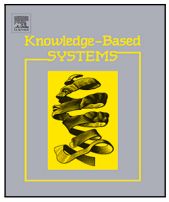
Knowledge-Based Systems 285 (2024) 111321

Contents lists available at ScienceDirect

Knowledge-Based Systems

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



Graphformer: Adaptive graph correlation transformer for multivariate long sequence time series forecasting

Yijie Wang, Hao Long, Linjiang Zheng *, Jiaxing Shang

College of Computer Science, Chongqing University, Chongqing 400044, China

ARTICLE INFO

Keywords:
Long sequence time series forecasting
Transformer
Dilated causal convolution
Self-attention
Graph convolution neural network

ABSTRACT

Accurate long sequence time series forecasting (LSTF) remains a key challenge due to its complex time-dependent nature. Multivariate time series forecasting methods inherently assume that variables are interrelated and that the future state of each variable depends not only on its history but also on other variables. However, most existing methods, such as Transformer, cannot effectively exploit the potential spatial correlation between variables. To cope with the above problems, we propose a Transformer-based LSTF model, called Graphformer, which can efficiently learn complex temporal patterns and dependencies between multiple variables. First, in the encoder's self-attentive downsampling layer, Graphformer replaces the standard convolutional layer with an dilated convolutional layer to efficiently capture long-term dependencies between time series at different granularity levels. Meanwhile, Graphformer replaces the self-attention mechanism with a graph self-attention mechanism that can automatically infer the implicit sparse graph structure from the data, showing better generality for time series without explicit graph structure and learning implicit spatial dependencies between sequences. In addition, Graphformer uses a temporal inertia module to enhance the sensitivity of future time steps to recent inputs, and a multi-scale feature fusion operation to extract temporal correlations at different granularity levels by slicing and fusing feature maps to improve model accuracy and efficiency. Our proposed Graphformer can improve the long sequence time series forecasting accuracy significantly when compared with that of SOTA Transformer-based models.

1. Introduction

Time series is a set of random variables in chronological order [1], which is usually the result of observing a potential process at a fixed sampling rate. Based on historical time series observations, time series forecasting aims to estimate and predict the variable state at future time steps, which is widely used in various fields, including statistics, operations research, and traffic accident prevention [2–4]. It is often necessary to obtain forecasts for multiple consecutive future time steps based on historical observations, i.e. multi-step forecasting. In contrast to classical multi-step forecasting, long-sequence time series forecasting encourages the longest possible range of forecasts. The growth in forecast horizon brings with it additional complexities such as error accumulation, reduced accuracy, and increased uncertainty [5]. Therefore, accurate long-sequence time series forecasting remains a challenging issue to the time series forecasting research community.

Similar to forecasting tasks in other fields, time series forecasting has benefited from the development of deep neural network research in recent years [6–8] and has been widely used in decision-making systems such as traffic accident prevention, cloud computing resource reprogramming and optimal investment strategies. In particular, based on the success of the Transformer [9] model in natural language processing, attention-based models have also made great progress in time series forecasting. However, existing methods do not effectively exploit the potential dependencies between variables. Recently proposed Transformer-based models, including Reformer [7], LogTrans [8], and Informer [6] focus on modeling temporal characteristics and do not explicitly model pairwise dependencies between different series, which weakens the interpretability and predictive power of the models.

In real life, sensors may track changes in traffic flow, energy use, temperature, and a variety of other types of data. Time series collected by several sensors can be combined to create multivariate time series. Multivariate time series forecasting [10] techniques implicitly presuppose that variables are connected to one another and that each variable's future state depends not only on its past value but also on other variables. The time series of any or all of the variables may need to be predicted by researchers. In transportation systems, knowing the

* Corresponding author.
E-mail addresses: 202214021045t@stu.cqu.edu.cn (Y. Wang), tlhgs286000@163.com (H. Long), zlj_cqu@cqu.edu.cn (L. Zheng), shangjx@cqu.edu.cn (J. Shang).

https://doi.org/10.1016/j.knosys.2023.111321
Received 25 September 2023; Received in revised form 12 December 2023; Accepted 20 December 2023
Available online 23 December 2023

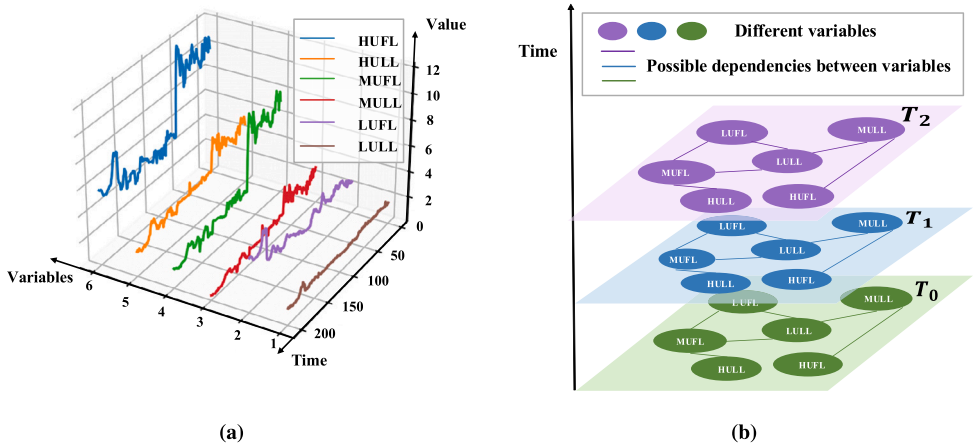


Fig. 1. Multivariable time series modeling for ETT datasets. Sensors can record different characteristics of power transformers, where HUFL, HULL are variables describing power resources. The time series recorded by different sensors can form a multivariate time series as shown in Fig. 1(a). Considering different variables as nodes of the graph and dependency pairs between variables as edges of the graph, the multivariate time series modeling is shown in Fig. 1(b).

future trend of traffic flow can help to reduce the occurrence of traffic accidents and congestion. However, the future values of traffic flow are not only related to their past values. They are also related to other variables (e.g., average speed of vehicles, space occupancy). Regardless of the above scenario, mining the implied correlations between different variables can improve the overall accuracy of time series forecasting.

How to extract the interaction features between different variables is a very challenging problem. A graph is a data type that describes the relationships between different entities in a network. Considering different variables as nodes of a graph and pairs of dependencies between variables as edges [11], the multivariate time series of the power resource dataset ETT, for example, can be naturally modeled as a structure as shown in Fig. 1, in which HUFL,HULL, and so on, are all variables describing the power resources. Recently, GCN [12–14] has achieved great success in processing graph-structured data. It is a promising approach for multivariate time series prediction using GCN to aggregate information from neighboring nodes and explore the implicit dependencies between time series while maintaining the time trajectory of the series.

However, GCN-based time series forecasting methods require a graph structure to be predefined by a distance or similarity metric. In most cases, multivariate time series do not have an explicit graph structure. The relationships between these variables are not provided as a priori knowledge but need to be mined from the data. Even for time series where explicit graph structures exist, these graphs rely heavily on the corresponding domain knowledge, are not applicable when the domain knowledge is unknown. It may also lead to the omission or bias of some implicit information due to manual definition factors.

Despite the existence of distance-based explicit graphs structures or data discovery-based implicit graph structures, existing graph construction methods result in a dense complete graph. Actually, variables are not strongly correlated over all time scales. For example, the traffic state of a road is largely influenced by its neighboring roads, while the influence from distant roads is relatively small. Moreover, feature interaction from uncorrelated variables can even introduce unwanted noises, leading to inferior predictive performance of the model while increasing the computational cost of the model [14]. On the other hand, existing graph convolution methods tend to learn temporal patterns shared among all nodes. Due to the different attributes of various data sources, variables may exhibit dissimilarity or even completely opposite patterns [15], and the shared parameter space is not conducive to the accurate extraction of specific node-level features.

In addition, most of the existing methods use temporal and spatial modules respectively to capture the trends and the dependencies between variables. In fact, the spatio-temporal correlation of time series data is so complex that the spatio-temporal features are interrelated and affect each other. In order to improve the accuracy, Transformer needs to be tightly integrated with GCN to jointly model the temporal and spatial features of the sequence.

To address the above challenges and limitations, we take into consideration extracting feature interactions between different variables and propose a Transformer-based model Graphformer for long sequence time series forecasting. Basically, Graphformer still follows the encoder–decoder structure, but in the self-attention down-sampling module, we use dilated causal convolution instead of the standard convolution layer, which can efficiently capture long-term dependencies between time series at different granularity levels. Second, we design a multi-scale feature fusion operation that slices feature maps through a multi-scale pyramid network. It extracts time dependencies of sequences at different scale levels, and finally performs the fusion operation to capture cross-scale feature information. This achieves equal or higher prediction accuracy while saving the computational cost of adding additional encoders. On this basis, we replace the self-attention mechanism with a graph self-attention mechanism that can automatically infer the implicit sparse graph structure from the data, showing better generality for time series without explicit graph structure and learning implicit spatial dependencies between sequences.

The main research contributions of this work are summarized as follows.

- We propose a Transformer-based model named Graphformer that follows an encoder–decoder structure, incorporating a sparse graph self-attention mechanism, multi-scale feature fusion, and a temporal inertia mechanism. It can achieve equal or higher prediction accuracy.
- We change the sparse self-attention mechanism to the sparse graph self-attention mechanism, which considers the spatial correlation between variables at different scales and realizes the close combination of Transformer architecture and GCN. Its internal adaptive graph convolution module does not rely on any prior knowledge and has better universality for multivariate time series without explicit graph structures.
- Extensive experiments are conducted on three real datasets in the fields of energy, transportation, and disease to evaluate the proposed mode. The experimental results show that our method outperforms the state-of-the-art approaches.

2. Related work

In this section, we review state-of-the-art approaches for long sequence time series forecasting. We also provide a brief summary of the

attention mechanism separately, as it is an essential foundation of the Transformer technique.

2.1. Long sequence time-series forecasting

Existing methods for time-series forecasting can be roughly grouped into four categories: traditional statistical time-series models, machine learning, traditional neural networks, and attention mechanisms. Although still being widely used, traditional time series forecasting models, such as Moving Average models (MA), Auto-Regressive models (AR), and Auto Regressive Moving Average models (ARMA) [16–18], are designed to estimate the model parameters based on determining the time series parametric model. Besides, these traditional time series models also require future trend data to be stable. Especially, the prediction results obtained from massive data samples are not ideal, which is not suitable for long sequence time series forecasting.

To tackle the aforementioned challenges, machine learning-based models have attracted the researchers' attention. Mellit et al. [19] used the support vector regression (SVR) model to find a hyperplane that minimizes the deviation of all samples based on regression loss and verified the effectiveness of the method in financial prediction scenarios and meteorological prediction scenarios. Das and Ghosh [20] integrated Bayesian Network (BN) into the existing network structure and verified the effectiveness of the method in the field of meteorological prediction. Yu et al. [21] proposed a temporal regularized matrix factorization (TRMF) technique, which uses regularization operators to express the dependencies of time series data along the time dimension, to deal with the high-dimensional time series prediction. There are other machine learning methods, such as Gaussian processes and automatic Bayesian covariance discovery [22]. Since machine learning methods require feature engineering to model time series data, there are limitations in fitting high-dimensional real-world data, and the generalization ability is limited. In addition, traditional machine learning methods do not take into account the correlation between time steps and focus on temporal feature extraction, which makes it difficult to improve the accuracy of long sequence time series forecasting.

With the development of deep neural network technology, some deep neural network models for speech recognition and computer vision are used for time series prediction problems. Sun et al. [23] proposed a deep memory network, MTnet, which models multivariate time series as a memory matrix and incorporates an encoder–decoder structure to capture the relationships between variables as well as long-term and short-term temporal dependencies. Salinas et al. [24] proposed an autoregressive recurrent neural network model DeepAR, which uses a stacked recurrent neural network (RNN) to model the probability distribution of future sequences. Lai et al. [25] proposed a multivariate time series modeling framework LSTnet, which uses Convolutional Neural Networks (CNN) to extract short-term dependencies among variables and uses RNN with jump connection structure to extract long-term dependencies in sequences, taking good advantage of CNN and RNN. Bai et al. [26] proposed a Temporal Convolution Network (TCN), which uses causal convolution and residual connection structure to simulate temporal causality. Compared with RNN and its variants, TCN has the strength of fast training speed, fewer parameters, and a stable gradient.

Naturally, traditional statistical models, machine learning models, and traditional neural network models for time series prediction all have their own advantages and limitations. Specifically, traditional statistical time series models perform well in mining the implicit linear relationship from data. Machine learning models are unique in dealing with the nonlinear relationship from low-dimensional data. And traditional neural network models make it possible to model complex spatial–temporal correlations among time series. However, it is difficult to determine whether the long sequence series data has a linear relationship. Also, traditional neural networks make it possible to model complex spatial–temporal correlations among time series, but with the increase in the number of network layers, the models suffer from the drawbacks of gradient vanishing or gradient explosion when dealing with long sequence scenes.

Different from the above methods, the attention mechanism can focus on important information with high weights and ignore irrelevant information with low weights, and the weights can be continuously adjusted so that important information can be selected in different situations, which has higher scalability and robustness. The core idea of the attention mechanism introduced is to calculate the influence of different positions in the input sequence on the output results, so as to pay attention to different time steps of different variables. For example, Shih et al. [27] proposed a multivariate time series modeling framework TPA-LSTM that combines attention mechanisms, long-term and short-term neural networks to extract corresponding time patterns. Based on this idea, a large number of scholars combined the attention mechanism and neural network method to construct a model to study the long sequence time series prediction problem and confirmed the strong memory ability of the attention mechanism.

2.2. Attention mechanism

Among the attention-based methods, Transformer is a representative architecture. Compared with traditional models, the time series prediction accuracy of Transformer has been greatly improved, but the quadratic time and space complexity $\mathcal{O}(L^2)$ limits the model to accept longer time series.

Recent research attempts to improve the calculation of self-attention for the computational efficiency of Transformer. Li et al. [8] introduced local convolution into Transformer and proposed LogTrans which uses the logarithmic sparse attention to limited select the time step at an exponential interval, reducing the complexity to $\mathcal{O}(L \ln L)$. Kitaev et al. [7] proposed Reformer using local sensitive hashing to divide the sequence into multiple buckets, calculate attention separately, and combine the reversible residual structure to reduce the complexity to $\mathcal{O}(L \ln L)$. Zhou et al. [6] extended Transformer with probabilistic sparse attention and proposed Informer to improve the encoder–decoder structure which achieves $\mathcal{O}(L \ln L)$ time and space complexity.

Besides, Transformers have shown great performance in time series forecasting [28–31]. Especially, Wu et al. [31] presented the Autoformer with Auto Correlation mechanism to capture the series-wise temporal dependencies based on the learned periods. Afterwards, FEDformer [32] employed the mixture-of-expert design to enhance the seasonal-trend decomposition and presents a sparse attention within the frequency domain. Note that these attention-based models are based on the vanilla Transformer and try to improve the self-attention mechanism to a sparse version. In this paper, our proposed Graphformer is based on Informer for the LSTF problem.

3. Methodology

3.1. Problem formulation

Consider a time series as a series of continuous observations with length L, dimensionality d_x, and the same time interval, which can be expressed as $X_t = \{x_1^t, x_2^t, \ldots, x_L^t | x_i^t \in R^{d_x}\}$. If $d_x > 1$, then the time series is called a multivariate time series, where X can be represented by a $L \times D$ two-dimensional matrix:

$$X = \begin{bmatrix} X_{1,1}^t & X_{1,2}^t & \cdots & X_{1,d_x}^t \\ X_{2,1}^t & X_{2,2}^t & \cdots & X_{2,d_x}^t \\ \vdots & \vdots & \ddots & \vdots \\ X_{L,1}^t & X_{L,2}^t & \cdots & X_{L,d_x}^t \end{bmatrix} \tag{1}$$

where $X_i^t = [X_{i,1}^t, X_{i,2}^t, \ldots, X_{i,d_x}^t]$ represents the value of X_t observed at time i, $X_{i,k}^t$ represents the kth variable value of X_i^t observed at time i. In particular, when the variable dimensionality $d_x = 1$, the time series

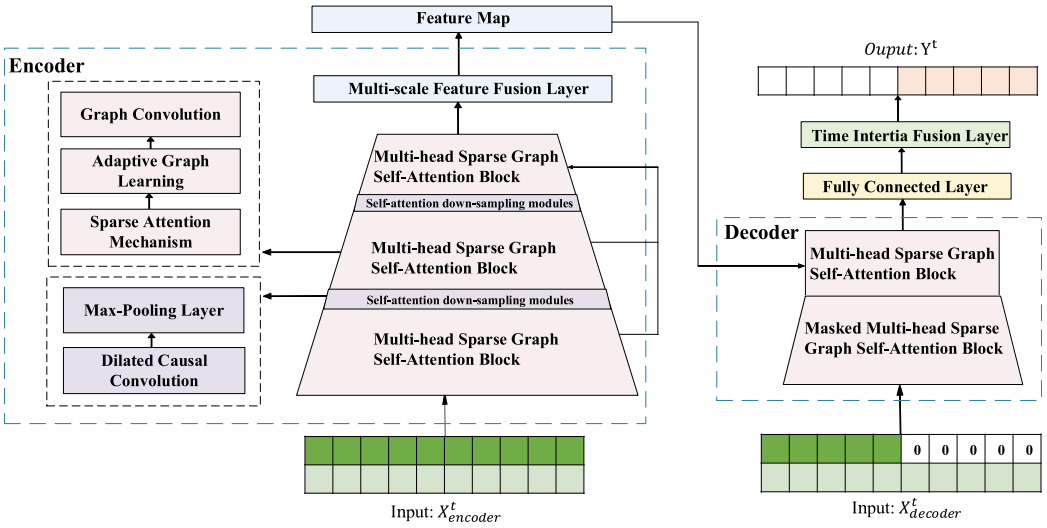


Fig. 2. Graphformer architecture. Graphformer follows the encoder–decoder structure, where the encoder stacks three multi-head sparse graph self-attention modules, two self-attention down-sampling modules and a multi-scale feature fusion layer. The decoder consists of a masked sparse graph self-attention module and a sparse graph self-attention module.

is called univariate time series. Obviously, multivariate time series can be regarded as a combination of multiple univariate time series, and the variables may influence each other. This paper mainly studies the prediction of long time series under multivariate conditions.

Graph G is a structure that describes the relationship between entities in the network. In this paper, we refer to the work of Graph WaveNet [33] to introduce an adaptive graph learning method to reflect the graph structure of sparse query matrices. Graph $G = (V, E)$ is composed of a set of vertex sets V and a set of edge sets E, which can be represented by an adjacency matrix A. If there is an edge e_{ij} between vertices v_i and v_j, then $A_{ij} > 0$, otherwise, if there is no edge, then $A_{ij} = 0$.

3.2. Overview architecture

Fig. 2 shows the architecture of the proposed Graphformer model, which follows the encoder–decoder architecture to extract the long-term temporal correlation among the input sequence. The encoder stacks three multi-head sparse graph self-attention modules, two self-attention down-sampling modules and a multi-scale feature fusion layer. The decoder consists of a masked sparse graph self-attention module and a sparse graph self-attention module.

In the encoder, each graph sparse self-attention module includes a sparse self-attention mechanism, an adaptive graph learning mechanism, and a graph convolutional network. The sparse self-attention mechanism computes the scaled dot product attention of the query-key. Afterwards, the adaptive graph learning mechanism captures the graph structure of the sparse query matrix and adaptively infers the hidden spatial dependencies in the sparse query matrix. Graph Convolutional Networks learn the features of nodes by aggregating and transforming their domain information on the basis of learning weights and predefined graph structures. For multivariate time series without explicit graph structures, the sparse graph self-attention module is also able to adaptively learn potential spatial correlations between sequences. Each two multi-headed sparse graph self-attention blocks are connected to each other by a self-attention down-sampling module. Graphformer builds on Informer's self-attention down-sampling module by replacing the standard convolutional layer with a dilated causal convolutional layer to efficiently capture long-term dependencies between time series at different granularity levels. A Max-pooling layer is followed that drastically reduces the network size.

In the field of CNN-based computer vision, finer-grained features are usually extracted using a feature pyramid network [34]. Yolov5 [35], a CNN network applied to target detection, uses a focus layer that takes feature maps from earlier networks and merges them with the final feature maps to obtain finer-grained information. We migrate this structure to the field of Transformer-based model to merge feature maps at different scales. Without adding additional encoders, the multi-scale feature fusion layer fuses three feature maps output from three sparse self-attentive blocks and then transforms them into a final output of appropriate dimensionality.

In the decoder, the elements to be predicated on the input sequence are padded with zeros. The learned features from masked multi-head self-attention blocks are fused with the feature map from the encoder. Then, the output from the decoder is fed into a fully connected layer and finally time patterns based on time period are discovered through the time inertia fusion layer and aggregated with the model output.

3.3. Self-attention down-sampling module

In order to extract deeper feature maps, several self-attention blocks are stacked, which brings higher time and space complexity. To address the issue, Graphformer utilizes Informer's self-attention down-sampling module by replacing the standard convolutional layer with a dilated causal convolutional layer to efficiently capture long-term dependencies between time series at different granularity levels. The dilated causal convolution samples the feature elements at a fixed step, pads zero before the elements along the time dimension to ensure the equal size of filters and conducts the convolution operation on the sampled feature elements. For each dilated causal convolution following the self-attention block, the convolution operation $DConv(\cdot)$ is formulated as Eq. (2).

$$DConv(x_n) = \begin{bmatrix} x_n \\ x_{n-i} \\ \cdot \\ \cdot \\ \cdot \\ x_{n-(s-1)\times i} \end{bmatrix} W \tag{2}$$

where $x_n \in R^d$ is the feature element of the sequence $X \in R^{L\times d}$ and $n \in \{0, 1, 2, \ldots, L\}$, s is the size of filter, $W \in R^{d\times d}$ is the learnable weight, and i is the dilation factor. As the number of layers increases, the convolution kernel of the ith dilated causal convolution layer skips $2^{i-1} - 1$ elements between two adjacent feature taps and slides over the input. In addition, according to the nature of causality, each element x

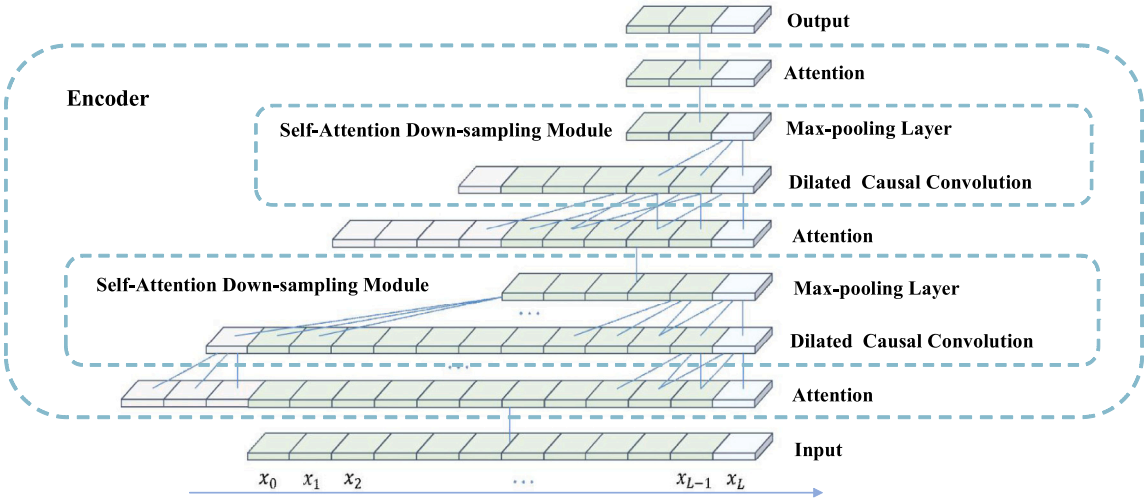


Fig. 3. The structure of the encoder network using self-attention down-sampling module is visualized, where the input is a time series of length L. The encoder contains three self-attention layers, with a down-sampling module in the middle of each of the two self-attention layers, and consists of a group of a dilated causal convolution layer and a max-pooling layer.

at time t is only convoluted with elements at or before t to ensure that there is no leakage from future information [15]. In particular, when $i = 1(2^0)$, the dilated causal convolution will degenerate to a normal causal convolution.

The feature maps from dilated causal convolution layer are obtained by the max-pooling layer for down-sampling, formulated as Eq. (3):

$$X_{i+1}^{t} = MaxPool(Dconv([X_i^t]_{MSA})) \tag{3}$$

where $[\cdot]_{MSA}$ represents the multi-headed sparse self-attention block, and $MaxPool(\cdot)$ represents the maximum pooling operation.

Fig. 3 shows the two dilated causal convolution layer with a kernel size of 3 in Graphformer encoder. Compared with standard convolution, it can capture larger receptive field and reduce the computational costs and memory overhead even if only two dilated causal convolutional layers are used. Besides, the combination of dilated causal convolution layer and the max-pooling layer not only eliminates the redundancies from the sparse self-attention mechanism block (caused by the value mean) and gives the dominant feature privilege, but also clips the input length of X_{i+1} to half than that of X_i, so that the memory overhead is also reduced to $\mathcal{O}(L \ln L)$.

3.4. Sparse graph self-attention mechanism

The sparse graph self-attention block consists of three parts: sparse self-attention mechanism, adaptive graph learning and graph convolution network, which aims to extract dominant attention and potential spatial dependence.

Sparse Self-Attention Mechanism: The probsparse self-attention in Informer is defined on receiving the tuple input (query, key, value) based on Transformer and performs the scaled dot-product as $\mathcal{A}(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}})V$, where $Q \in \mathbb{R}^{L_Q \times d}$, $K \in \mathbb{R}^{L_K \times d}$, $V \in \mathbb{R}^{L_k \times d}$ and d is the input dimension. Following the formulation above, the attention weight of the ith query can be converted into a probabilistic kernel smoother [36]:

$$Attention(q_i,K,V) = \sum_j p\left(k_j \middle| q_i\right) v_j = \sum_j \frac{k\left(q_i,k_j\right)}{\sum_l k\left(q_i,k_j\right)} v_j \tag{4}$$

Based on Informer, we use KL (Kullback–Leibler) divergence to measure the similarity between the probability $p(k_j|q_i)$ and the uniform distribution $q(k_j|q_i)$ of the ith query $KL(q_i,K)$. By calculating the KL divergence of all queries and ranking them in descending order, the first u queries $\bar{Q} \in \mathbb{R}^{u \times d_{sq}}$ are selected as the sparse query matrix where $u = clnL_Q$, and then the sparse self-attention score is calculated for all keys.

Adaptive graph learning: To further discuss the sparse self-attention mechanism, referring to the research work of Graph WaveNet [33], we introduce an adaptive graph learning method to reflect the graph structure G_{sq} of sparse query matrix $\bar{Q}$, and adaptively infer the hidden spatial dependencies in $\bar{Q}$. For G_{sq}, the process is:

$$G_{sq} = Softmax\left(Elu\left(E_{\bar{q}} - E_{\bar{q}}^T\right)\right) \tag{5}$$

where each row of $E_{\bar{q}} \in \mathbb{R}^{d_e \times d_{sq}}$ represents the embedding of a query, and d_e represents the embedded dimension. During training, the embedded $E_{\bar{q}}$ of a trainable query is randomly initialized and assigned to all nodes, the hidden spatial dependencies between each pair of queries are inferred by matrix multiplication, and the $E_{\bar{q}}$ is automatically updated by gradient during training.

Unlike Graph WaveNet, Graphformer selects the ELU activation function to eliminate weak connections:

$$f(x) = \begin{cases} \alpha \times (e^x - 1), x \le 0 \\ x, x > 0 \end{cases} \tag{6}$$

where α is the set parameter, the $(e^x - 1)$ function part makes the $ELU(\bullet)$ more robust to input noise, and the linear part enables the $ELU(\bullet)$ to avoid the gradient vanishing problem. The average value of $ELU(\bullet)$ output is close to zero, so its convergence speed is faster. Finally, softmax is used to normalize and generate the adaptive adjacency matrix G_{sq} of $\bar{Q}$, the number of nodes d_{sq} of G_{sq}. In this way, the adaptive adjacency matrix of sparse query matrix $\bar{Q}$ can be learned without any prior knowledge, and the dominant query is mapped to a low-dimensional dense graph structure, which can be well extended to the modeling tasks of various time series data.

Graph Convolution Network: For the time series input $X \in \mathbb{R}^{L \times d}$, GCN [34] can be well approximated by the first-order Chebyshev polynomial expansion, and extended to high-dimensional GCN. The graph convolution operation GC based on spectral graph can be defined as:

$$X_g = GC \times X = \widetilde{A} X \Theta \tag{7}$$

where $X_g \in \mathbb{R}^{P \times d}$ represents the output of GCN, $\widetilde{A} \in \mathbb{R}^{d \times d}$ represents the self cyclic normalized adjacency matrix of the graph, and $\Theta \in \mathbb{R}^{L \times P}$ represents the parameter matrix. After expanding GC, we have:

$$X_g = GC \times X = \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) X\Theta + b \tag{8}$$

where I_N represents the unit matrix, $D \in \mathbb{R}^{d \times d}$ represents the degree matrix of the graph, $A \in \mathbb{R}^{d \times d}$ represents the adjacency matrix of

the graph, $D_{ii} = \sum_j A_{ij}$, $\Theta \in \mathbb{R}^{L\times P}$ and $b \in \mathbb{R}^P$ represent the trainable weights and biases, and $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ represents the normalized Laplacian matrix. Graphformer uses the above self-adaptively generated graph structure G_{sq} to replace $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ in Eq. (8) to extract the dependencies of the dominant query, and introduces the node adaptive parameter learning module proposed by AGCRN to mine the unique spatial characteristics of each node. The calculation process is as follows:

$$X_g = GC \times X = \left(I_{sq} + G_{sq}\right) X E_{\bar{q}} W_g + E_{\bar{q}} b_g \tag{9}$$

where I_{sq} represents the unit matrix, W_g and b_g represent the trainable hyperparameters. The node adaptive parameter learning module solves the overfitting phenomenon caused by too large Θ which is difficult to optimize due to the allocation of parameters for each node. Specifically, by drawing on the idea of matrix decomposition, GCN learns two relatively small matrices W_g and b_g to replace the weights $\Theta \in \mathbb{R}^{(k\times u)\times(k\times u)\times d_{sq}}$ and the deviations $b \in \mathbb{R}^{(k\times u)\times d_{sq}}$. Here $\Theta = E_{\bar{q}} \cdot W_g$, $b = E_{\bar{q}} \cdot b_g$, where $W_g \in \mathbb{R}^{(k\times u)\times(k\times u)\times d_e}$ represents the weight pool, $b_g \in \mathbb{R}^{(k\times u)\times d_e}$ represents the deviation pool, $E_{\bar{q}} \in \mathbb{R}^{d_e\times d_{sq}}$ represents the trainable query embedding, d_e represents the embedded dimension, d_{sq} represents the number of nodes in the graph, that is, the sequence feature dimension, and k represents the number of multi-head sparse attention heads. Through this decomposition method, each node extracts its own parameters from the shared weight pool W_g and deviation pool b_g according to $E_{\bar{q}}$, reducing the parameter size and training burden while helping the model learn the hidden and unique spatial dependencies of time series.

To sum up, in order to find the hidden association between the nodes on the sparse query matrix $\bar{Q}$, the adaptive graph learning layer calculates the adjacency matrix of the graph, and then uses the matrix as the input of the GCN, which is extended to a sparse query matrix $\bar{Q}_g$ with potential spatial correlation:

$$\bar{Q}_g = GC \times \bar{Q} = \left(I_{sq} + Softmax\left(Elu\left(E_{\bar{q}} \cdot E_{\bar{q}}^T\right)\right)\right)\bar{Q}\Theta \tag{10}$$

where $\bar{Q}_g \in \mathbb{R}^{d_{sq}\times P}$, Θ denotes the set of all trainable parameters. According to the approximate method of measuring sparsity in Graphformer, the similarity between the probability distribution $p\left(k_j \middle| q_{ig}\right)$ and the uniform distribution $q\left(k_j \middle| q_{ig}\right)$ of the ith query in the sparse query space matrix $\bar{Q}_g$, that is, the sparsity MaxMean $\left(q_{ig}, K\right)$ of the ith query can be rewritten as follows:

$$MaxMean(q_{ig}, K) = max_j\left\{\frac{q_{ig}k_j^T}{\sqrt{d}}\right\} - \frac{1}{L_K}\sum_{j=1}^{L_K}\frac{q_{ig}k_j^T}{\sqrt{d}} \tag{11}$$

In practice, Graphformer only needs to randomly sample $U = L_Q \ln L_K$ dot product pairs (the remaining dot product is assigned a value of 0), calculate the maximum mean metric $MaxMean\left(q_i, K\right)$ and arrange them from large to small, select the first u queries as the sparse query matrix $\bar{Q}$, generate $\bar{Q}_g after$ adaptive graph learning and GCN extraction of spatial dependence, update $MaxMean\left(q_{ig}, K\right)$, and then calculate all key to obtain the sparse graph self-attention score SparseGraphAttention :

$$SparseGraphAttention(Q, K, V) = Softmax(\frac{\bar{Q}_g K^T}{\sqrt{d}})V \tag{12}$$

Compared with the traditional sparse self-attention mechanism, the sparse graph self-attention mechanism calculates the hidden spatial correlation in the learned high-dimensional space, and combines with the multi-head self-attention to explore multiple spatial dependencies in the potential relational space of different scales.

3.5. Multi-scale feature fusion

Inspired by the design of focus layer in computer vision [35], our Graphformer model introduces the design into the encoder, named multi-scale feature fusion, applied on long sequence time series prediction. Fig. 4 shows how the multi-scale feature fusion layer is designed in Graphformer encoder, where three feature maps at different levels of granularity from each self-attention block are fused into one feature map. Specifically, when the encoder stacks n self-attention blocks, each self-attention block will generate a feature map, where the length of the kth $(k = 1, 2, \ldots, n)$ feature map is $L/2^{k-1}$, and the dimension is d. In order to fuse the feature maps at different levels of granularity, the kth feature map is input into the multi-scale feature fusion to be sliced into 2^{n-k} feature maps with the length of $L/2^{n-1}$. After the slicing operation, all the feature maps are concatenated along the dimension into a fusion feature map with a dimension of $(2^n - 1)\times d$. Finally, a 1D-convolution layer is added to ensure that the entire encoder outputs a feature map with the appropriate dimension d to form the hidden state $H^t \epsilon \mathbb{R}^{\frac{L}{4}\times d}$.

Note that the function of the multi-scale feature fusion operation is similar to that of the distilling operation in Informer. However, the distilling operation needs to construct many additional encoders, the number of which is the same as the number of self-attention blocks, resulting in considerable computational overhead. Our Graphformer model requires only one encoder using multi-scale feature fusion, which can reduce the computational cost and extract various fine-grained features.

3.6. Time inertia fusion layer

The time-inertial block directly takes the subsequence with the nearest length of the input sequence X^t as the prediction range P as the output, that is, $\bar{Y}^t = \left\{x^t_{L-P+1}, x^t_{L-P+2}, \cdots, x^t_L\right\}$. The specific process is shown in Fig. 5. It is not difficult to see that time inertia requires that the length of the predicted sequence to be less than the length of the input sequence, i.e. $P \le L$. In the practical application scenarios of long-time series forecasting, the data set is usually several orders of magnitude larger than the prediction range P, which is easy to implement. After obtaining the time inertia output, the output of the deep learning model is weighted and summed to obtain the final prediction result.

Similarity between predicted and input sequences is essential for time inertia. For long time series, the temporal pattern of the predictable series is more stable and reflects a more complete periodicity, which is manifested by the similarity in phase and amplitude between the predicted and input series. At this point, applying temporal inertia may yield a large improvement in prediction performance, especially if the prediction range is divisible by the time series period length.

Sophisticated deep learning models take this inertia into account to some extent by embedding cycle timestamps as input sequences. However, the non-linear nature of the self-attention mechanism and the convolution operation results in their outputs being insufficiently sensitive to inputs in the nearest range [25]. It means that temporal inertia will play an important role in certain long sequential time series cycles with relatively gentle variations. Therefore, it can be beneficial to directly utilize the temporal inertia information to improve the final prediction results.

4. Experiments

4.1. Datasets

We extensively conduct experiments and evaluate the proposed Graphformer on three real-world datasets, including three application scenarios: traffic, energy, and illness.

- PeMs (Caltrans Performance Measurement System): It is a traffic dataset collected by the California department of transportation from 2015 to 2016. It describes road occupancy rates (between 0 and 1) measured by sensors on 862 freeways in the San Francisco Bay Area, with a total of 17,544 records.

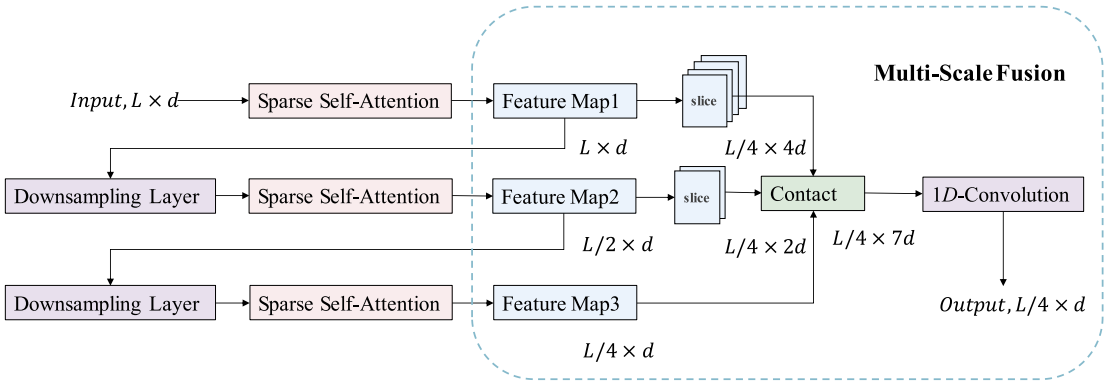


Fig. 4. An encoder structure stacked with three self-attention blocks is demonstrated, each of which will generate a feature map, and multi-scale feature fusion can merge feature maps at different scales, allowing access to finer-grained information.

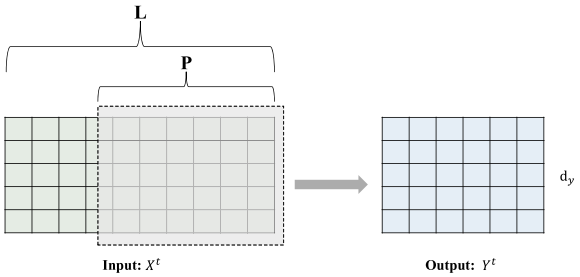


Fig. 5. Schematic diagram of the inputs and outputs of the temporal inertia mechanism.

Table 1
Description of the three datasets (PeMs,ETTm1 and ILI) including number of samples, number of nodes and sampling rate.

Dataset	Number of samples	Number of nodes	Sampling rate
PeMs	17,544	862	1 h
ETTm1	69,680	7	15 min
ILI	966	7	1 week

- ETTm1 (Electricity Transformer Temperature) : It is a dataset of long-term deployment of power resources recorded between 2016 and 2018 in two cities in China, and it describes information on seven power load variables such as oil temperature, with a total of 69,680 records.
- ILI (Influenza-like Illness): It consists of weekly data on influenza-like illness patients recorded by the U.S. Centers for Disease Control and Prevention from 2002 to 2021. It describes the proportion and number of patients in each category. Each data item consists of 7 variables, with a total of 966 items.

Table 1 summarizes the statistics of these three datasets. We first use Z-score normalization to process the original data as the distribution with zero mean and unit variance, so as to reduce the influence of different values' magnitude in data. Then the datasets are divided into the training set, the validation set, and the test set along the time dimension: For the PeMs dataset, the training set contains the first 14 months of samples, the validation set contains the next 5 months of samples after the first 14 months, and the test set contains the last 5 months samples. For the ETTm1 dataset, the training set contains the first 12 months' data, the validation set contains the next 4 months' data, and the test set contains the last 4 months' data. For the ILI dataset, the training set contains the first 14 years' samples, the validation set contains the next 2 years' samples after the first 14 years, and the test set contains the last 4 years' samples.

4.2. Baseline methods

The following 7 time-series forecasting methods are used for a fair comparison, including LSTnet, Reformer, LogTrans, LSTMa, Informer, Pyraformer and Dlinear.

- LSTMa [37]: A deep learning model combines LSTM and attention mechanism. After generating hidden vectors by recurrent neural networks, attention is used to select the relevant parts of the original sequence.
- LSTnet [38]: A deep learning model combines convolutional neural networks and recurrent neural networks to extract long- and short-term dependencies among variables and model complex temporal-dependent patterns in multivariate time series prediction tasks.
- Reformer [7]: A Transformer-based variant model uses local-sensitive hashing to divide the sequence into multiple buckets, calculates attention separately, and combines the reversible residual structure to further reduce memory overhead.
- LogTrans [8]: A Transformer-based variant model that utilizes convolutional operations to replace linear transformations in the self-attention mechanism, thereby introducing local information, while selecting sparse attention in exponential steps to reduce computational effort.
- Informer [6]: A variant model based on Transformer utilizes probsparse self-attention mechanism to replace the conventional self-attention mechanism, and proposes generative decoder to change the decoding method to directly output the result in one step, which greatly improves the prediction speed of the long sequence.
- Dlinear [39]: A model uses a combination of a location coding strategy with a linear layer. It first decomposes the raw data into moving average components, and seasonal trend components. Then two single linear layers are applied to each component and the two features are summed to get the final prediction.
- Pyraformer [28]: A variant model based on Transformer introduces the pyramidal attention module in which the inter-scale tree structure summarizes features at different resolutions and the intra-scale neighboring connections model the temporal dependencies of different ranges.

4.3. Experimental settings

In this paper, the mean square error (MSE) and mean absolute error (MAE) are used to evaluate the performance of different prediction models, formulated as Eq. (13) and Eq. (14):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^p)^2 \tag{13}$$

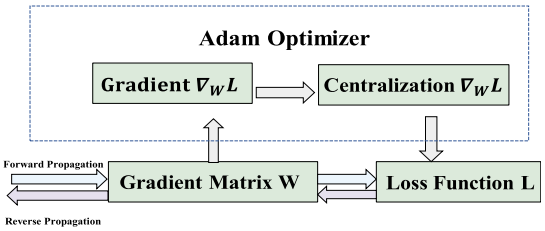


Fig. 6. Gradient centralization, which computes the mean of the gradient vector directly and concentrates the gradient to the zero mean, and is a projective gradient descent method with bounded loss function effects.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - {y_i}^p\right| \tag{14}$$

where n represents the length of the predicted sequence, y_i represents the true value, and ${y_i}^p$ represents the predicted value. MSE is sensitive to outliers, reflecting the degree of dispersion of samples. MAE represents the mean of the absolute error between the predicted value and the true value. The smaller two metrics are, the higher the prediction accuracy of the model is. In addition, in the multivariate time series prediction task, the mean of MSE/MAE of all variables on multiple repeated experiments are treated as the final evaluation results.

All experiments are based on the Pytorch framework, on an Intel Xeon Gold 6133 @ 2.50 GHz CPU, 128 GB memory, NVIDIA Tesla V100 32 GB GPU server. In addition, we improve the Adam optimizer by using gradient centralization. As shown in Fig. 6, appropriate optimization strategies are conducive to the training of large-scale data sets on deep neural networks. Gradient centralization is proposed by Yong et al. [40] This technique directly calculates the average value of the gradient vector and concentrates the gradient on the zero means. It is a projection gradient descent method with the effect of the constraint loss function. The details of the gradient centralization and other parameter settings are described in Appendix.

4.4. Experiment results

In the multivariate time series forecasting task, we compare the performance of the proposed Graphformer model with other baseline models. The results are shown in Table 2. In this table, the best results are in bold while the second best results are underlined. It can be observed that compared with the traditional deep learning models LSTMa and LSTnet, Graphformer reduces the MSE on ETTm1 by an average of 34.78% (with a prediction length of 24), 37.31% (with a prediction length of 48), 37.11% (with a prediction length of 96), and 22.09% (with a prediction length of 288), which proves the significant advantages of the Transformer's model has a significant advantage in solving the multivariate time series prediction problem. In addition, Graphformer's performance is better than the Transformer-based models, including Reformer, LogTrans, Informer and Pyraformer. In the PeMs dataset, the Graphformer model outperforms Dlinear, but in the other two datasets, the Graphformer model slightly underperforms Dlinear. Transformer-based models mainly utilize sub-regression for forecasting, and the long-term forecasting ability of forecasting solutions using autoregression is usually poor due to the unavoidable cumulative effect of errors. DLinear decomposes the time series into a trend series and a periodic series, and uses two single-layer linear networks to model both series for forecasting tasks, with better performance.

4.5. Ablation studies

In order to verify the effectiveness of the key components in the model proposed in this paper that are conducive to improving the prediction results, the following models are designed based on Graphformer.

Table 2
Multivariate long-sequence time-series forecasting results with our proposed Graphformer model and other baselines models. We use prediction lengths $T \in \{24, 36, 48, 60\}$ for ILI dataset, $T \in \{24, 48, 96, 288\}$ for ETTm1 dataset and $T \in \{96, 192, 336, 720\}$ for the PeMs dataset. The best results are in bold and the second best are underlined.

Datasets	Methods	MSE				MAE			
		96	192	336	720	96	192	336	720
PeMs	LSTMa	0.843	0.847	0.853	1.5	0.453	0.453	0.455	0.805
	LSTnet	1.107	1.157	1.216	1.481	0.685	0.706	0.73	0.805
	Reformer	0.732	0.733	0.742	0.755	0.423	0.42	0.42	0.423
	LogTrans	0.684	0.685	0.733	0.717	0.384	0.39	0.408	0.396
	Informer	0.735	0.702	0.786	0.89	0.413	0.389	0.436	0.491
	Pyraformer	0.867	0.869	0.881	0.896	0.468	0.467	0.469	0.473
	Dlinear	0.65	**0.598**	**0.605**	**0.645**	0.396	0.37	0.373	**0.394**
	Graphformer	**0.617**	0.619	0.624	**0.645**	**0.374**	**0.369**	**0.347**	**0.394**
	Methods	MSE				MAE			
		24	48	96	288	24	48	96	288
ETTm1	LSTMa	0.621	1.392	1.339	1.74	0.629	0.939	0.913	1.124
	LSTnet	1.968	1.999	2.762	1.257	1.17	1.215	1.542	2.076
	Reformer	0.724	1.098	1.433	1.82	0.607	0.777	0.945	1.094
	LogTrans	0.419	0.507	0.768	1.462	0.412	0.583	0.792	1.32
	Informer	0.362	0.522	0.699	1.065	0.375	0.524	0.648	0.845
	Pyraformer	0.543	0.557	0.754	0.908	0.51	0.537	0.655	1.124
	Dlinear	0.345	**0.38**	**0.413**	**0.474**	**0.372**	**0.389**	**0.413**	**0.453**
	Graphformer	**0.317**	0.497	0.532	0.778	0.378	0.499	0.517	0.652
	Methods	MSE				MAE			
		24	36	48	60	24	36	48	60
ILI	LSTMa	5.914	6.631	6.736	6.87	1.734	1.845	1.857	1.879
	LSTnet	6.026	5.34	6.08	5.548	1.77	1.668	1.787	1.72
	Reformer	4.44	4.783	4.832	4.882	1.382	1.448	1.465	1.483
	LogTrans	4.48	4.799	4.8	5.278	1.444	1.467	1.468	1.56
	Informer	5.829	4.991	4.994	5.701	1.712	1.482	1.487	1.59
	Pyraformer	7.394	7.551	7.662	7.931	2.012	2.031	2.057	2.1
	Dlinear	**2.398**	**2.646**	**2.614**	**2.804**	**1.04**	**1.088**	**1.086**	1.464
	Graphformer	3.958	3.653	3.531	3.737	1.308	1.286	1.267	**1.289**

- Proformer: The Proformer is a model for Graphformer to replace the sparse graph self-attention mechanism with the sparse self-attention mechanism. The Proformer model of dilated causal convolution is adopted, and the encoder is connected by multi-scale feature fusion operation.
- Proformer w/GC: The Proformer model with parameter optimization using the gradient-centered improved Adam optimizer.
- Proformer w/AGCN: Proformer model integrating adaptive graph learning and GCN.
- Inertia: The model directly use the time inertia output as the prediction result.
- Graphformer: Graphformer model adopts a sparse graph self-attention mechanism, introduces time inertia fusion operation, and uses gradient-centered improved Adam optimizer to optimize model parameters.

Fig. 7 shows the ablation experimental results of Proformer and Graphformer on the ETTm1 dataset. It can be observed that the Proformer w/GC trained by the gradient-centered improved Adam optimizer improves the prediction accuracy of the model and achieves about 0.98% performance improvement. Through the introduction of adaptive graph convolution module, Proformer w/AGCN has better performance than Proformer, reflecting that the sparse graph self-attention mechanism can capture the potential and complex spatial correlation between variables from deep representation. However, the improvement of model performance is limited, which may be because sparse attention destroys the correlation between some variables and blocks the propagation of information flow between isolated but interdependent nodes. On the other hand, GCN with node adaptive parameter learning brings higher complexity, which makes the model difficult to optimize.

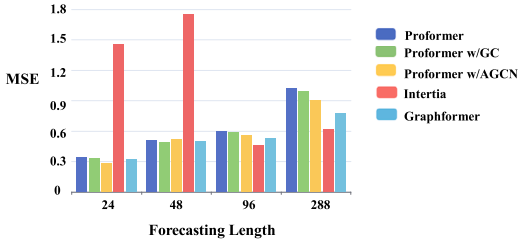


Fig. 7. Ablation study of the Graphformer components on the ETTm dataset. 4 cases are included: Proformer, Proformer w/GC, Proformer w/AGCN and Inertia. And MSE is used as a measure of model performance is brought up in Section 4.3.

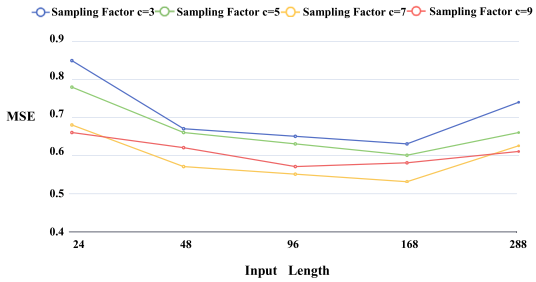


Fig. 8. The parameter sensitivity of sampling factor c in Proformer w/AGCN on the ETTm dataset. By fixing the prediction range to 96, the input length of the encoder is increased in the form of {24,48,96,168,288}, and the size of the sampling factor is increased in the form of {3,5,7,9}.

When the prediction range is 24 and 48, the prediction performance of Inertia is poor, because the sampling rate of the ETTm1 data set is 15 min, the short prediction range is difficult to reflect the periodic pattern, and the time inertia seriously affects the prediction results. However, when the prediction range is between 96 and 288, the improvement of Inertia prediction performance is very significant. At this time, the prediction range completely covers the clear periodic mode (1 day) in the data set, and the time inertia of obtaining similar phases greatly improves the prediction accuracy.

4.6. Parameter sensitivity

In order to explore the influence of the size of the sampling factor c in the sparse graph self-attention mechanism on the prediction performance of Graphformer, by fixing the prediction range to 96, the input length of the encoder is increased in the form of {24,48,96,168,288}, and the size of the sampling factor is increased in the form of {3,5,7,9}. The parameter sensitivity experiment is carried out on the ETTm1 dataset under multivariate conditions. Notably, this group of experiments used the Proformer w/AGCN model to eliminate the effect of time inertia, as shown in Fig. 8, It can be observed that the model achieves the best performance when the sampling factor is 7. Smaller or larger sampling factors will reduce the prediction accuracy of the model, because the smaller sampling bandwidth may not be enough to propagate effective information between potential neighbor nodes, which destroys the integrity of the sparse graph self-attention. The attention score of the contribution of the redundant dot product caused by the larger sampling bandwidth can be ignored. According to the experimental results of parameter sensitivity, the sampling factor c of Graphformer is set to 7.

Another key parameter in the self-attention mechanism of sparse graphs is the embedding dimension d_e, which not only affects the quality of adaptively generated graphs but also determines the parameter diversity of node learning. By fixing the prediction range to 96, the encoder input length increases in the form of {24,48,96,168,288}, and the embedding dimension increases in the form of {16,32,48,64}, the

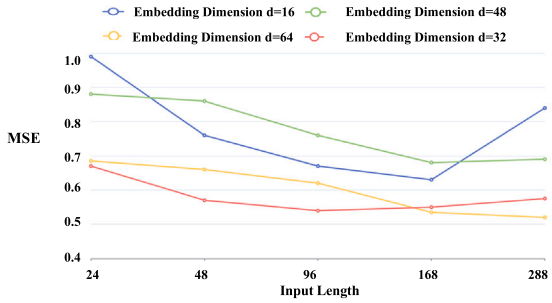


Fig. 9. The parameter sensitivity of embedding dimension in Proformer w/AGCN on the ETTm dataset. By fixing the prediction range to 96, the encoder input length increases in the form of 24,48,96,168,288, and the embedding dimension increases in the form of {16,32,48,64}.

parameter sensitivity experiment is carried out on the ETTm1 dataset under multivariate conditions, and the results are shown in Fig. 9. It can be observed that with the increase of embedding dimension, the prediction accuracy of the model will be improved to a certain extent, which indicates that too small embedding dimension will limit the extraction of potential correlation between variables; when the embedding dimension is further increased, this performance gain begins to flatten. This is because too large an embedding dimension will increase the complexity of the model, making the model difficult to train and more prone to overfitting. According to the results of parameter sensitivity experiments, this paper sets the embedding dimension d_e of Graphformer to 32.

5. Conclusion

In this paper, we studied the long sequence time series forecasting (LSTF) problem, and proposed Graphformer to predict long sequences time series. Specifically, we improve the sparse self-attention mechanism to a sparse graph self-attention mechanism, which considers the spatial correlation between variables at different scales and realizes the close integration of Transformer architecture and GCN. Moreover, we develop a self-attention down-sampling module, a multi-scale feature fusion operation, and a temporal inertia fusion operation. Extensive empirical analysis shows that Graphformer outperforms the previous state-of-the-art approaches by a considerable margin.

Our research is not immune from its limitations, which can be addressed in future studies. First, the model proposed in this paper takes the input of the nearest range as part of the output component, and despite the simplicity of the inertia mechanism of this heuristic, it is often ignored by complex long-series time series forecasting models. The extent to which our findings can be generalized requires further research. In the future, we will first use decomposition methods such as STL, EEMD (Ensemble Empirical Mode Decomposition), and EWT (Empirical Wavelet Transform) to preprocess the time series. Based on this, for a series of separated eigenmodes, we will use different models for prediction, and finally improve the model accuracy by adaptively combining the model outputs through automatic machine learning (AutoML) techniques.

CRediT authorship contribution statement

Yijie Wang: Writing – original draft, Methodology. **Hao Long:** Methodology. **Linjiang Zheng:** Writing – review & editing. **Jiaxing Shang:** Writing – review & editing.

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

The authors do not have permission to share data.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China Civil Aviation Joint Fund Key Program(U2033213), the National Key R&D Program of China under Grant(2022YFC3006400), the Fundamental Research Funds for the Central Universities, China (2022CDJDX-003), the Science and Technology Innovation Key R&D Program of Chongqing, China (CSTB2022TIAD-KPX0099), and the Transportation Science and Technology Program of Chongqing, China (2022-09).

Appendix

A.1. Multi-head self-attention mechanism

In the process of attention calculation, if there is a deviation in weight calculation, it will seriously affect the performance of the model. On the basis of the self-attention mechanism, the multi-head self-attention mechanism allows the model to extract features from different subspaces at the same time and aggregate this information in different historical ranges, thereby enhancing the representation learning ability of the model. Inspired by the multi-headed self-attention structure, sparse self-attention can be transformed into a multi-headed sparse self-attention structure, as shown in Fig. A.1. The multi-head sparse self-attention block contains a linear projection layer query, key, and value keeping both input and output dimensions as d. Each head has the same dimension and independently focuses on input from different representation subspaces in different historical scales to reveal richer potential information. For the results of different head stitching, linear projection processing needs to be used again as the output of the multi-head sparse self-attention block:

$$MultiHead(Q,K,V) = Concat\left(head_1, head_2, \ldots, head_h\right) \tag{A.1}$$

where $head_i = SparseAttention\left(q_i, k_i, v_i\right)$, $Concat(\bullet)$ represents a connection that preserves the original spatio-temporal features as completely as possible. Finally, a residual connection structure and LayerNorm processing are added to the multi-head sparse self-attention layer and the linear sub-layer. The LayerNorm is the default input normalization in the Transformer class structure, which consists of a feedforward neural network containing a linear transformation and a nonlinear activation function to prevent model overfitting. And residual structure by adding an identity connection, skipping the middle layer is to retain the input, to accelerate the convergence speed of the model, to avoid the effect of vanishing gradient.

A.2. Generative decoder

In the prediction phase, the encoder of the Transformer class model encodes the input X^t into a hidden state $H^t = h_1^t, h_2^t, \ldots, h_{L_H}^t$, and the decoder performs a dynamic decoding multi-step operation on it to decode H^t into the output Y^t. Specifically, the decoder calculates the hidden state h_{k+1}^t of step $k+1$ based on the hidden state h_k^t and the output y_k^t of step k, and outputs the predicted value y_{k+1}^t. Inspired by Informer, Proformer contains a generative inference decoder that can output all predicted values through only one forward propagation. The input vector of the decoder consists of the start token and the target sequence:

$$X_{decoder}^t = Concat\left(X_{token}^t, X_0^t\right) \in \mathbb{R}^{(L_{token}+P)\times d} \tag{A.2}$$

Compared with the starting flag set to a special identifier in dynamic decoding, $X_{token}^t \in \mathbb{R}^{L_{token}\times d}$ in Proformer represents the latest L_{token} time slice in the input sequence, and $X_0^t \in \mathbb{R}^{P\times d}$ represents the reserved placeholder of the target sequence to be predicted (the scalar is set to 0). In this way, the classical decoder is extended to a generative decoder, which outputs all the predicted values only through one forward propagation, alleviating the speed bottleneck of the encoder–decoder structure. In addition, the masked multi-head sparse self-attention block replaces the masked multi-head self-attention block and combines with a multi-head self-attention block to form the basic structure of the decoder. On the basis of the normal self-attention block, the masked dot product is set to $-\infty$, and the purity of the feature is maintained before the key and value are fed, so as to avoid the model directly using the future real value in the decoding process to predict, and ensure that the future time step will not produce information leakage to the past, which is conducive to solving the autoregressive problem. Finally, a fully connected layer (FC) is added to obtain the output sequence Y^t with length P and dimension d_y.

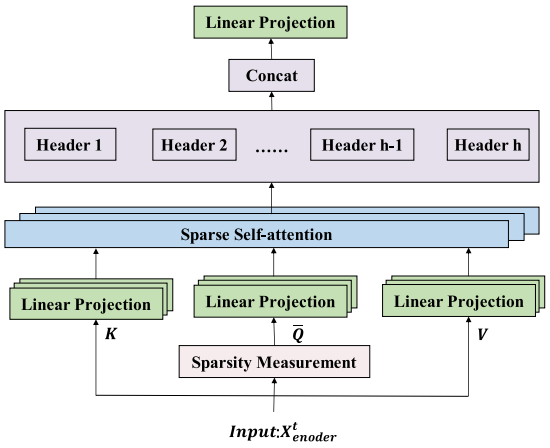


Fig. A.1. Multi-head self-attention mechanism.

A.3. Distilling operation

Informer uses a full distilling operation to stack n-1 additional encoders that gradually reduce the input length and the number of self-attention blocks, so that the final output feature map retains some details. The ith additional encoder is stacked with n-i self-attention blocks, and its input is the last $\frac{1}{2^i}$ of the main encoder input, so all encoders have the same output length. Informer's encoder consists of a fully-distilling structure and three encoders, which contains a main encoder stacked with three sparse self-attention blocks (input length is L, output length is $\frac{L}{4}$), and two additional encoders with shorter input and fewer self-attention blocks(input lengths are $\frac{L}{2}$ and $\frac{L}{4}$, output length is $\frac{L}{4}$). Besides, the convolution layer and the max-pooling layer are used to connect each two self-attention blocks, and all the feature maps output from three encoders are fused to the feature map provided to the decoder.

A.4. Gradient centralization

Specifically, for a back-propagating gradient $\nabla_{w_i} L$ $(i = 1, 2, \ldots, N)$ is a gradient vector of w_i the gradient centralization operation $GC(\bullet)$ can be defined as:

$$GC\left(\nabla_{w_i} L\right) = \nabla_{w_i} L - \mu\nabla_{w_i} L \tag{A.3}$$

where $\mu\nabla_{w_i} L = \frac{\sum_{j=1}^{M} W_{i,j} L}{M}$ denotes the gradient mean of the ith column of the gradient matrix $W \in \mathbb{R}^{M\times N}$ and L denotes the loss function. $GC(\bullet)$ generates a centralized gradient matrix by calculating the mean of the gradient matrix column vectors and subtracting their respective

mean values from each column vector. The gradient centralization operation can also be converted into the form of matrix operation:

$$GC\left(\nabla_W L\right) = P\nabla_W L \tag{A.4}$$

where $P = I - ee^T$ represents the projection matrix on the hyperplane of the same size as W, I represents the unit matrix of size $M \times M$, and e represents a unit vector of size $M \times 1$. After obtaining the centralized gradient $GC\left(\nabla_W L\right)$, the weight matrix is updated directly with it and embedded into the Adam optimizer to smooth and accelerate the training process and improve the generalization ability of the model.

References

[1] Q. Wen, T. Zhou, C. Zhang, et al., Transformers in time series: A survey, 2022.
[2] S. Ishak, H. Al-Deek, Performance evaluation of short-term time series traffic prediction model, J. Transp. Eng. 128 (6) (2002) 490–498.
[3] S. Papadimitriou, P. Yu, Optimal multi-scale patterns in time series streams, in: ACM SIGMOD, Vol. 2006, ACM, 2006, 647658.
[4] Y. Matsubara, Y. Sakurai, W.G. van Panhuis, C. Faloutsos, FUNNEL: automatic mining of spatially coevolving epidemics, in: ACM SIGKDD, Vol. 2014, 2014, pp. 105–114.
[5] A. Sorjamaa, J. Hao, N. Reyhani, Methodology for long-term prediction of time series, Neurocomputing 70 (1618) (2007) 2861–2869.
[6] Haoyi. Zhou, Shanghang. Zhang, Jieqi. Peng, Shuai. Zhang, Jianxin. Li, Hui. Xiong, Wancai. Zhang, Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, AAAI, 2021.
[7] Nikita Kitaev, Lukasz Kaiser, Anselm Levskaya, Reformer: The Efficient Transformer, ICLR, 2020.
[8] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, Xifeng Yan, Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting, NeurIPS, 2019.
[9] A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need, 2017, arXiv.
[10] L. Bai, L. Yao, C. Li, Adaptive graph convolutional recurrent network for traffic forecasting, in: Proceedings of the 34th Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc, 2020.
[11] J. Yin, W. Rao, M. Yuan, Experimental study of multivariate time series forecasting models, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), ACM, Beijing, China, 2019, pp. 2833–2839.
[12] M. Li, S. Chen, Y. Zhao, IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE Comput. Soc. 2020 (2020) 214–223.
[13] J. Bruna, W. Zaremba, A. Szlam, Spectral networks and locally connected networks on graphs, in: Proceedings of the 6st International Conference on Learning Representations (ICLR), OpenReview.net, Alberta, Canada, 2018.
[14] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc, Barcelona, Spain, 2016, pp. 3844–3852.
[15] Z. Wu, S. Pan, G. Long, Connecting the dots: multivariate time series forecasting with graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM, California, USA, 2020, pp. 753–763.
[16] Z. Pan, Y. Liang, W. Wang, Urban traffic prediction from spatio-temporal data using deep meta learning, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM, Alaska, USA, 2019, pp. 1720–1730.
[17] David Salinas, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Deepar: Probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. (2020).
[18] G.E.P. Box, Gwilym M. Jenkins, Time series analysis, forecasting and control, 1970.
[19] George E.P. Box, Gwilym M. Jenkins, Some recent advances in forecasting and control, J. R. Stat. Soc. (Ser.-C) (1968).
[20] A. Mellit, P. Massi, Benghanem, Least squares support vector machine for short-term prediction of meteorological time series, Theor. Appl. Climatol. 111 (1) (2013) 297–307.
[21] M. Das, S.K. Ghosh, A probabilistic approach for weather forecast using spatio-temporal inter-relationships among climate variables, in: 2014 9th International Conference on Industrial and Information Systems (ICIIS), IEEE, 2015.
[22] H.F. Yu, N. Rao, I.S. Dhillon, Temporal regularized matrix factorization for high-dimensional time series prediction, in: Neural Information Processing Systems, Curran Associates Inc., 2016.
[23] F. Tobar, T.D. Bui, R.E. Turner, Learning stationary time series using Gaussian processes with nonparametric kernels, in: Neural Information Processing Systems, MIT Press, 2015.
[24] F.Y. Sun, A memory-network based solution for multivariate time-series forecasting, 2018, [EB/OL]. https://arxiv.org/abs/1809.02105.
[25] D. Salinas, V. Flunkert, J. Gasthaus, Deepar: probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.
[26] S.J. Bai, K. Zico, K. Vladlen, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, [EB/OL], https://arxiv.org/abs/1803.01271.
[27] S.Y. Shih, F.K. Sun, H.Y. Lee, Temporal pattern attention for multivariate time series forecasting, Mach. Learn. 108 (8) (2019) 1421–1441.
[28] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, Schahram Dustdar, Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting, ICLR, 2021a.
[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV, 2021b.
[30] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, Mingsheng Long, Flowformer: Linearizing Transformers with Conservation Flows, ICML, 2022.
[31] Haixu Wu, Jiehui Xu, Jianmin Wang, Mingsheng Long, Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting, NeurIPS, 2021.
[32] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, Rong Jin, FEDformer: Frequency Enhanced Decomposed Transformer for Long-Term Series Forecasting, ICML, 2022.
[33] Z. Wu, S. Pan, G. Long, Graph wavenet for deep spatial–temporal graph modeling, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), AAAI Press, Macao, China, 2019, pp. 1907–1913.
[34] W. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31th International Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc, New York, USA, 2017, pp. 1025–1035.
[35] J. Glenn, YOLOv5, 2022, [EB/OL]. https://github.com/ultralytics/yolov5.
[36] Y. Tsai, S. Bai, M. Yamada, Transformer dissection: a unified understanding of transformer's attention via the lens of kernel, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP & IJCNLP), ACL SIGDAT, Hong Kong, China, 2019, pp. 4335–4344.
[37] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Proceedings of the 3th International Conference on Learning Representations (ICLR), OpenReview.net, San Diego, USA, 2015.
[38] G. Lai, W.C. Chang, Y. Yang, Modeling long-and short-term temporal patterns with deep neural networks, in: Proceedings of the 41th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), ACM, Michigan, USA, 2018.
[39] Ailing Zeng, Muxi Chen, Lei Zhang, Qiang Xu, Are transformers effective for time series forecasting?, 2022, arXiv preprint arXiv:2205.13504.
[40] H. Yong, J. Huang, X. Hua, Gradient centralization: a new optimization technique for deep neural networks, in: Proceedings of the 16th European Conference on Computer Vision (ECCV), Springer, Glasgow, UK, 2020, pp. 632–652.