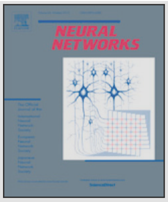
Neural Networks 180 (2024) 106745

Contents lists available at ScienceDirect

Neural Networks

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



Multi-scale convolution enhanced transformer for multivariate long-term time series forecasting

Ao Li [a], Ying Li [a], Yunyang Xu [a], Xuemei Li [a,*], Caiming Zhang [a,b]

[a] School of Software, Shandong University, Jinan 250101, China
[b] Shandong Provincial Laboratory of Future Intelligence and Financial Engineering, Yantai 264005, China

ARTICLE INFO

Keywords:
Multivariate time series
Transformer
Multi-scale segmentation
Long-term time series
Forecasting
Attention

ABSTRACT

In data analysis and forecasting, particularly for multivariate long-term time series, challenges persist. The Transformer model in deep learning methods has shown significant potential in time series forecasting. The Transformer model's dot-product attention mechanism, however, due to its quadratic computational complexity, impairs training and forecasting efficiency. In addition, the Transformer architecture has limitations in modeling local features and dealing with multivariate cross-dimensional dependency relationship. In this article, a Multi-Scale Convolution Enhanced Transformer model (MSCformer) is proposed for multivariate long-term time series forecasting. As an alternative to modeling the time series in its entirety, a segmentation strategy is designed to convert the input original series into segmented forms with different lengths, then process time series segments using a new constructed multi-Dependency Aggregation module. This multi-Scale segmentation approach reduces the computational complexity of the attention mechanism part in subsequent models, and for each segment of length corresponds to a specific time scale, it also ensures that each segment retains the semantic information of the data sequence level, thereby comprehensively utilizing the multi-scale information of the data while more accurately capturing the real dependency of the time series. The Multi-Dependence Aggregate module captures both cross-temporal and cross-dimensional dependencies of multivariate long-term time series and compensates for local dependencies within the segments thereby captures local series features comprehensively and addressing the issue of insufficient information utilization. MSCformer synthesizes dependency information extracted from various temporal segments at different scales and reconstructs future series using linear layers. MSCformer exhibits higher forecasting accuracy, outperforming existing methods in multiple domains including energy, transportation, weather, electricity, disease and finance.

1. Introduction

Studies on time series forecasting research aim to predict the future values of time series based on historical observations and are extensively applied in a variety of fields such as traffic management (Li et al., 2015; Shen et al., 2023), energy management (Pang et al., 2022), financial investment (Song et al., 2023), and disease spread analysis (Matsubara et al., 2014). The research can be divided into two main types: Univariate Time Series Forecasting and Multivariate Time Series Forecasting. In practical applications, a system often involves complex interactions between multiple variables. A multivariate time series forecasting method that comprehensively analyzes the relationships within and between multiple variables can improve our understanding of the behavior of variables in real-world systems, improving resource allocation, increasing efficiency, and supporting better decision-making (Deihim et al., 2023). In comparison with univariate forecasting models, multivariate forecasting models have a wider application prospect. Furthermore, multivariate time series forecasting models have increased requirements for predictable time horizons, especially for long-term planning and early warning applications. Increasing the forecast time horizon means that models can make more accurate forecasting about trends and changes over a longer period of time, allowing more time to take measures to minimize risks. However, the increasing length of forecasted series also places more demands on time series forecasting models.

[*] Corresponding author.

E-mail addresses: aoli@mail.sdu.edu.cn (A. Li), liying2022@mail.sdu.edu.cn (Y. Li), xuyunyang@mail.sdu.edu.cn (Y. Xu), xmli@sdu.edu.cn (X. Li), czhang@sdu.edu.cn (C. Zhang).

https://doi.org/10.1016/j.neunet.2024.106745
Received 31 January 2024; Received in revised form 1 August 2024; Accepted 15 September 2024
Available online 23 September 2024

Research on multivariate long-term time series forecasting currently faces the following key challenges: (1) Robustness in modeling non-stationarity data (Kim et al., 2021; Zhao et al., 2023): The non-stationarity of time series is manifested by its statistical properties (such as mean and variance) changing over time, so the model must be capable of capturing and adapting to these changes as they change over time. (2) Capture complex temporal dependencies within each variable (Zhang et al., 2022): Each variable in a multivariate time series often exhibits multiple dependencies, including local dependencies reflecting short-term fluctuations and global dependencies revealing long-term trends. It is essential to capture these multiple dependencies effectively to improve forecast accuracy. (3) Capture complex inter-dependencies among variables: Multivariate time series often exhibit complex inter-dependencies among variables. When these inter-dependencies are neglected, it directly impacts the overall accuracy of the multivariate series forecasting. (4) Maintain efficient computation while maintaining forecasting performance (Ma et al., 2024): As the forecasted series lengthens, the model needs to effectively handle long-term dependencies while maintaining efficient computation. In addressing these challenges, the traditional statistical methods, relying on assumptions of data stationarity, often lack precision when addressing dynamically changing time series problems. Furthermore, these methods are inefficient and inflexible when it comes to capturing long-term dependencies and handling large-scale datasets. While deep learning methods have a significant advantage when dealing with non-linearity and complex patterns, time series forecasting research has steadily shifted towards this field. In terms of deep learning methods, Recurrent Neural Networks (RNN) models show potential for capturing internal temporal dependencies, but their efficiency is limited by vanishing gradients and lack of parallelism. Temporal Convolutional Networks (TCN), with deeper structures to expand their receptive fields, attempt to capture long-term dependencies, but they have limited ability to handle complex dependencies within variables. Transformer-based models, with their unique self-attention mechanism, are capable of effectively calculating the relevance of all positions in the input series, demonstrating exceptional ability to model long sequence dependencies. This has garnered widespread attention in long-term series forecasting problems. However, Transformer-based models still have some shortcomings, mainly related to computational complexity and capturing cross-temporal and cross-dimensional dependencies.

Transformer's self-attention mechanism requires calculating the similarity between any two elements, leading to quadratic growth in computational and spatial complexity with the length of the time series. Researchers typically focus on finding ways to reduce computational complexity in order to solve this issue. LogTrans (Zhang et al., 2021), Informer (Zhou et al., 2021), and Pyraformer (Liu et al., 2021) all use sparse attention mechanisms to reduce computational demands. However, these models still perform dot-product attention calculations between each time step and rely on point-wise connections to capture temporal dependencies, resulting in high computational complexity when processing long time series, making their application on large-scale data difficult. Autoformer (Chen et al., 2021) computes self-correlations on top-k time-lagged sequences, and then applies aggregation operations across the entire lagged sequence to uncover inter-sequence dependencies, which requires complex Fourier transformations to calculate sequence correlations. The aforementioned methods typically perform correlation calculations at the point level or at the overall sequence level, leading to a high degree of redundancy in the computational process. Furthermore, the goal of time series forecasting is to understand relationships between different time steps, but unlike words in natural language sentences, individual time steps in a time series often lack explicit semantic meaning. Directly establishing dependencies between time points makes it more challenging to fully reflect the true internal dependencies of the time series (Nie et al., 2022).

Modeling time series with Transformer introduces another critical issue: The Transformer model's global self-attention mechanism lacks local context perception in capturing cross-temporal dependencies. The global self-attention mechanism might overfocus on distant correlations while neglecting equally important closer local dependencies in time series, which can lead to decreased performance in capturing cross-temporal dependencies within a series despite the importance of local contextual dependencies. On the other hand, cross-dimensional dependencies in multivariate long-term time series forecasting are also important, meaning that the changes in one variable of a multivariate time series may be influenced by variables in other dimensions. For example, when predicting future humidity, a solely reliance on historical humidity data is insufficient; wind speed and temperature data must also be considered. However, Transformer-based deep learning models typically capture dependencies between variables by embedding data points from all dimensions at the same time step into a feature vector. Although this approach effectively captures temporal dependencies, it confuses the cross-dimensional dependencies between variables, resulting in Transformer models not functioning well in multivariate long-term time series forecasting.

In summary, existing research methods for multivariate long-term time series forecasting based on Transformer models face several key issues: (1) Standard Transformer models grows quadratically in computational complexity with an increase in the length of the time series to be processed, and directly establishing correlations between time points does not adequately reflect the internal dependencies within the time series. (2) Transformer's global self-attention mechanism has limited sensitivity to local context. (3) There is a lack of ability to distinguish cross-dimensional dependencies among variables. We propose the Multi-Scale Convolution Enhanced Transformer model (MSCformer) for multivariate long-term time series forecasting. The model initially segments the time series into different lengths using a newly designed multi-scale time series segmentation module. This approach not only helps replace the traditional dot-product attention mechanism with a multi-scale segmented attention mechanism in subsequent modules, thereby reducing the quadratic increase in computational complexity associated with the length of the series, but also helps these segments better reflect represent the local characteristics and changing patterns of the time series. This provides a new way to capture the true internal dependencies of the time series, and utilizes the multi-scale nature of the data to enhance predictive performance. The multi-dependence aggregation module of the model combines attention mechanisms and convolutional structures for explicitly modeling both global and local dependencies of individual time series, enhancing the model's capability for feature representation. A cross-dimensional dependency extraction network within this module identifies complex relationships and interactions among various variables, enabling the extraction of global dependency across dimensions. Comparing MSCformer model with state-of-the-art methods, we have validated its effectiveness on numerous benchmarks. The main contributions of this article are as follows:

- Multi-Scale Segmented Attention Mechanism: We propose an attention mechanism based on multi-scale time series segmentation in place of the traditional dot-product attention mechanism. As the segments at multiple scales retain the associative information at the series level, this mechanism effectively reduces computational complexity while better representing the patterns of change in the time series, as well as enhancing forecasting performance by utilizing multi-scale information.
- Local Dependency Compensation Strategy for Cross-Temporal Dependencies: A convolutional structure is introduced as a local dependency compensation network, enhancing the extraction of local features of time series. This offers a new solution to the issue of insufficient local context perception in the Transformer model's global self-attention mechanism.

- Enhanced Modeling of Cross-Dimensional Dependencies: Taking into account the uniqueness of multivariate time series, we have bolstered the modeling of global cross-dimensional dependencies on top of capturing global temporal dependencies. This makes the model more suitable for multivariate long-term time series requiring complex modeling and precise forecasting.

The article is organized as follows. Section 2 reviews some mainstream methods for time series forecasting. Section 3 provides a detailed introduction to the construction of the MSCformer model. Section 4 demonstrates the advantages of the proposed method through extensive experiments on numerous real datasets. Section 5 concludes the work and offers prospects for future research directions.

2. Related work

Time series forecasting techniques, due to their widespread application across various fields, have garnered significant attention, ranging from classic statistical methods to the latest deep learning technologies. Despite significant improvements in the performance of multivariate long-term time series forecasting methods, there remain challenges to be addressed in the current state-of-the-art approaches. Methods for multivariate long-term time series forecasting can generally be categorized into classical statistical methods, methods based on RNN models, TCN models and Transformer models.

Classical statistical methods, such as Autoregressive (AR) models (Yule, 1927), Moving Average (MA) models (Walker, 1931), and Autoregressive Moving Average (ARMA) models (Box George et al., 1976), have gained wide attention due to their simplicity and interpretability. However, these methods often rely on a series of strict assumptions, such as stationarity. Moreover, when dealing with large-scale multivariate long-term time series, they exhibit poor scalability and struggle to effectively capture complex dependencies within the time series, thus limiting their predictive performance on complex data.

In recent years, with the development of deep learning, an increasing number of studies have explored the application of deep learning models in time series forecasting. Recurrent Neural Networks (RNN) (Elman, 1990) and their variants, such as LSTM (Lai et al., 2018; Sutskever et al., 2014) and GRU (Dey & Salem, 2017), rely on their recursive structure to extract effective temporal dependencies from complex real-world data. The DeepAR model (Salinas et al., 2020) utilizes an autoregressive RNN model, effectively addressing the issue of inconsistent time series scales and select likelihood functions based on data feature to predict the probability distribution of time series. Although some works (Qin et al., 2017) have introduced additional attention mechanisms into RNN models to improve their predictive performance, these RNN-based models have not fundamentally solved issues such as vanishing gradients and lack of parallelism. The vanishing gradient problem restricts the RNN models' ability to learn from distant data points in long time series, while the lack of parallelism hinders their application on large-scale data. Furthermore, when capturing complex interdependencies among variables in multivariate data, RNN models face challenges in processing high-dimensional input data.

Given the inherent limitations of recurrent structures that are difficult to fundamentally resolve, researchers have begun exploring novel deep learning approaches to address time series forecasting challenges. The success of Convolutional Neural Networks (CNN) in other domains has prompted researchers to explore their potential in multivariate long-term time series forecasting (Semenoglou et al., 2023; Sen et al., 2019). To adapt to time series datasets, Temporal Convolutional Networks (TCN) (Hewage et al., 2020) have been proposed. TCN-based models typically model from a local perspective, where convolutional kernels effectively extract local information from inputs. By stacking causally dilated convolutional layers with exponentially increasing dilation factors, the receptive field can be expanded to encompass the entire input space, thereby achieving overall information aggregation. Within this framework, MICN (Wang et al., 2022) introduced a multi-scale branching structure employing down-sampling convolutions and equidistant convolutions to better integrate local features and global correlations in time series. SCInet (Liu, Zeng et al., 2022) identified two limitations of dilated convolutions in conventional TCN networks. First, convolutional layers within the same layer utilize the same size convolutional kernels, primarily extracting average temporal features. Second, TCN's intermediate layers have limited effective receptive fields, resulting in the loss of temporal relationships during feature extraction. Therefore, SCInet uses a binary tree structural design to extract time features at various resolutions, incorporating different convolutional filters. Although CNN-based models excel in capturing local dependencies, and many models also focus on expanding the feature extraction range of dilated kernels, they fail to explicitly capture the dependency between two distant locations as a result of the convolution kernel's limited receptive field. It takes a long path to transmit information between two distant locations, which makes it still challenging to capture long-term dependencies, especially in the field of long time series analysis and prediction, where neglecting long-term dependencies may compromise the accuracy of data modeling.

Benefitting from the self-attention mechanism in Transformer models, these models have shown exceptional performance in various fields since (Vaswani et al., 2017) introduced the Transformer architecture in 2017. Unlike previous methods that require multiple layers to obtain global information, the self-attention mechanism in Transformer models enables the calculation of correlations across all input sequences in one go, efficiently capturing global information. In recent years, many studies have attempted to apply Transformer models to multivariate long-term time series forecasting. However, due to the quadratic relationship between the computational complexity of Transformer models and the length of the input time series, much research has been devoted to reducing this complexity. For instance, LogTrans (Zhang et al., 2021) uses a convolutional self-attention layer with a LogSparse design to capture local information, reducing spatial complexity. Informer (Zhou et al., 2021) introduces ProbSparse self-attention and distillation techniques to effectively extract key information. Autoformer (Chen et al., 2021) draws inspiration from traditional time series analysis with decomposition and autocorrelation concepts. FEDformer (Zhou et al., 2022) utilizes Fourier or wavelet transforms for self-attention operations in the frequency domain, reducing the computational complexity of the Transformer model. Pyraformer (Liu et al., 2021) employs a pyramid-style attention module with inter-scale and intra-scale connections, also reducing computational complexity. ETSformer (Woo et al., 2022) replaces the self-attention mechanism with exponential smoothing attention and frequency attention, improving model accuracy and efficiency. These models primarily focus on reducing the computational complexity of Transformer-like models but typically encode input data from all dimensions into a specified dimension when modeling dependencies in time series, overlooking the explicit modeling of cross-dimensional dependencies crucial in multivariate time series forecasting.

Recent studies have focused on cross-dimensional dependencies in time series data. An example like STformer (Xiao et al., 2024) considers cross-dimensional dependencies and is capable of handling multivariate time series. However, because it flatten two-dimensional time series into one-dimensional sequences, it has low computational efficiency and difficulties handling high-dimensional data and long-term forecasting. In contrast, Crossformer (Zhang & Yan, 2023) uses a hierarchical encoder–decoder architecture and a two-stage attention mechanism at different layers to extract cross-temporal and cross-dimensional dependencies. Moreover, TSmixer (Chen et al., 2023) uses a time mixing multilayer perceptron (MLP) for modeling temporal patterns of time series, as well as a feature mixing MLP for modeling covariate information.

Unlike the aforementioned models, MSCformer does not use the traditional encoder–decoder structure for feature extraction, instead using

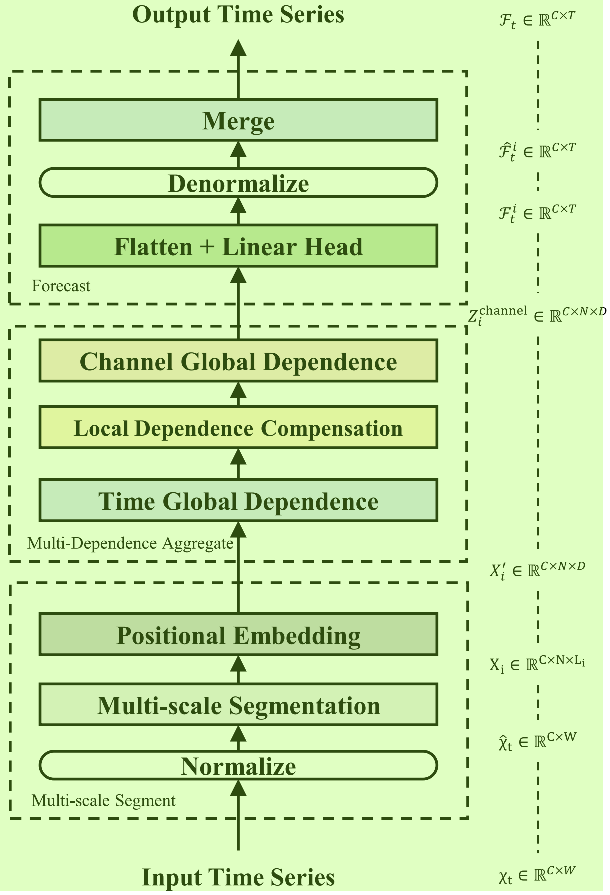

Output Time Series
$\mathcal{F}_t \in \mathbb{R}^{C\times T}$
Merge
$\hat{\mathcal{F}}_t^i \in \mathbb{R}^{C\times T}$
Denormalize
$\mathcal{F}_t^i \in \mathbb{R}^{C\times T}$
Flatten + Linear Head
Forecast
$Z_i^{\text{channel}} \in \mathbb{R}^{C\times N\times D}$
Channel Global Dependence
Local Dependence Compensation
Time Global Dependence
Multi-Dependence Aggregate
$X_i' \in \mathbb{R}^{C\times N\times D}$
Positional Embedding
$X_i \in \mathbb{R}^{C\times N\times L_i}$
Multi-scale Segmentation
$\hat{\chi}_t \in \mathbb{R}^{C\times W}$
Normalize
Multi-scale Segment
Input Time Series
$\chi_t \in \mathbb{R}^{C\times W}$

Fig. 1. The overview of MSCformer.

only an encoder. By introducing a multi-scale segmentation module to change the data input format, MSCformer reduces the computational complexity of traditional Transformer models and intuitively utilizes the multi-scale features of data. Furthermore, considering the advantage of Transformer-like models in capturing global dependencies and their relative insensitivity to local dependencies, we believe that merely using feature embedding operations to capture local dependencies is inadequate. Therefore, our model not only comprehensively considers cross-temporal global dependency and cross-dimensional global dependency, but also adds local dependency, which provides a better solution to the problem of multivariate long time series modeling.

3. Methodology

This section describes the proposed MSCformer in detail. MSCformer's overall structure is shown in Fig. 1. In the framework, three main modules are included: multi-scale segmentation module, multi-dependence aggregation module, and output forecasting module. MSCformer segments time series as input to the Transformer model using a newly designed multi-scale segmentation module. Our multi-scale segmentation strategy not only addresses the problem of manually selecting segment lengths but also provides multiple scale signals for subsequent modules to extract dependencies at various granularities. Moreover, since the number of segments is significantly smaller than the number of original time series points, this segmentation strategy effectively reduces the computational complexity for subsequent modules. Multi-dependence aggregation module is composed of a Time Global Dependence network (TGD), a Local Dependence Compensation network (LDC) and a Channel Global Dependence network (CGD). The module comprehensively captures global and local dependencies across time, as well as global dependencies across dimensions, thus greatly enhancing the model's ability to understand the internal dynamics of time series and capture interactions between variables. The output forecasting module of the model consolidates information modeled at different scales to produce final forecasting. Detailed descriptions of each part of the model will be provided in the following sections.

3.1. Problem formulation

The core objective of time series forecasting is to predict future values based on known historical observations. Specifically, given a set of historical time series observations $\chi_t \in \mathbb{R}^{C\times W}$, where W denotes the size of the look-back window and C represents the dimensions of the time series (the number of features), the goal is to predict the values of C variables in the time series for the next T time steps, which can be expressed mathematically as follows:

$$\mathcal{F}_t = F\left(\chi_t\right) \in \mathbb{R}^{C\times T} \tag{1}$$

where $\mathcal{F}_t$ represents the time series to be predicted, T denotes the length of the time series to be forecast, and when the dimension of the time series $C > 1$, $F(\cdot)$ signifies a multivariate time series forecasting model. The task of a multivariate time series forecasting model is to use past variables to predict future values of the same dimensional variables. In scenarios where the number of time steps T to be forecasted is significantly greater than the size of the look-back window W, $F(\cdot)$ is defined as a model for multivariate long-term time series forecasting.

In the context of multivariate time series forecasting, data dependency analysis from different perspectives and scales is required. This paper defines these as cross-temporal global dependences, local dependences and cross-dimensional global dependences specifically. Cross-temporal global dependencies Z_i^{time} reflect the overall trends and cyclical fluctuations of the time series at a specific scale, used to reveal the data's long-period patterns and global fluctuation trends; local dependencies Z_i^{local} reflect a detailed analysis of short-term patterns within the time series at a specific scale, aimed at reveal responding sensitivity to sudden events or short-term changes; cross-dimensional global dependencies $Z_i^{channel}$ demonstrate the interactions and dependencies between different features at a specific scale, focusing on the synergistic effects among various variables. The extraction of the model's dependencies can be represented as:

$$\begin{aligned} Z_i^{\text{time}} &= f\left(\chi_t\right) \\ Z_i^{\text{local}} &= g\left(\chi_t\right) \\ Z_i^{\text{channel}} &= w\left(\chi_t\right) \end{aligned} \tag{2}$$

where χ_t represents the input data of the model, and $f(\cdot)$, $g(\cdot)$, $w(\cdot)$ are defined as the respective dependency extraction modules. The goal of the multivariate long-time series prediction model is to capture these dependencies from the data and reconstruct future sequences based on them.

3.2. The multi-scale segmentation module

A time series' statistical properties, such as mean and variance, often change over time. It is known as the distribution shift effect in time series, and implies that the data distribution may change significantly at different time points or periods. Those distribution shifts often make it challenging for models to capture features that adapt to the current data, thereby affecting forecasting performance. This adaptability issue is also one of the most challenging aspects of multivariate time series forecasting. To address this problem, normalization techniques are commonly used to process input data, employing the mean and variance of each variable in a multivariate series to fix the distribution of the model's input data, thus reducing model learning difficulty. Recently, this normalization method has been proven to effectively

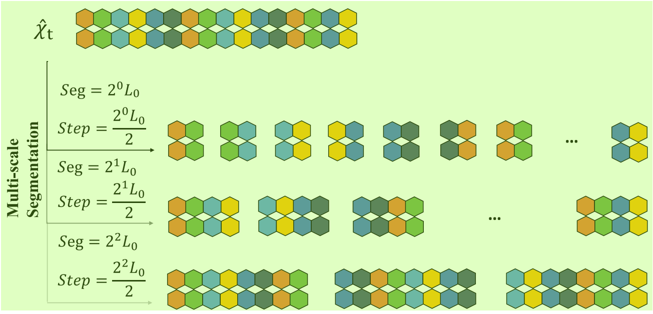


Fig. 2. The detail of multi-scale segmentation (Suppose that $L_0 = 2, W = 16, C = 2$).

enhance forecasting accuracy (Kim et al., 2021). The normalization formula is based on the following:

$$\hat{\chi}_t = \left(\chi_t - E\left[\chi_t\right]\right) / \sqrt{\operatorname{Var}\left[\chi_t\right] + \epsilon} \tag{3}$$

where $\hat{\chi}_t$ represents the time series after normalization, while χ_t is the original time series. $E\left[\chi_t\right]$ and $\operatorname{Var}\left[\chi_t\right]$ denote the mean and variance of each variable in the input data χ_t. However, normalization operations can cause varying degrees of inconsistency in data metrics within the model during the learning process. This inconsistency prevents the model from capturing the original distribution, which can adversely affect predictive performance. To address this issue, a strategy of denormalization at the output is considered. This approach re-injects the information removed during the normalization process back into the data, ensuring that the metrics of the data at the output are consistent with those of the original data and the model does not have to reconstruct the original distribution itself while keeping the advantage of normalizing the input. Based on Eq. (3), the denormalization formula is as follows:

$$\hat{\mathcal{F}}_t^i = \left(\sqrt{\operatorname{Var}\left[\chi_t\right] + \epsilon}\right) \mathcal{F}_t^i + E\left[\chi_t\right] \tag{4}$$

where $E\left[\chi_t\right]$ and $\operatorname{Var}\left[\chi_t\right]$ represent the mean and variance of each variable in the original data calculated at the input end. $\mathcal{F}_t^i$ is the predictive result generated by the model at each scale, and $\hat{\mathcal{F}}_t^i$ denotes the denormalized model input forecasting result. Using Eqs. (3) and (4) can effectively mitigate the issue of inconsistency in the distribution metrics of the time series.

Subsequent operations are based on the normalized time series $\hat{\chi}_t$. Transformer has demonstrated outstanding long-term dependency modeling capabilities in numerous fields. While when processing time series, it faces the problem of high computational complexity. Thus, it is necessary to find a method that both leverages the Transformer model's advantages and reduces its computational complexity effectively.

The divide-and-conquer approach is an effective strategy for reducing complexity, thus sequence segmentation is introduced. In the multi-scale segmentation module, the input time series points are transformed into segments. Fig. 2 illustrates the specific segmentation process. Starting from a smaller initial segment length L_0, the segment length increases exponentially. The formula is given as:

$$L_i = 2^i L_0 \tag{5}$$

where $i \in \{0, 1, \dots, l_{max}\}$, $l_{\max} = \lfloor \log_2 \left(W / L_0\right) - 1 \rfloor$ and W is the size of the time series look-back window. Using this segment length, the original time series points are divided into segments of varying lengths, with a step length of $L_i / 2$ for each segment. Consequently, the normalized time series matrix $\hat{\chi}_t \in \mathbb{R}^{C \times W}$ is unfolded into an input X^i with segment structures using Eq. (6).

$$X_i = \text{ Multi-Scale Segmentation } \left(\hat{\chi}_t\right), X^i \in \mathbb{R}^{C \times N \times L_i} \tag{6}$$

where L_i denotes the segment length at each scale, and $N = \lfloor 2W / L_i - 1 \rfloor$ is the number of segments for the corresponding scale. When the look-back window W cannot be evenly divided by L_i, in order to ensure uniformity of segments, the last value within the look-back window is replicated $L_i / 2$ times and appended to the end of the original series, ensuring all segments are uniform.

Through this segmentation process, the original time series is transformed into several segments of length L_i. This segmentation strategy significantly optimizes the feature extraction performance of the subsequent multi-dependence aggregate module. On the one hand, as the number of segments is far less than the number of original time series points, the number of input tokens is reduced from W in the dot-product attention mechanism to W / L_i, significantly reducing the computational burden of the subsequent Transformer model. Memory consumption and computational complexity for calculating the attention map have been significantly reduced. Further, by using segmented structures as inputs to the model, each segment retains semantic information at a sequence level, allowing adjacent time points with similar features to be aggregated effectively, better reflecting the true internal dependencies of the series. Simultaneously, the length of the time series segments determines the granularity of the segment attention mechanism. A larger segment length helps capture coarse-grained temporal dependencies in the time series, whereas a smaller segment length focuses on capturing fine-grained dependencies. In this way, the multi-scale segment attention mechanism maintains low complexity while minimizing the impact of hyperparameter segment length selection on model performance, and integrates multi-scale information to achieve accurate forecasting.

Subsequently, a linear layer $W_{mlp} \in \mathbb{R}^{L_i \times D}$ projects these segments $X_i \in \mathbb{R}^{C \times N \times L_i}$ into a latent space of model dimension D and a learnable positional encoding $W_{pos} \in \mathbb{R}^{C \times N \times D}$ is applied to compensate for temporal information within the time series, resulting in $X_i' \in \mathbb{R}^{C \times N \times D}$, which serves as the input fed into the subsequent model. The data transformation process is specifically expressed as follows:

$$X_i' = X_i W_{mlp} + W_{pos} \tag{7}$$

3.3. Multi-dependence aggregate module

To address the limitations of existing methods in modeling multivariate long-term time series, particularly their lack of focus on local and cross-dimensional information, this section introduces the Multi-Dependence Aggregate module (MDA). The constructed Time Global Dependence network (TGD) is used to extract cross-temporal global dependencies. Building upon TGD, a Local Dependence Compensation network (LDC) is introduced to compensate for the local dependency information within time series segments. The global dependency extraction by TGD and the local dependency compensation by LDC work in tandem, allowing the model to simultaneously capture fine-grained local changes and broad global trends in the time series. Additionally, to further enhance the model's understanding of interactions between different variables, a Channel Global Dependence network (CGD) has been developed to extract cross-dimensional global dependencies, thereby significantly improving the accuracy and efficiency of multivariate long-time series predictions. The information extraction process can be represented as:

$$\begin{aligned} Z_i^{\text{time}} &= TGD\left(X_i'\right) \\ Z_i^{\text{local}} &= LDC\left(Z_i^{\text{time}}\right) \\ Z_i^{\text{channel}} &= CGD\left(Z_i^{\text{local}}\right) \end{aligned}$$

where X_i' is the data after segmentation processing and dimensional embedding. The model's information processing flow adopts a sequential processing strategy to progressively deepen its data analysis capability, with Z_i^{time}, Z_i^{local} and $Z_i^{channel}$ representing the extracted cross-temporal global dependencies, local dependencies and cross-dimensional global dependencies, respectively. Compared to multimodal fusion methods, the sequential processing structure offers richer interpretability.

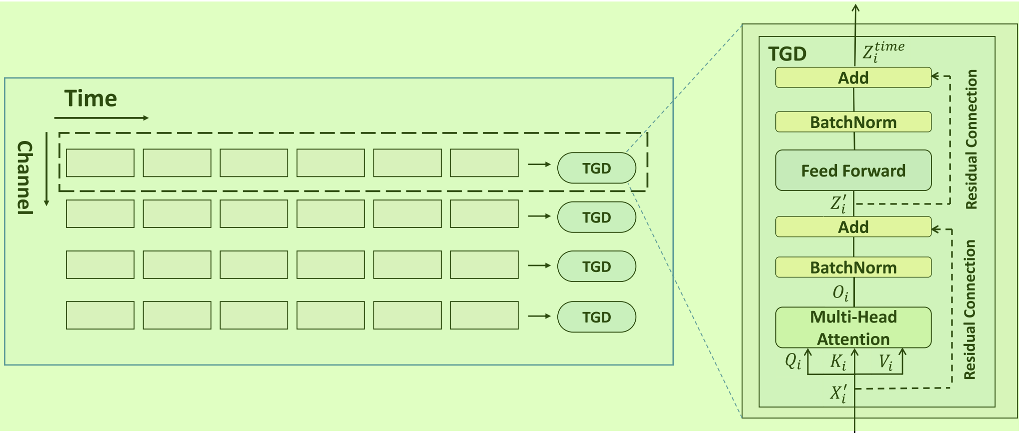


Fig. 3. The detail of Time Global Dependence.

3.3.1. *Time Global Dependence extraction network*

Multivariate time series exhibit unique features and dynamics at different time points. Capturing cross-temporal dependencies means better extraction of the data's evolving patterns over time, thereby deeply understanding the essence and dynamic characteristics of the data. Convolutional networks, fully connected networks, and other methods were used in early research to capture long-term dependencies in multivariate time series. In CNN-based methods, however, the size of the convolutional kernel is usually much smaller than the input data length. Therefore, many convolutional layers must be stacked consecutively to aggregate the overall information, but it is uncertain whether continuous stacking can completely extract long-term dependencies is uncertain, which limits the model's ability to model long-term dependencies. In contrast, simple linear models, such as fully connected layers, struggle to capture complex long-term dependencies in time series due to their limited expressive power. Due to these limitations, the aforementioned models are unable to fully understand and predict the dynamic features of multivariate time series. In comparison, Transformer's attention mechanism considers the feature representation of each time step in the time series comprehensively and correlates it with features of other time steps, effectively mining global dependencies across time.

In contrast to conventional attention mechanism applied directly to data points, we propose using attention mechanisms on time series segments divided into different scales by the multi-scale segmentation module. In addition to the previously mentioned reduction in computational complexity, this approach also makes time series modeling more explanatory.

Specifically, each time series segment obtained in Section 3.2 is treated as data within a time window, corresponding to patterns or relationships within that period of time. It is easier to associate with actual events or phenomena. For example, in weather forecasting, each segment corresponds to data over a period of time, making the model's forecasting more easily linked to actual weather conditions; in stock market forecasting, each segment corresponds to market dynamics over a period of time, preserving the temporal information of the original data. Based on this, we constructed the Time Global Dependence extraction network (TGD), whose structure is shown in Fig. 3. TGD encodes time series segments using a self-attention mechanism. First, the query matrix Q_i, key matrix K_i, value matrix V_i of the output X_i' of the multi-scale segmentation module are generated through linear projection, where Q_i, K_i, V_i $\in \mathbb{R}^{C\times N\times D}$, F_q, F_k, F_v are all linear layers. When performing linear transformation to obtain Q_i, K_i, V_i, Transformer introduces learnable parameters with a well-defined structure. This significantly enhances the network's learning capability and enables the model to capture contextual information more comprehensively.

$$Q_i = F_q\left(X_i'\right), \quad K_i = F_k\left(X_i'\right), \quad V_i = F_v\left(X_i'\right) \tag{8}$$

Subsequently, according to the principles of multi-head attention, Q_i, K_i, V_i are partitioned to generate $q_i^{(j)}$, $k_i^{(j)}$, $v_i^{(j)}$ $\in \mathbb{R}^{C\times N\times d_{\text{head}}}$, where $j = 1, 2, \ldots, H$, H and d_{head} respectively represent the number of heads and the dimension of each head in the multi-head attention mechanism.

$$q_i^{(j)}, k_i^{(j)}, v_i^{(j)} = \text{Multi–Head}\left(Q_i, K_i, V_i\right) \tag{9}$$

Furthermore, as multivariate time series comprise multiple time series variables, each with its own independent cross-temporal dependencies, to independently model the dependencies of time series in the temporal dimension, a variable independence setting named Channel-Independent (CI) is adopted. This transform $q_i^{(j)}$, $k_i^{(j)}$, $v_i^{(j)}$ into $q_i^{(j),(c)}$, $k_i^{(j),(c)}$, $v_i^{(j),(c)}$ $\in \mathbb{R}^{N\times d_{head}}$, for $j = 1, 2, \ldots, H$, $c = 1, 2, \ldots, C$, where each input of the Transformer model comes solely from the information of a single variable.

$$q_i^{(j),(c)}, k_i^{(j),(c)}, v_i^{(j),(c)} = \text{CI}\left(q_i^{(j)}, k_i^{(j)}, v_i^{(j)}\right) \tag{10}$$

Next, $q_i^{(j),(c)}$, $k_i^{(j),(c)}$, $v_i^{(j),(c)}$ are used for the calculation of the attention mechanism. Multiplying $q_i^{(j),(c)}$, $k_i^{(j),(c)}$ and scaling by d_{head} produces the attention distribution $E_i^{(j),(c)}$ for the segments of the time series. Then, the attention distribution is normalized using the softmax function.

Finally, the processed attention distribution is multiplied by the corresponding $v_i^{(j),(c)}$ to obtain the output of the attention mechanism $\text{Attn}_i^{(j),(c)} \in \mathbb{R}^{N\times d_{\text{head}}}$. The outputs of all heads and variables are then concatenated to form the final output of the multi-head attention $O_i \in \mathbb{R}^{C\times N\times D}$. The calculation process of the multi-head attention mechanism is summarized in Fig. 4, and the data transmission process is expressed as follows:

$$\begin{gathered}
E_i^{(j),(c)} = \left(q_i^{(j),(c)}\right)\left(k_i^{(j),(c)}\right)^T / \sqrt{d_{\text{head}}} \\
\text{Attn}_i^{(j),(c)} = \text{Softmax}\left(E_{(i),(c)}\right) v_i^{(j),(c)} \\
O_i^{(c)} = \text{Concat}\left(\text{Attn}_i^{1,(c)}, \text{Attn}_i^{2,(c)}, \ldots, \text{Attn}_i^{H,(c)}\right) \\
O_i = \text{Concat}\left(O_i^1, O_i^2, \ldots, O_i^C\right)
\end{gathered} \tag{11}$$

The Time Global Dependence extraction network also includes a BatchNorm layer and a feed-forward network with residual connections as shown in Eq. (12). By doing so, vanishing gradients are alleviated and a faster training process is possible, leading to deeper network architectures and reduced overfitting. The output of the cross-temporal dependence capture module at each scale is defined as Z_i^{time}. Therefore, the final output form of the Time Global Dependence extraction network is $Z_i \in \mathbb{R}^{C\times N\times D}$.

$$\begin{gathered}
Z_i' = \text{BatchNorm}\left(O_i\right) + X_i' \\
Z_i^{time} = \text{BatchNorm}\left(\text{FFN}\left(Z_i'\right)\right) + Z_i'
\end{gathered} \tag{12}$$

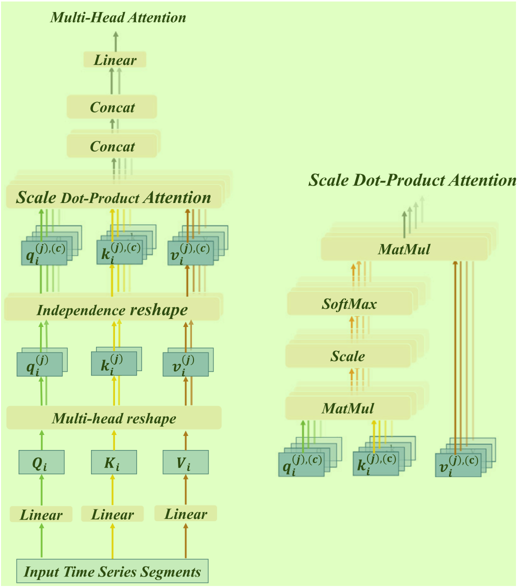


Fig. 4. Algorithm flow of the multi-head attention mechanism.

3.3.2. Local dependence compensation network

To compensate for the limitations of self-attention mechanism in focusing on local dependencies, multi-dependence aggregate module introduces the Local Dependence Compensate (LDC) network within segments to extract local dependency characteristics of each segment. Since convolution operations effectively capture local correlations, the LDC employs a convolutional structure to build this network, as illustrated in Fig. 5.

The convolution module consists of a gating mechanism composed of Pointwise Convolution and GLU (Gated Linear Unit). Pointwise Convolution, by using a convolution kernel for dimension expansion, enhances the model's representational capability, enabling it to capture complex features more effectively. The GLU introduces a gating mechanism that allows the network to selectively focus on and learn important features in the input while suppressing unimportant ones. Through this mechanism, the model is able to better understand local features and correlations in the data, improving its efficiency, representation capability, and non-linear modeling capability, while reducing the impact of redundant data. Depthwise Convolution is then introduced, a process similar to regular convolution, for extracting local features. LayerNorm and BatchNorm are also incorporated to facilitate training, and residual connections are used to accelerate training. With gating mechanism (GLU), pointwise convolution, and depth-wise convolution, and regularization and residual connections added, a network structure is constructed that efficiently captures local dependencies. Using this structure, the model's complexity and computational cost are reduced while its performance is enhanced. The computation process can be represented as follows:

$$\begin{aligned}
Z_i^1 &= \mathrm{LayerNorm}\left(Z_i^{time}\right)\\
Z_i^2 &= \mathrm{GLU}\left(\mathrm{PointwiseConvolution}\left(Z_i^1\right)\right)\\
Z_i^3 &= \mathrm{DepthwiseConvolution}\left(Z_i^2\right)\\
Z_i^4 &= \mathrm{Swish}\left(\mathrm{BatchNorm}\left(Z_i^3\right)\right)\\
Z_i^{local} &= \mathrm{PointwiseConvolution}\left(Z_i^4\right) + Z_i^{time}
\end{aligned} \tag{13}$$

GLU and Swish are two different activation functions. As shown in Eq. (14), the GLU activation function first splits the input vector x into two parts: x_1 and x_2, with a gating mechanism selectively focusing on one part. This process is achieved by mapping each element in x_1 through a Sigmoid function to a range between 0 and 1, and then multiplying it by x_2. This mechanism allows the network to selectively focus on different parts of the input, helping to dynamically and adaptively weight different parts of the input.

$$\mathrm{GLU} = \mathrm{Sigmoid}\left(x_1\right) * x_2 \tag{14}$$

The Swish activation function consists simply of an input vector x and a sigmoid function, defined as follows:

$$\mathrm{Swish}(x) = x * \mathrm{Sigmoid}(x) \tag{15}$$

3.3.3. Channel global dependence extraction network

In multivariate time series forecasting, it is crucial to understand not only the historical sequence features within each variable, but to also consider the inter-variable correlations to fully capture their dependencies, thereby enhancing forecasting accuracy. The Channel Global Dependence network (CGD) applies a self-attention mechanism in the variable domain C to obtain cross-dimensional dependencies between different variables, as shown in Fig. 6.

The computation process of CGD first involves using linear projections to generate linear projection matrices based on the output of the Local Dependence Compensate network Z_i^{local}, and then splitting them according to the principles of multi-head attention to generate $q_i^{(j)}$, $k_i^{(j)}$, $v_i^{(j)} \in \mathbb{R}^{C\times N\times d_{\text{head}}}$.

$$\begin{gathered}
Q_i = F_q\left(Z_i^{local}\right), \quad K_i = F_k\left(Z_i^{local}\right), \quad V_i = F_v\left(Z_i^{local}\right)\\
q_i^{(j)}, k_i^{(j)}, v_i^{(j)} = \mathrm{Multi{-}Head}\left(Q_i, K_i, V_i\right)
\end{gathered} \tag{16}$$

In processing the input of the Transformer model, unlike the previous approach that focused on information from a single variable, here each input comes from specific time segments of each variable. An independence setting named Time-Independent (TI) is adopted, which transforms $q_i^{(j)}$, $k_i^{(j)}$, $v_i^{(j)}$ into $q_i^{(j),(n)}$, $k_i^{(j),(n)}$, $v_i^{(j),(n)} \in \mathbb{R}^{C\times d_{\text{head}}}$:

$$q_i^{(j),(n)}, k_i^{(j),(n)}, v_i^{(j),(n)} = \mathrm{TI}\left(q_i^{(j)}, k_i^{(j)}, v_i^{(j)}\right) \tag{17}$$

Multiplying $q_i^{(j),(n)}$ and $k_i^{(j),(n)}$ generates the attention distribution for time series segments, scaled by the factor d_{head}. This attention distribution is then normalized using the softmax function. Subsequently, it is multiplied by the corresponding $v_i^{(j),(n)}$, yielding the output $\mathrm{Attn}_i^{(j),(n)} \in \mathbb{R}^{C\times d_{\text{head}}}$:

$$\begin{aligned}
E_i^{(j),(n)} &= \left(q_i^{(j),(n)}\right)\left(k_i^{(j),(n)}\right)^T / \sqrt{d_{\text{head}}}\\
\mathrm{Attn}_i^{(j),(n)} &= \mathrm{Softmax}\left(E_{(i),(n)}\right) v_i^{(j),(n)}
\end{aligned} \tag{18}$$

The outputs are concatenated as the output of multi-head attention $O_i \in \mathbb{R}^{C\times N\times D}$. The output of O_i after passing through a BatchNorm layer and a feed-forward network with residual connections, serves as the final output of the module $Z_i \in \mathbb{R}^{C\times N\times D}$. The computation process can be represented as follows.

$$\begin{aligned}
O_i^{(n)} &= \mathrm{Concat}\left(\mathrm{Attn}_i^{1,(n)}, \mathrm{Attn}_i^{2,(n)}, \ldots, \mathrm{Attn}_i^{H,(n)}\right)\\
O_i &= \mathrm{Concat}\left(O_i^1, O_i^2, \ldots, O_i^n\right)\\
Z_i' &= \mathrm{BatchNorm}\left(O_i\right) + Z_i^{local}\\
Z_i^{channel} &= \mathrm{BatchNorm}\left(\mathrm{FFN}\left(Z_i'\right)\right) + Z_i'
\end{aligned} \tag{19}$$

With its two global dependence extraction networks, the multi-dependence aggregate module effectively captures the dependencies in two key dimensions of multivariate time series: cross-temporal dependence for a single variable and cross-dimensional dependence for different variables. This comprehensive analysis of dependencies enables a better understanding of time series' integrated characteristics. The cross-temporal dependence analysis reveals the trends and behavior patterns of a single variable over time, which are crucial for predicting future trajectories. The cross-dimensional dependence analysis, on the other hand, reveals the interactions and influences between variables. Further, by combining these global dependencies with local dependence networks, multivariate time series forecasting models are provided with more detailed and accurate. In this way, key characteristics of multivariate time series can be uncovered, improving the forecasting accuracy and model robustness.

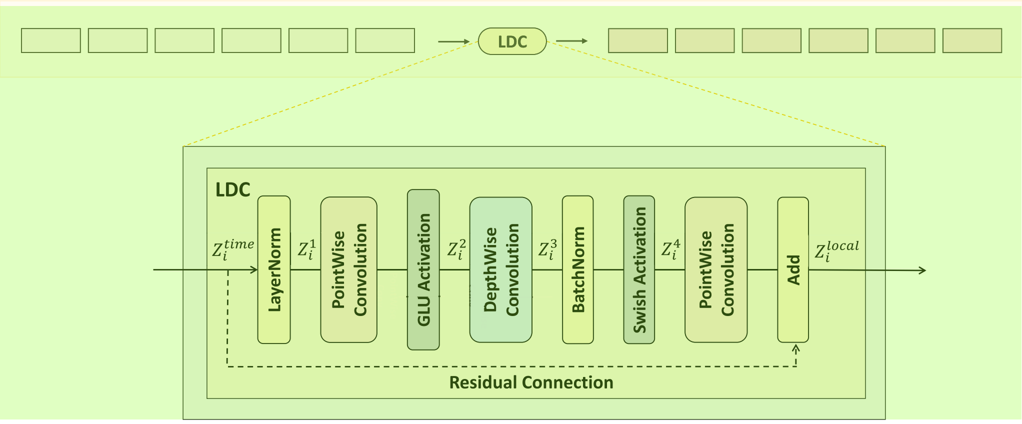


Fig. 5. The detail of Local Dependence Compensation.

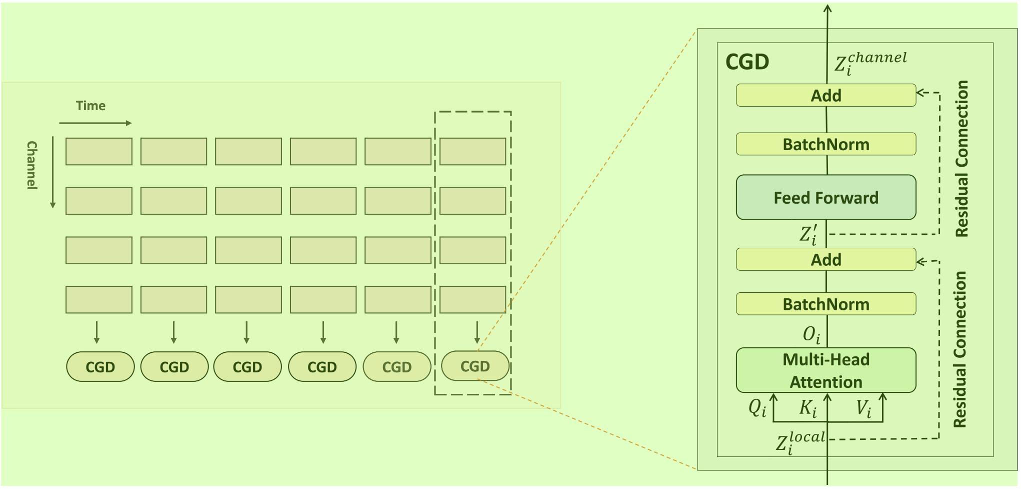


Fig. 6. The detail of Channel Global Dependence.

3.4. The output forecasting module

The MSCformer model employs a multi-scale segmentation module, enabling it to consider dependencies across various time scales and better adapt to the diversity of time series. In the output forecasting module, it follows a simple yet effective principle to fuse dependencies at various scales to reconstruct future sequences. Specifically, it first uses a Flatten layer to transform $Z_i^{channel} \in \mathbb{R}^{C\times N\times D}$ into $Z_i^{channel} \in \mathbb{R}^{C\times N*D}$, and then a single linear layer is used instead of a traditional decoder to generate forecasting results at each scale $\mathcal{F}_t^i \in \mathbb{R}^{C\times T}$.

$$\mathcal{F}_t^i = \text{Linear}\left(\text{Flatten}\left(Z_i^{channel}\right)\right) \tag{20}$$

Subsequently, denormalization is applied at the $\mathcal{F}_t^i$, yielding denormalized model input forecasting results $\hat{F}_t^i \in \mathbb{R}^{C\times T}$.

$$\hat{\mathcal{F}}_t^i = \left(\sqrt{\text{Var}\left[\chi_t\right]+\epsilon}\right)\mathcal{F}_t^i + \text{E}\left[\chi_t\right] \tag{21}$$

Here, $E\left[\chi_t\right]$ and $\text{Var}\left[\chi_t\right]$ represent the mean and variance of the original data calculated at the input, with $\hat{\mathcal{F}}_t^i \in \mathbb{R}^{C\times T}$ being the forecasting result produced by the model at each scale.

The model employs four different methods for aggregating outputs across multiple scales, including equal weight aggregation, exponential decay weight aggregation, learnable weight aggregation, and weighted average aggregation. The equal weight aggregation assigns the same weight to dependencies at different scales, emphasizing the contribution of each scale to provide a more comprehensive forecast; exponential decay weight aggregation assigns greater weight to information at lower scales, emphasizing the importance of lower-scale information in prediction and enhancing the model's sensitivity to specific scale information; the learnable weight aggregation approach defines learnable weights for outputs at each scale, which are updated through network training, offering the flexibility to automatically adjust weights to adapt to data across different scales, thus emphasizing data-driven weight optimization; and weighted average aggregation assigns weights based on the segment length at each scale, balancing the global perspective from higher scales with the detailed feedback from lower scales. The specific formulas for these four aggregation methods are as follows:

Equal weight aggregation:

$$\begin{gathered}\mathcal{F}_t = \text{Avg}\left(\hat{\mathcal{F}}_t^i\right) = \alpha_0\hat{\mathcal{F}}_t^0 + \alpha_1\hat{\mathcal{F}}_t^1 + \cdots + \alpha_{l_{\max}}\hat{\mathcal{F}}_t^{l_{\max}} \\ \alpha_0 = \alpha_1 = \cdots = \alpha_{l_{\max}} = 1/\left(l_{\max}+1\right)\end{gathered} \tag{22}$$

Exponential decay weight aggregation:

$$\begin{gathered}\mathcal{F}_t = \text{Exp–Decay}\left(\hat{\mathcal{F}}_t^i\right) = \alpha_0\hat{\mathcal{F}}_t^0 + \alpha_1\hat{\mathcal{F}}_t^1 + \cdots + \alpha_{l_{\max}}\hat{\mathcal{F}}_t^{l_{\max}} \\ \alpha_0 = 2^0/\sum_{l=0}^{l_{\max}} 2^l, \alpha_1 = 2^1/\sum_{l=0}^{l_{\max}} 2^l, \ldots, \alpha_{l_{\max}} = 2^{l_{\max}}/\sum_{l=0}^{l_{\max}} 2^l\end{gathered} \tag{23}$$

Learnable weight aggregation:

$$\mathcal{F}_t = \text{Learnable}\left(\hat{\mathcal{F}}_t^i\right) = \alpha_0\hat{\mathcal{F}}_t^0 + \alpha_1\hat{\mathcal{F}}_t^1 + \cdots + \alpha_{l_{\max}}\hat{\mathcal{F}}_t^{l_{\max}} \tag{24}$$

where $\alpha_0 \ldots \alpha_i$ are all learnable weight factors.

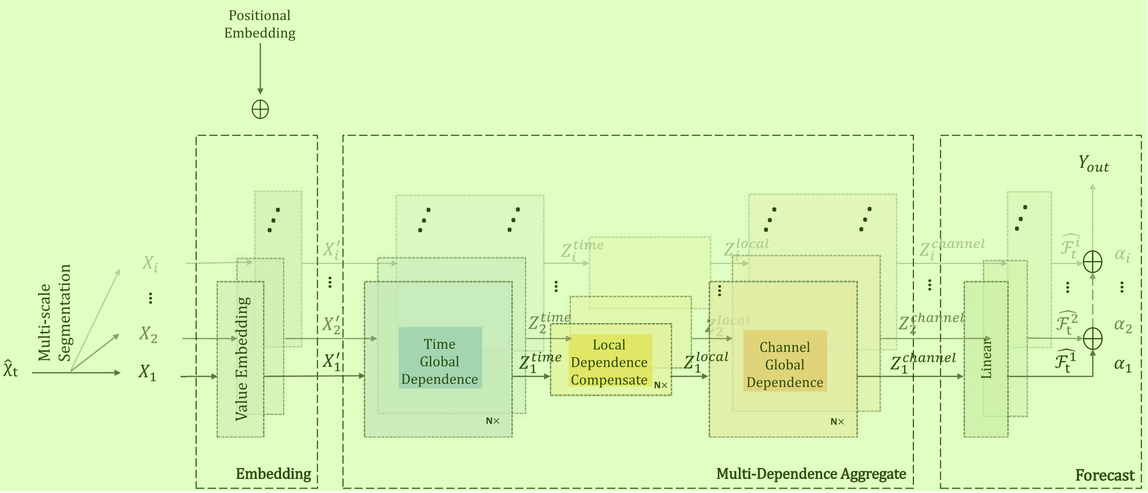


Fig. 7. The detail of MSCformer.

Weighted average aggregation:

$$\mathcal{F}_t = \text{Weighted}-\text{Avg}\left(\hat{\mathcal{F}}_t^{i}\right) = \alpha_0\hat{\mathcal{F}}_t^0 + \alpha_1\hat{\mathcal{F}}_t^1 + \cdots + \alpha_{l_{\max}}\hat{\mathcal{F}}_t^{l_{\max}}$$
$$\alpha_0 = \frac{\mathrm{L}_0}{\sum_{i=0}^{l_{\max}}\mathrm{L}_\mathrm{i}}, \alpha_1 = \frac{\mathrm{L}_1}{\sum_{i=0}^{l_{\max}}\mathrm{L}_\mathrm{i}}, \ldots, \alpha_{l_{\max}} = \frac{\mathrm{L}_{\max}}{\sum_{i=0}^{l_{\max}}\mathrm{L}_\mathrm{i}} \tag{25}$$

$\mathcal{F}_t \in \mathbb{R}^{C\times T}$ is the final predictive output of the model. This multi-scale information aggregation strategy enhances MSCformer's forecasting accuracy and robustness, making it highly effective in multivariate long-term time series forecasting. The overall framework of the model is illustrated in Fig. 7.

3.5. Algorithm flow

The following section provides a detailed description of the entire framework of MSCformer. To summarize the model process, Algorithm 1 constructs an algorithmic flowchart, concisely outlining the MSCformer method's workflow.

4. Experiment

To validate the predictive performance of the proposed MSCformer, extensive experiments were conducted on six real-world multivariate datasets, with a detailed analysis of the results. This section first introduces the experimental setup, datasets, evaluation metrics, and benchmark comparison algorithms used in the experiments. The proposed method is then validated across multiple experiments.

4.1. Experimental settings

MSCformer was implemented using the Pytorch deep learning framework and trained on an Nvidia RTX 4090 GPU (24 GB). To enhance reproducibility of results, random seeds were fixed. By default, MSCformer comprises 3 encoder layers, with the number of heads (H) set to 16 and the model hidden dimension (D) to 128. The feedforward network consists of two linear layers with Swish activation functions: one layer projects the hidden representation $D = 128$ to a new dimension $F = 256$, and the other projects it back to $D = 128$. For particularly small datasets like Illness and Exchange-rate, reduced-size parameters ($H = 4$, $D = 16$, $F = 128$, $Dropout = 0.3$) were used to mitigate potential overfitting. All experiments, except on the Illness and Exchange-rate datasets, employed a Dropout rate of 0.2 in the encoder and used Mean Squared Error (MSE) as the loss function. An Adam optimizer was used, with other parameter settings consistent with most comparison methods.

Algorithm 1 MSCformer

```
Input:
    Input multivariate time series $\chi_t \in \mathbb{R}^{C\times W}$, initial segment length $L_0$, aggregate weight $\alpha_0, \alpha_1, \ldots, \alpha_{l_{\max}}$
Output:
    Output multivariate time series $\mathcal{F}_t \in \mathbb{R}^{C\times T}$
Normalize the original time series points with $\hat{\chi}_t = \chi_t - E[\chi_t] / \sqrt{\mathrm{Var}[\chi_t] + \epsilon}$.
$X_i = \text{Multi-scale Segment}(\hat{\chi}_t)$. Transform input time series points into segments.
for $i$ in $\{0, \ldots, \log_2(W/L_0) - 1\}$ do
    $X'_i = X_i W_{mlp} + W_{pos}$. Perform position embedding and input embedding.
    $Z_i^{time} = \mathrm{TGD}(X'_i)$. Capture cross-temporal dependencies of the multivariate time series.
    $Z_i^{local} = \mathrm{LDC}(Z_i^{time})$. Compensate for local dependencies.
    $Z_i^{local} = \mathrm{Transpose}(Z_i^{local})$. Transpose $Z_i^{local}$ for subsequent cross-dimensional dependency extraction.
    $Z_i^{channel} = \mathrm{CGD}(Z_i^{local})$. Capture cross-dimensional dependencies among variables of the multivariate time series.
    $\mathcal{F}_t^i = \mathrm{Linear}(Z_i^{channel})$. Obtain forecasting results.
    $\widehat{\mathcal{F}_t^i} = \left(\sqrt{\mathrm{Var}[\chi_t] + \epsilon}\right)\mathcal{F}_t^i + E[\chi_t]$. Denormalize the forecasting results.
end for
$\mathcal{F}_t = \alpha_0\mathcal{F}_t^0 + \alpha_1\mathcal{F}_t^1 + \cdots + \alpha_{l_{\max}}\mathcal{F}_t^i$. Aggregate results across different scales to form the final forecasting
return $\mathcal{F}_t$
Use Mean Squared Error (MSE) to compute the loss function
Update parameters through gradient descent.
```

4.2. Datasets

To validate the robustness of MSCformer's predictive performance, six popular multivariate datasets Table 1 were selected for experiments in multivariate time series forecasting. The datasets used in the experiments include (1) Electricity[1]: recording hourly electricity consumption of 321 clients from 2012–2014. (2) Traffic[2]: occupancy

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

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

Table 1
Datasets used for multivariate time series forecasting task.

Dataset	Length	Variates	Frequency
Electricity	26 304	321	1 h
Traffic	17 544	862	1 h
Illness	966	7	1 week
ETT-h	17 420	7	1 h
Weather	52 696	21	10 min
Exchange-rate	7588	8	1 day

rates on San Francisco freeways recorded by sensors from January 2015 to December 2016. (3) Illness[3]: patient counts and influenza-like illness ratios recorded weekly by the US Centers for Disease Control and Prevention from 2002 to 2021. (4) ETT-h (Zhou et al., 2021): load and oil temperature data of a power transformer in China, recorded from July 2016 to July 2018. (5) Weather[4]: 21 meteorological indicators, such as humidity and temperature, collected every 10 min in 2020 by the Max Planck Institute for Biogeochemistry in Germany. (6) Exchange-rate (Lai et al., 2018): daily exchange rate records from eight countries from 1990 to 2016. The preprocessing scheme for datasets was consistent with previous studies (Zeng et al., 2023; Zhou et al., 2022, 2021). Specifically, the ETT-h dataset was divided into training/validation/test sets in a 6:2:2 ratio. In contrast, other datasets employed a split ratio of 7:1:2.

4.3. Baseline

To ascertain MSCformer's state-of-the-art performance, it was compared with current advanced models. These models include Autoformer (Chen et al., 2021), Informer (Zhou et al., 2021), Non-stationary Transformer (Liu, Wu et al., 2022), FEDformer (Zhou et al., 2022), Dlinear (Zeng et al., 2023), TimesNet (Wu et al., 2022), PatchTST (Nie et al., 2022), NBEATS (Oreshkin et al., 2019), TiDE (Das et al., 2023), TSMixer (Chen et al., 2023), MultiWaveNet (Tian et al., 2024), Vector Autoregression (Cryer, 1986), Vector Exponential Smoothing (Cryer, 1986), ETSformer (Woo et al., 2022) and the simple Repeat-C model, which uses the last value in the look-back window as the forecast, provide 15 baseline models available for comprehensive comparison.

4.4. Evaluation metric

To facilitate a clear comparison of the predictive performance of MSCformer with other methods, two widely used metrics, Mean Squared Error (MSE) and Mean Absolute Error (MAE), are employed as evaluation indices for multivariate time series forecasting. The Mean Squared Error (MSE) represents the average of the squared differences between predicted and actual values, while the Mean Absolute Error (MAE) represents the average of the absolute differences between predicted and actual values. The definitions of MSE and MAE are as follows:

$$\mathrm{MSE} = 1/T \sum_{i=1}^{T} \left(y_i - \hat{y}_i\right)^2$$
$$\mathrm{MAE} = 1/T \sum_{i=1}^{T} \left|y_i - \hat{y}_i\right|$$

where T is the length of the time series to be predicted, $\hat{y}_i$ represents the predicted value by MSCformer, and y_i denotes the actual value.

Furthermore, to more accurately assess model's performance relative to naive forecasting, the Mean Absolute Scaled Error (MASE) has been introduced as a comparative metric. MASE provides a standardized measurement of error by comparing the model's Mean Absolute Error (MAE) to the MAE of a naive forecast. It allows the model's performance to be assessed relative to a baseline model. Specifically, a MASE value less than 1 indicates the model's predictive MAE is lower than that of the naive forecast, which means that the model has better predictive performance than a naive forecast; on the other hand, a MASE value greater than 1 indicates that the model has inferior predictive performance to the naive approach:

$$\mathrm{MASE} = \frac{\mathrm{MAE}}{\mathrm{MAE}_{\mathrm{naive}}}$$

4.5. Experimental results of multivariate time series forecasting

Table 2 showcases the comparative results of MSCformer with other multivariate long-term time series prediction methods. The findings indicate that MSCformer possesses superior long-term predictive performance for multivariate time series compared to other methods. In 48 forecasts across six benchmark datasets, MSCformer performed best in 29 cases and best or second-best in 43 experiments. Compared to the current leading PatchTST model, MSCformer significantly reduced MSE across different prediction lengths. Specifically, the average predictive accuracy computed across four different prediction lengths indicates that MSCformer reduced MSE on Electricity, Traffic, Illness, ETT-h, Weather, and Exchange-rate datasets by 8.90%, 7.92%, 24.9%, 1.05%, 1.93% and 3.81% respectively. On large-scale datasets such as Traffic and Electricity, MSCformer achieved particularly notable performance improvements, reflecting the advantage of large datasets in providing abundant data samples, enabling the model to capture finer variable relationships. Compared to past methods, MSCformer more comprehensively utilized information across multiple time scales, accurately capturing the internal dependency structure of time series. This multi-scale modeling approach is particularly important for handling large-scale datasets rich in variation and complexity. MSCformer also demonstrated excellent performance on small-scale datasets like Illness, thanks to the effective application of the Local Dependence Compensation module, enhancing predictive capabilities on smaller datasets. Moreover, for the Exchange-rate, the simple Repeat-C model outperformed other models, it can be attributed to the high noise, ambiguity and randomness of the exchange-rate dataset, which makes complex models ineffective at capturing true trends, while simple persistence models may be better adapted to such random fluctuations (Meese & Rogoff, 1983; Moosa & Burns, 2014).

Overall, MSCformer's significant performance enhancement is attributed to its comprehensive dependency capturing modules, including across-variable and across-time dependency capture, and the local dependence compensation module. These innovations render MSCformer more comprehensive in understanding and modeling the intrinsic dependency structure of multivariate time series, allowing it to excel in prediction tasks and provide reliable prediction support for practical applications.

To further evaluate our model's performance relative to the naive model Repeat-C, the MASE metric was further explored. The best results are highlighted in bold. As shown in Table 3, across 24 benchmark tests on six datasets, MSCformer achieved the best results in 20 cases, demonstrating its superior performance. Compared to the naive model, our model is clearly superior, further confirming the significant improvements in prediction accuracy offered by MSCformer.

4.6. More results on ablation study

4.6.1. Main ablation experient

The MSCformer model relies on three key modules: cross-temporal dependency extraction in independent time series, local dependency compensation in time series segmentation, and cross-dimensional dependency extraction among multivariate data. To validate the impact of these modules on model performance, the study constructed three variants: MSCformer-a, which removes the cross-temporal dependency

[3] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.
[4] https://www.bgc-jena.mpg.de/wetter/.

Table 2

MSCformer's long-term multivariate forecasting results. For the illness dataset, the look-back window size $W = 36$, with prediction lengths $T \in \{24, 36, 48, 60\}$; for other datasets $W = 96$, with prediction lengths $T \in \{96, 192, 336, 720\}$. The best results are highlighted in red, and the second-best in blue.

Model		MSCformer		TiDE		TSMixer		MultiWaveNet		ETSformer		PatchTST		Timesnet		DLinear		NBEATS		FEDformer		Stationary		Informer		Autoformer		VAR		VES		Repeat-C	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	0.147	0.241	0.232	0.317	0.204	0.308	0.170	0.271	0.187	0.304	0.167	0.252	0.168	0.272	0.197	0.282	0.145	0.247	0.193	0.308	0.169	0.273	0.274	0.368	0.201	0.307	0.503	0.421	1.568	0.942	1.588	0.946
	192	0.161	0.253	0.232	0.321	0.218	0.329	0.182	0.281	0.199	0.315	0.175	0.261	0.184	0.289	0.196	0.285	0.180	0.283	0.201	0.315	0.182	0.286	0.296	0.386	0.222	0.334	0.760	0.531	1.567	0.945	1.595	0.950
	336	0.178	0.273	0.226	0.311	0.239	0.350	0.197	0.297	0.212	0.329	0.191	0.278	0.198	0.300	0.209	0.301	0.200	0.308	0.214	0.329	0.200	0.304	0.300	0.394	0.231	0.338	2.074	0.722	1.622	0.963	1.617	0.961
	720	0.208	0.301	0.260	0.333	0.271	0.373	0.228	0.324	0.233	0.345	0.231	0.312	0.220	0.320	0.245	0.333	0.266	0.362	0.246	0.355	0.222	0.321	0.373	0.439	0.254	0.361	7.711	0.968	1.655	0.981	1.647	0.975
	Avg	0.174	0.267	0.238	0.321	0.233	0.340	0.194	0.293	0.208	0.323	0.191	0.276	0.192	0.295	0.212	0.300	0.198	0.300	0.214	0.327	0.193	0.296	0.311	0.397	0.227	0.338	2.762	0.661	1.603	0.958	1.612	0.958
Traffic	96	0.394	0.274	0.731	0.464	0.528	0.359	0.559	0.379	0.607	0.392	0.446	0.283	0.593	0.321	0.650	0.396	0.398	0.282	0.587	0.366	0.612	0.338	0.719	0.391	0.613	0.388	1.402	0.709	2.813	1.081	2.723	1.079
	192	0.422	0.281	0.714	0.458	0.563	0.382	0.571	0.382	0.621	0.399	0.453	0.286	0.617	0.336	0.598	0.370	0.409	0.293	0.604	0.373	0.613	0.340	0.696	0.379	0.616	0.382	2.814	0.876	2.841	1.087	2.756	1.087
	336	0.434	0.286	0.749	0.468	0.579	0.393	0.581	0.385	0.622	0.396	0.468	0.292	0.629	0.336	0.605	0.373	0.449	0.318	0.621	0.383	0.618	0.328	0.777	0.420	0.622	0.337	4.369	1.077	2.943	1.108	2.791	1.095
	720	0.469	0.305	0.811	0.484	0.623	0.421	0.613	0.401	0.632	0.396	0.501	0.310	0.640	0.350	0.645	0.394	0.589	0.391	0.626	0.382	0.653	0.355	0.864	0.472	0.660	0.408	7.985	1.486	2.915	1.101	2.811	1.097
	Avg	0.430	0.287	0.751	0.469	0.573	0.389	0.581	0.387	0.621	0.396	0.467	0.293	0.620	0.336	0.625	0.383	0.461	0.321	0.610	0.376	0.614	0.340	0.764	0.416	0.628	0.379	4.143	1.037	2.878	1.094	2.770	1.090
Illness	24	1.215	0.728	5.797	1.804	3.219	1.233	1.787	0.847	2.527	1.020	1.921	0.855	2.317	0.934	2.398	1.040	1.879	0.886	3.228	1.260	2.294	0.945	5.764	1.677	3.483	1.287	1.939	0.981	2.614	1.154	6.587	1.701
	36	1.382	0.743	5.116	1.638	3.677	1.287	1.762	0.837	2.615	1.007	1.994	0.883	1.972	0.920	2.646	1.088	2.210	1.018	2.679	1.080	1.825	0.848	4.755	1.467	3.103	1.148	5.801	1.685	7.435	1.941	7.130	1.884
	48	1.411	0.767	4.200	1.458	3.991	1.364	2.118	0.917	0.359	0.972	2.026	0.872	2.238	0.940	2.614	1.086	2.440	1.088	2.622	1.078	2.010	0.900	4.763	1.469	2.669	1.085	4.549	1.434	5.784	1.659	6.575	1.798
	60	1.726	0.835	4.324	1.494	4.592	1.496	2.027	0.906	2.487	1.016	1.744	0.849	2.027	0.928	2.804	1.146	2.547	1.057	2.857	1.157	2.178	0.963	5.264	1.564	2.770	1.125	3.759	1.129	4.678	1.401	5.893	1.677
	Avg	1.434	0.768	4.859	1.599	3.870	1.345	1.924	0.877	2.497	1.004	1.921	0.865	2.139	0.931	2.616	1.090	2.269	1.012	2.847	1.144	2.077	0.914	5.137	1.544	3.006	1.161	4.012	1.307	5.128	1.539	6.546	1.765
ETT	96	0.294	0.343	0.303	0.355	0.548	0.536	0.299	0.349	0.340	0.391	0.292	0.342	0.340	0.374	0.333	0.387	0.201	0.343	0.358	0.397	0.476	0.458	3.755	1.525	0.346	0.388	0.448	0.420	0.514	0.454	0.438	0.425
	192	0.374	0.394	0.377	0.399	0.872	0.631	0.359	0.392	0.430	0.439	0.377	0.393	0.402	0.414	0.477	0.476	0.222	0.368	0.429	0.439	0.512	0.493	5.602	1.931	0.456	0.452	0.452	0.433	0.500	0.456	0.524	0.469
	336	0.404	0.419	0.358	0.395	0.835	0.664	0.418	0.436	0.485	0.479	0.418	0.428	0.452	0.452	0.594	0.541	0.329	0.454	0.496	0.487	0.552	0.551	4.721	1.835	0.482	0.486	0.481	0.459	0.535	0.482	0.582	0.504
	720	0.423	0.440	0.426	0.444	0.975	0.725	0.423	0.442	0.500	0.497	0.423	0.442	0.462	0.468	0.831	0.657	0.365	0.477	0.463	0.474	0.562	0.560	3.647	1.625	0.515	0.511	0.463	0.461	0.544	0.495	0.583	0.514
	Avg	0.374	0.399	0.366	0.398	0.808	0.639	0.375	0.405	0.439	0.452	0.378	0.401	0.414	0.427	0.559	0.515	0.279	0.411	0.437	0.449	0.526	0.516	4.431	1.729	0.450	0.459	0.461	0.443	0.523	0.472	0.532	0.478
Weather	96	0.171	0.216	0.236	0.262	0.181	0.252	0.185	0.240	0.197	0.281	0.177	0.219	0.172	0.220	0.196	0.255	0.206	0.227	0.217	0.296	0.173	0.223	0.300	0.384	0.266	0.336	0.595	0.467	0.248	0.258	0.259	0.254
	192	0.219	0.258	0.264	0.286	0.218	0.291	0.241	0.282	0.237	0.312	0.223	0.258	0.219	0.261	0.237	0.296	0.255	0.261	0.276	0.336	0.245	0.285	0.598	0.544	0.307	0.367	0.653	0.503	0.295	0.297	0.309	0.292
	336	0.272	0.297	0.323	0.306	0.262	0.321	0.290	0.317	0.298	0.353	0.278	0.297	0.280	0.306	0.283	0.355	0.308	0.304	0.339	0.380	0.321	0.338	0.578	0.523	0.359	0.395	0.689	0.526	0.349	0.336	0.377	0.338
	720	0.351	0.350	0.361	0.352	0.320	0.367	0.356	0.354	0.352	0.388	0.354	0.346	0.365	0.359	0.345	0.381	0.368	0.359	0.403	0.428	0.414	0.410	1.059	0.741	0.419	0.428	0.699	0.537	0.437	0.391	0.465	0.394
	Avg	0.253	0.280	0.296	0.302	0.245	0.308	0.268	0.298	0.271	0.334	0.258	0.280	0.259	0.287	0.265	0.317	0.284	0.288	0.309	0.360	0.288	0.314	0.634	0.548	0.338	0.382	0.659	0.508	0.332	0.321	0.353	0.320
Exchange	96	0.084	0.203	0.094	0.217	0.228	0.385	0.086	0.205	0.085	0.204	0.087	0.205	0.107	0.234	0.088	0.218	0.098	0.206	0.148	0.278	0.111	0.237	0.847	0.752	0.197	0.323	1.508	0.569	0.263	0.266	0.081	0.196
	192	0.181	0.303	0.202	0.324	0.386	0.513	0.177	0.299	0.182	0.303	0.175	0.296	0.226	0.344	0.176	0.315	0.225	0.329	0.271	0.380	0.219	0.335	1.204	0.895	0.300	0.369	1.577	0.792	0.435	0.448	0.167	0.289
	336	0.319	0.408	0.350	0.431	0.492	0.575	0.353	0.428	0.348	0.428	0.343	0.421	0.367	0.448	0.313	0.427	0.493	0.482	0.460	0.500	0.421	0.476	1.672	1.036	0.509	0.524	2.402	1.025	1.324	0.803	0.305	0.396
	720	0.826	0.680	0.860	0.700	0.705	0.702	0.875	0.705	1.025	0.774	0.863	0.688	0.964	0.746	0.839	0.695	1.108	0.804	1.195	0.841	1.092	0.769	2.478	1.310	1.447	0.941	2.777	1.349	4.006	1.374	0.823	0.681
	Avg	0.353	0.399	0.377	0.418	0.453	0.544	0.373	0.409	0.410	0.427	0.367	0.403	0.416	0.443	0.354	0.414	0.481	0.455	0.519	0.500	0.461	0.454	1.550	0.998	0.613	0.539	2.066	0.934	1.507	0.723	0.344	0.391
Count		29		0		2		0		0		2		0		0		7		0		0		0		0		0		0		8	

Table 3

MSCformer's long-term multivariate forecasting results (MASE). For the illness dataset, the look-back window size $W = 36$, with prediction lengths $T \in \{24, 36, 48, 60\}$; for other datasets, $W = 96$, with prediction lengths $T \in \{96, 192, 336, 720\}$. The best results are highlighted in red.

TYPE		Transformer		MLP			CNN	RNN	Mathematical
Model		Ours	PatchTST	TiDE	TSMixer	Dlinear	Timesnet	LSTM	VAR
Metric		MASE	MASE	MASE	MASE	MASE	MASE	MASE	MASE
Electricity	96	0.255	0.266	0.335	0.326	0.298	0.288	0.462	0.445
	192	0.266	0.275	0.338	0.346	0.300	0.304	0.498	0.559
	336	0.284	0.289	0.324	0.364	0.313	0.312	0.492	0.751
	720	0.309	0.320	0.342	0.383	0.342	0.328	0.835	0.993
	Avg	0.279	0.288	0.335	0.355	0.313	0.308	0.573	0.689
Traffic	96	0.254	0.262	0.430	0.333	0.367	0.297	0.420	0.657
	192	0.259	0.263	0.421	0.351	0.340	0.309	0.417	0.806
	336	0.261	0.267	0.427	0.359	0.341	0.307	0.416	0.984
	720	0.278	0.283	0.441	0.384	0.359	0.319	0.734	1.355
	Avg	0.263	0.269	0.430	0.357	0.352	0.308	0.497	0.952
Illness	24	0.428	0.503	1.061	0.725	0.611	0.549	1.019	0.577
	36	0.394	0.469	0.869	0.683	0.577	0.488	0.979	0.894
	48	0.427	0.485	0.811	0.759	0.604	0.523	1.033	0.798
	60	0.498	0.506	0.891	0.892	0.683	0.553	1.120	0.673
	Avg	0.435	0.490	0.906	0.762	0.618	0.527	1.036	0.741
ETT-h	96	0.807	0.805	0.835	1.261	0.911	0.880	3.007	0.988
	192	0.840	0.838	0.851	1.345	1.015	0.883	2.951	0.923
	336	0.831	0.849	0.784	1.317	1.073	0.897	2.754	0.911
	720	0.856	0.860	0.864	1.411	1.278	0.911	2.640	0.897
	Avg	0.835	0.839	0.833	1.337	1.077	0.893	2.828	0.927
Weather	96	0.850	0.862	1.031	0.992	1.004	0.866	1.598	1.839
	192	0.884	0.884	0.980	0.997	1.014	0.894	1.490	1.723
	336	0.879	0.879	0.906	0.950	1.050	0.905	1.343	1.556
	720	0.888	0.878	0.893	0.931	0.967	0.911	1.320	1.363
	Avg	0.877	0.876	0.944	0.963	0.992	0.898	1.420	1.591
Exchange	96	1.036	1.046	1.107	1.964	1.112	1.194	5.352	2.903
	192	1.048	1.024	1.121	1.775	1.090	1.190	4.080	2.740
	336	1.030	1.063	1.088	1.452	1.078	1.131	3.109	2.588
	720	0.999	1.010	1.028	1.031	1.021	1.095	2.095	1.981
	Avg	1.020	1.031	1.070	1.392	1.060	1.134	3.128	2.391
Count		20	4	0	0	0	0	0	0

Table 4
The ablation studies on Illness dataset.

Forecasting length		24		36		48		60	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Illness	Origin	**1.215**	**0.728**	**1.382**	**0.743**	**1.411**	**0.767**	**1.726**	**0.835**
	MSCformer-a	1.447	0.771	1.445	0.783	1.419	0.772	1.762	0.863
	MSCformer-b	1.391	0.733	1.485	0.768	1.417	0.782	1.731	0.865
	MSCformer-c	1.287	0.698	1.395	0.750	1.411	0.763	1.735	0.850

Table 5
The ablation studies on Electricity dataset.

Forecasting length		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	Origin	**0.147**	**0.241**	**0.161**	**0.253**	0.178	0.273	**0.208**	**0.301**
	MSCformer-a	0.290	0.382	0.298	0.387	0.305	0.393	0.336	0.411
	MSCformer-b	0.148	0.241	0.163	0.254	**0.175**	**0.269**	0.211	0.302
	MSCformer-c	0.167	0.253	0.175	0.261	0.191	0.278	0.231	0.312

Table 6
The ablation studies on ETT-h dataset.

Forecasting length		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT-h	Origin	**0.294**	**0.343**	**0.374**	0.394	**0.404**	**0.419**	**0.423**	**0.440**
	MSCformer-a	0.351	0.382	0.428	0.428	0.435	0.443	0.446	0.456
	MSCformer-b	0.294	0.343	0.379	**0.393**	0.417	0.426	0.427	0.444
	MSCformer-c	0.316	0.359	0.379	0.396	0.427	0.437	0.429	0.445

Table 7
The ablation studies on Weather dataset.

Forecasting length		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	Origin	**0.171**	**0.216**	**0.219**	**0.258**	**0.272**	**0.297**	**0.351**	0.350
	MSCformer-a	0.200	0.251	0.246	0.286	0.295	0.318	0.363	0.361
	MSCformer-b	0.180	0.226	0.220	0.260	0.277	0.302	0.352	0.352
	MSCformer-c	0.182	0.219	0.231	0.262	0.280	0.299	0.355	**0.349**

extraction module, neglecting the extraction of dependencies within variables. MSCformer-b, which omits the local dependency compensate module, ignoring the supplementation of internal local dependencies in segments. MSCformer-c, which removes the cross-dimensional dependency extraction module between multivariate data, disregarding the dependencies among different variables, to assess their respective roles.

Tables 4–7 show that MSCformer exhibits superior predictive performance compared to its sub-models, highlighting the importance of integrating cross-temporal, local dependency compensate, and cross-dimensional dependencies. Ablation study results on the Electricity, ETT-h and Weather datasets, shown in Tables 5–7, demonstrate the significance of cross-temporal dependency extraction and cross-dimensional dependency extraction modules. By removing these two modules, model performance is significantly reduced, suggesting that these modules enable the model to identify subtle correlations between data and enhance its ability to capture complex variable relationships in large-scale datasets effectively. Conversely, the ablation study on the Illness dataset Table 4 reveals the significant impact of the local dependency compensation module. The model's performance notably declines after removing this module, indicating that capturing fine-grained local time series features is crucial for prediction accuracy in small-scale datasets. This emphasizes MSCformer's capability in capturing short-term dependencies within time series to improve prediction performance in smaller datasets.

In summary, MSCformer, through multi-faceted dependency extraction, achieves significant performance improvements in large-scale datasets while also demonstrating robust predictive capabilities in small-scale datasets. This research not only confirms the role of key modules in MSCformer for multivariate time series prediction but also further emphasizes the importance of considering various dependencies to enhance the accuracy of time series prediction.

4.6.2. *Varying segment length*

Additionally, the performance enhancement of the MSCformer model is attributed to its comprehensive utilization of information across multiple time scales. As demonstrated in Tables 8 and 9, experiments conducted on the ETT-h and Weather datasets with multi-scale time segmentation and under single time scale conditions reveal the model's performance at different scales. The results indicate that under single time scale conditions, the model shows dataset-specific sensitivity to the choice of segment length. For example, in the ETT-h dataset, the model exhibits superior predictive performance when the time series segment length is set to 8, while in the Weather dataset, optimal performance is achieved with a segment length of 4. However, compared to single-scale time segmentation, the MSCformer employing multi-scale time segmentation demonstrates more outstanding performance in both datasets.

These experimental results conclusively show that the choice of segment length under a single time scale impacts predictive performance. Nevertheless, by integrating information across multiple time scales, MSCformer more comprehensively captures the dynamic variations of time series, thereby effectively enhancing the overall forecasting accuracy of the model. This further confirms the importance and potential value of using multi-time scale information in multivariate time series forecasting to improve model performance.

4.6.3. *Impact of normalization and denormalization*

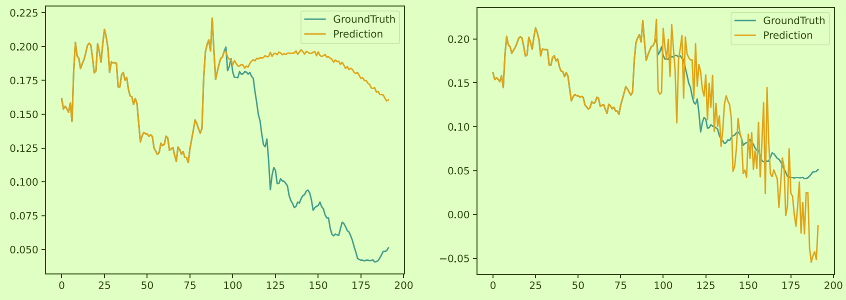
To validate the effectiveness of normalization and denormalization methods in handling non-stationary time series data, we visualized the forecasting of the MSCformer model on the Weather dataset with $W = \{96\}$ and $T = \{96\}$, as shown in Fig. 8. Fig. 8(a) displays the forecasting results of the MSCformer model without normalization and

Table 8
The results with different single segment lengths on ETT-h dataset.

Forecasting		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT-h	Origin	0.294	**0.343**	**0.374**	**0.394**	**0.404**	**0.419**	**0.423**	**0.440**
	Segment = 4	**0.291**	0.349	0.379	0.395	0.415	0.426	0.425	0.450
	Segment = 8	0.298	0.347	0.375	0.394	0.408	0.423	0.427	0.449
	Segment = 16	0.294	0.345	0.380	0.396	0.421	0.430	0.433	0.447

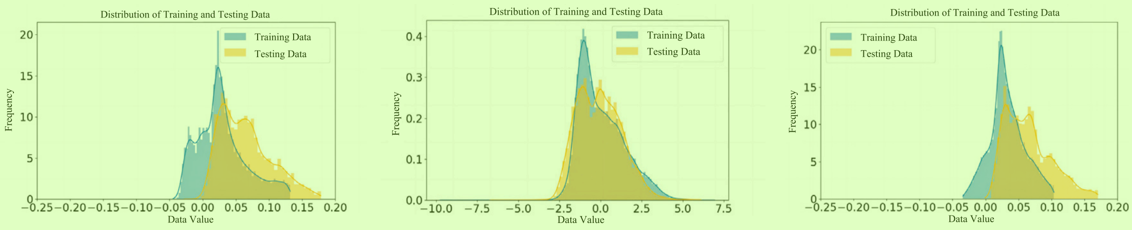
Table 9
The results with different single segment lengths on Weather dataset.

Forecasting		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	Origin	**0.171**	**0.216**	**0.219**	**0.258**	**0.272**	**0.297**	0.351	0.350
	Segment = 4	0.172	0.219	0.219	0.260	0.277	0.301	**0.347**	**0.349**
	Segment = 8	0.173	0.217	0.219	0.261	0.274	0.295	0.348	0.349
	Segment = 16	0.172	0.219	0.221	0.262	0.276	0.302	0.352	0.352



(a) Without Norm/Denorm. (b) With Norm/Denorm.

Fig. 8. The forecasting results on Weather dataset.



(a) Model Input. (b) Normalization Input. (c) Denormalization Input.

Fig. 9. The impact of normalization and denormalization on the differences in training and test data distributions, analyzed based on the MSCformer model's forecasting for the Weather dataset with a forecasting Length of 96. The figure shows the distribution of training and test data at each step of the training process, from left to right.

denormalization, while Fig. 8(b) shows the results after applying these two processes. It can be observed that when dividing the Weather dataset (variable: OT) into training and test data at a specific time point, there is a significant issue of distribution shift, meaning the distributions of training and test data differ. Consequently, the model often mispredicts future values, as shown in Fig. 8(a), where the average value of the data continuously decreases and the model fails to keep up with changes in data trends, leading to forecasting offsets. However, as illustrated in Fig. 8(b), by applying normalization and denormalization methods, this distribution difference is mitigated and the accuracy of model forecasting is enhanced.

Additionally, we further analyzed the data distribution of the Weather dataset during the experimental process, as shown in Fig. 9. We compared the data distributions of training and test data at each step of the sequence processing, presenting Fig. 9(a): The model's original input. Fig. 9(b): The input after normalization. Fig. 9(c): The output distribution after denormalization. In Fig. 9(a), a disparity in the distributions of the original training and test data was observed. In Fig. 9(b), the normalization method transformed the data distribution into a mean-centered distribution, effectively alleviating the issue of distribution shift in the input data. Subsequently, in Fig. 9(c), the denormalization technique successfully returned the final output distribution to a state consistent with the original distribution shown in Fig. 9(a), ensuring consistency in the model's input and output distributions.

4.7. Varying look-back window

In general, an effective time series prediction model should possess strong temporal relationship extraction capabilities. The size of the input length represents how much historical information the model can utilize. A model with the ability to model long-term temporal dependencies should perform better under a bigger look-back window (Zeng et al., 2023). To this end, we compared the predictive performance of MSCformer across various input lengths.

Specifically, we explored different input lengths on the ETT-h and Weather datasets. Tables 10 and 11 show the results. On the ETT-h dataset, the predictive performance of the model generally improves as forecast length increases. Weather dataset error results at input lengths

Table 10
The results with different input lengths on ETT-h dataset.

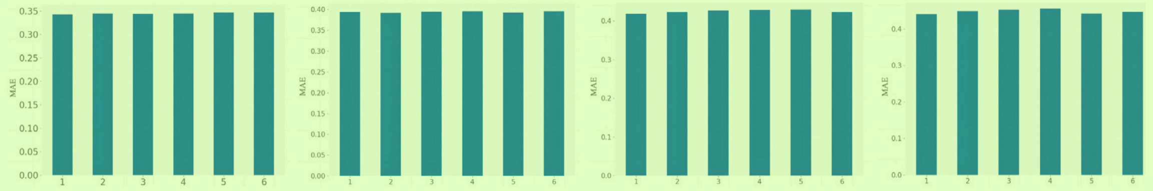
Input length		24		48		96		192		336		420		512	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT-h	96	0.328	0.359	0.300	0.344	0.294	0.343	0.292	0.345	0.296	0.361	0.284	0.342	0.282	0.340
	192	0.421	0.410	0.390	0.397	0.374	0.394	0.377	0.403	0.360	0.389	0.355	0.385	0.347	0.384
	336	0.464	0.444	0.428	0.429	0.404	0.419	0.407	0.424	0.386	0.415	0.339	0.386	0.332	0.386
	720	0.471	0.461	0.438	0.445	0.423	0.440	0.447	0.460	0.391	0.426	0.382	0.423	0.381	0.425
	Avg	0.421	0.419	0.389	0.404	0.374	0.399	0.381	0.408	0.358	0.398	0.340	0.384	0.336	0.384

Table 11
The results with different input lengths on Weather dataset.

Input length		24		48		96		192		336		420		512	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
WTH	96	0.221	0.247	0.217	0.248	0.171	0.216	0.158	0.204	0.154	0.205	0.153	0.204	0.149	0.199
	192	0.267	0.282	0.259	0.281	0.219	0.258	0.204	0.246	0.200	0.248	0.197	0.245	0.197	0.250
	336	0.327	0.323	0.315	0.320	0.272	0.297	0.258	0.291	0.253	0.291	0.252	0.293	0.248	0.289
	720	0.406	0.375	0.384	0.367	0.351	0.350	0.335	0.347	0.339	0.350	0.327	0.340	0.326	0.342
	Avg	0.305	0.307	0.294	0.304	0.253	0.280	0.239	0.272	0.237	0.274	0.232	0.271	0.230	**0.270**

Table 12
The results with different random seed on Weather dataset.

Random seed		Random seed = 2019		Random seed = 2020		Random seed = 2021		Random seed = 2022		Random seed = 2023	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.171	0.217	0.170	0.215	0.171	0.216	0.169	0.215	0.172	0.217
	192	0.217	0.259	0.235	0.267	0.219	0.258	0.221	0.262	0.219	0.261
	336	0.274	0.300	0.294	0.310	0.272	0.297	0.277	0.302	0.277	0.303
	720	0.350	0.350	0.374	0.368	0.351	0.350	0.373	0.367	0.352	0.352
	Avg	0.253	0.282	0.268	0.290	0.253	0.280	0.260	0.287	0.255	0.283



(a) ETT-h(T=96). (b) ETT-h(T=192). (c) ETT-h(T=336). (d) ETT-h(T=720).

Fig. 10. MAE scores under different model parameters, with each line representing an MAE score for a parameter combination. The combinations (L, D) = (3, 128), (3, 256), (4, 128), (4, 256), (5, 128), (5, 256) are labeled from 1 to 6 in the figure in sequence.

of 192 and 336, however, show fluctuations. It is primarily due to the particularities of the weather dataset, where the accuracy of weather forecasts is greatly dependent on recent observational data, and a longer input window might introduce too much unnecessary information, causing local fluctuations. Overall, the MSCformer achieved better experimental results in longer look-back windows, demonstrating its effectiveness in extracting information from longer look-back windows.

4.8. Hyperparameter analysis

4.8.1. Random seed

The final results reported by the model were run with a fixed random seed $\mathrm{RandomSeed} = 2021$. To test the robustness of the results, we trained the MSCformer model with 5 different $\mathrm{RandomSeed} = \{2019, 2020, 2021, 2022, 2023\}$ and calculated the MSE and MAE scores for each random seed. The predictive results for the Weather dataset under different random seeds, as shown in Table 12, demonstrate the robustness of our model to the choice of random seed.

4.8.2. Model hyperparameter

To understand whether MSCformer is sensitive to choices in deep learning settings, we experimented with different model parameters. We varied the number of model training layers $L = \{3, 4, 5\}$, chose model dimensions $D = \{128, 256\}$, and set the hidden dimension of the feedforward network to $F = 2D$, resulting in 6 different sets of model hyperparameters. Fig. 10 shows these combinations' MAE scores at different prediction lengths $T = \{96, 192, 336, 720\}$ on the ETT-h dataset, indicating the dataset's robustness to the choice of model hyperparameters.

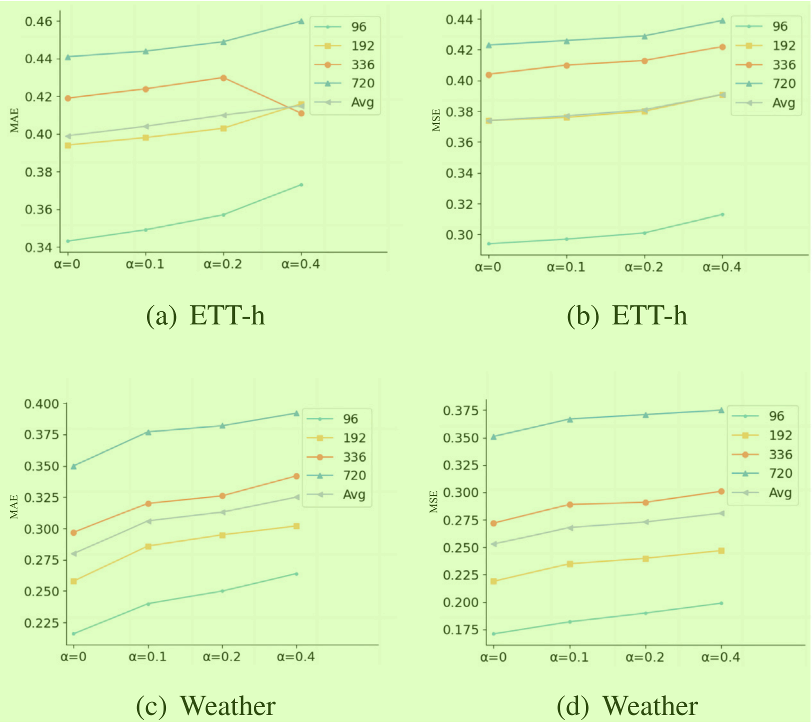
4.9. Impact of different aggregation methods

In exploring the impact of aggregation strategies at different time scales, four main aggregation methods were employed: equal weight aggregation, exponential decay weight aggregation, learnable weight aggregation, and weighted average aggregation. The experimental results are shown in Table 13.

As compared with other methods, exponential decay aggregation demonstrated superior predictive capabilities. The reasons analyzed include: unlike equal weight aggregation, the exponential decay method assigns differentiated weights to data segments across different time scales, enabling the model to enhance information extraction and utilization within key time periods while maintaining an overall understanding of the data, thus improving prediction accuracy and model sensitivity; compared to methods that require training to learn optimal weights, exponential decay aggregation uses a predefined, fixed weight reduction strategy that simplifies the training process and enhances the model's generalization ability on unseen data; in contrast to weighted average aggregation, which assigns weights based on segment

Table 13
Performance of different aggregation methods.

Dataset	Metric	Average		Exponential reduced		Learnable		Weighted average	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	0.147	0.241	0.147	0.241	0.147	0.241	0.148	0.241
	192	0.161	0.254	0.161	0.253	0.162	0.254	0.162	0.254
	336	0.179	0.272	0.178	0.273	0.179	0.272	0.180	0.273
	720	0.213	0.305	0.208	0.301	0.223	0.311	0.222	0.310
	Avg	0.175	0.268	0.174	0.267	0.178	0.270	0.178	0.270
ETT-h	96	0.296	0.345	0.294	0.343	0.295	0.344	0.297	0.346
	192	0.375	0.395	0.374	0.394	0.375	0.392	0.377	0.393
	336	0.407	0.421	0.404	0.419	0.390	0.411	0.392	0.419
	720	0.426	0.442	0.423	0.440	0.428	0.441	0.430	0.442
	Avg	0.376	0.401	0.374	0.399	0.372	0.397	0.374	0.400
Weather	96	0.173	0.220	0.171	0.216	0.171	0.218	0.173	0.218
	192	0.219	0.259	0.219	0.258	0.219	0.261	0.219	0.261
	336	0.277	0.302	0.272	0.297	0.275	0.300	0.277	0.301
	720	0.351	0.349	0.351	0.350	0.353	0.350	0.353	0.352
	Avg	0.255	0.283	0.253	0.280	0.255	0.282	0.256	0.283



(a) ETT-h (b) ETT-h

(c) Weather (d) Weather

Fig. 11. The impact of different noise scales on prediction performance.

length, exponential decay aggregation allocates heavier weights to lower scales with shorter segment lengths and more segments, enhancing the model's sensitivity to key changes and subtle trends in time series segments. Therefore, given the clear advantages of exponential decay aggregation in predictive performance, this method was chosen for the final experimental setup of the model, thereby ensuring the accuracy of the predictions.

4.10. Impact of adding noise information

To validate the performance of MSCformer under various noise input levels, we introduced noise of different scales into the original input data and studied its impact on predictive performance. Specifically, we added Gaussian noise at levels of 0.1, 0.2 and 0.4 to the input data, which can be represented as follows:

$$X'_{\text{input}} = X_{\text{input}} + \alpha \times \mu$$

where X_{input} represents the initial input data, α is the noise level coefficient and $\mu \sim N(0,1)$ is a random variable following a normal distribution. Observing the experimental results Fig. 11, the MSCformer model demonstrates robustness to small-scale noise across different datasets. Under noise levels of 0.1, 0.2 and 0.4, the decline in MSCformer's predictive performance is gradual. Particularly at a noise level of 0.1, there is almost no noticeable impact on predictive performance. Even when the noise is increased to 0.4, MSCformer's predictive performance remains at an acceptable level. Based on these experimental results, we conclude that MSCformer can effectively utilize multi-scale information and accurately capture cross-temporal and cross-variable dependencies in multivariate long-term time series. Especially when facing small-scale noise (such as noise levels of 0.1, 0.2, 0.4), MSCformer exhibits robustness and maintains its ability to accurately model multivariate long-term time series.

4.11. Efficiency analysis

To evaluate the efficiency of our model, we compared MSCformer with other models in terms of parameter count, memory usage and

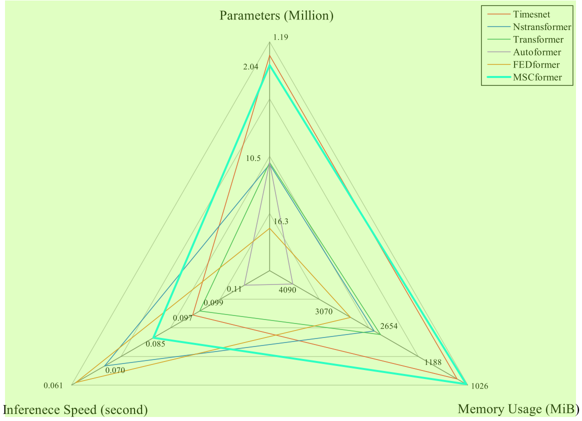


Fig. 12. The parameter count, memory usage and inference speed of different models.

average inference speed. For fairness, the experiments were conducted on the same condition with a look-back window $W = 96$ and forecast length $T = 96$ on the ETT-h dataset.

Fig. 12 illustrates the comparison between the MSCformer model with other models in terms of parameter count, memory usage, and inference speed. The results indicate that MSCformer has a significant advantage in parameter count and memory usage. While MSCformer does not have the traditional encoder–decoder structure of the Transformer and retains only the attention mechanism as its primary modeling approach, it still demonstrates significant performance advantages with a smaller parameter count. Specifically, MSCformer's memory consumption and parameter count are only 2/5 and 1/5 of those of the Autoformer's respectively, and its inference speed is about 24.5% faster. Despite MSCformer's moderate inference speed, this is primarily due to the need to integrate complex multi-scale and multi-dependency relationships during inference, however, with similar inference speeds, MSCformer still achieves optimal prediction results, demonstrating its effectiveness.

4.12. Statistical testing

To strengthen the persuasiveness of the model's experimental results, additional statistical tests were conducted. A total of 16 benchmark models were retrained using the Weather and ETT-h datasets, and multiple experiment results were collected for each model. Using these data, a one-way ANOVA was performed on the average MSE for different forecast lengths, revealing statistically significant differences between the models in MSE errors. Afterwards, the Dunnett test was used for multiple comparisons (Multiple Comparison with the Baselines — MCB), which was designed specifically to compare the performance differences between one baseline group (MSCformer) and multiple comparison groups.

The Dunnett test calculated the t-statistic and adjusted p-values for each group. The t-statistic measures the magnitude of the difference between two groups, Positive values indicate that MSCformer performs better than the comparison models, and negative values indicates the opposite. The p-value reflects the probability of observing the current or more extreme differences under the assumption that there are no differences between the groups. Lower p-values (adjusted by the Benjamini–Hochberg method) indicate statistically significant differences between the groups. If the adjusted p-value is less than the set significance level ($\alpha = 0.05$), there are significant differences between the groups; if it is greater, there are no significant differences.

To visualize the results of the Dunnett test, Fig. 13 was drawn. According to the figures, the pink area represents areas in which our model significantly outperforms the comparison models, and the light blue area represents areas where our model performs worse, both with statistically significant differences. In the area to the right of the dashed line, there are no significant differences between the models. These results show that in most cases, MSCformer's performance is superior to the existing benchmark models, and there are significant differences between them. These statistical analysis results not only highlight the advantages of the MSCformer model, but also strengthen the rigor and credibility of the research findings.

4.13. Conformal predictions

In order to enhance the practicality of the MSCformer model and expand its application scenarios, this study has also introduced an interval forecasting technique that uses simple conformal prediction techniques to extend point predictions to interval predictions., without changing the original architecture of the model. The method has been applied to a 96-step forecasting task on the Electricity dataset.

Based on the predetermined confidence levels, during the testing phase, the conformal prediction method is used to generate prediction intervals for the data in the test set, thereby providing confidence intervals that cover forecast uncertainty.

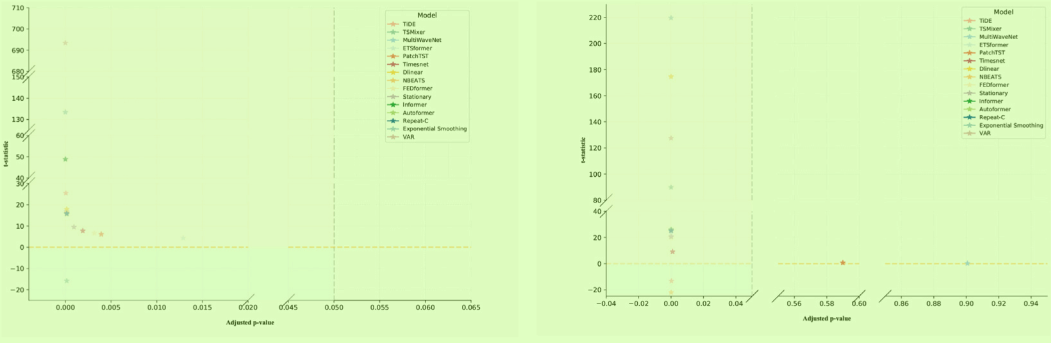
This experiment selected three different confidence levels: $\alpha = 0.5$, $\alpha = 0.9$ and $\alpha = 0.95$. Confidence levels correspond to the range of coverage of the model's prediction uncertainty, where lower confidence levels correspond to narrower prediction intervals, and higher confidence levels correspond to broader coverage ranges. With this approach, MSCformer not only provides precise point predictions, but also reveals the uncertainty of the forecasts through the confidence intervals, thus enhancing the model's reliability, confidence intervals enable decision-makers to conduct more cautious risk assessments and make decisions based on the uncertainty of the predictions.

Fig. 14 shows the point predictions for the 'OT' variable column in the Electricity dataset, along with their corresponding interval results. The 50% confidence intervals have covered most of the actual values, and all actual observations are within the 95% confidence interval. This enriches the practicality of the MSCformer model.

5. Conclusion

In this paper, we introduce MSCformer, a novel model for multivariate long-term time series forecasting. MSCformer effectively overcomes the challenges of computational efficiency, local context awareness, and cross-dimensional dependency recognition encountered by standard Transformer models in handling complex time series. A multi-scale temporal segmentation module, a cross-temporal global dependency extraction network, a local dependency compensation network, and a cross-dimensional global dependency extraction network are integrated in MSCformer, which reduces computational complexity while capturing local characteristics and changing patterns in time series more accurately. It offers a new perspective on capturing deep dependencies within time series. Furthermore, MSCformer fully integrates global and local dependencies across time, as well as global dependencies across dimensions, enabling it to fully comprehend and express the complex dynamics of time series. As a result of this comprehensive feature representation strategy, MSCformer has significantly improved performance on several widely available multivariate time series forecasting datasets.

Innovations introduced in this research focus primarily on optimizing the feature representation of multivariate time series through key modules in the model architecture. As we explore multivariate time series further, we will employ more interpretable methods to model the complex relationships among variables. We also aim to enhance the model's memory efficiency, especially when dealing with long sequences and a large number of variables, so that it can be widely deployed and applied in more practical settings.



(a) Weather.

(b) ETT-h.

Fig. 13. The Dunnet result.

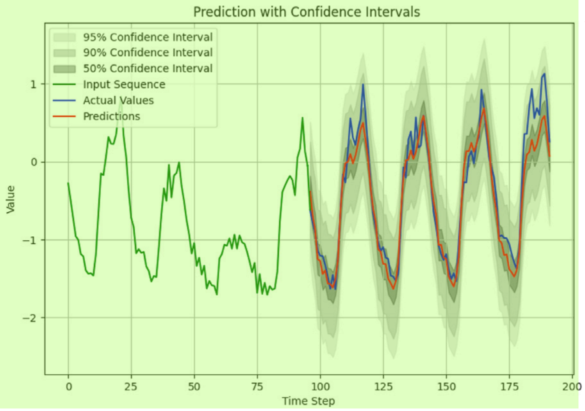


Fig. 14. The result of interval forecasts on Electricity dataset.

CRediT authorship contribution statement

Ao Li: Writing – original draft. **Ying Li:** Visualization. **Yunyang Xu:** Validation, Software. **Xuemei Li:** Writing – review & editing, Methodology, Funding acquisition. **Caiming Zhang:** Investigation, Conceptualization.

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.

Acknowledgments

Supported by the National Natural Science Foundation of China (NSFC) Joint Fund with Zhejiang Integration of Informatization and Industrialization, China under Key Project (Grant No. U22A2033) and NSFC, China (Grant No. 62072281).

References

Box George, E., Jenkins Gwilym, M., Reinsel Gregory, C., & Ljung Greta, M. (1976). *Time series analysis: forecasting and control*. San Francisco: Holden Bay.

Chen, S. A., Li, C. L., Yoder, N., Arik, S. O., & Pfister, T. (2023). Tsmixer: An all-mlp architecture for time series forecasting. arXiv preprint arXiv:2303.06053.

Chen, M., Peng, H., Fu, J., & Ling, H. (2021). Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12270–12280).

Cryer, J. D. (1986). *Vol. 286*, *Time series analysis*. Duxbury Press Boston.

Das, A., Kong, W., Leach, A., Mathur, S., Sen, R., & Yu, R. (2023). Long-term forecasting with tide: Time-series dense encoder. arXiv preprint arXiv:2304.08424.

Deihim, A., Alonso, E., & Apostolopoulou, D. (2023). Sttre: A spatio-temporal transformer with relative embeddings for multivariate time series forecasting. Available at SSRN 4404879.

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems* MWSCAS, (pp. 1597–1600). IEEE.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., & Liu, Y. (2020). Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, *24*, 16453–16482.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J. H., & Choo, J. (2021). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*.

Lai, G., Chang, W. C., Yang, Y., & Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 95–104).

Li, L., Su, X., Zhang, Y., Lin, Y., & Li, Z. (2015). Trend modeling for traffic time series analysis: An integrated study. *IEEE Transactions on Intelligent Transportation Systems*, *16*, 3430–3439.

Liu, Y., Wu, H., Wang, J., & Long, M. (2022). Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, *35*, 9881–9893.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., & Dustdar, S. (2021). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., & Xu, Q. (2022). Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, *35*, 5816–5828.

Ma, X., Li, X., Fang, L., Zhao, T., & Zhang, C. (2024). U-mixer: An unet-mixer architecture with stationarity correction for time series forecasting. arXiv preprint arXiv:2401.02236.

Matsubara, Y., Sakurai, Y., Van Panhuis, W. G., & Faloutsos, C. (2014). Funnel: automatic mining of spatially coevolving epidemics. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 105–114).

Meese, R. A., & Rogoff, K. (1983). Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, *14*, 3–24.

Moosa, I., & Burns, K. (2014). A reappraisal of the meese–rogoff puzzle. *Applied Economics*, *46*, 30–40.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.

Pang, Y., Zhou, X., Zhang, J., Sun, Q., & Zheng, J. (2022). Hierarchical electricity time series prediction with cluster analysis and sparse penalty. *Pattern Recognition*, *126*, Article 108555.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*, 1181–1191.

Semenoglou, A. A., Spiliotis, E., & Assimakopoulos, V. (2023). Image-based time series forecasting: A deep convolutional neural network approach. *Neural Networks*, *157*, 39–53.

Sen, R., Yu, H. F., & Dhillon, I. S. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*, *32*.

Shen, L., Wei, Y., & Wang, Y. (2023). Gbt: Two-stage transformer framework for non-stationary time series forecasting. *Neural Networks*, *165*, 953–970.

Song, G., Zhao, T., Wang, S., Wang, H., & Li, X. (2023). Stock ranking prediction using a graph aggregation network based on stock price and stock relationship information. *Information Sciences*, Article 119236.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *27*.

Tian, G., Zhang, C., Shi, Y., & Li, X. (2024). Multiwavenet: A long time series forecasting framework based on multi-scale analysis and multi-channel feature fusion. *Expert Systems with Applications*, *251*, Article 124088.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Walker, G. T. (1931). On periodicity in series of related terms. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, *131*, 518–532.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., & Xiao, Y. (2022). Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). Etsformer: Exponential smoothing transformers for time-series forecasting. arXiv preprint arXiv:2202.01381.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2022). Timesnet: Temporal 2d-variation modeling for general time series analysis. arXiv preprint arXiv:2210.02186.

Xiao, Y., Liu, Z., Yin, H., Wang, X., & Zhang, Y. (2024). Stformer: A dual-stage transformer model utilizing spatio-temporal graph embedding for multivariate time series forecasting. *Journal of Intelligent & Fuzzy Systems*, 1–17.

Yule, G. U. (1927). Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, *226*, 267–298.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11121–11128).

Zhang, C., Wang, X., Zhang, H., Zhang, H., & Han, P. (2021). Log sequence anomaly detection based on local information extraction and globally sparse transformer model. *IEEE Transactions on Network and Service Management*, *18*, 4119–4133.

Zhang, Y., & Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., & Li, J. (2022). Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. arXiv preprint arXiv:2207.01186.

Zhao, T., Ma, X., Li, X., & Zhang, C. (2023). Mpr-net: Multi-scale pattern reproduction guided universality time series interpretable forecasting. arXiv preprint arXiv:2307.06736.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning* (pp. 27268–27286). PMLR.

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. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11106–11115).