probability distribution of the input time series and predict the future time series with the specified time step $\tau$. In *long-term forecasting*, the $\tau$ is greater than in normal time series forecasting. Our objective can be expressed as follows:

$$Z^i_{t_p:t_p+\tau} = \Phi(X^n_{1:t_p-1}, \theta)$$

$$\theta = argmin(Y^i_{t_p+\tau}, Z^i_{t_p+\tau})$$

Here, $i$ represents the dimensions to be predicted, and $\Phi$ represents our model. The $\theta$ denotes the learnable parameters.

## 3. Methodology

In Section 3.1, we will start by presenting the comprehensive architecture of PWDformer, as depicted in Fig. 1. Subsequently, Section 3.2 will introduce the structure of PWD correlation, which encompasses DL aggregation and Position Weights. The detailed specifics of DL aggregation can be found in Section 3.2.1, while Section 3.2.2 will elaborate on the working principles of Position Weights. Lastly, in Section 3.3, we will introduce the Frequency-Selection module.

### 3.1. Pwdformer structure

As depicted in Fig. 1, PWDformer comprises two major components: Encoder and Decoder. The input to the Encoder is denoted as $X_{en} \in \mathbb{R}^{L \times D}$, where $L$ represents the length of inputs, and $D$ represents the number of variables in the time series. For the Decoder input, denoted as $X_{de} \in \mathbb{R}^{L \times D}$, we use the second half of the original inputs $X \in \mathbb{R}^{L \times D}$ filled with zeros to match the length of the Encoder input. Subsequently, the series decompose module decomposes $X_{de}$ into seasonal items $X_s \in \mathbb{R}^{L \times D}$ and residual items $X_r \in \mathbb{R}^{L \times D}$.

$$X_{de} = Padding(X_{\frac{I}{2}:I}, 0)$$

The process of the series decompose module can be described as follows:

$$X_{mean} = Mean(X_{\frac{I}{2}:I}),\ X_{mean} \in \mathbb{R}^{\frac{L}{2}/2 \times D}$$

$$X_r = AvgPool(Padding(X_{de}, 0))$$

$$X_s = Padding(X_{de} - X_r, X_{mean})$$

Here, $x_r$ represents the residual item, and $x_s$ denotes the seasonal item. The idea of decomposing the series is derived from some early time series modeling methods [33,34], which believe that the seasonal term contains a clear change law and the residual term is a combination of trend and inherent noise, reflecting the general trend of the series. Some recent works [29,35] have proved that this decomposition based prediction method can effectively improve the performance of the model. The operation $Padding(x, \alpha)$ denotes filling the series $x$ with a constant $\alpha$. Meanwhile, $AvgPool$ refers to the moving average operation. The Data Embedding step utilizes Temporal Embedding [29] to transform the input into a high-dimensional space with the assistance of linear layers.

### 3.2. Position weight deformable (PWD) correlation

The existing Transformer-based models suffer from two main draw-backs:

Firstly, when dealing with complex patterns in *long-term forecasting*, sacrificing the ability of information aggregation is not an ideal solution. Time information aggregation refers to combining information on historical time points within a certain time window size and extracting potential features, while traditional self-attention can only aggregate features on a single time point, so the aggregated feature information is not rich and sufficient, which is difficult to deal with long-term prediction. Our aim is to design a mechanism that enhances the information aggregation capability of the self-attention mechanism without significantly increasing computational complexity.

Secondly, the self-attention operation is order-insensitive, meaning that changing the order of input information produces the same attention result. This limitation hinders the self-attention mechanism from fully leveraging its advantages in time series prediction tasks, especially in *long-term forecasting*. We believe that the model should be able to assign varying degrees of attention to inputs based on the time distance between them and the predicted point. For instance, in a stock prediction task, the model should focus more on recent data, while for stable and regular datasets, distant time points might be more critical than recent ones. To measure the information aggregation ability of Transformer-based models, we calculate the amount of information that can be aggregated at each time point to be predicted. As shown in Table 1, a higher number indicates a stronger information aggregation ability. For example, in Autoformer, each point to be predicted is aggregated by $n$ input time points.

**Table 1**
Comparison of information aggregation ability of different models.

| Method | Time aggregation capability |
|---|---|
| Ours | $\Delta k \cdot c \cdot \log_L$ |
| Autoformer | $c \cdot \log_L$ |
| Informer | $c \cdot \log_L$ |
| Logtrans | $c \cdot \log_L$ |
| Reformer | $c \cdot \log_L$ |

* $\Delta k$ represents the deformable scale, $c$ is a constant, $c \cdot \log L$ represents the number of Topk QK pairs in the attention module. For example, in Autoformer [29] the $c \cdot \log L$ represents Topk autocorrelations.

In response to these challenges, we propose the PWD correlation to tackle the above problems. As shown in Fig. 2, PWD correlation comprises two main components: DL aggregation and Position Weights. First, PWD calculates the correlation between Queries ($Q$) and Keys ($K$) using Wiener–Khinchin theorem [36]. We select the Top-$\sigma$ pairs of Query-Keys ($QK$) with the highest correlation, where $\sigma$ is a manually set constant. For these Top-$\sigma$ $QK$ pairs, DL aggregation is applied to process the original Values ($V$) and generate new $V'$ with enriched local information and complex time patterns. Then, the model's attention is adjusted to the new $V'$ through Position Weights. Finally, the outputs of PWD correlation are obtained.

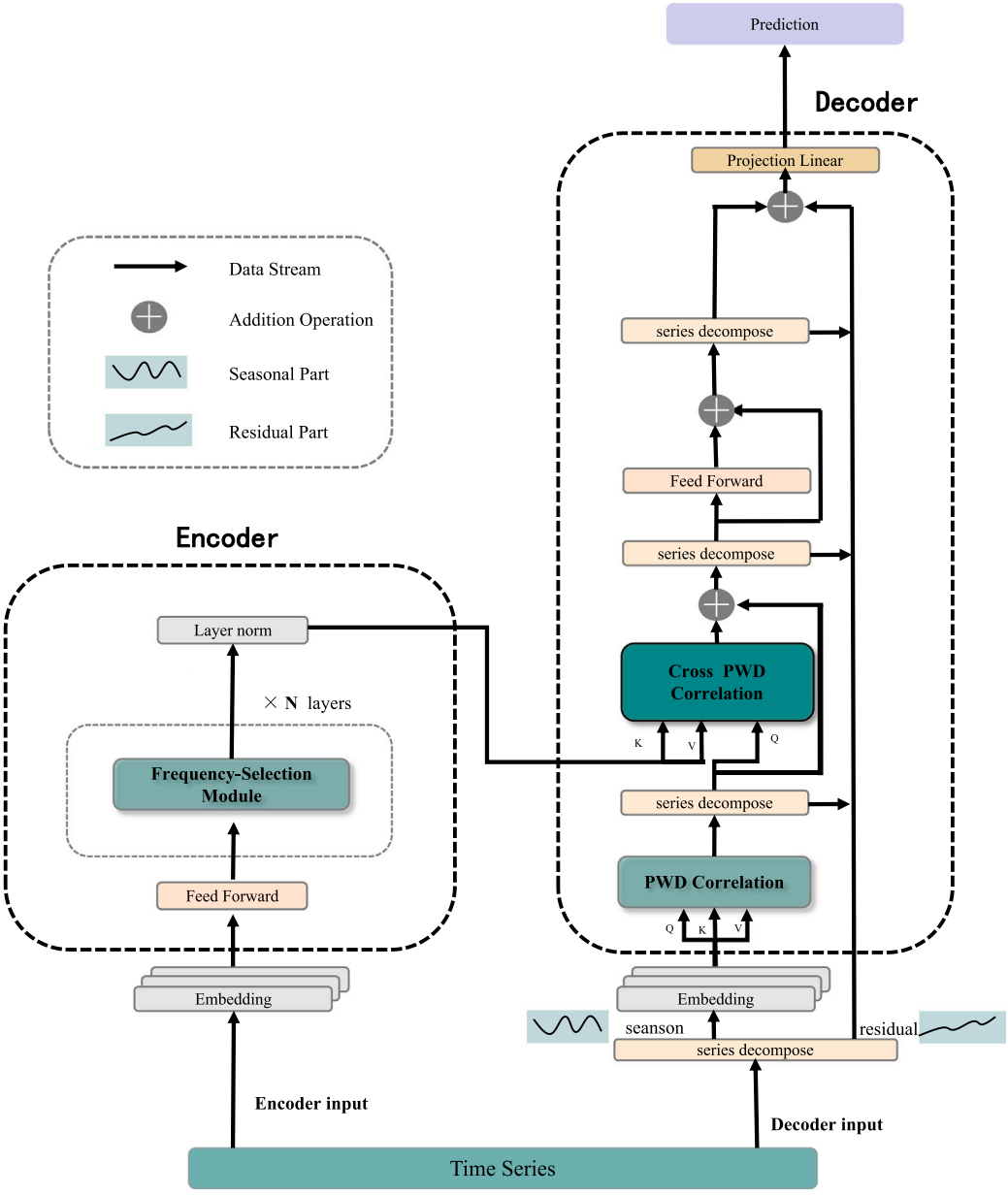



**Fig. 1.** The overall structure of PWDformer.

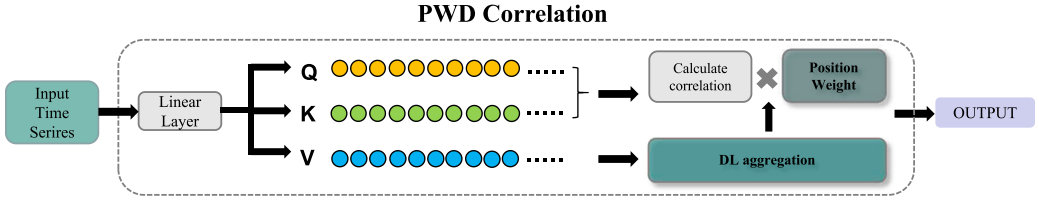



**Fig. 2.** The PWD correlation involves generating three matrices of the same shape from the input time series through a linear layer: $Q$, $K$, and $V$.

*3.2.1. Deformable local(DL) aggregation mechanism*

To enable the model to capture more complex patterns in time series data, we emphasize that in the self-attention mechanism [24], it should aggregate temporal information at the level of time periods rather than fragmented and incomplete information at individual time points. However, different application scenarios and time series data may require varying sizes of aggregated time periods. Even for different time points within the same time series dataset, the required time period for prediction may differ.

Therefore, we propose the DL aggregation, which allows the model to dynamically adjust the size of the aggregated time period information based on the characteristics of the time points to be predicted. This adaptive approach facilitates better capturing of relevant temporal patterns.

The vanilla self-attention [24] mechanism comprises Querys, Keys, and Values (abbreviated as $Q$, $K$, and $V$). To facilitate further discussion on self-attention, we establish the notation that $Q$, $K$, and $V$ represent the Querys, Keys, and Values, respectively. Based on this notation, we can formally define the vanilla self-attention as follows:

$$A(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})V$$

Additionally, we introduce $v_i$ as the representation of the $i$th row of $V$, where $i$ corresponds to the $i$th time point. Drawing inspiration from the Deformable Convolution [37,38] utilized in *object detection*, we propose that each $v_i$ in the *value* matrix should also encompass temporal information from its neighboring time points within a certain time span. Leveraging the characteristics of time series data, we devise the DL aggregation, which is elaborated step by step below.

To elucidate the process of DL aggregation, let us consider a simple case of time aggregation with an immutable-size window. As shown in Fig. 3, we assume the aggregated window size to be $n$ (where each $v_i$ can aggregate information from $n-1$ points around it, and $n$ is usually an odd number).

We define Values as $V$ in the self-attention, the length of $V$ is $L_V$. For each point $v_i$, $i = 1, 2, 3, \dots, L_V$, We gather information around $v_i$, and the aggregated information is used as the new value of $v_i$. which can be expressed as:

$$\hat{v_i} = \sum_{j=i-(\frac{n-1}{2})}^{i+(\frac{n-1}{2})} w_j * v_j$$

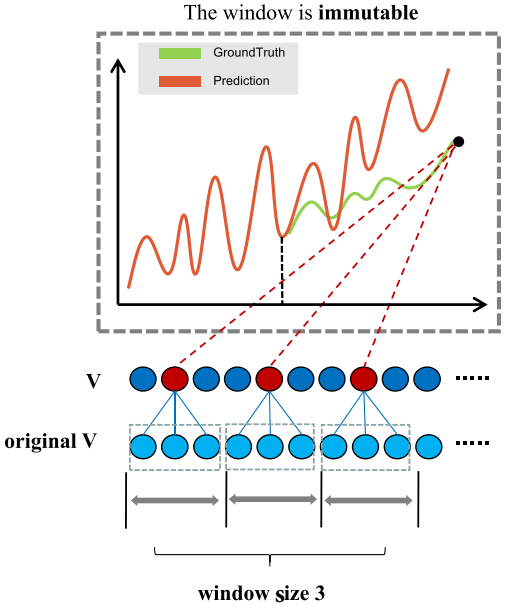



**Fig. 3.** Time aggregation with size-immutable window

where $w_i$ is the weight that can be learned during the process.

However, when dealing with time series data exhibiting complex patterns, the model should possess the capability to dynamically adjust the size of the time aggregation window based on the input data.

Furthermore, we propose the deformable local aggregation mecha-nism, as illustrated in Fig. 4.

Our general idea can be expressed as follows: for each $v_i$ in every time aggregation window, our mechanism learns an offset value $\Delta s_i$. Given that the same $v_i$ will learn $n$ offset values (due to the step size of the time aggregation window being 1, resulting in overlap between windows), we further represent $\Delta s_i$ as $\Delta s_{ij}$. Here, $\Delta s_{ij}$ indicates the $j$th offset value of $v_i$. The new value of time point $v_i$ can be expressed as follows:

$$v_i = v_{\hat{i}}$$

$$\hat{i} = i + \sum_{j=1}^{n} p_{ij} * \Delta s_{ij}$$

Where $p_{ij}$ represents the learnable weight corresponding to each offset value $\Delta s_{ij}$.

However, the final coordinate $\hat{i}$ is often a decimal value. To obtain the ultimate value of $v_{\hat{i}}$, we will utilize unilinear interpolation. The rationale behind choosing unilinear interpolation is the continuity of time series data and the feasibility of backpropagation. Thus, the final value of $v_{\hat{i}}$ can be determined by the following formula:

$$v_{\hat{i}} = G(ceil(\hat{i}), i) * v_{ceil(\hat{i})} + G(floor(\hat{i}), i) * v_{floor(\hat{i})}$$

$G$ represents the linear interpolation kernel function. $ceil$ denotes the operation of rounding up to the nearest integer, and $floor$ denotes the operation of rounding down to the nearest integer.

Each $v_i$ in the original $V$ undergoes processing using the aforementioned offset mechanism, resulting in the transformed $V$, denoted as $\mathcal{V}$.

Thus, the final $\mathcal{V}$ is represented as $\mathcal{V}\ |\hat{v}_i, i = 0, 1, 2, 3..., L_v$. The core workflow of the DL aggregation can be observed in Fig. 5. In this context, we employ a fixed-size time aggregation window to slide (with a stride of 1) on $\mathcal{V}$, thereby aggregating information for each point to implement our mechanism.

This step can be expressed as follows:

$$\hat{v}_i \in \hat{V}$$

$$\hat{v}_i = \sum_{j=i-(\frac{n-1}{2})}^{i+(\frac{n-1}{2})} w_j * \hat{v}_j$$

With the aid of information entropy theory [39], we posit that if the extracted feature exhibits more diverse time patterns, its information entropy will be higher.

We conducted a statistical analysis on the characteristics ($\hat{v}_j$) to which each predicted point was aggregated. Our findings reveal that the information entropy [39] of the outputs aggregated by DL aggregation is consistently higher than the feature distribution that was not processed by DL aggregation. The feature distribution obtained through DL aggregation was subjected to a statistical comparison with the final feature distribution used by autocorrelation in Autoformer. Some of the visualization results are displayed in Fig. 6. Additionally, we calculated the average information entropy of each head with an input length of 96 and an output length of 96 on the four datasets of Autoformer and PWDformer under the multivariable prediction task. Further details and comprehensive results are provided in Table 2.

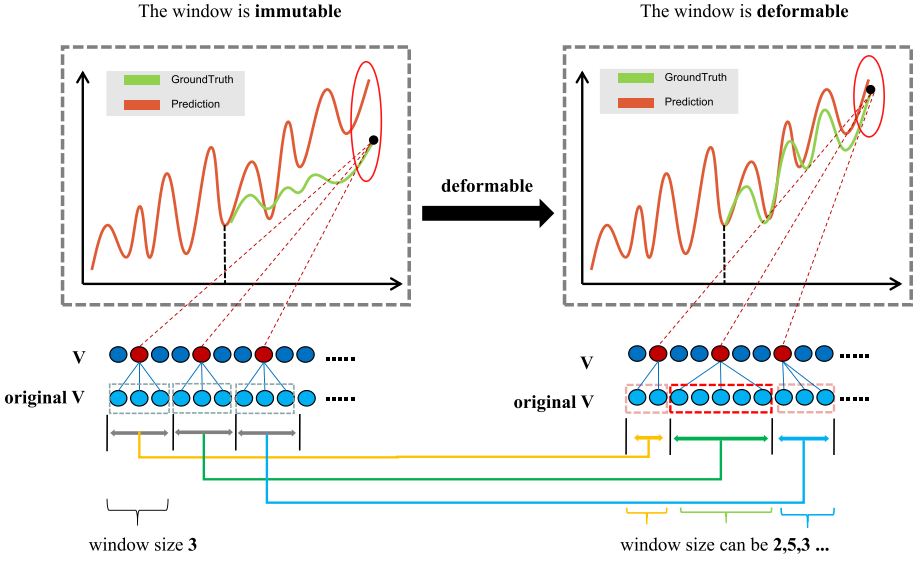



**Fig. 4.** Deformable time aggregation window.

**Table 2**
The statistical comparison results of the feature information entropy extracted between our model and Autoformer. The input length is 96 and the prediction length is 96. Head 1 represents the first head of the Multi-headed attention. The values in the table represent the average information entropy within each head. The larger the value of information entropy, the more complex information pattern can be extracted. The bolded value represents the larger average information entropy on the same dataset.

| Dataset | Method | Head 1 | Head 2 | Head 3 | Head 4 | Head 5 | Head 6 | Head 7 | Head 8 |
|---|---|---|---|---|---|---|---|---|---|
| ETTh1 | Autoformer | 1.2164 | 1.2187 | 1.2181 | 1.2146 | 1.2144 | 1.2178 | 1.2191 | 1.2181 |
| | Ours | **1.3957** | **1.3923** | **1.3835** | **1.4000** | **1.4073** | **1.3990** | **1.3991** | **1.3880** |
| ETTh2 | Autoformer | 1.2620 | 1.2609 | 1.2584 | 1.2581 | 1.2578 | 1.2606 | 1.2569 | 1.2521 |
| | Ours | **1.5118** | **1.4939** | **1.4830** | **1.4781** | **1.4907** | **1.4934** | **1.5023** | **1.5033** |
| ETTm1 | Autoformer | 1.8391 | 1.8445 | 1.8356 | 1.8448 | 1.8417 | 1.8421 | 1.8468 | 1.8357 |
| | Ours | **1.9008** | **1.8950** | **1.9060** | **1.8972** | **1.9009** | **1.8930** | **1.8725** | **1.8954** |
| ETTm2 | Autoformer | 0.7558 | 0.7597 | 0.8193 | 0.7593 | 0.8056 | 0.7658 | 0.7826 | 0.7703 |
| | Ours | **0.9173** | **0.9192** | **0.9312** | **0.9406** | **0.9203** | **0.9299** | **0.9295** | **0.9336** |
| Exchange | Autoformer | 1.3780 | 1.3688 | 1.3773 | 1.3711 | 1.3864 | 1.3730 | 1.3752 | 1.3408 |
| | Ours | **1.7527** | **1.7507** | **1.7454** | **1.7490** | **1.7504** | **1.7469** | **1.7511** | **1.7516** |

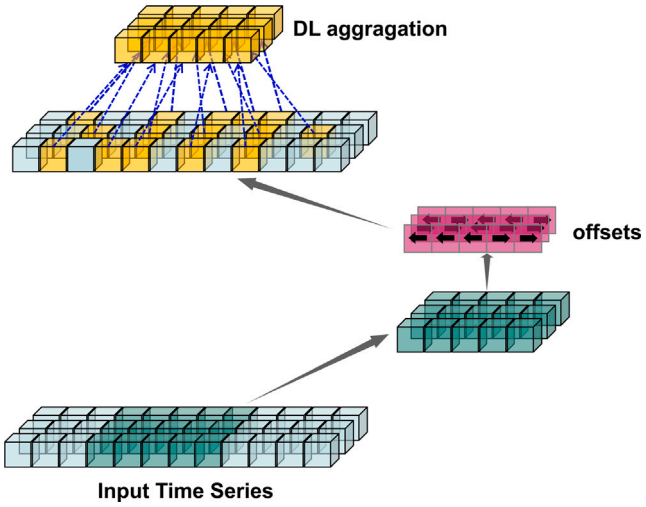
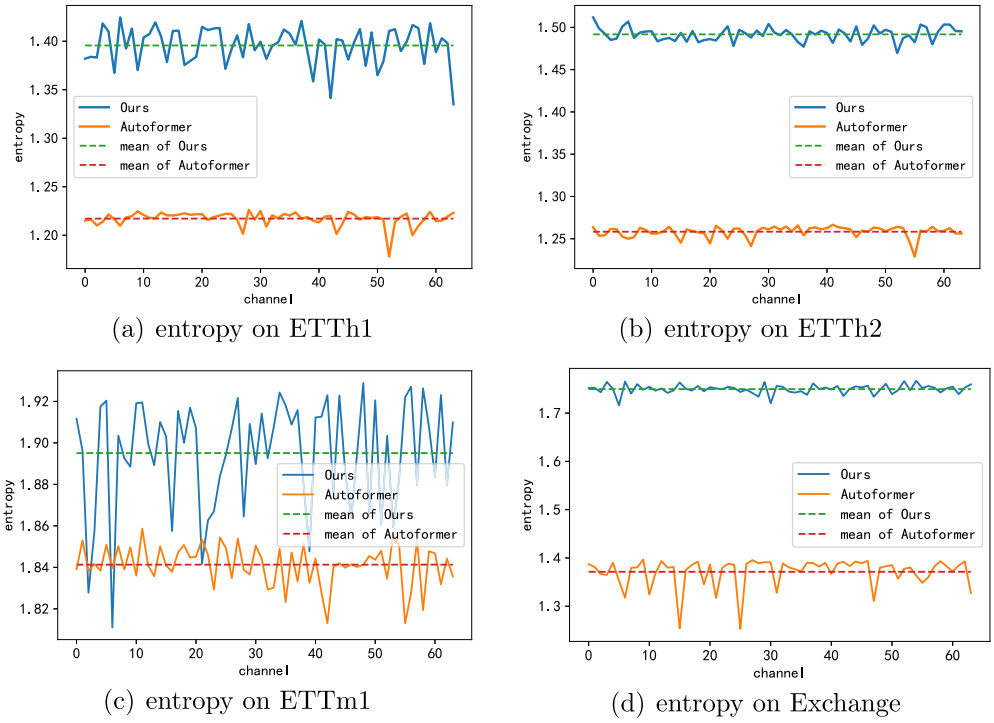



**Fig. 5.** The core working flow of deformable local aggregation.




(a) entropy on ETTh1 (b) entropy on ETTh2

(c) entropy on ETTm1 (d) entropy on Exchange

**Fig. 6.** The information entropy of the extracted feature was calculated and analyzed. For visual display purposes, we specifically selected the information entropy of the first 8 channels in each head. In the figure, the *y*-axis represents the information entropy, while the *x*-axis represents the channel number. Notably, all the input lengths in the figure are set to 96, and the prediction length is also 96. Moreover, it is essential to mention that the predictions made are for a multivariable scenario.

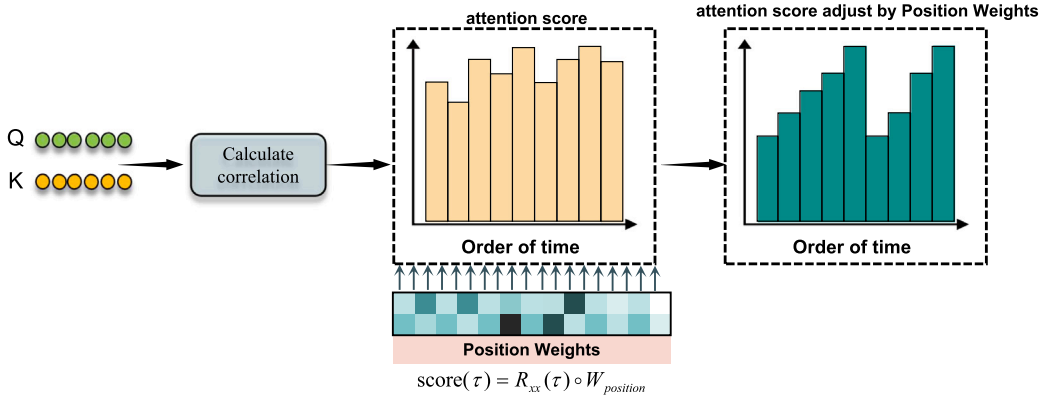



**Fig. 7.** The working flow of Position Weights.

### 3.2.2. Position weights

As most of the existing *long-term forecasting* models based on the self-attention mechanism overlook the significance of temporal order in time series prediction tasks, we propose the concept of Position Weights. This novel approach takes into account the sequential nature of time series data and can be mathematically expressed as follows:

$$Corr = R(Q, K)$$

$$score(\tau) = Softmax(\mathcal{W}_{h,c} \circ Corr_{h,c})$$

Where $R(Q, K)$ represents the correlation coefficient between $Q$ and $K$, and $\tau$ denotes the lag or delay of the autocorrelation coefficient. To delve deeper into complex time patterns, we propose that the model assigns different weights to various positions within each channel. This allows the model to discern the varying importance of different time points in capturing temporal dependencies and patterns. $\mathcal{W}$ represents a set of learnable parameters with the shape of $h \times c$, where $h$ denotes the number of heads in the multi-head self-attention mechanism, and $c$ represents the number of channels within each head. These learnable parameters allow the model to adaptively assign different weights to different positions and channels, facilitating the extraction of relevant temporal patterns and correlations for the task at hand.

In terms of computing the autocorrelation coefficient $R(Q, K)$, we employ the Wiener–Khinchin theorem [36]. This theorem allows us to efficiently calculate the autocorrelation coefficient between sequences. By utilizing this method, we can effectively analyze the relationships and dependencies within the sequences without incurring excessive computational overhead (see Fig. 7). We define the autocorrelation coefficient between sequences as:

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

Based on the Wiener–Khinchin theorem, we can efficiently obtain a function for calculating the autocorrelation coefficient of a sequence:

$$S_{\mathcal{X}\mathcal{X}}(f) = \mathcal{F}\left(\mathcal{X}_t\right) \mathcal{F}^*\left(\mathcal{X}_t\right) = \int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi t f}\, \mathrm{d}t \int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi t f}\, \mathrm{d}t$$

$$\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau) = \mathcal{F}^{-1}\left(S_{\mathcal{X}\mathcal{X}}(f)\right) = \int_{-\infty}^{\infty} S_{\mathcal{X}\mathcal{X}}(f) e^{i2\pi f \tau} \mathrm{d}f$$

$L$ represents the length of the sequence, and $\tau \in 1, \ldots, L$ denotes the delay or lag of the sequence autocorrelation coefficient. $\mathcal{F}$ represents the fast Fourier transform. In conclusion, the expression for Position Weights can be represented as follows:

$$\tau \in \{1, \ldots, L\}$$

$$score(\tau) = Softmax(\mathcal{W}_{h,c} \circ Corr_{h,c})$$

$$\hat{\tau_1}, \ldots, \hat{\tau_k} = argTopK(score(\tau))$$

$$PW(Q, K, V) = \sum_{i=1}^{factor} Gather(\mathcal{V}, \hat{\tau}) \circ Softmax(score(\hat{\tau}))$$

Where:

- $Gather(\mathcal{V}, \hat{\tau})$ refers to the value of $\mathcal{V}$ after applying the delay $\hat{\tau}$.
- $W \in \mathbb{R}^{H\times C\times L}$ represents the set of learnable parameters.
- $score(\tau) \in \mathbb{R}^{H\times C\times L}$ represents the attention scores for different delays.
- $\circ$ denotes the dot product operation.

With the application of Position Weights, the attention scores for different positions within each channel are adjusted, allowing the model to effectively capture complex time patterns and dependencies in the time series data (see Fig. 8).

### 3.3. Frequency-selection module

We believe that the task of time series prediction may not necessarily require a complex coding process for the encoder. Instead, we posit that the decoder's ability to extract complex temporal patterns plays a crucial role. The encoder's primary responsibility should be focused on tasks such as noise reduction [40] or filtering. With these ideas in mind, we have designed the Frequency Selection module for the encoder. The main purpose of this module is to enhance or weaken specific frequencies of the time series data in the frequency domain by employing a set of learnable weights. To elaborate, let us specify the Fourier transform operation as $fft$, and the inverse Fourier transform as $ifft$. The input time series data is denoted as $x$ ($x \in \mathbb{R}^{L\times D}$), where $L$ is the sequence length, and $D$ represents the number of dimensions or channels in the data. The corresponding learnable weights are denoted as $w$ ($w \in \mathbb{R}^{L\times D}$). Considering the above, the workflow of the entire module can be succinctly expressed as follows:

$$\hat{x} = fft(x)$$

$$output = ifft(w \circ \hat{x})$$

The operator ∘ is the Hadamard product. In our module, the first step is to obtain the frequency domain representation $\hat{x}$ of the input time series data $x$ using the fast Fourier transform. Next, we perform the Hadamard product between the learnable weights $w$ and $\hat{x}$ effectively modifying the frequency components. Finally, we invert the Fourier transform on the result to bring it back to the time domain, resulting in the modified time series data with the desired frequency adjustments. To summarize, the Frequency Selection module achieves its objective by transforming the input data into the frequency domain, applying learnable weights to the frequency components using the Hadamard product, and then bringing the modified data back to the time domain through the inverse Fourier transform. This process allows the module to selectively enhance or weaken specific frequencies in the time series data, contributing to noise reduction or other filtering tasks in the encoder's workflow.

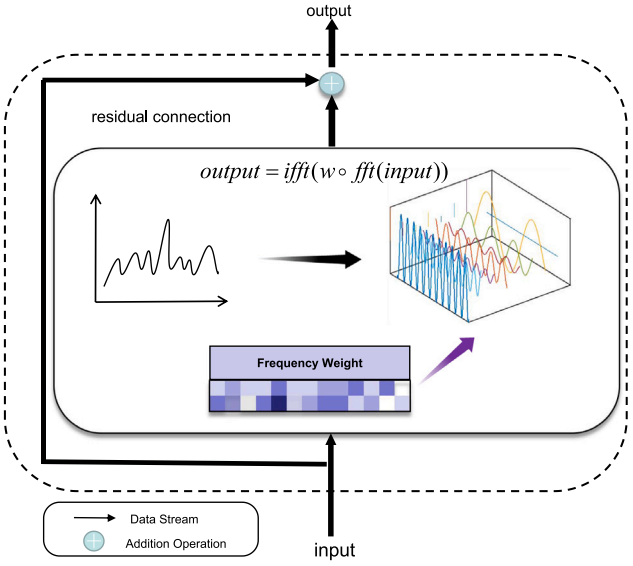



**Fig. 8.** The working flow of Frequency Selection.

## 4. Experiments

In this section, we will comprehensively verify the performance of PWDformer through a series of extensive experiments. To achieve this, we carefully selected six datasets that represent various application scenarios for most long-term forecasting tasks. These datasets encompass five different scenarios: energy, economy, transportation, weather, and disease prediction. During our study, we discovered that traditional time series prediction models, such as ARIMA [16], RNN [8], and CNN [7]-based models, performed inadequately on these datasets. This observation is further corroborated by the findings presented in the studies of Wu et al. [29] and Zhou et al. [25]. To conduct a fair comparison, we selected the best-performing models on these datasets, with most of them being based on the Transformer architecture [24]. For the multivariate forecasting task, we compared the performance of PWD-former against several prominent models, including Autoformer [29], Informer [25], LogTrans [27], Reformer [26], LSTNet [41], LSTM [42], and TCN [43]. Additionally, for the univariate forecasting task, we contrasted the performance of PWDformer against Autoformer [29], Informer [25], LogTrans [27], and Reformer [26].

Through these comprehensive comparative experiments, we aim to demonstrate the superiority of PWDformer in tackling a wide range of long-term forecasting scenarios. The results will highlight PWDformer's ability to outperform existing state-of-the-art Transformer-based models in both univariate and multivariate forecasting tasks across diverse application domains.

### *4.1. Datasets and experimental settings*

In this section, the details of the experiment datasets are summarized as follows: (1) ETT [29] dataset contains four sub-dataset: ETTh1, ETTh2, ETTm1 and ETTm2. Among them, ETTh1 and ETTm1 come from the power transformer of the same site.ETTh2, ETTm2 from another site. ETTh1 and ETTh2 represent the collection frequency in hours. ETTm1 and ETTm2 represent the collection frequency in the unit of 15 min. ETT dataset contains multiple series of loads and one series of oil temperatures. (2) The Electricity[1] dataset is a dataset on electricity consumption that collects the electricity consumption of 322 clients over time (one client per column). (3) The Exchange [41] dataset is about the exchange rates and contains exchange rate data for exchanges of eight countries. (4) The Traffic[2] dataset contains the traffic bring collected on highway in California. (5) Weather[3] dataset has 21 meteorological indicators for a range of 1 year in Germany. (6) Illness[4] dataset contains the influenza-like illness patients in the United States. We follow the Autoformer [29] setup for partitioning the dataset and split all datasets into training, validation and test set in chronological order by the ratio of 6:2:2 for the ETT dataset and 7:1:2 for all others.

In terms of experimental setting, we are also consistent with Auto-former: The method is trained with the L2 loss, using the ADAM [44] optimizer with an initial learning rate of 1e-4. Batch size is set to 32. The training process is early stopped within 10 epochs. All of our experiments were repeated three times, implemented in PyTorch [45] and carried out on a single NVIDIA GeForce RTX 3060 12 GB GPU.

[1] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[2] http://pems.dot.ca.gov
[3] https://www.bgc-jena.mpg.de/wetter/
[4] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

### 4.2. Multivariate results

As depicted in Table 4, our model demonstrates superior perfor-mance compared to the existing state-of-the-art (SOTA) model (Auto-former) across all four subdatasets of ETT. Notably, on the ETTh1 dataset, our model achieved an impressive **9.7%** improvement in Mean Squared Error (MSE) (0.490 → 0.442) and a significant **5.8%** improvement in Mean Absolute Error (MAE) (0.481 → 0.453). Similarly, on the ETTh2 dataset, our model achieved an **11.2%** improvement in MSE (0.457 → 0.406) and a notable **7%** improvement in MAE (0.455 → 0.423). Furthermore, on the ETTm1 dataset, our model demonstrated a remarkable **22.2%** improvement in MSE (0.634 → 0.491) and a substantial **9.5%** improvement in MAE (0.453 → 0.41). On the ETTm2 dataset, our model showcased impressive improvements of **18.4%** in MSE (0.255 → 0.208) and **14.5%** in MAE (0.339 → 0.29).

Similarly, as evident from Table 5, our model outperforms the existing SOTA model (Autoformer) across the remaining five datasets as well. On the Electricity dataset, our model achieved a noteworthy **14.9%** improvement in MSE (0.222 → 0.189) and a significant **9.6%**improvement in MAE (0.334 → 0.302). For the Exchange dataset, our model demonstrated an impressive **27.1%** improvement in MSE (0.509 → 0.371) and a substantial 14.7% improvement in MAE (0.524 → 0.447). On the Traffic dataset, our model showcased a considerable 13.8% improvement in MSE (0.660 → 0.569) and a notable 13.7%improvement in MAE (0.408 → 0.352). Moreover, on the Weather dataset, our model achieved a noteworthy 10.5% improvement in MSE (0.266 → 0.238) and a significant 7.7% improvement in MAE (0.336 → 0.31). Lastly, on the ILI dataset, our model demonstrated a remarkable 16.9% improvement in MSE (3.103 → 2.578) and a substantial 9.0%improvement in MAE (1.148 → 1.045).

Overall, our model consistently achieved a substantial performance improvement compared to the existing models. Particularly noteworthy is the improvement of over **20%** observed on some datasets (e.g., Ex-change, ETTm1). Notably, our model showcased exceptional performance on the Traffic dataset, achieving a **13.8%** MSE improvement with a forecast length of 720, and on the Exchange dataset, where a **21.4%** MSE improvement was achieved with the same forecast length.

We also provide the Pearson correlation coefficient and the Spear-man correlation coefficient for some datasets. Our model achieved the best results in both of these evaluation metrics. As shown in Table 3.

These compelling results further underscore the advantages of our model for handling *long-term forecasting*.

**Table 3**
Pearson correlation coefficient and the Spearman correlation coefficient.

| Datasets | | ETTh1 | | | | | ETTh2 | | | | | ETTm1 | | | | | ETTm2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PredictionLength | | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 96 | 288 | 672 | 24 | 48 | 96 | 288 | 672 |
| Ours | Pearson | **0.682** | **0.68** | **0.578** | **0.516** | **0.451** | **0.633** | **0.535** | **0.358** | **0.246** | **0.161** | **0.744** | **0.7** | **0.682** | **0.576** | **0.513** | **0.785** | **0.755** | **0.702** | **0.566** | **0.417** |
| | Spearman | **0.343** | **0.334** | **0.288** | **0.254** | **0.237** | **0.309** | **0.265** | **0.182** | **0.123** | **0.084** | **0.362** | **0.342** | **0.332** | **0.296** | **0.269** | **0.382** | **0.369** | **0.348** | **0.286** | **0.213** |
| Autoformer | Pearson | 0.624 | 0.633 | 0.515 | 0.506 | 0.388 | 0.557 | 0.529 | 0.34 | 0.24 | 0.118 | 0.651 | 0.617 | 0.57 | 0.508 | 0.502 | 0.779 | 0.74 | 0.679 | 0.548 | 0.409 |
| | Spearman | 0.307 | 0.317 | 0.257 | 0.252 | 0.2 | 0.274 | 0.263 | 0.173 | 0.121 | 0.067 | 0.316 | 0.299 | 0.282 | 0.268 | 0.259 | 0.375 | 0.362 | 0.338 | 0.28 | 0.21 |

**Table 4**
Multivariate prediction results on the ETT dataset. We follow the SOTA models' (Autoformer [29], Informer [25] etc.) settings: set the input length as 96. A lower MSE or MAE indicates a better prediction.

| Models | | Ours | | Autoformer | | Informer | | LogTrans | | Reformer | | LSTNet | | LSTMa | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 24 | **0.357** | **0.409** | 0.384 | 0.425 | 0.577 | 0.549 | 0.686 | 0.604 | 0.991 | 0.754 | 1.293 | 0.901 | 0.650 | 0.624 |
| | 48 | **0.38** | **0.419** | 0.392 | 0.419 | 0.685 | 0.625 | 0.766 | 0.757 | 1.313 | 0.906 | 1.456 | 0.960 | 0.702 | 0.675 |
| | 168 | **0.442** | **0.453** | 0.490 | 0.481 | 0.931 | 0.752 | 1.002 | 0.846 | 1.824 | 1.138 | 1.997 | 1.214 | 1.212 | 0.867 |
| | 336 | **0.481** | **0.482** | 0.505 | 0.484 | 1.128 | 0.873 | 1.362 | 0.952 | 2.117 | 1.280 | 2.655 | 1.369 | 1.424 | 0.994 |
| | 720 | 0.503 | 0.511 | **0.498** | **0.500** | 1.215 | 0.896 | 1.397 | 1.291 | 2.415 | 1.520 | 2.143 | 1.380 | 1.960 | 1.322 |
| ETTh2 | 24 | **0.238** | **0.323** | 0.261 | 0.341 | 0.720 | 0.665 | 0.828 | 0.750 | 1.531 | 1.613 | 2.742 | 1.457 | 1.143 | 0.813 |
| | 48 | **0.297** | **0.36** | 0.312 | 0.373 | 1.457 | 1.001 | 1.806 | 1.034 | 1.871 | 1.735 | 3.567 | 1.687 | 1.671 | 1.221 |
| | 168 | **0.406** | **0.423** | 0.457 | 0.455 | 3.489 | 1.515 | 4.070 | 1.681 | 4.660 | 1.846 | 3.242 | 2.513 | 4.117 | 1.674 |
| | 336 | **0.466** | **0.475** | 0.471 | 0.475 | 2.723 | 1.340 | 3.875 | 1.763 | 4.028 | 1.688 | 2.544 | 2.591 | 3.434 | 1.549 |
| | 720 | **0.458** | **0.476** | 0.474 | 0.484 | 3.467 | 1.473 | 3.913 | 1.552 | 5.381 | 2.015 | 4.625 | 3.709 | 3.963 | 1.788 |
| ETTm1 | 24 | 0.34 | 0.391 | 0.383 | 0.403 | **0.323** | **0.369** | 0.419 | 0.412 | 0.724 | 0.607 | 1.968 | 1.170 | 0.621 | 0.629 |
| | 48 | **0.383** | **0.41** | 0.454 | 0.453 | 0.494 | 0.503 | 0.507 | 0.583 | 1.098 | 0.777 | 1.999 | 1.215 | 1.392 | 0.939 |
| | 96 | **0.398** | **0.429** | 0.481 | 0.463 | 0.678 | 0.614 | 0.768 | 0.792 | 1.433 | 0.945 | 2.762 | 1.542 | 1.339 | 0.913 |
| | 288 | **0.491** | **0.489** | 0.634 | 0.528 | 1.056 | 0.786 | 1.462 | 1.320 | 1.820 | 1.094 | 1.257 | 2.076 | 1.740 | 1.124 |
| | 672 | **0.559** | **0.516** | 0.606 | 0.542 | 1.192 | 0.926 | 1.669 | 1.461 | 2.187 | 1.232 | 1.917 | 2.941 | 2.736 | 1.555 |
| ETTm2 | 24 | **0.149** | **0.255** | 0.153 | 0.261 | 0.173 | 0.301 | 0.211 | 0.332 | 0.333 | 0.429 | 1.101 | 0.831 | 0.580 | 0.572 |
| | 48 | **0.171** | **0.269** | 0.178 | 0.280 | 0.303 | 0.409 | 0.427 | 0.487 | 0.558 | 0.571 | 2.619 | 1.393 | 0.747 | 0.630 |
| | 96 | **0.208** | **0.29** | 0.255 | 0.339 | 0.365 | 0.453 | 0.768 | 0.642 | 0.658 | 0.619 | 3.142 | 1.365 | 2.041 | 1.073 |
| | 288 | **0.309** | **0.353** | 0.342 | 0.378 | 1.047 | 0.804 | 1.090 | 0.806 | 2.441 | 1.190 | 2.856 | 1.329 | 0.969 | 0.742 |
| | 672 | **0.403** | **0.407** | 0.434 | 0.430 | 3.126 | 1.302 | 2.397 | 1.214 | 3.090 | 1.328 | 3.409 | 1.420 | 2.541 | 1.239 |

**Table 5**
Multivariate prediction results on the Five dataset. We follow the SOTA models' (Autoformer [29], Informer [25] etc.) settings: set the input length I as 36 for ILI and 96 for the others. A lower MSE or MAE indicates a better prediction.

| Models | | Ours | | Autoformer | | Informer | | LogTrans | | Reformer | | LSTNet | | LSTM | | TCN | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| Electricity | 96 | **0.185** | **0.299** | 0.201 | 0.317 | 0.274 | 0.368 | 0.258 | 0.357 | 0.312 | 0.402 | 0.680 | 0.654 | 0.375 | 0.437 | 0.985 | 0.813 |
| | 192 | **0.189** | **0.302** | 0.222 | 0.334 | 0.296 | 0.386 | 0.266 | 0.368 | 0.348 | 0.433 | 0.725 | 0.676 | 0.442 | 0.473 | 0.996 | 0.821 |
| | 336 | **0.198** | **0.31** | 0.231 | 0.338 | 0.300 | 0.394 | 0.280 | 0.380 | 0.350 | 0.433 | 0.828 | 0.727 | 0.439 | 0.473 | 1.000 | 0.824 |
| | 720 | **0.219** | **0.328** | 0.254 | 0.361 | 0.373 | 0.439 | 0.283 | 0.376 | 0.340 | 0.420 | 0.957 | 0.811 | 0.980 | 0.814 | 1.438 | 0.784 |
| Exchange | 96 | **0.153** | **0.286** | 0.197 | 0.323 | 0.847 | 0.752 | 0.968 | 0.812 | 1.065 | 0.829 | 1.551 | 1.058 | 1.453 | 1.049 | 3.004 | 1.432 |
| | 192 | **0.245** | **0.368** | 0.300 | 0.369 | 1.204 | 0.895 | 1.040 | 0.851 | 1.188 | 0.906 | 1.477 | 1.028 | 1.846 | 1.179 | 3.048 | 1.444 |
| | 336 | **0.371** | **0.447** | 0.509 | 0.524 | 1.672 | 1.036 | 1.659 | 1.081 | 1.357 | 0.976 | 1.507 | 1.031 | 2.136 | 1.231 | 3.113 | 1.459 |
| | 720 | **1.137** | **0.825** | 1.447 | 0.941 | 2.478 | 1.310 | 1.941 | 1.127 | 1.510 | 1.016 | 2.285 | 1.243 | 2.984 | 1.427 | 3.150 | 1.458 |
| Traffic | 96 | **0.58** | **0.365** | 0.613 | 0.388 | 0.719 | 0.391 | 0.684 | 0.384 | 0.732 | 0.423 | 1.107 | 0.685 | 0.843 | 0.453 | 1.438 | 0.784 |
| | 192 | **0.578** | **0.353** | 0.616 | 0.382 | 0.696 | 0.379 | 0.685 | 0.390 | 0.733 | 0.420 | 1.157 | 0.706 | 0.847 | 0.453 | 1.463 | 0.794 |
| | 336 | **0.569** | **0.346** | 0.622 | 0.337 | 0.777 | 0.420 | 0.733 | 0.408 | 0.742 | 0.420 | 1.216 | 0.730 | 0.853 | 0.455 | 1.479 | 0.799 |
| | 720 | **0.569** | **0.352** | 0.660 | 0.408 | 0.864 | 0.472 | 0.717 | 0.396 | 0.755 | 0.423 | 1.481 | 0.805 | 1.500 | 0.805 | 1.499 | 0.804 |
| Weather | 96 | **0.238** | **0.31** | 0.266 | 0.336 | 0.300 | 0.384 | 0.458 | 0.490 | 0.689 | 0.596 | 0.594 | 0.587 | 0.369 | 0.406 | 0.615 | 0.589 |
| | 192 | **0.291** | **0.346** | 0.307 | 0.367 | 0.598 | 0.544 | 0.658 | 0.589 | 0.752 | 0.638 | 0.560 | 0.565 | 0.416 | 0.435 | 0.629 | 0.600 |
| | 336 | 0.368 | 0.399 | **0.359** | **0.395** | 0.578 | 0.523 | 0.797 | 0.652 | 0.639 | 0.596 | 0.597 | 0.587 | 0.455 | 0.454 | 0.639 | 0.608 |
| | 720 | **0.407** | **0.414** | 0.419 | 0.428 | 1.059 | 0.741 | 0.869 | 0.675 | 1.130 | 0.792 | 0.618 | 0.599 | 0.535 | 0.520 | 0.639 | 0.610 |
| ILI | 24 | **3.055** | **1.193** | 3.483 | 1.287 | 5.764 | 1.677 | 4.480 | 1.444 | 4.400 | 1.382 | 6.026 | 1.770 | 5.914 | 1.734 | 6.624 | 1.830 |
| | 36 | **2.578** | **1.045** | 3.103 | 1.148 | 4.755 | 1.467 | 4.799 | 1.467 | 4.783 | 1.448 | 5.340 | 1.668 | 6.631 | 1.845 | 6.858 | 1.879 |
| | 48 | 2.685 | 1.096 | **2.669** | **1.085** | 4.763 | 1.469 | 4.800 | 1.468 | 4.832 | 1.465 | 6.080 | 1.787 | 6.736 | 1.857 | 6.968 | 1.892 |
| | 60 | **2.7** | **1.111** | 2.770 | 1.125 | 5.264 | 1.564 | 5.278 | 1.560 | 4.882 | 1.483 | 5.548 | 1.720 | 6.870 | 1.879 | 7.127 | 1.918 |

### *4.3. Univariate results*

The results of the univariate prediction are presented in Table 6. A comparison with Autoformer reveals that our model has exhibited improved accuracy in the majority of cases. Remarkably, on the ETTm1 dataset, our model achieved a remarkable **32.1%** increase in MSE for a predicted length of 96, and an impressive **30.9%** increase in MSE for a predicted length of 192, respectively. These significant improvements underscore the superiority of our model in tackling univariate forecasting tasks, particularly on the ETTm1 dataset.

**Result analysis:** Our model consistently outperforms Autoformer in both multivariate and univariate prediction tasks, as evidenced by lower MSE and MAE values, indicating closer proximity to the true values. Moreover, our model exhibits higher Spearman and Pearson coefficients, suggesting a stronger correlation between predicted and true values. These superior performance metrics are attributed to the incorporation of three proposed modules: DL aggregation, Position Weights and Frequency-Selection. By leveraging diverse historical features from multiple perspectives, our model excels at capturing the general trend of sequences with remarkable accuracy, as demonstrated in Table 3 where our model achieves the highest correlation coefficients across all ETT datasets compared to Autoformer.

For cases such as 336 and 720, where the prediction length is more extreme, we believe that further analysis is necessary to clearly demonstrate our contribution. When the prediction step size is very long, we believe that more rich and diverse feature information is the key to improve the performance of the model. This is similar to Bengio et al. [46] mentioned that feature representation from multiple sources helps to improve model performance. The DL aggregation module enriches the acquisition of features from a local perspective. The Position Weights provides more information to the model from a sequential perspective. Frequency-Selection module performs feature selection from the frequency domain perspective. All of these factors contribute to our model's ability to accurately capture the underlying patterns of the original sequence even under extreme prediction step sizes.

As shown in Fig. 9, our model outperforms Autoformer in capturing the general trend of the series, even when facing a prediction step size of 720. The higher Spearman and Pearson coefficients reported in Table 3 indicate that our predicted results are more similar to the ground truth compared to Autoformer. While our model may have slightly higher MSE and MAE indices than Autoformer for ETTh1 at a prediction step size of 720, it is important to note that these metrics do not fully represent the quality of predictions as low MSE values can still result in irrelevant predictions. Combining this with correlation evaluation results (Table 3), we observe that our proposed model achieves significantly higher Pearson and Spearman coefficients at 0.451 and 0.237 respectively, compared to Autoformer's scores of 0.388 and 0.2 respectively. This is further supported by Fig. 9 which shows that while Autoformer may make 'lucky' predictions, its overall trend prediction is completely wrong.

**Table 6**
Univariate prediction results on the ETT dataset. We set the input length as 96. A lower MSE or MAE indicates a better prediction.

| Models | | Ours | | Autoformer | | Informer | | LogTrans | | Reformer | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | 96 | 0.073 | 0.211 | **0.071** | **0.206** | 0.193 | 0.377 | 0.283 | 0.468 | 0.532 | 0.569 |
| | 192 | **0.92** | **0.234** | 0.114 | 0.262 | 0.217 | 0.395 | 0.234 | 0.409 | 0.568 | 0.575 |
| | 336 | **0.106** | **0.257** | 0.107 | 0.258 | 0.202 | 0.381 | 0.386 | 0.546 | 0.635 | 0.589 |
| | 720 | **0.123** | **0.279** | 0.126 | 0.283 | 0.183 | 0.355 | 0.475 | 0.628 | 0.762 | 0.666 |
| ETTh2 | 96 | **0.144** | **0.294** | 0.153 | 0.306 | 0.213 | 0.373 | 0.217 | 0.379 | 1.411 | 0.838 |
| | 192 | **0.194** | **0.342** | 0.204 | 0.351 | 0.227 | 0.387 | 0.281 | 0.429 | 5.658 | 1.671 |
| | 336 | **0.242** | **0.389** | 0.246 | 0.389 | 0.242 | 0.401 | 0.293 | 0.437 | 4.777 | 1.582 |
| | 720 | 0.275 | 0.422 | **0.268** | **0.409** | 0.291 | 0.439 | 0.218 | 0.387 | 2.042 | 1.039 |
| ETTm1 | 96 | **0.038** | **0.151** | 0.056 | 0.183 | 0.109 | 0.277 | 0.049 | 0.171 | 0.296 | 0.355 |
| | 192 | **0.056** | **0.186** | 0.081 | 0.216 | 0.151 | 0.310 | 0.157 | 0.317 | 0.429 | 0.474 |
| | 336 | **0.073** | **0.214** | 0.076 | 0.218 | 0.427 | 0.591 | 0.289 | 0.459 | 0.585 | 0.583 |
| | 720 | **0.91** | **0.235** | 0.110 | 0.267 | 0.438 | 0.586 | 0.430 | 0.579 | 0.782 | 0.730 |
| ETTm2 | 96 | **0.077** | **0.213** | 0.065 | 0.189 | 0.088 | 0.225 | 0.075 | 0.208 | 0.076 | 0.214 |
| | 192 | **0.107** | **0.254** | 0.118 | 0.256 | 0.132 | 0.283 | 0.129 | 0.275 | 0.132 | 0.290 |
| | 336 | **0.138** | **0.289** | 0.154 | 0.305 | 0.180 | 0.336 | 0.154 | 0.302 | 0.160 | 0.312 |
| | 720 | **0.18** | **0.329** | 0.182 | 0.335 | 0.300 | 0.435 | 0.160 | 0.321 | 0.168 | 0.335 |

### *4.4. Ablation study*

In order to further explore and demonstrate the validity of our proposed three modules, we conducted ablation experiments on two different datasets.

Tables 7–8 show the results on the ETTm2 and Exchange datasets respectively. For the sake of illustration, we have added a column (for short) to the far left of the table to mark the four different module configurations. We set Mode A as the baseline model Autoformer. Mode B represents the individual Frequency Selection module. Mode C is the individual DL Aggregation module. Mode D is the individual Position Weights module. Mode E combines the Frequency Selection module with the DL Aggregation module. Mode F combines the Frequency Selection module with the Position Weights module. Mode G combines the DL Aggregation module with the Position Weights module. Mode H includes all three modules, representing our complete model. The horizontal axis represents the predicted length and the vertical axis represents the MSE.

Ablation experiments on two different datasets demonstrate the effectiveness of our proposed three modules. From the perspective of individual modules, the Frequency Selection module generally provides performance improvement in most cases. However, it only weakens the baseline model's performance on the ETT dataset when the prediction stride is 48. The DL Aggregation module performs well on the Exchange dataset, improving the model's performance for all four prediction strides. The improvement becomes more evident with increasing prediction stride. For instance, at a prediction stride of 96, the baseline model's MSE improves from 0.197 to 0.167 (17.9%). At a prediction stride of 336, it improves from 0.509 to 0.401 (21.2%). However, the DL Aggregation module's performance on the ETT dataset is not satisfactory, as it even weakens the model's performance with shorter prediction strides (24, 48). Its performance on the ETT dataset improves gradually with the increase of prediction stride. On the other hand, the Position Weights module provides positive assistance to the model in all cases and both datasets, regardless of the prediction stride.

From the perspective of combining two modules, let us start with the combination of the Frequency Selection module and DL Aggregation module. This combination slightly weakens the model's performance on the ETT dataset at a prediction stride of 48. Similarly, the combination of the Frequency Selection module and Position Weights module also slightly weakens the model's performance on the ETT dataset with a prediction stride of 48. This may be related to the subpar performance of the individual Frequency Selection module in this particular case. Now, the combination of the Frequency Selection module with Position Weights module has the smallest improvement effect compared to other dual-module combinations. In some cases, it does not improve the model or even has a negative impact. For example, on the ETT dataset with prediction strides of 24 and 48. Finally, the combination of the DL Aggregation module with Position Weights module is the most beneficial among the dual-module combinations. Especially as the prediction stride increases, its effect becomes more prominent. For instance, on the Exchange dataset with a prediction stride of 96, this combination's improvement effect is still smaller than Mode E, but as the prediction stride increases, its improvement effect surpasses Mode E.Those further demonstrate the capability of our model for the *long-term forecasting*.

### *4.5. Position weights visualization*

As demonstrated in Figs. 10 and 11, we have selected a subset of data from the Exchange dataset to visualize the effect of Position Weights. In each graph, the first subgraph depicts the original correlation, while the second subgraph represents the correlation after applying our proposed Position Weights. The third subgraph visualizes the relevance of attention with the application of Position Weights processing.

As we discussed earlier, existing Transformer-based models tend to be order-insensitive, which may not be ideal for time series prediction tasks. In contrast, our proposed Position Weights take into account the importance of different positions in the time series data. As evident from Figs. 10–11, the original correlation appears regular and order-insensitive. However, once adjusted using the Position Weights, the models can capture more intricate and complex time patterns, which are crucial for accurate time series predictions. The enhanced ability to capture these complex temporal dependencies contributes to the improved performance of our model over existing Transformer-based approaches.

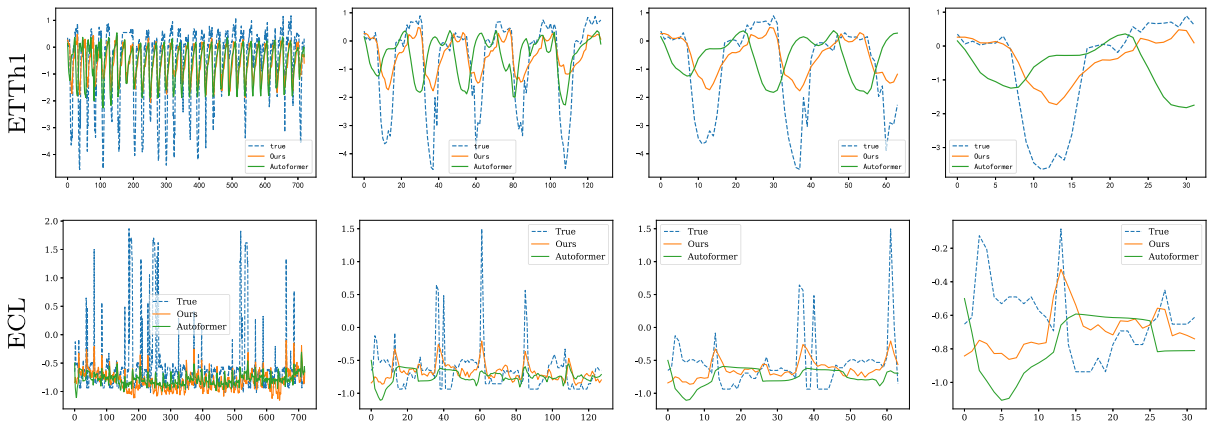



**Fig. 9.** 720 step prediction detail display. When the predicted step size is 720, the results are gradually enlarged from left to right.

**Table 7**
Results of the ablation experiment on the Exchange dataset. We set the input length as 96. A lower MSE or MAE indicates a better prediction.

| For short | Frequency selection | DL aggregation | Position weights | Predictions lengths | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 96 | | 192 | | 336 | | 720 | |
| | | | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| Mode A | | | | 0.197 | 0.323 | 0.3 | 0.369 | 0.509 | 0.524 | 1.447 | 0.941 |
| Mode B | ✓ | | | 0.155 | 0.286 | 0.274 | 0.383 | 0.467 | 0.511 | 1.2 | 0.851 |
| Mode C | | ✓ | | 0.167 | 0.294 | 0.278 | 0.389 | 0.401 | 0.469 | 1.199 | 0.849 |
| Mode D | | | ✓ | 0.186 | 0.317 | 0.299 | 0.365 | 0.437 | 0.491 | 1.314 | 0.877 |
| Mode E | ✓ | ✓ | | 0.141 | 0.273 | 0.272 | 0.381 | 0.409 | 0.473 | 1.156 | 0.838 |
| Mode F | ✓ | | ✓ | 0.155 | 0.283 | 0.27 | 0.384 | 0.423 | 0.499 | 1.192 | 0.844 |
| Mode G | | ✓ | ✓ | 0.149 | 0.277 | 0.251 | 0.379 | 0.378 | 0.46 | 1.141 | 0.829 |
| Mode H (ours) | ✓ | ✓ | ✓ | 0.153 | 0.286 | 0.245 | 0.37 | 0.371 | 0.447 | 1.137 | 0.825 |

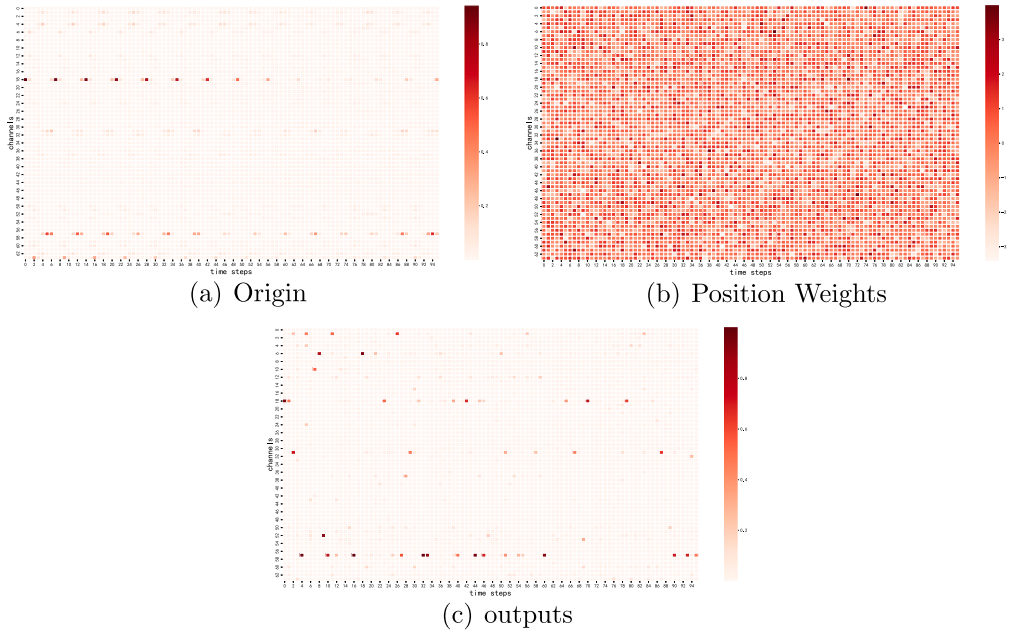
**Table 8**
Results of the ablation experiment on the ETTm2 dataset. We set the input length I as 96. A lower MSE or MAE indicates a better prediction.

| For short | Frequency selection | DL aggregation | Position weights | Predictions lengths | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 24 | | 48 | | 96 | | 288 | | 672 | |
| | | | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| Mode A | | | | 0.153 | 0.261 | 0.178 | 0.28 | 0.255 | 0.339 | 0.342 | 0.378 | 0.434 | 0.43 |
| Mode B | ✓ | | | 0.15 | 0.26 | 0.181 | 0.282 | 0.246 | 0.331 | 0.332 | 0.376 | 0.427 | 0.421 |
| Mode C | | ✓ | | 0.155 | 0.264 | 0.183 | 0.289 | 0.254 | 0.341 | 0.319 | 0.371 | 0.412 | 0.427 |
| Mode D | | | ✓ | 0.151 | 0.259 | 0.177 | 0.284 | 0.247 | 0.331 | 0.328 | 0.369 | 0.429 | 0.426 |
| Mode E | ✓ | ✓ | | 0.144 | 0.259 | 0.179 | 0.283 | 0.221 | 0.307 | 0.312 | 0.365 | 0.408 | 0.411 |
| Mode F | ✓ | | ✓ | 0.149 | 0.262 | 0.183 | 0.29 | 0.252 | 0.339 | 0.34 | 0.373 | 0.424 | 0.419 |
| Mode G | | ✓ | ✓ | 0.146 | 0.26 | 0.176 | 0.281 | 0.211 | 0.305 | 0.309 | 0.361 | 0.405 | 0.409 |
| Mode H (ours) | ✓ | ✓ | ✓ | 0.149 | 0.255 | 0.171 | 0.269 | 0.208 | 0.29 | 0.309 | 0.353 | 0.403 | 0.407 |

### *4.6. Forecasting showcases*

To provide a more intuitive demonstration of our model's performance, we have randomly selected a portion of the data for visualization. A detailed comparison with the Autoformer model is presented to showcase the superiority of our approach.

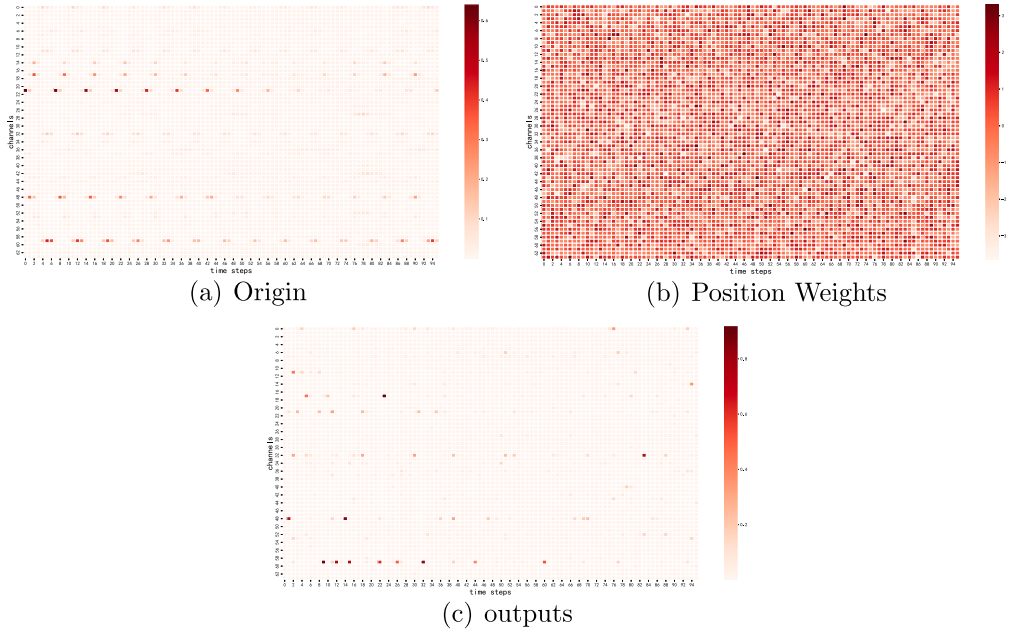
As depicted in Fig. 12, and Fig. 13, our model exhibits a notable reduction in prediction error compared to the Autoformer model. Additionally, the prediction curves generated by our model are notably smoother. These results further emphasize the enhanced predictive capabilities of our model, particularly in terms of achieving more accurate and refined predictions compared to the Autoformer model. The visualizations offer a clear and intuitive representation of our model's performance, validating its efficacy in handling time series forecasting tasks.




(a) Origin (b) Position Weights

(c) outputs

**Fig. 10.** Multihead Position Weights' heatmap with 8 heads on the Exchange dataset. This is the head 1. Since the dimension of the data in the model is 512 and the number of heads is 8, the number of channels per head is 64. The $y$ axis represents the channel and the horizontal axis represents the point in time to be predicted. The shade of each pixel represents the amount of attention (or weight).




(a) Origin (b) Position Weights

(c) outputs

**Fig. 11.** Multihead Position Weights' heatmap with 8 heads on The Exchange dataset. This is the head 2.

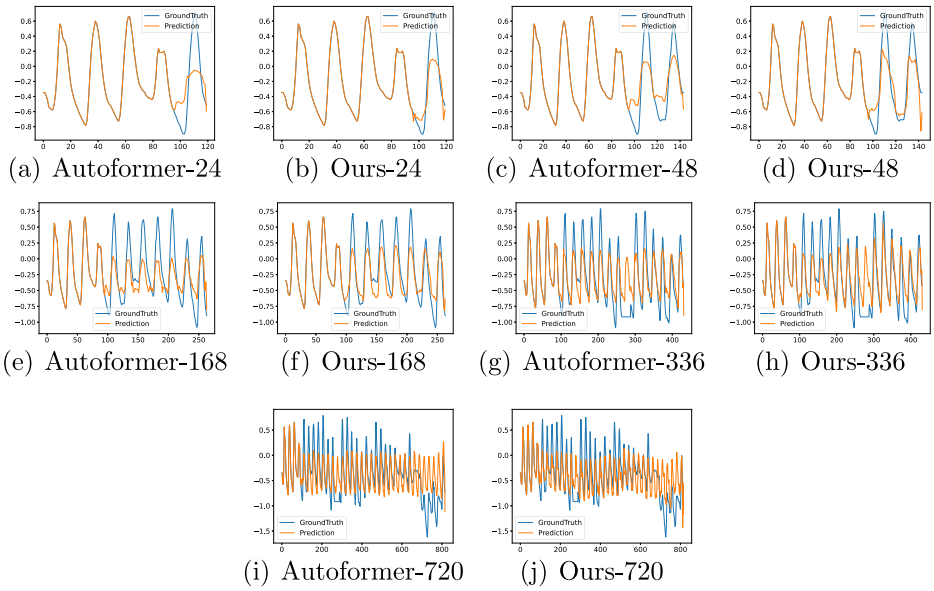



**Fig. 12.** Visualization results of multivariate prediction on ETTh2 dataset. Input step size is 96. Ours-x represents the performance of Ours under the setting of prediction step size x.

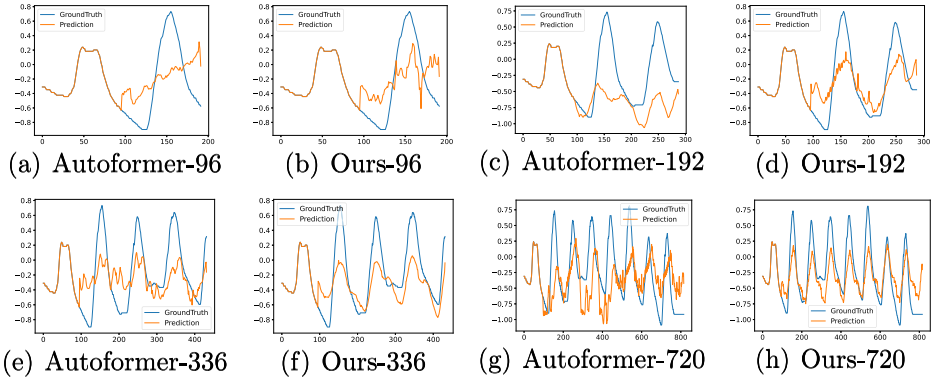



**Fig. 13.** Visualization results of multivariate prediction on ETTm2 dataset. Input step size is 96. Ours-x represents the performance of Ours under the setting of prediction step size x.

## 5. Conclusion

We present a novel deep learning model, PWDformer, for long-term forecasting by integrating signal processing techniques with an autoregressive model based on deep learning. The experimental results, both for multivariate and univariate prediction tasks, support our proposed mechanisms. PWDformer achieved state-of-the-art results on six popular datasets, demonstrating its strong potential for long-term forecasting tasks. Moreover, we provided evidence of DL aggregation's ability to extract complex time patterns using information entropy analysis. A detailed ablation experiment further validated the effectiveness of the three proposed modules: DL aggregation, Position Weights, and Frequency Selection. Visualization results of Position Weights confirmed its role in solving the order-insensitivity problem.

Despite significant progress and the proposal of the novel deep learning model, PWDformer, our study acknowledges several limitations and shortcomings that require attention and future research: Dataset Selection: In this study, we utilized multiple publicly available datasets to validate the effectiveness of PWDformer. However, these datasets might not fully represent the diversity and complexity of real-world time series data. Future research should consider incorporating more diverse and domain-specific datasets to ensure the model's generalizability. Computational Complexity: PWDformer incorporates DL aggregation and other advanced techniques, which might introduce increased computational complexity. While our model has demonstrated competitive performance on existing hardware, further efforts are needed to optimize the model's efficiency for large-scale and real-time applications.

In the future, we recommend exploring the integration of more deep learning methods with traditional signal processing techniques. For instance, incorporating comparative learning to mitigate the impact of noise on predictions could yield promising results. Additionally, developing a low-complexity Transformer-based model would be valu-able for extending the length of input and output sequences. Exploring spectrum analysis technology in deep learning models can enable the mining of even more intricate time patterns.

Overall, our research paves the way for further improvements in long-term forecasting by synergizing deep learning and signal processing methodologies. Addressing the identified limitations and exploring new avenues of research will undoubtedly contribute to advancing the field and enhancing the performance of predictive models for complex time series data.

## Declaration of competing interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Data availability

Data will be made available on request.

## Acknowledgment

The authors wish to acknowledge the support for the research work from the National Natural Science Foundation of China (CN) under grant Nos. [62076180], and [62376189].

## References

[1] G. Merkuryeva, A. Valberga, A. Smirnov, Demand forecasting in pharmaceutical supply chains: A case study, Procedia Comput. Sci. 149 (2019) 3–10, http://dx.doi.org/10.1016/j.procs.2019.01.100, ICTE in Transportation and Logistics 2018 (ICTE 2018). URL https://www.sciencedirect.com/science/article/pii/S1877050919301061.

[2] D. Cheng, F. Yang, S. Xiang, J. Liu, Financial time series forecasting with multi-modality graph neural network, Pattern Recognit. 121 (2022) 108218, http://doi.org/10.1016/j.patcog.2021.108218, URL https://www.sciencedirect.com/science/article/pii/S003132032100399X.

[3] Y. Pang, X. Zhou, J. Zhang, Q. Sun, J. Zheng, Hierarchical electricity time series prediction with cluster analysis and sparse penalty, Pattern Recognit. 126 (2022) 108555, http://dx.doi.org/10.1016/j.patcog.2022.108555, URL https: //www.sciencedirect.com/science/article/pii/S003132032200036X.

[4] H. Hu, Y. Li, X. Zhang, M. Fang, A novel hybrid model for short-term prediction of wind speed, Pattern Recognit. 127 (2022) 108623, http://dx.doi.org/10.1016/patcog.2022.108623, URL https://www.sciencedirect.com/science/article/pii/S0031320322001042.

[5] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Adv. Neural Inf. Process. Syst. 27 (2014).

[6] F. Giuliari, I. Hasan, M. Cristani, F. Galasso, Transformer networks for trajectory forecasting, in: 2020 25th International Conference on Pattern Recognition, ICPR, 2021, pp. 10335–10342, http://dx.doi.org/10.1109/ICPR48806.2021.9412190.

[7] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, Handb. Brain Theory Neural Netw. 3361 (10) (1995) 1995.

[8] K. ichi Funahashi, Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Netw. 6 (6) (1993) 801–806, http://dx.doi.org/10.1016/S0893-6080(05)80125-X, URL https://www.sciencedirect. com/science/article/pii/S089360800580125X.

[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8)(1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735, arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

[10] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.

[11] S.S. Rangapuram, M.W. Seeger, J. Gasthaus, L. Stella, Y. Wang, T. Januschowski, Deep state space models for time series forecasting, Adv. Neural Inf. Process. Syst. 31 (2018).

[12] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, T. Januschowski, Deep factors for forecasting, in: International Conference on Machine Learning, PMLR, 2019, pp. 6607–6617.

[13] J. Durbin, S.J. Koopman, Time Series Analysis By State Space Methods, Vol. 38, OUP Oxford, 2012.

[14] H.-F. Yu, N. Rao, I.S. Dhillon, Temporal regularized matrix factorization for high-dimensional time series prediction, Adv. Neural Inf. Process. Syst. 29 (2016).

[15] R. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, Forecasting with Exponential Smoothing: The State Space Approach, Springer Science & Business Media, 2008.

[16] G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.

[17] C. Li, J.-W. Hu, A new ARIMA-based neuro-fuzzy approach and swarm intelligence for time series forecasting, Eng. Appl. Artif. Intell. 25 (2) (2012) 295–308, http://dx.doi.org/10.1016/j.engappai.2011.10.005, Special Section: Local Search Algorithms for Real-World Scheduling and Planning. URL https://www.sciencedirect.com/science/article/pii/S095219761100203X.

[18] R. Frigola, Bayesian Time Series Learning with Gaussian Processes (Ph.D. thesis), University of Cambridge, 2015.

[19] R. Umatani, T. Imai, K. Kawamoto, S. Kunimasa, Time series clustering with an EM algorithm for mixtures of linear Gaussian state space models, Pattern Recognit. 138 (2023) 109375, http://dx.doi.org/10.1016/j.patcog.2023.109375, URL https://www.sciencedirect.com/science/article/pii/S0031320323000766.

[20] C. Aicher, N.J. Foti, E.B. Fox, Adaptively truncating backpropagation through time to control gradient bias, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 799–808.

[21] T. Trinh, A. Dai, T. Luong, Q. Le, Learning longer-term dependencies in rnns with auxiliary losses, in: International Conference on Machine Learning, PMLR, 2018, pp. 4965–4974.

[22] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &Amp; Data Mining, KDD '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 753–763, http://dx.doi.org/10.1145/3394486.3403118.

[23] Z.L. Li, G.W. Zhang, J. Yu, L.Y. Xu, Dynamic graph structure learning for mul-tivariate time series forecasting, Pattern Recognit. 138 (2023) 109423, http://dx.doi.org/10.1016/j.patcog.2023.109423, URL https://www.sciencedirect.com/science/article/pii/S0031320323001243.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[25] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 12, 2021, pp. 11106–11115.

[26] N. Kitaev, Ł. Kaiser, A. Levskaya, Reformer: The efficient transformer, 2020, arXiv preprint arXiv:2001.04451.

[27] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, Adv. Neural Inf. Process. Syst. 32 (2019).

[28] C. Li, H. Li, B. Sun, Comprehensive evaluation of multi-energy complementary combined cooling heating and power system based on analytic hierarchy process, in: Chinese Control Conference, Vol. 2019-July, CCC, 2019, pp. 5243–5248, http://dx.doi.org/10.23919/ChiCC.2019.8865520, Cited by: 1. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0- 85074396379&doi=10.23919%2fChiCC.2019.8865520&partnerID=40&md5= 6d6d76eadd9187eebe3c4fcfa342f0a8.

[29] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.

[30] A. Tamkin, D. Jurafsky, N. Goodman, Language through a prism: A spectral approach for multiscale language representations, Adv. Neural Inf. Process. Syst. 33 (2020) 5492–5504.

[31] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, et al., Spectral temporal graph neural network for multivariate time-series forecasting, Adv. Neural Inf. Process. Syst. 33 (2020) 17766–17778.

[32] I. Beltagy, M.E. Peters, A. Cohan, Longformer: The long-document transformer, 2020, arXiv preprint arXiv:2004.05150.

[33] J. Brownlee, How to decompose time series data into trend and seasonality, 2017, Machinelearningmastery. com, Jan,

[34] P.J. Brockwell, R.A. Davis, Introduction to Time Series and Forecasting, Springer, 2002.

[35] G. Woo, C. Liu, D. Sahoo, A. Kumar, S. Hoi, CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting, 2022, arXiv preprint arXiv:2202.01575.

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

[37] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 764–773.

[38] X. Zhu, H. Hu, S. Lin, J. Dai, Deformable convnets v2: More deformable, better results, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9308–9316.

[39] C. Shannon, Claude Shannon, Inf. Theory 3 (1948) 224.

[40] S. Singh, Noise impact on time-series forecasting using an intelligent pattern matching technique, Pattern Recognit. 32 (8) (1999) 1389–1398, http://dx. doi.org/10.1016/S0031-3203(98)00174-5, URL https://www.sciencedirect.com/science/article/pii/S0031320398001745.

[41] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[42] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8)(1997) 1735–1780.

[43] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, arXiv preprint arXiv:1803.01271.

[44] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).

[46] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.