Big Data Research 25 (2021) 100237

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Time-Series Graph Network for Sea Surface Temperature Prediction

Yongjiao Sun [a], Xin Yao [a,*], Xin Bi [b], Xuechun Huang [a], Xiangguo Zhao [c], Baiyou Qiao [a]

[a] School of Computer Science and Engineering, Northeastern University, Shenyang, 110819, China
[b] Key Laboratory of Ministry of Education on Safe Mining of Deep Metal Mines, Northeastern University, Shenyang, 110819, China
[c] College of Software, Northeastern University, Shenyang, 110819, China

ARTICLE INFO

Article history:
Received 6 May 2021
Received in revised form 24 May 2021
Accepted 7 June 2021
Available online 15 June 2021

Keywords:
Graph neural network
Time-series graph
SST prediction

ABSTRACT

Sea surface temperature (SST) is an important indicator for balancing surface energy and measuring sea heat. Various effects caused by the sea temperature field significantly affect human activities to a large extent. It is important to predict the sea surface temperature efficiently and accurately. Existing SST prediction methods regard each sensor as an independent individual, without considering the structure or topology of the sensor network and ignoring the connectivity of the sensor network. To address this problem, this study investigates the SST prediction problem from the perspective of graph learning. We propose a time-series graph network (TSGN) that can jointly capture graph-based spatial correlation and temporal dynamics. TSGN uses a long short-term memory network to aggregate the features of time series data and establishes a graph neural network model to complete the SST prediction task. Experiments using SST data from the Pacific Northwest from 2001 to 2005 show that this method is more efficient and accurate when dealing with data containing time-series information and is superior to the existing SST prediction methods.

1. Introduction

The sea surface temperature (SST) [1] is a physical quantity reflecting the thermal conditions of seawater and a key parameter in the field of marine research. The various effects of the seawater temperature field largely affect the distribution of fishery resources, the direction of marine pollution, the development of oil and gas resources, and military activities. In addition, global climate anomalies are inseparably related to changes in the seawater surface temperature. Therefore, it is important to make accurate and efficient predictions of seawater surface temperatures.

The methods for predicting seawater surface temperature can be broadly classified into two types: numerical and statistical learning models.

The numerical model [2] establishes a prediction model using kinetic and thermal equations based on the required initial and boundary conditions, describes the physical states using partial differential equations, and makes predictions of future seawater surface temperature after conducting a large number of calculations to derive numerical solutions. However, owing to the stochasticity and uncertainty of seawater motion, it is an extremely difficult and complicated task to accurately predict the seawater surface temperature, even with a large amount of abundant observation data.

Statistical learning models [3] directly uncover potential change patterns from data without the need to understand the factors and mechanisms affecting SST variability. Common statistical learning models can be broadly classified into time-series analysis models, regression models, and deep learning models. The existing statistical learning model methods only take a time series perspective and treat each sensor as an independent individual, dividing sensor networks into segments or grids, without considering the structure or topology of the sensor network, and ignoring the sensor network in terms of connectivity. Even if a two-dimensional convolution on the grid is used, it can only roughly capture the spatial locality owing to a compromise of the data modeling.

In recent years, graph learning methods have been widely used in different domains [4], which can effectively solve the problem of the overall connectivity of a sensor network. By exploiting the graph structure of the sensor network, the application of a graph convolution directly to the graph structure data can help extract highly meaningful patterns and features in the spatial domain. Therefore, we used a graph learning approach to investigate the SST prediction problem.

* Corresponding author.
E-mail address: yaoxin@stumail.neu.edu.cn (X. Yao).

https://doi.org/10.1016/j.bdr.2021.100237
2214-5796/

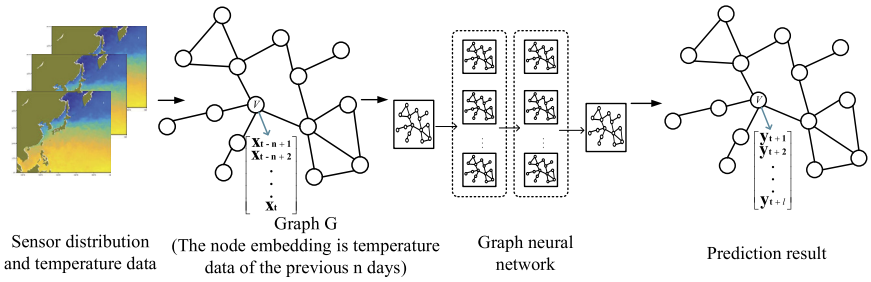

Fig. 1. Illustration of SST prediction problem.

For the complex dependencies of sea surface temperature data in the time and space dimensions, we propose a graph neural network called a time-series graph network (TSGN) by combining the advantages of a long short-term memory (LSTM) network in processing temporal information. The model is based on the graph structure of the sensor network and uses the topology between sensors to model the geometric structure of the sensors. LSTM aggregation is adopted for SST data containing temporal information. This model combines graph convolution and gated time convolution to extract the most useful spatial features and coherently capture the most basic temporal features.

The main contributions of this paper are as follows:

(1) A graph learning network, i.e., a TSGN, is designed, which has strong learning ability for data prediction tasks containing time series information.
(2) A TSGN employs an LSTM to aggregate the node features to better capture the time correlation and improve the prediction accuracy.
(3) A TSGN is applied to an SST prediction task, which is a regression fitting, and the experimental results show good prediction results.

The remainder of this paper is organized as follows. Section 2 introduces the definition and related studies on the SST prediction problem based on a TSGN. Section 3 provides an overview of the overall framework, and then introduces the graphic convolution of the TSGN, including sampling and LSTM-based aggregation. Section 4 introduces the experimental setup and results of the performance evaluation, comparison, and discussion. Finally, Section 5 provides some concluding remarks regarding this research.

2. Preliminaries

2.1. Problem definition

The sea surface temperature prediction is based on historical temperature data obtained from marine sensor observations to predict future temperatures. In this section, we first provide a definition of the sea surface temperature prediction problem. Given the previous n consecutive SST observations $X = x_{t-n+1}, x_{t-n+2}, \cdots, x_t$, the l consecutive SST predicted values in the future can be defined as $Y = y_{t+1}, y_{t+2}, \cdots, y_{t+l}$.

Because considering only time series data ignores the spatial dependencies between sensors, we propose a graph-based method that enables sea-surface temperature prediction. We have to construct a sensor network, which is defined for the sensor network as follows: temperature sensors do not have directional information, and thus the sensor network can be modeled as an undirected graph $G = (V, E)$ with structured time series data, where V is the set of sensor nodes and E is the connectivity between nodes.

As shown in Fig. 1, the task of graph-based SST prediction can be defined as follows: build a graph neural network, input historical temperature data $X \in R^{N \times T_{in} \times D}$ (where N is the number of sensors, T_{in} is the length of the time window, and D is the input feature dimension) and a graph G reflecting the spatial connectivity between temperature sensors, and output the temperature for a future period $Y \in R^{N \times T_{out} \times d}$ (where T_{out} is the prediction step and d is the number of temperatures).

2.2. Related work

2.2.1. SST prediction

SST prediction is not only of significant theoretical value but also of practical application in many ocean-related fields because SST series are typical long time series data, the prediction of SST has been classified as a time-series [5] regression problem by many researchers, and many time series prediction methods have been applied to the prediction of SST.

Time series models such as the autoregressive model (AR), sliding average model (MA), and autoregressive sliding model [6], which are linear models, cannot accept feature inputs other than serial data and cannot meet the realistic needs of SST prediction. Regression models include linear regression models and support vector machines (SVMs). Lins et al. [7] used an SVM to achieve SST forecasting by studying the seasonality of SST and the pattern of intra-seasonal variation, combined with the curvature and slope information of the acquired SST sequences, which improves the prediction accuracy, but requires complex feature engineering.

In recent years, deep learning has rapidly developed, and neural networks have a wide range of application in marine science research because they can learn complex features in the data to build reliable nonlinear relational models. For example, Aparna et al. [8] proposed the use of an artificial neural network (ANN) to forecast the SST in the northeastern Arabian Sea; however, in traditional neural networks, the model does not focus on what information will be available for the next moment from the previous moment of processing, and only focuses on the current moment of processing in each case. The emergence of recurrent neural networks (RNNs) [9] solves this problem by multiple replications of the same neural network, where each moment of the neural network passes information to the next moment. Therefore, an RNN has some memory function and can be used to solve many different problems, such as speech recognition, language modeling, and machine translation. However, it does not deal well with the long-time dependence problem.

In 1997, Hochreiter and Schmidhuber [10] proposed an LSTM network, which is a special type of an RNN. It adds the gate mechanism, which can effectively solve the problem of long-distance dependence information in time-series data. An LSTM network, similar to a BP neural network, does not rely on physical mechanisms to obtain the results in practical applications, and is also a

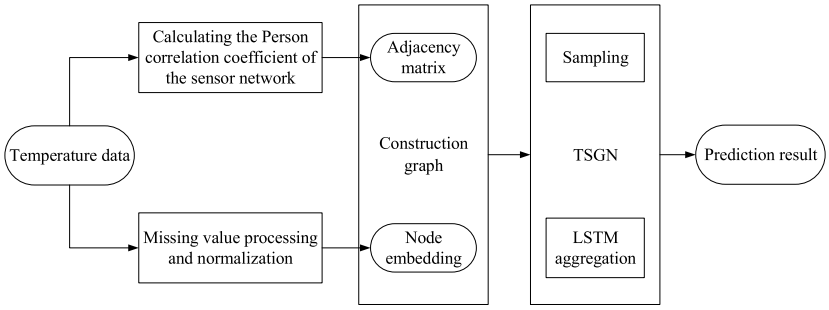

Fig. 2. Flowchart of the prediction model.

statistical learning-based approach. The input of an LSTM network can contain physical parameters, and the model can quantify the effects of these physical parameters on the time series, which is equivalent to borrowing the idea of numerical models to improve the accuracy of the model as well as the interpretability of the model results. An LSTM network is also widely used in prediction models. In 2017, Zhang et al. [11] first proposed the application of an LSTM network to SST prediction. The temporal features of SST sequences are first extracted by the LSTM, and the output is then mapped to the final prediction result through full connectivity. Owing to the multi-branching structure of the LSTM, the model can capture the features at different scales in the time series, which results in highly accurate prediction results; however, this method has an input-output dimension of 1 and ignores the spatial features of the data.

In 2019, He et al. [12] proposed a sea surface temperature prediction method under the Spark platform, and in 2020, an STL-based sea-level temperature prediction algorithm [13] was proposed to make full use of the periodicity, continuity, and non-stationarity of SST, which improved the accuracy of SST prediction to a certain extent. Subsequently, Qi et al. [14] proposed a sea surface temperature prediction method based on the EMD-GRU model to reduce the prediction difficulty. These methods only consider the time correlation of sea surface temperature data and ignore the spatial correlation of the sensor network.

Shi et al. [15] proposed the ConvLSTM model, which was later widely used in temperature prediction tasks, which enables the model to learn spatial features along with temporal features of the data by adding convolution to the LSTM. However, the spatial features are still local spatial features, and the overall connectivity of the sensor network is not considered.

2.2.2. Graph neural networks

In recent years, there has been increasing interest in graphs. However, traditional neural networks cannot consider graph structure data because they cannot handle non-European structures. Therefore, the graph neural network (GNN) has become a new research hotspot driven by the success of multiple factors.

One of the largest application areas of a graph neural network is computer vision, such as scene graph generation, point cloud classification and segmentation, action recognition, and many other aspects using the graph structure approach. In addition, many scholars have provided new ideas for solving problems such as recommendation systems [16], traffic volume prediction [17], surface water quality index prediction [18], and brain network classification [19].

Gori et al. [20] first proposed the concept of a graph neural network in 2005, and Scarselli et al. [21] defined the theoretical basis of graph neural networks in 2009. The earliest GNN [22] was studied based on immobile point theory and mainly solved strict graph theoretical problems such as molecular structure classification. Until 2013, based on graph signal processing, Bruna [23] proposed the first spectral- and spatial-domain based graph convolutional neural network (GCN). It has broad application prospects. For example, Han et al. [24] propose a new framework for disease-gene association task by combining GCN and matrix factorization, named GCN-MF. A spectral-domain-based GCN [25] treats a graph as an undirected graph for processing and introduces filters to define the graph convolution from the perspective of graph signal processing. The spatial-domain-based GCN [26] defines the graph convolution operation based on the spatial relationship of the nodes, that is, by pooling the information of neighboring nodes to construct the graph convolution. GraphSAGE, proposed by Hamilton [27], is a classical null-domain-based graph model, which makes distributed training of large-scale graph data possible. Liu et al. [28] proposed a GraphSAGE-based traffic speed prediction for sparse data, which reflects the superiority of GraphSAGE in link prediction performance on graphs. In 2017, Velikovic et al. [29] proposed a model-GAT that learns the edge weights through an attention mechanism. Liu et al. [29] proposed a GB-CNN, the graph-based model provides the initial supervision for training the convolutional neural networks (CNNs). In 2018, Errica et al. [30] proposed a CGMM to study the interpretability of the probability based on NN4G.

However, most SST prediction methods either learn from artificially extracted shallow features or learn from deep features in a Euclidean space. There have been no studies on the SST prediction problem based on the graph topology.

3. Framework overview

To implement the sea surface temperature prediction function based on graph learning, we propose a prediction model framework, as shown in Fig. 2.

The key implementations in the framework are described as follows: constructing a graph based on the distribution of sensors on the sea surface and the covariance matrix calculated from the observed data, modeling the complex spatial dependencies of the sensors, and preprocessing the historically monitored data. The temperature data corresponding to each moment are used as the node feature input, and the creation of edges is based on the Pearson correlation coefficient between each sensor. Then, a TSGN, which is a graph neural network, was used to implement the information interaction between nodes, and all nodes were put in during sampling, feature aggregation was implemented using LSTM networks, and the final output prediction results were obtained. In the following section, we explain in detail the contents of the main modules in the framework.

3.1. Undirected graph construction for sensor networks

The sensor network can be mathematically modeled as a graph. However, in previous studies [15], spatial features were extracted

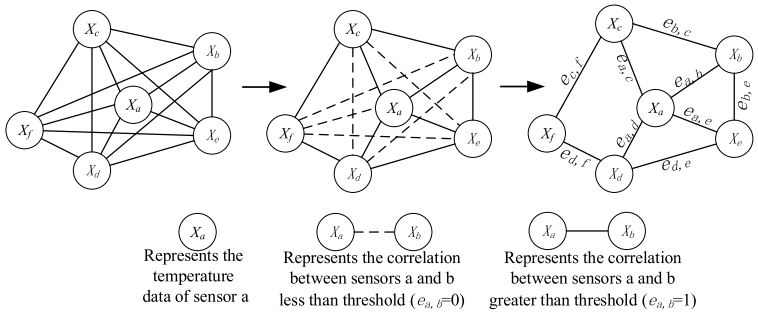

Fig. 3. Construction of sensor graph.

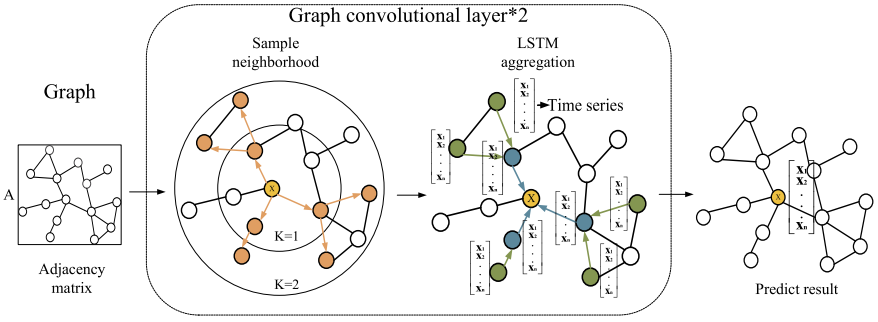

Fig. 4. Framework of TSGN.

using neural networks without considering the spatial dependence of the sensor network; therefore, when the sea area is partitioned into multiple grids, the connectivity of the network is ignored.

The sensor network was first modeled as an undirected graph. The distribution of nodes on the graph corresponds to the distribution of sensors within the sea area, and the edges indicate the relationship between the two sensors. Fig. 3 shows the method for constructing the undirected graph G. Here, X_a denotes sensor a; $e_{a,b}$ denotes whether there is an edge between sensor and sensor b, that is, $e_{a,b} = 1$ indicates that a and b are connected, and there is an edge; $e_{a,b} = 0$ shows that a and b are not connected and there is no edge; X_a consists of the temperature data at the corresponding moment; and $e_{a,b}$ is related to the Pearson correlation coefficient between two points.

The Pearson correlation coefficient is one of the most popular statistics for measuring the linear correlation between two normally distributed variables [31], and is calculated as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{1}$$

where r is the Pearson correlation coefficient, X_i and Y_i are the sample values, $\bar{X}$ and $\bar{Y}$ are the sample means, and n is the number of samples.

The Pearson correlation coefficients can be calculated to obtain the covariance matrix, after which some edges whose correlation is less than the threshold R_0 can be removed from the complete graph of the sensor network by setting the threshold R_0. Thus, $e_{a,b}$ is calculated as follows:

$$e_{a,b} = \begin{cases} 1, & |r| > R_0 \\ 0, & \text{others} \end{cases} \tag{2}$$

where R_0 denotes the correlation coefficient threshold, and r denotes the Pearson correlation coefficient. An edge with a correlation coefficient less than the threshold was considered to have less spatial dependence between the two sensors.

3.2. Graph convolution

This section describes the specific implementation of the TSGN model for predicting sea surface temperatures, and where TSGN model is implemented using node features and sensor networks as model inputs. The model generates node representations from its local neighborhood, thereby considering local spatial effects, and then predicts the future temperature values using historical temperature data. The following is a description of a TSGN spatial convolution.

3.2.1. Spatial convolution

The TSGN model has the following advantages: (1) The model solves the problem of a GCN memory explosion by neighbor sampling and can effectively predict the temperature in large-scale graphs. (2) The introduction of neighbor sampling transforms the direct-push node to represent only one local structure into a node-inductive representation corresponding to multiple local structures, which can effectively prevent an overfitting during the training and enhance the model generalization capability. (3) The TSGN generates embeddings in the local neighborhoods of the nodes, which can consider the local spatial features in the transmission network.

Fig. 4 shows the structural diagram of the TSGN model. The model was first sampled based on adjacency matrix A. Subsequently, the sorted node features are input into the LSTM network to aggregate information from the node neighbors to realize the propagation of information between different layers, where the node features are a sequence in chronological order. Finally, the return value of the aggregation function is output as the prediction result.

The TSGN model generates node embeddings in the following steps.

(1) Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges in the graph. The adjacency matrix A of graph G and the node features $\{x_v, \forall v \in V\}$ with the number of graph convolution layers K are input as parameters for the model. The input node features x_v are used as the initial representations of the nodes h_0 (line 1);
(2) A total of K layers are propagated, and in each layer, each node ($\forall v \in V$) of the graph G is sampled for its neighbors, by $f_{sample}(A, m)$ returning the top-level neighbors $N(v)$ of node v according to the adjacency matrix A and the number of samples m (line 3);
(3) Each node v aggregates the node representations in its immediate neighborhood $N(v)$ into a neighborhood vector $h^k_{N(v)}$ by means of an aggregation function, which depends on the representation generated in the previous iteration (line 5);
(4) The current embedding of a node h^{k-1}_v with its aggregated neighborhood vector $h^k_{N(V)}$ is summed and the results obtained are subsequently sent to a fully connected layer with a nonlinear activation function Relu, which will then be used in the next step of the representation (line 6);
(5) The final output Z_v, $\forall v \in V$ is the node embedding of depth K (line 7).

This procedure is described in Algorithm 1.

Algorithm 1: Graph convolution of TSGN.

Input: Graph $G(V, E)$; input features $x_v, \forall v \in V$; number of graph convolutional layers K; sample number m
Output: Vector representations $Z_v, \forall v \in V$

```
1 h_0 = x_v, ∀v ∈ V ;
2 for k = 1, ..., K do
3     N(v) = f_sample(A, m) ;
4     for v ∈ V do
5         h^k_{N(v)} = AGG_k ;
6         h^k_v = relu(W^k · sum(h^{k-1}_v, h^k_{N(v)})) ;
7 Z_v = h^K_v, ∀v ∈ V
```

The TSGN model can be said to be an implementation of a GCN from the spatial domain, and its core is neighbor sampling and feature aggregation, which will be briefly described below.

3.2.2. Neighborhood sampling

A TSGN transforms a GCN from a full-graph training approach to a node-centered, small-batch training approach through a neighbor sampling strategy, which makes distributed training of large-scale graph data possible.

TSGN sampling is achieved by randomly selecting m nodes instead of using all nodes, and thus the number of nodes to be sampled becomes $1 + m + m^2 + \cdots + m^k$, which makes $m^k \ll N^k$; thus, the computational complexity is greatly reduced, and the generalizability is further improved by not sampling all neighboring nodes.

3.2.3. LSTM-based feature aggregation

3.2.3.1. Symmetric aggregation The properties required for the aggregation neighbor operation are as follows: (1) The aggregation operation must be adaptive to the number of aggregated nodes. The dimensionality of the output after the aggregation operation must be consistent, regardless of how the number of neighbors of the nodes changes. (2) The aggregation operation has an invariant alignment for the aggregated nodes, which requires that the output result must be the same regardless of the order of the neighboring nodes. (3) The optimization of the model requires a backpropagation algorithm, and use of the backpropagation algorithm requires

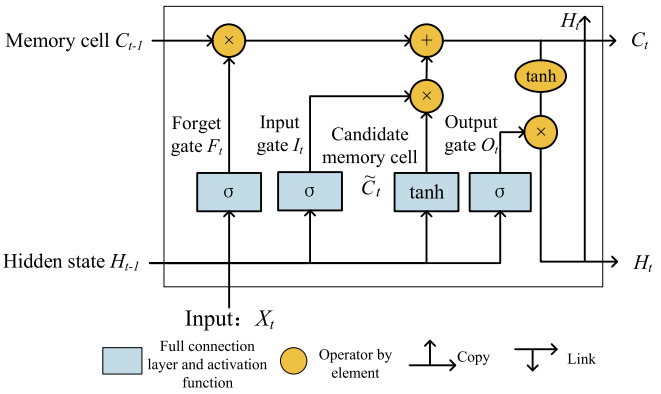

Fig. 5. LSTM structure.

that this aggregation operation be derivable. Based on these three conditions, there are two relatively simple operators.

(1) Mean/sum aggregation operator: Element-by-element summation and averaging is the most straightforward type of aggregation operator, which is a linear approximation of the graph convolution operation in a GCN. The mean/sum aggregation method stitches together the $(k-1)_{th}$ layer vectors of the target vertex and the neighboring vertices, and then applies a mean/sum operation for each dimension of the vector, which produces the k_{th} layer representation vector of the target vertex, which is expressed as follows:

$$AGG^{mean/sum} = \sigma(Mean/Sum\{Wh^{k-1}_v + b, \forall v \in N(v)\}) \quad (3)$$

(2) Pooling aggregation operator: This operator draws on the pooling operation in a CNN to achieve an aggregation, such as the maximum pooling operation, which takes the maximum value in an element by element manner. The k_{th} layer representation vector of the target vertex is obtained through a nonlinear transformation, and the formula is expressed as follows:

$$AGG^{pool} = MAX\{\sigma(Wh^{k-1}_v + b), \forall v \in N(v)\} \quad (4)$$

In formula (3)(4), W is the learnable parameter, b is the bias vector, h^{k-1}_v is the result of the previous level of aggregation, and $N(v)$ is the set of neighbor nodes.

3.2.3.2. LSTM aggregation An LSTM aggregation is more expressive than the above two aggregation methods, and the LSTM structure is carefully designed with a GAT structure, as shown in Fig. 5.

In Fig. 5, C_t indicates the cell state

$$C_t = F_t \times C_{t-1} + I_t \times \widetilde{C}_t \quad (5)$$

where C_{t-1} is the cell state at the previous moment, and F_t and I_t are the unique forgetting and renewal gates, respectively.

The forgetting gate F_t is calculated from the hidden state at the previous moment H_{t-1} and the input at the current moment X_t. A sigmoid is typically used as the activation function, which outputs a value of between zero and 1 for each number in the cell state C_{t-1}, where 1 indicates that the information is completely maintained and zero means the information is completely discarded, and determines what information we will be discarded from the cell state. The forgetting gate F_t is represented as follows, where W is the learnable parameter, b is the bias vector, and σ is

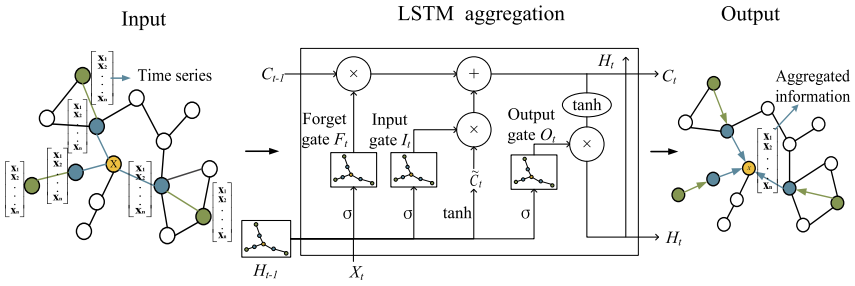

Fig. 6. Time-series aggregation.

the activation function, the formulas of which are all the following:

$$F_t = \sigma(W_F \cdot [H_{t-1}, X_t + b_F]) \tag{6}$$

The input gate I_t is similarly calculated from the hidden state of the previous moment H_{t-1} and the input of the current moment X_t. It is used to update the old cell state, where the activation function I_t is also used as a sigmoid, whereas the cell state update value $\widetilde{C}_t$ generally uses tanh as the activation function, which is expressed as follows:

$$I_t = \sigma(W_I \cdot [H_{t-1}, X_t + b_I]) \tag{7}$$

$$\widetilde{C}_t = tanh(W_C \cdot [H_{t-1}, X_t + b_C]) \tag{8}$$

The calculation of the output gate O_t is highly similar to that of the other two gates, and mainly relies on the output result to update the hidden state H_{t-1}, which is expressed as follows:

$$O_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o) \tag{9}$$

$$H_t = O_t \times tanh(C_t) \tag{10}$$

The LSTM aggregation method is shown in Fig. 6. The neighboring node features within the input time window are arranged in temporal order, and the composed sequence is input into the LSTM model for aggregation; finally, the output hidden layer features undergo a nonlinear transformation to produce the k_{th} layer representation vector of the target vertex, which is expressed by the following equation:

$$AGG^{lstm} = \sigma(LSTM\{Wh_v^{k-1} + b, \forall v \in N(v)\}) \tag{11}$$

where W is the learnable parameter, b is the bias vector, h_v^{k-1} is the result of the previous level of aggregation, and $N(v)$ is the set of neighbor nodes.

3.2.3.3. Supervised parameters learning The LSTM aggregator was trained using supervised learning. The weight matrix and parameters of the aggregation function were adjusted using a stochastic gradient descent. The loss function uses the mean square error (MSE) to describe the magnitude of the difference between the predicted and true values of the model, which is expressed through the following equation:

$$Loss_{mse} = \sum_{i=1}^{n} (x_i - \widetilde{x})^2 \tag{12}$$

where x_i is the sample true value, $\widetilde{x}$ is the predicted value, MSE is the expected value of the squared difference between the parameter estimate and the parameter value, and the MSE loss function is usually used in regression fitting problems.

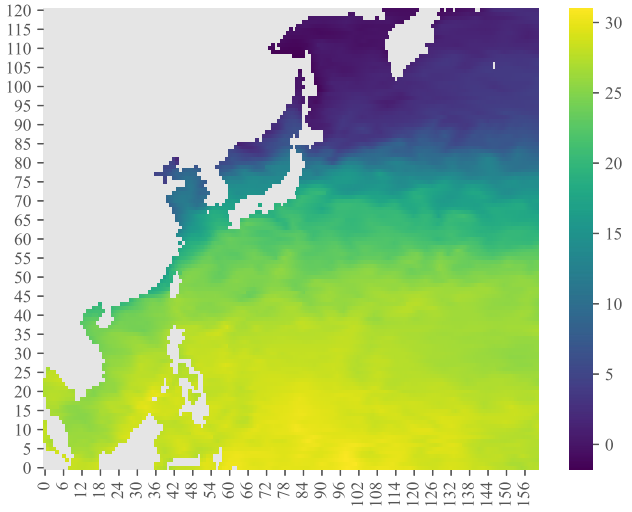

Fig. 7. Heat map of sea surface temperature in the northwest area of the Pacific Ocean at 0:00 on January 1, 2005. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4. Experiment

4.1. Dataset and data pre-processing

The experimental data are the sea surface temperature data of the Northwest Pacific Ocean ($range: 0 - 60°N, 100°E - 180°E$) from 2001 to 2005. There are four variables, and the variable names and data formats are as follows:

(1) Time: This indicates January 1, 2001, 0:00 to December 31, 2005, 18:00, where the time step is 6 h, with a total of 7304 moments, and text-type data, such as "20011010100" and "20011010106";
(2) Latitude: This indicates the distribution of latitude 0 $60°N$, with a step size of 0.5°, and is an array of 121 × 1;
(3) Longitude: The distribution of longitude $100°E$ $180°E$, with a step size of 0.5°, is an array of 161 × 1;
(4) sst01_05: This indicates the sea surface temperature, which is an array of 121×161×7304, corresponding to the latitude, longitude, and time. The sea surface temperature of the northwest Pacific Ocean at 0:00 on January 1, 2005, was selected as a sample map, as shown in Fig. 7.

The actual data collection and storage process will encounter numerous problems, which may lead to missing and abnormal final data; thus, the obtained data should be pre-processed by the abnormal and missing values. In the above dataset, not all data belong to the observed area. We filtered out 13447 observation points belonging to the Northwest Pacific Ocean. Each observa-

tion point contained a sensor. The sensors and graph nodes have a one-to-one relation and therefore, there are 13447 nodes in the undirected graph of the sensor network.

There are large differences in the distribution of the model input data. In the field of machine learning, to eliminate the influence of magnitudes between indicators, different evaluation indicators often have different magnitudes and magnitude units, which will affect the results of a data analysis; in addition, data normalization is required, and thus the experimental data are normalized, and the normalization calculation formula is as follows:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{13}$$

where X_{norm} is the normalized data, X is the original data, and X_{max} and X_{min} are the maximum and minimum values of the original dataset, respectively.

4.2. Experimental setup

The experimental code was written in Python, and the PyTorch framework [32] was used to build the neural network model, which can seamlessly access the Python ecosystem, realize tensor, variable, and Numpy interconversion, and can encapsulate variables on top of the tensor to facilitate the construction of neural networks with powerful back-end functions. The PC configuration for training the model was as follows: a 3.7-GHz Intel Core CPU, NVIDIA GeForce RTX 2080 Ti graphics card, 32 GB 2400 mHz DDR 4 RAM, and 500-GB solid-state drive.

The hyperparameters of the model are set as follows: There are 2 graph convolutional layers of the TSGN model, and 10 neighbor sampling points per order. The length of the hidden state of the LSTM network used for aggregation was 28, and the number of layers in the hidden layer was 1. The model parameters were optimized using the Adam optimizer [33], the learning rate was set to 0.001, the training epoch was set to 2, and the batch size was set to 4.

The prediction task in our experiments was defined as using 28 ($T_{in} = 24$, sampling period of 6 h) historical temperature data of all nodes to predict the next 4 ($T_{out} = 4$, sampling period of 6 h) temperature values, that is, the temperature values of the first 7 consecutive days are input to predict the temperature values of the next day. A sliding window with a window size of 32 was used to generate 7304 data samples with a step size of 1. The data are divided into a training set, an evaluation set, and a test set at a ratio of 7:1:2 in chronological order.

We used samples from the training set to optimize the model parameters. The performance of the model was evaluated by optimizing the hyperparameters of the execution results with the evaluation set and validating them with the test set.

We used three performance evaluation metrics: the root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE).

(1) RMSE: Within the range of $[0, +\infty)$, when the predicted and true values of the exact match are equal to zero, that is, a perfect model, the greater the error, the greater the value. The order of magnitude of the RMSE is more intuitive. The formula is expressed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - y_i)^2} \tag{14}$$

(2) MAE: Within the range of $[0, \infty)$, when the predicted and true values of an exact match are equal to zero, that is, a perfect model, the greater the error, the greater the value. The formula is expressed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | \bar{y}_i - y_i | \tag{15}$$

(3) MAPE: Within the range of $[0, +\infty)$, 0% indicates a perfect model, and a MAPE of greater than 100% means an inferior model. Compared to RMSE, MAPE is equivalent to normalizing the error at each point, reducing the impact of absolute error from individual outliers. The formula is expressed as follows:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n} | \frac{\bar{y}_i - y_i}{y_i} | \tag{16}$$

In Equations (14)–(16), $\bar{y}_i$ is the predicted value, and y_i is the true value, which are often used for the performance evaluation of a regression prediction.

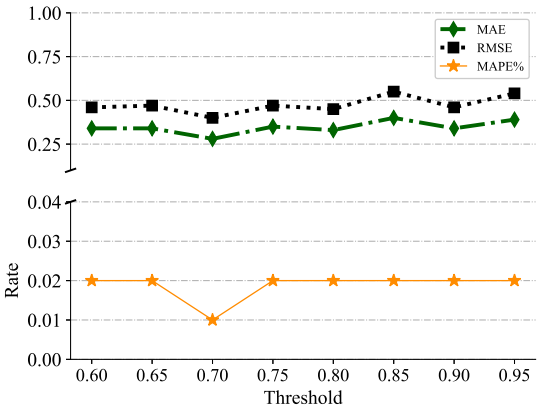

Fig. 8. Effects of correlation coefficient threshold R_0 on evaluation indexes.

4.3. Results

4.3.1. Effect of Person correlation coefficient threshold R_0

The size of the Pearson correlation coefficient [34] threshold R_0 directly affects the connection density of the constructed graph edges. When R_0 is large, it is beneficial to preserve the features of the nodes during the graph convolution calculation, but it does not adequately model the spatial dependencies between sensors. When R_0 is small, almost all nodes in the graph are connected, that is, there are edges between every point, and the signals of all nodes on the graph may tend to be the same after a computation through a graph convolution, thus losing the features of the nodes, that is, generating an over smoothing phenomenon. The trends of the performance evaluation metrics RMSE, MAE, and MAPE with R_0 are shown in Fig. 8.

As shown in Fig. 8, the optimization effect is best when $R_0 = 0.7$ for either of the evaluation indexes. Therefore, in the subsequent experiments, R_0 was set to 0.7.

4.3.2. Effect of data sequence length on prediction accuracy

To make the experimental results more valid, we set up the following comparison experiments: To investigate the effect of different prediction steps on the prediction accuracy when inputting the same sequence length, we used 7 days to predict 1 day (denoted as 7-1), 7 days to predict 3 (denoted as 7-3) days, and 7 days to predict 7 days (denoted as 7-7), the experimental results of which are shown in Table 1.

Table 1
Comparison of results for different prediction lengths.

Evaluation indicators	7-1	7-3	7-7
MAE	0.2812	0.3509	0.4824
MAPE	0.0141%	0.0171%	0.0244%
RMSE	0.3882	0.4737	0.6304

Table 2
Comparison of the results of different sequence lengths.

Method	Evaluation indicators	7-1	10-5	15-7
GraphSAGE_Mean	MAE	0.4787	0.4896	0.5737
	MAPE	0.0236%	0.0244%	0.0293%
	RMSE	0.6442	0.6563	0.7608
TSGN	MAE	0.2812	0.4438	0.5106
	MAPE	0.0141%	0.0216%	0.0273%
	RMSE	0.3882	0.5999	0.6916

Table 3
Comparison of prediction results of different models.

Method	MAE	MAPE	RMSE
CNN_LSTM	0.8712	0.0434%	1.2570
ConvLSTM	0.3052	0.0146%	0.3888
GCN	0.8676	0.0570%	1.2092
GraphSAGE_Mean	0.4787	0.0236%	0.6442
GraphSAGE_Pool	1.2360	0.0694%	1.5941
TSGN	0.2812	0.0141%	0.3882

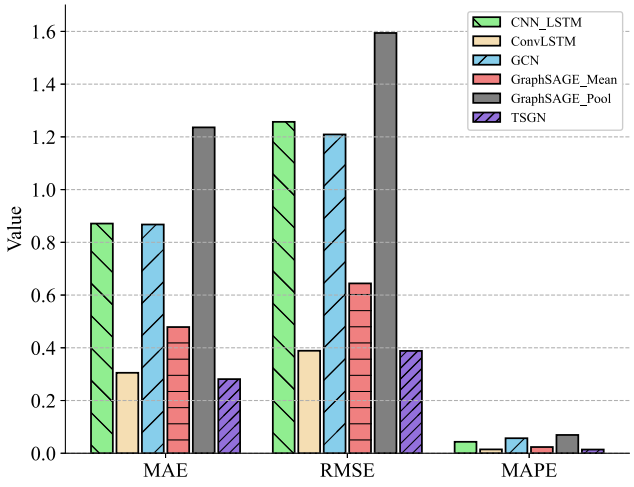

Fig. 9. Comparison of prediction results of different models.

From Table 1, it can be seen that the prediction accuracy varies for different prediction steps. When the same length of historical data is input, the longer the number of prediction days, the lower the prediction accuracy.

In addition, we conducted two sets of supplementary experiments by setting different input and output lengths, using 10 days to predict 5 days (denoted as 10-5) and 15 days to predict 7 days (denoted as 15-7), the experimental results of which are shown in Table 2.

As can be seen from Table 2, the prediction accuracy decreases as the prediction length increases. Moreover, in the case of sea surface temperature prediction with the same input and output, a TSGN not only considers the temporal sequential nature of the temperature data, it also better considers the spatial dependence of the sensor network from a spatial perspective, which yields better prediction results.

4.3.3. Performance comparison

To verify the effectiveness of the prediction model in this study, we compared it with the mainstream time series prediction models, a CNN_LSTM and ConvLSTM, with a spectral convolution-based GCN, and with GraphSAGE methods that use different aggregation methods. The performances of the models were evaluated by comparing the MAE, MAPE, and RMSE metrics obtained using different forecasting methods.

The comparison results of different model evaluation indexes are shown in Table 3.

From Table 3 and Fig. 9, we can see that (1) ConvLSTM achieves a better prediction than CNN_LSTM. Although both models are able to capture temporal and spatial information, the CNN_LSTM learns spatial features first, followed by temporal features, whereas ConvLSTM learns both temporal and spatial features. (2) Compared with the ConvLSTM model and the TSGN, the prediction effect of the TSGN is somewhat better because, although the input of the ConvLSTM model is a graph, it ignores the overall connectivity of the sensor network, whereas the TSGN solves this problem by constructing an undirected graph for the sensor network. (3) The prediction effect of the TSGN is significantly better than that of a GCN, which is based on the spectral domain, whereas a TSGN is based on the spatial domain, and the results show that the spatial domain convolution is better than spectral domain convolution in performing sea surface temperature prediction. (4) The prediction accuracy varied with the different aggregation methods. The TSGN uses an LSTM network to aggregate neighboring node features, and the prediction effect is significantly better than the mean aggregation method and pool aggregation method, which is due to the fact that an LSTM network can capture temporal information well; in addition, the temperature data itself are a time series, and thus can obtain better prediction results. We selected one of the sensors with node coordinates of (0.0°, 104.0°) (longitude 0.0°, latitude 104.0°) and visualized the prediction results of different aggregation methods by comparing their predicted temperature values for 1 year with the actual temperature values.

As shown in Fig. 10 and Table 4, the prediction error of LSTM aggregation is significantly less than that of the mean aggregation and pool aggregation in the sea surface temperature prediction task. It can be seen that TSGN has significant advantages for time-series data processing.

5. Summary

In this paper, we propose a sea surface temperature prediction model based on a time-series graph network. The model uses a graph learning approach to model the complex dependencies of a sea temperature sensor network in space. In the feature aggregation, the sorted feature sequence is used as the input of the long and short-term memory network to model the complex dependence of the historical temperature sequence on time. The model demonstrated a strong learning capability in handling data containing temporal information. The experimental results show that this model significantly improves the prediction performance compared to existing SST prediction methods.

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.

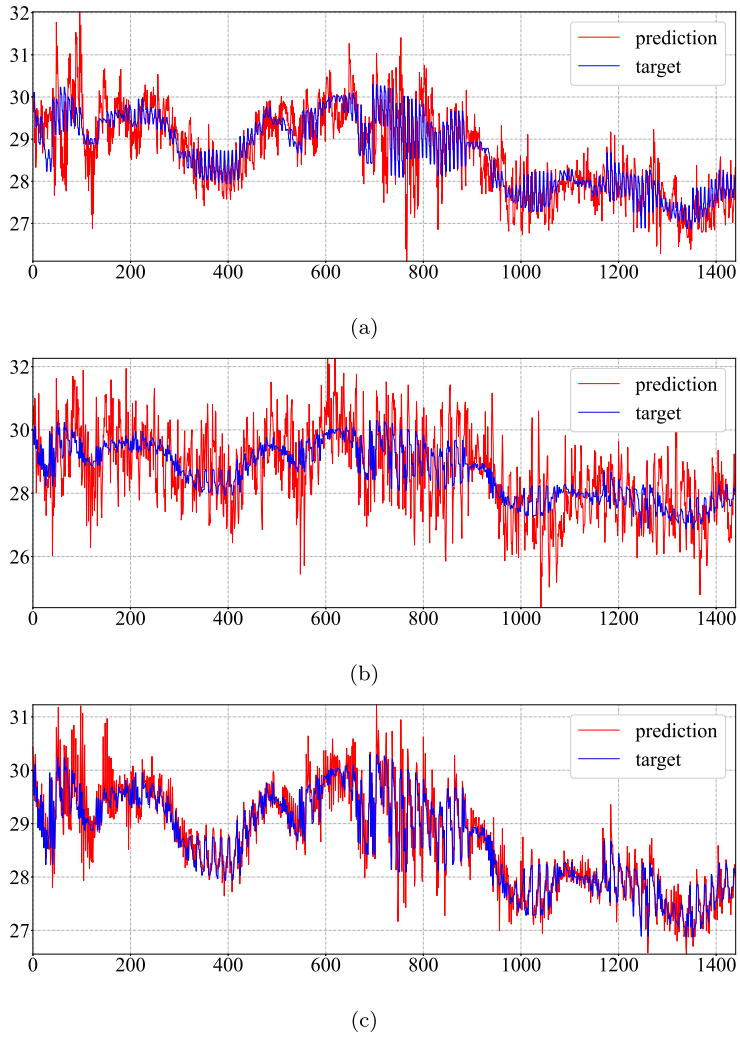

Fig. 10. Comparison of true and predicted values for (0.0°, 104.0°): (a) mean aggregation, (b) pool aggregation, and (c) LSTM aggregation.

Table 4
Comparison of results for different aggregation methods.

Evaluation indicators	Mean aggregation	Pool aggregation	LSTM aggregation
Maximum error	9.8689	28.3961	6.2845
Mean Absolute Error	0.4787	1.2360	0.2812
Accuracy	99.9764%	99.9305%	99.9859%

Acknowledgement

This work is partially supported by the National Key R&D Program of China (Grant No. 2016YFC1401902), the National Natural Science Foundation of China (Grant No. 61972077 and 62072087), the Liaoning Revitalization Talents Program (Grant No. XLYC2007079), the Fundamental Research Funds for the Central Universities (Grant No. N2101044).

References

[1] L. Xiao, J. Shi, G.R. Jiang, Z.L. Liu, The influence of ocean waves on sea surface current field and sea surface temperature under the typhoon background, Marine Science Bulletin 37 (4) (2018) 396–403.
[2] C.A. Repelli, P. Nobre, Statistical prediction of sea-surface temperature over the tropical Atlantic, Int. J. Climatol. 24 (1) (2004) 45–55.
[3] V. Mendoza, E. Villanueva, J. Adem, Numerical experiments on the prediction of sea surface temperature anomalies in the Gulf of Mexico, J. Mar. Syst. 13 (1) (1997) 83–99.
[4] Big graph classification frameworks based on extreme learning machine, Neurocomputing 330 (2019) 317–327.
[5] Explainable time-frequency convolutional neural network for microseismic waveform classification, Inf. Sci. 546 (2021) 883–896.
[6] L.X. Zhang, M. Zhao, X.S. Jiang, The change trend of happened frequency and the R/S forecast of frequently happened year for red tide in China, Marine Science Bulletin 29 (1) (2010) 72–77.
[7] I.D. Lins, M. Araújo, M. das Chagas Moura, M.A. Silva, E.L. Droguett, Prediction of sea surface temperature in the tropical Atlantic by support vector machines, Comput. Stat. Data Anal. 61 (2013) 187–198.
[8] S.G. Aparna, S. D'Souza, N.B. Arjun, Prediction of daily sea surface temperature using artificial neural networks, Int. J. Remote Sens. 39 (12) (2018) 4214–4231.
[9] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization, CoRR, arXiv:1409.2329 [abs], 2014.
[10] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[11] Q. Zhang, H. Wang, J. Dong, G. Zhong, X. Sun, Prediction of sea surface temperature using long short-term memory, IEEE Geosci. Remote Sens. Lett. 14 (10) (2017) 1745–1749.
[12] H. Q, C. Cheng, S. Miao, J. Xiaoyi, Q. Fuming, H. Dongmei, S. Wei, Parallel sea surface temperature prediction algorithm under spark platform, Ocean Bull. 38 (03) (2019) 280–289.

[13] H. Qi, C. Cheng, S. Wei, Q. Fuming, H. Zengzhou, H. Dongmei, STL-based algorithm for sea surface temperature prediction, Mar. Environ. Sci. 39 (06) (2020) 918–925.
[14] H. Q, H. Zeyu, X. Huifang, S. Wei, D. Yanling, A sea surface temperature prediction method based on EMD-GRU model, Adv. Lasers Optoelectron. (2021) 1–16.
[15] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, W. Woo, Convolutional LSTM network: a machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, 2015, pp. 802–810.
[16] P. Han, Z. Li, Y. Liu, P. Zhao, S. Shang, Contextualized point-of-interest recommendation, in: Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20, 2020.
[17] C. Chen, K. Li, S.G. Teo, X. Zou, K. Wang, J. Wang, Z. Zeng, Gated residual recurrent graph neural networks for traffic prediction, in: The Thirty-Third AAAI Conference on Artificial Intelligence, 2019, pp. 485–492.
[18] X. Jiahui, W. Jingchang, C. Ling, W. Yong, A prediction model of surface water quality based on graph neural network, J. Zhejiang Univ. (Eng. Ed.) (2021) 1–7.
[19] X. Bi, Z. Liu, Y. He, X. Zhao, Y. Sun, H. Liu, GNEA: a graph neural network with ELM aggregator for brain network classification, Complexity 2020 (2020) 8813738.
[20] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005, vol. 2, 2005, pp. 729–734.
[21] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80.
[22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24.
[23] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, Comput. Sci. (2013).
[24] P. Han, P. Yang, P. Zhao, S. Shang, P. Kalnis, GCN-MF: disease-gene association identification by graph convolutional networks and matrix factorization, in: The 25th ACM SIGKDD International Conference, 2019.
[25] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, 2017.
[26] A. Micheli, Neural network for graphs: a contextual constructive approach, IEEE Trans. Neural Netw. 20 (3) (2009) 498–511.
[27] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 1024–1034.
[28] J. Liu, G.P. Ong, X. Chen, GraphSAGE-based traffic speed forecasting for segment network with sparse data, IEEE Trans. Intell. Transp. Syst. (2020) 1–12.
[29] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2017.
[30] D. Bacciu, F. Errica, A. Micheli, Contextual graph Markov model: a deep and generative approach to graph processing, in: International Conference on Machine Learning (ICML2018), 2018.
[31] A.J. Bishara, J. Hittner, Testing the significance of a correlation with nonnormal data: comparison of Pearson, Spearman, transformation, and resampling approaches, Psychol. Methods 17 (3) (2012) 399–417.
[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 2019, pp. 8024–8035.
[33] D. Kingma, J. Ba, Adam: a method for stochastic optimization, Comput. Sci. (2014).
[34] Gerhard Nahler, Pearson correlation coefficient, Springer, Vienna, 2009, p. 132 (Chapter 1025).