Applied Intelligence (2023) 53:26067–26078
https://doi.org/10.1007/s10489-023-04632-2

Recurrent neural networks integrate multiple graph operators for spatial time series prediction

Bo Peng[1,2] · Yuanming Ding[1,2] · Qingyu Xia[1,2] · Yang Yang[1,2]

Accepted: 10 April 2023 / Published online: 18 August 2023

Abstract
For multivariate time series forecasting problems, entirely using the dependencies between series is a crucial way to achieve accurate forecasting. Real-life multivariate time series often have complex time dependence, spatial dependence and high nonlinearity simultaneously, so Euclidean space is no longer sufficient to describe them. graph neural network presents a vital idea to solve this problem by modelling multivariate time series as graphs. Using the nature of graphs makes it possible to capture the dependencies between multivariate time series. However, no graph structure can perfectly characterize the relationships among multivariate time series; the facts underlying multivariate time series are much more complex. Therefore, we propose an integrated model (iGoRNN), which improves the model's understanding of the deep relationships of multivariate time series by fusing the information captured by multiple graph operators through an integrator with a specific structure. In addition, we conducted experiments on the Metr-LA and PeMS-BAY datasets. The experimental results show that the proposed model outperforms the baseline model in three evaluation metrics, MAE, MAPE and RMSE, and can forecast complex multivariate time series.

Keywords Multivariate time series · Graph neural networks · Graph operator integrator · Space dependence · Time dependence

1 Introduction

A time series is a set of random variables arranged in time order. Time series forecasting, on the other hand, uses known experience to estimate unknown future values of random variables based on the analysis of historical observations. In the field of traditional time series analysis, most of the studies are on univariate time series. The classical ones are Auto-Regressive (AR) [1] model, Moving Average (MA) [2], and Auto Regressive Moving Average (ARMA) [3] model. The AR model focuses on the historical observations themselves. The MA model focuses on the cumulative error in the forecast, eliminating the random fluctuations in the forecast. The ARMA model combines the two in a simple linear fashion, allowing the model to be analyzed from both perspectives in an integrated manner. However, this model can only be used to deal with small-scale smooth time series in univariate, homoskedasticity settings. Later, the proposal of the AutoRegressive Integrated Moving Average(ARIMA) [4] model made dealing with non-stationary time series possible. The main idea of this model is to transform the non-stationary series into a stationary series by differencing it to study it. On the other hand, many models emerged in heteroskedasticity, multivariate, and nonlinear time series analysis. For example, the Threshold Autoregressive (TAR) [5] model and the Auto-regressive conditional heteroskedasticity (ARCH) [6] model. The TAR model can divide the state space by setting a threshold and then use various linear forms to deal with nonlinear time series. The ARCH model applies to the case of heteroskedasticity. Its basic idea is to assume that the variance is a random variable obeying a normal distribution and

✉ Yuanming Ding
dingyuanming@dlu.edu.cn

Bo Peng
pengboag@gmail.com

Qingyu Xia
xiaqingyu0315@163.com

Yang Yang
yang_dldx@163.com

[1] Communication and Network Laboratory, Dalian University, 116622 Dalian, China

[2] School of Information Engineering, Dalian University, 116622 Dalian, China

to use a linear combination of the squared values of past finite duration series for regression analysis. Another central idea of traditional time series analysis is transforming it to the frequency domain. The spectral analysis method is a powerful tool for mining the hidden periodicity of time series. The main idea is to model the time series with a linear combination of sine and cosine by Fourier transform [7]. Modern spectral analysis methods are still applied in many fields and perform excellently.

Traditional models are usually effective only for specific temporal patterns. Machine learning models are more generalizable than traditional models and can often be used to deal with more complex time series. The common ones are Support Vector Regression (SVR) [8], Random Forests (RF) [9], and Extreme Gradient Boosting (Xgboost) [10, 11]. SVR belongs to the application of SVM to regression problems, and it can handle linear and nonlinear regression problems by selecting different kernel functions. RF is an integrated model that chooses a decision tree as its base model. It combines the outputs of the base models with specific rules to obtain the final predicted values. Xgboost, like Random Forest, is also an integration algorithm. However, instead of simply combining the outputs of the base model, the tree is continuously added to fit the residuals of the previous base model predictions so that the predicted values keep approximating the actual values.

In recent times, with the development of hardware technology, the arithmetic power of computers has been significantly improved. Neural networks became popular and developed. Many models for processing time series have also been born in this field. For example, an LSTM [12, 13] model with temporal information enhancement (T-LSTM) was proposed by Mou et al. [14]. The model is based on the original LSTM model and improves prediction accuracy by capturing the intrinsic correlation between traffic flow and temporal information. The above model relies purely on recurrent neural networks. Later, Wen et al. [15] made the first attempt to use CNNs [16–18] in time series prediction problems. The main idea is to model the anomaly detection problem in time series as an image segmentation problem and then use a U-net-like[19, 20] model architecture to process the time series in a convolutional manner, and it also achieves good results.

Although the above models have satisfied most of the time series prediction tasks, the multivariate prediction still needs improvements. The birth of graph neural networks (GNN)[21, 22] has dramatically impacted the dominance of traditional neural networks. Due to the graph structure that can represent non-Euclidean data, it has received attention from several research fields, such as computer vision [23, 24] and natural language processing [16, 25]. Breakthroughs have also occurred in the field of multivariate time series prediction. In 2020, Wu et al.[26] proposed a Multivariate Time series forecasting with Graph Neural Networks (MTGNN) model specifically for multivariate time series forecasting problems. In the same year, Song et al. [27] proposed the Spatial-Temporal Synchronous Graph Convolutional Networks (STSGCN) model. This model can effectively capture graphs' complex correlations and heterogeneities through a spatiotemporal synchronous modelling mechanism. Time series analysis models based on graph neural networks have flourished.

Graphical neural networks have been an essential tool for mining the spatial dependence of multivariate time series. Many scholars have attempted to improve modeling methods to exploit the inherent dependencies among multivariate time series to improve prediction accuracy. As a result, various graph models have emerged. These graph operators act similarly to the different convolution kernels in convolutional networks. As we know, in image processing, it is often necessary to match different convolution kernels, such as dilation convolution[28, 29], grouped convolution [30], and separable convolution. [31, 32], as needed if we want to achieve the desired results. Different convolutional kernels can extract different levels of features from the image. Inspired by convolutional networks, this paper is the first attempt to use multiple graph operators in feature extraction, using different graph operators to explore the spatial dependencies between nodes from different perspectives and then improve the accuracy of the model prediction.

2 Research background

Traffic flow is a typical multivariate time series, which refers to the number of traffic entities or other traffic indicators passing through a location, section, or road lane during a specific period. Most traffic flows are highly nonlinear [33], time-dependent and uncertain, making it difficult for traditional time series models to meet the needs of practical applications. In recent years, graph neural networks have seen rapid development. Their powerful representations can use to model multivariate time series spatially and achieve accurate predictions. Furthermore, this has become an important method for studying multivariate series problems. Typical ones are Diffusion Convolutional Recurrent Neural Networks (DCRNN) [34] and Gated Attention Networks for Learning on Large and Spatiotemporal Graphs (GaAN) [35]. The DCRNN model is based on a diffusion mechanism that captures spatial dependencies between nodes by randomly wandering around the graph; the GaAN is based on a multi-headed attention mechanism but adds a gated value to each head to adjust the importance of each attention head.

Most of the above algorithms only model traffic flow from a single perspective. However, considering the complexity of traffic road networks, there may be multiple spatial

dependencies at their underlying layers, so a single type of network cannot fully exploit the dependency information among nodes. This paper proposes an integrated model — Recurrent Neural Networks Integrating Graph gated attention and Graph diffusion convolution Operators (iGoRNN) to address the above problems. The model is based on an encoder-decoder architecture with a core building block for a graph operator integrator that efficiently fuses information captured by different graph operators to improve the network's ability to understand the spatial dependence of multivariate time-series data.

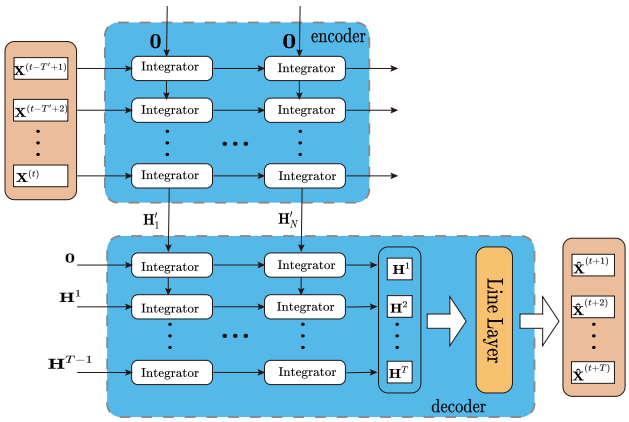


Fig. 1 Model Body Architecture

3 Research methodology

3.1 Problem description

In this paper, the sensor network in the road section is abstract as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. where $\mathcal{V}$ is the set of sensors in the road network; $N = |\mathcal{V}|$ denotes the number of sensors; $\mathcal{E}$ denotes the set of edges, representing the existence of links between sensors. $\mathbf{W} \in \mathbb{R}^{N\times N}$ is a weighted proximity matrix representing nodes' proximity.

To facilitate the description of the problem, we define the traffic flow signals collected by all sensors at time t as $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times P}$, where P denotes the number of traffic indicators detected by the sensors; and define $\hat{\mathbf{X}}^{(t)} \in \mathbb{R}^{N\times P}$ denotes the predicted value of the traffic indicators at time t. In turn, $\mathcal{X} = \left\{\mathbf{X}^{(t-T'+1)}, \mathbf{X}^{(t-T'+2)}, \cdots, \mathbf{X}^{(t)}\right\}$ can represent the observed values for the historical T' timestamps; $\hat{\mathcal{X}} = \left\{\hat{\mathbf{X}}^{(t+1)}, \hat{\mathbf{X}}^{(t+2)}, \cdots, \hat{\mathbf{X}}^{(t+T)}\right\}$ can represent the predicted values for the future T timestamps. Therefore, the ultimate goal of this paper is to learn a function $\mathscr{H}(\cdot)$ that maps T' historical timestamp data to future T timestamp data.

$$\hat{\mathcal{X}} = \mathscr{H}(\mathcal{X}) \tag{1}$$

3.2 Model architecture

In this paper, we design a encoder-decoder architecture, Fig. 1. Its encoding and decoding process is mainly done through graph operator integrators(Go-Integrator), and the vertical flow of information in the graph can be understood as the encoding and decoding process. The horizontal flow can be understood as the feature capture process. For more complex multivariate timing data, the depth of information mining can be increased by stacking integrator layers horizontally.

The encoder's input in the model is the historical observation $\mathcal{X}$; the output of the decoder is the future prediction $\hat{\mathcal{X}}$; $\mathbf{H} \in \mathbb{R}^{N\times Q}$ denotes the input or output of the intermediate hidden layer.

3.2.1 Graph operator integrator

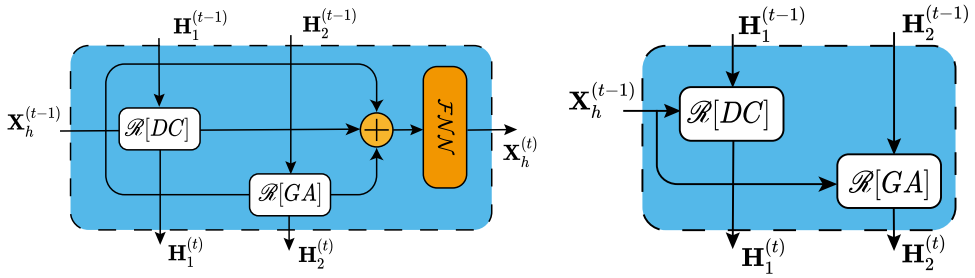
The Graph Operator Integrator (Go-Integrator) is the core building block of iGoRNN, which is used upwards to control the overall flow of information and downwards to fuse the information captured by each graph operator. In order to enable the integrator to perform information fusion efficiently, We present for the first time a feasible integrator architecture, see Fig. 2. Subfigure (a) is a standard information integrator for the start and intermediate nodes of the codec. At the same time, since the last layer of the encoder does not need to obtain an output, we have designed a simplified architecture subfigure (b), which can save computational resources to a certain extent.

Two graph networks are used inside the integrator for information mining, a diffusion convolutional network based on static graphs and a gated attention network based on dynamic graphs. The static graph can objectively portray the physical distance between sensors, and the dynamic graph can filter out the neighbouring nodes that have a high impact on the central node. The two graph networks, one based on physical space and one based on value space, each capture the spatial dependencies between nodes. Finally, an integrator is used to achieve the fusion of multiple features.

In the complete integrator unit, historical data is sent to two Graph Gated Recurrent Units (GGRU) [36, 37] along with implicit data for computation. Then its output is stitched with the original input and finally mapped to the specified dimension by the Feed Forward layer. Equation (2) gives the detailed process of fusion.

$$\hat{\mathbf{H}}_o^{(t)} = \mathscr{F}_{d_o}^{\tau}\left(\mathbf{X}^{(t)} \parallel \coprod_{i=1}^{I} \mathscr{R}[\Gamma_i]\left(\mathbf{X}^{(t)}, \mathbf{H}_i^{(t-1)}\right)\right) \tag{2}$$

Fig. 2 Go-integrator unit



a. Standard Structure b. Simplified Structure

Here, $\mathbf{X}^{(t)}$ denotes the historical observation at the current moment; I denotes the number of GGRU units involved in the calculation. $\mathbf{H}_i^{(t-1)}$ denotes the output of the i-th GGRU unit from the iGGRU of the previous layer, which is also used as the input of the i-th GGRU unit in this layer, and $\hat{\mathbf{H}}_o^{(t)}$ denotes the final output of the Go-Integrator.$\|$ denotes cascade, and $\coprod$ indicates sequential cascade. $\mathscr{R}[\Gamma_i]$ denotes the GGRU unit using the Γ_i graph operator, $\mathscr{F}_{d_o}^{\tau}$ denotes the feedforward neural network, d_o represents the dimension of model input, and τ denotes using the tanh activation function.

3.2.2 Graph gated recurrent unit

The Graph Gated Recurrent Unit (GGRU) is a particular type of Gated Recurrent Unit (GRU). GGRU uses a graph operator instead of the fully connected layer in the GRU, see Fig. 3. Compared with the classical GRU model, GGRU does not only learn time series patterns but also mines spatial dependencies between sequences by graph operators, making it applicable to spatial time series data.

The macro architecture of GGRU is consistent with that of a normal GRU, Fig. 3. where $\mathbf{X}^{(t)}$, $\mathbf{H}^{(t)}$ denotes the input and output of the t timestamp; $\mathbf{r}^{(t)}$, $\mathbf{u}^{(t)}$ denotes the gating state of the Reset gate and Update gate of the t timestamp; and Θ_r, Θ_u, Θ_C is the different filter parameters. $\Gamma_{\mathcal{G}}$ denotes the execution of the Γ operator in the specified graph network

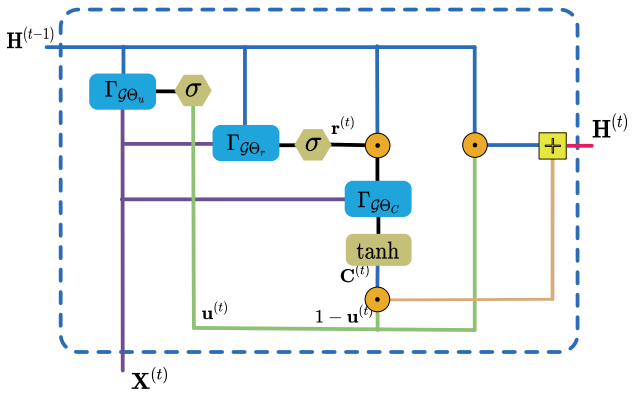


Fig. 3 Graph gated recurrent unit

$\mathcal{G}$, and $\odot$ denotes the Hadamard product. The specific calculation process is in the following equations.

$$\mathbf{r}^{(t)} = \sigma(\Gamma_{\mathcal{G}\Theta_r}[\mathbf{X}^{(t)} \parallel \mathbf{H}^{(t-1)}] + b_r) \tag{3}$$

$$\mathbf{u}^{(t)} = \sigma(\Gamma_{\mathcal{G}\Theta_u}[\mathbf{X}^{(t)} \parallel \mathbf{H}^{(t-1)}] + b_u) \tag{4}$$

$$\mathbf{C}^{(t)} = \tanh(\Gamma_{\mathcal{G}\Theta_C}[\mathbf{X}^{(t)} \parallel (\mathbf{r}^{(t)} \odot \mathbf{H}^{(t-1)})] + b_c) \tag{5}$$

$$\mathbf{H}^{(t)} = \mathbf{u}^{(t)} \odot \mathbf{H}^{(t-1)} + (1 - \mathbf{u}^{(t)}) \odot \mathbf{C}^{(t)} \tag{6}$$

Equations (3) and (4) are the update process of the gated state. The GGRU first fuses the input data $\mathbf{X}^{(t)}$ of the current moment with the output data $\mathbf{H}^{(t-1)}$ of the last moment and then feeds it into the graph operator network for capturing the spatial dependence. Finally, After activation by the sigmoid function, the gating states $\mathbf{r}^{(t)}$ and $\mathbf{u}^{(t)}$ at the t time are obtained, where $\mathbf{r}, \mathbf{u} \in (0, 1)$. After the gating signal update, the reset and update gates can capture the essential features in the current message.

Equation (5) is the process of capturing essential features using reset gates. First, the reset gate resets the hidden matrix $\mathbf{H}^{(t-1)}$. The reset process is similar to the forgetting process, in which the historical information retained in the matrix $\mathbf{H}^{(t-1)}$ will further reduce. Then it is fused with the observed data $\mathbf{X}^{(t)}$ at the current moment and fed into the graph operator network for the spatial feature extraction. Finally, after the tanh function activation will obtain the candidate matrix $\mathbf{C}^{(t)}$.

The final step of the algorithm is the update process of updating the historical information $\mathbf{H}^{(t-1)}$ using the candidate matrix $\mathbf{C}^{(t)}$, (6). Thus, we obtain the output $\mathbf{H}^{(t)}$ at the current moment.

In practice, We embed the diffusion convolution and gated attention operators into GGRU to obtain two types of feature capturers, GGRU(DC) and GGRU(GA), respectively. These two GGRUs can capture node information from different perspectives and thus improve the network's understanding of spatial-temporal data.

3.2.3 Graph diffusion convolution operator

Graph Convolutional Networks (GCN) [38–40] introduces convolution to general graph-structured data. For such non-

gridded data, instead of predicting each node individually, GCN aggregates useful information from neighboring nodes as much as possible for each node in a process similar to image convolution. Like CNN, GCN has features such as feature learning, parameter binding, and invariance.

However, GCN uses only the direct neighbors of each node. Nevertheless, we cannot simply interpret the nodes in the graph as a binary relationship. The basic facts between nodes are much more complex than that. The proposed Diffusion Convolutional Neural Networks (DCNN) [41, 42] breaks through this layer of limitation and is a special kind of GCN network. DCNN aims at mining the spatial dependencies between nodes at a deeper level. In 2018, Li et al. [34] constructed the first DCRNN model based on the DCNN model to solve the prediction problem of spatial timing. Equation (7) expresses the diffusion convolution process.

$$\begin{aligned}\mathbf{H} &= \coprod_{q=1}^{Q}\left(a\sum_{p=1}^{P}(\Theta_O\mathcal{P}_O^K+\Theta_I\mathcal{P}_I^K)\mathbf{X}_{[:,p]}\right)\\ \mathcal{P}_O^K &= \sum_{k=0}^{K}\alpha(1-\alpha)^k(\mathbf{D}_O^{-1}\mathbf{W})^k\\ \mathcal{P}_I^K &= \sum_{k=0}^{K}\beta(1-\beta)^k(\mathbf{D}_I^{-1}\mathbf{W})^k\end{aligned} \tag{7}$$

Here, $\mathbf{X}\in\mathbb{R}^{N\times P}$ represents the model's input, and $\mathbf{H}\in\mathbb{R}^{N\times Q}$ represents the model's output. The diffusion process is truncated using a finite number of K steps and eventually maps the inputs of P-dimensional features to Q-dimensional outputs. $\mathbf{D}_O, \mathbf{D}_I$ denotes each node's outgoing and incoming degree matrices in the graph, respectively, and further $\mathbf{D}_O^{-1}\mathbf{W}, \mathbf{D}_I^{-1}\mathbf{W}$ represents the forward state transfer matrix and the reverse state transfer matrix. $\alpha, \beta \in [0, 1]$ denotes the restart probability of random wandering, $\Theta_O, \Theta_I \in \mathbb{R}^{Q\times P\times K}$ indicates the model parameters, and $\Theta_{O[q,p,k]} = \alpha(1-\alpha)^k, \Theta_{I[q,p,k]} = \beta(1-\beta)^k$. $\mathcal{P}_{O[i,:]}^K, \mathcal{P}_{I[i,:]}^K$ denotes the probability of performing K-step diffusion from node i to its neighboring nodes, where the former denotes the forward diffusion probability and the latter denotes the reverse diffusion probability.

Equation (7) describes a two-way diffusion process, and using this two-way diffusion process allows the model to have the ability to capture both upstream and downstream traffic impacts. Exceptionally, if the restart probability value $\beta = 0$, the model will change to a one-way diffusion process that can apply to capture features on particular road segments. They are giving the model more flexibility and adaptability.

Since graph diffusion convolution operator is based on static graphs, its adjacency matrix is constructed using a distance-based Gaussian kernel[43], see (8).

$$w_{ij} = \begin{cases}\exp\left(-\dfrac{\mathrm{dist}(v_i, v_j)^2}{\sigma^2}\right) & \text{if } \mathrm{dist}(v_i, v_j) \le \kappa\\ 0 & \text{otherwise}\end{cases} \tag{8}$$

Here, v_i denotes the i-th sensor in the road, $\mathrm{dist}(v_i, v_j)$ and w_{ij} denote the actual distance and weight between two sensors, respectively, and σ is the standard deviation of the distance set. κ is the threshold parameter; by setting a higher threshold value, the graph can be made sparse, and thus the convergence speed of the model can be improved.

3.2.4 Graph gated attention operator

Graph-gated attention networks(GaAN) are special graph attention networks (GAT)[44, 45]. Classical attentional mechanisms emerged in the field of computer vision and flourished in the field of natural language processing. Instead of focusing on the information as a whole, the attention mechanism focuses limited attention on the critical information. The advantage of introducing the attention mechanism is that it has fewer parameters to get the most helpful information and avoid wasting resources. However, at the same time, it may cause information loss. To solve this problem, Vaswani et al.[46] propose the Multi-head Attention mechanism[47, 48], which is to add multiple attention heads to the network, each expanding into a separate subspace, allowing the model to capture information more comprehensively about several different perspectives simultaneously. However, the multi-headed attention mechanism needs to be revised in that it needs to consider the differences in importance between multiple heads. Later, Zhang et al.[35] proposed the gated attention mechanism, which can explore multiple representation subspaces between the central node and its neighbouring nodes while focusing on the value of these subspaces to dynamically adjust each contribution subspace to the outcome in a gated manner.

Suppose the K-head attention mechanism is applied to Graph Gated Attention networks for feature capture. For each node i, a K-dimensional gating vector $\mathbf{g}_i$ is used to regulate the contribution of each attention head. Equation (9) gives the specific calculation of the gating vector $\mathbf{g}_i$.

$$\begin{aligned}\mathbf{g}_i &= [g_i^{(1)}, \cdots, g_i^{(K)}]\\ &= \mathscr{L}_{\theta_K}^{\sigma}\left(\mathbf{x}_i \parallel \underset{j\in\mathcal{N}_i}{\mathbf{Max}}\left(\{\mathscr{L}_{\theta_m}(\mathbf{z}_j)\}\right) \parallel \frac{\sum_{j\in\mathcal{N}_i}\mathbf{z}_j}{|\mathcal{N}_i|}\right)\end{aligned} \tag{9}$$

Given node i, all its neighboring nodes are represented by the set $\mathcal{N}_i$. $\mathbf{x}_i = \mathbf{X}_{i,:}$ denotes the input feature vector of this node; $\mathbf{z}_{\mathcal{N}_i} = \{\mathbf{z}_j \mid j \in \mathcal{N}_i\}$ means the set of reference vectors of all its adjacent nodes, where $\mathbf{z}_i = \mathscr{L}_{\theta_h}(\mathbf{x}_i)$. **Max** indicates the maximum value element-wise. $\mathscr{L}^{\sigma}_{\theta_K}$ denotes the mapping of the vectors to the K dimension and scaling the result to the [0, 1] interval using the sigmoid function.

Since Graph Gated Attention is a dynamic network, it needs to calculate the weight relationship between node i and its neighbors in real-time. The specific calculation process is shown in (10).

$$
\begin{aligned}
w_{ij}^{k} &= \frac{\exp\left(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_j)\right)}{\sum_{l=1}^{|\mathcal{N}_i|} \exp\left(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_l)\right)} \\
\phi_w^{(k)}(\mathbf{x}, \mathbf{z}) &= \langle \mathscr{L}_{\theta_{xa}^{(k)}}(\mathbf{x}), \mathscr{L}_{\theta_{za}^{(k)}}(\mathbf{z}) \rangle
\end{aligned} \tag{10}
$$

Since there are a total of K attention heads in the Graph Gated Attention Operator algorithm, we must compute a weight matrix for each head. Here w_{ij}^{k} denotes the weights of node i and node j under the k-th attention head. $\mathscr{L}_{\theta_{xa}^{(k)}}, \mathscr{L}_{\theta_{za}^{(k)}}$ is used to generate the query and key vectors of dimension d_a. $\langle \cdot, \cdot \rangle$ denotes the inner product operation.

Once the gating values and weight matrices of each attention head are obtained, the graph aggregation process of node i can be completed. The information captured by the K attention heads is first multiplied by their respective gating vectors and then stitched together with the original input. Finally, the output vector y_i is obtained after mapping by the fully connected layer. Equation (11) describes the specific implementation process.

$$
\mathbf{y}_i = \mathscr{L}_{\theta_o}\left(\mathbf{x}_i \parallel \coprod_{k=1}^{K}\left(g_i^{(k)} \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} \mathscr{L}_{\theta_v^{(k)}}^{\iota}(\mathbf{z}_j)\right)\right) \tag{11}
$$

Here, $\mathscr{L}_{\theta_v^{(k)}}$ generates a value vector of dimension d_v, $\mathscr{L}_{\theta_o}$ is responsible for mapping the final output to a specific dimension, and ι denotes the LeakRelu activation function.

4 Experimental design

4.1 Experimental setup

The METR-LA and PeMS-BAY datasets used in the experiment were obtained from Li et al. [34]. The dataset contains traffic information for the Los Angeles and California Freeway. The nodes in the dataset represent the measured traffic speed sensor IDs, and the edges are the proximity calculated based on the distance between sensors in the road network using (8). The sampling interval of the sensor is 5 minutes. Table 1 records the detailed statistical information of the data.

Table 1 Statistical information of the experimental dataset

Dataset	#Nodes	#Edges	#Timestamps	E(X)	D(X)
METR-LA	207	1722	34272	54.41	19.49
PeMS-BAY	325	2694	52116	62.74	9.44

Figure 4 shows the traffic speed variation over 48 hours on three road segments for both data sets.

This paper uses the first 70% of the data set as the training set, the middle 10% as the validation set, and the last 20% as the test set. The sliding window is used to group the sequences, and the window size is set to 12. The grouped data is used as the input to the model to predict the traffic flow in the next hour. In the integrator, the diffusion convolution operator uses a bi-directional diffusion mode with truncation steps set to 2. The number of attention heads in the gated attention operator is set to 2. The learning rate is initialized to 0.01 and dynamically adjusted using the Adam optimizer with a multiplier of 0.99^{epoch}.

Before the model starts training, the data are transformed into a distribution with mean 0 and variance 1 based on the expectation and sample variance of the training set. The advantage of performing normalization is that the data set is limited to a smaller range of values, which facilitates the computation of gradients in the post-order and ensures fast convergence of the model. The uniformity of the data scale can avoid the influence of different representations on the training results. Equations (12) and (13) correspond to the normalization and denormalization processes, respectively.

$$
\mathbf{X}^{*} = \frac{\mathbf{X} - E(\mathbf{X})}{\sqrt{D(\mathbf{X})}} \tag{12}
$$

$$
\mathbf{X} = \mathbf{X}^{*} \cdot \sqrt{D(\mathbf{X})} + E(\mathbf{X}) \tag{13}
$$

Here, $\mathbf{X}$ denotes the training set sample, and $E(\mathbf{X})$, $D(\mathbf{X})$ indicates the training set sample mean and sample variance, respectively.

4.2 Comparison experiments

In this section, we selected four significant classes of standard time series analysis models in the field of time series. These include the statistical learning model ARIMA [49], machine learning model LSVM [50], deep learning model FC-lSTM [51] and graph-based deep learning models DCRNN[34], STGCN [52], GaAN [35], ASTGCN [53], and GMAN [54]. Experiments use Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error

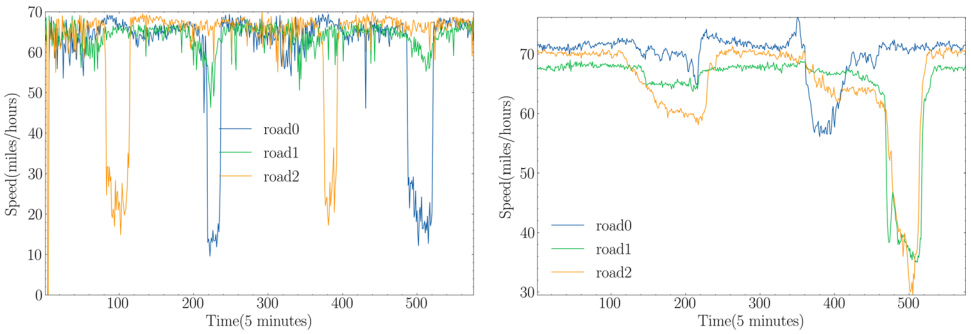


Fig. 4 Traffic flow changes over 48 hours on three road sections of the METR-LA and PeMS-BAY datasets

(MAPE) is used as evaluation metrics to assess the prediction performance of the models.

$$\mathrm{MAE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} |x_i - \hat{x_i}| \tag{14}$$

$$\mathrm{RMSE}(\mathbf{x}, \hat{\mathbf{x}}) = \sqrt{\frac{1}{|\Omega|} \left| \sum_{i \in \Omega} (x_i - \hat{x_i})^2 \right|} \tag{15}$$

$$\mathrm{MAPE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left| \frac{x_i - \hat{x}}{x_i} \right| \tag{16}$$

Tables 2 and 3 compare the prediction results of the iGoRNN model with the other baseline models. By comparing the scores of each model on long-term and short-term time series predictions, we can see that the graph-based neural network model achieves better prediction accuracy. For smoother sequences (PeMS-BAY), some traditional models achieved good results even for short-term predictions (15 min) but performed poorly for long-time predictions (60 min). For unstable sequences (Metr-LA), conventional models perform poorly in both short- and long-term predictions. On the other hand, the GaAN model performs well on the Metr-LA dataset. GMAN is more suitable for smoother series. All these illustrate the importance of introducing graph structure in complex time series forecasting. Moreover, the model we designed shows optimal or suboptimal performance in long-term and short-term forecasting, especially for more complex time series, because it simultaneously introduces the advantages of multiple graph structures.

4.3 Ablation experiments

We designed several similar Integrator structures (Fig. 5) to evaluate our model comprehensively and did further ablation experiments using the METR-LA dataset.

Figure 6 shows the reduction of MAE loss values on the training and validation sets for different prediction forecast lengths. The figure shows that most integrators can reduce the forecast accuracy to an approximate value, except for the individual model, which requires better convergence. Fig. 7-a shows the magnitude of the MAE loss values for the final eight models, from which we can see that iGoRNN$^{\ddagger\downarrow\prime}$ achieves the best results for short-term predictions. In the long-term forecast, iGoRNN$^{\dagger\downarrow\prime}$ works best. Fig. 7-b shows the number of Floating Point Operations (FLOPs) required for each model. The parallel structure is generally less computationally intensive than the series structure, mainly because the second graph operator of the latter is susceptible to the intermediate dimensionality of the model. Moreover, the parallel structure is more conducive to deploying par-

Table 2 Performance comparison of different traffic speed prediction models on the METR-LA dataset

Models/T	15 min			30 min			60 min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
ARIMA	3.99	8.21	9.60%	5.15	10.45	12.70%	6.52	10.11	15.80%
LSVR	3.99	8.45	9.30%	5.05	10.87	12.10%	6.72	13.76	16.70%
FC-LSTM	3.44	6.30	9.60%	3.77	7.23	10.90%	4.37	8.69	13.20%
DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.59	10.50%
STGCN	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
GaAN	2.71	5.24	6.99%	3.12	6.36	8.56%	3.64	7.65	10.62%
ASTGCN	2.75	5.62	7.51%	3.31	6.98	9.32%	4.52	9.24	12.62%
GMAN	2.69	5.55	7.42%	3.15	6.78	9.02%	4.03	8.11	11.72%
iGoRNN	**2.67**	**5.22**	**6.91%**	**3.09**	**6.20**	**8.51%**	**3.24**	**7.43**	**9.97%**

Table 3 Performance comparison of different traffic speed prediction models on the PeMS-BAY dataset

Models/T	15 min			30 min			60 min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
ARIMA	1.62	3.30	3.50%	2.33	4.76	5.40%	3.38	6.50	8.30%
LSVR	1.85	3.59	3.80%	2.48	5.18	5.50%	3.28	7.08	8.00%
FC-LSTM	2.05	4.19	4.80%	2.20	4.55	5.20%	2.37	4.96	5.70%
DCRNN	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
GaAN	1.41	3.21	3.11%	1.70	3.83	3.86%	2.12	4.80	5.34%
ASTGCN	1.32	2.78	2.75%	1.75	3.98	3.95%	2.32	5.41	5.51%
GMAN	1.34	2.82	2.81%	**1.62**	3.72	3.63%	1.86	4.32	4.31%
iGoRNN	**1.30**	**2.73**	**2.80%**	1.63	**3.66**	**3.62%**	**1.83**	**4.29**	**4.26%**

allelized computations and thus has better advantages in large-scale computations.

Due to the limitation of experimental resources, here we only compare the performance and efficiency of the iGoRNN

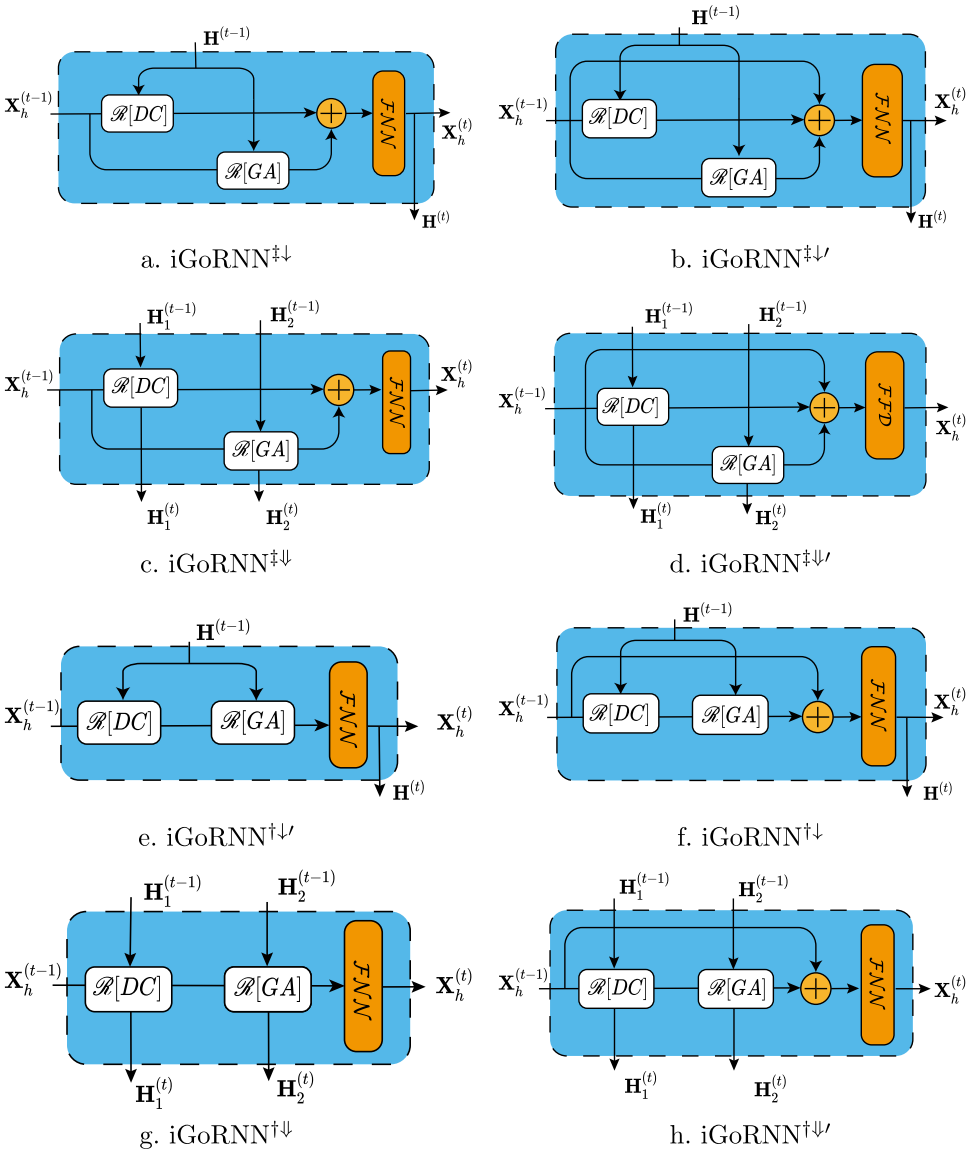


Fig. 5 Eight different Integrator structures. † indicates the use of a serial structure, ‡ indicates the use of a parallel structure, ⇓ indicates that each GGRU unit has separate hidden layer integrator inputs and outputs, and ↓ indicates that all GGRU units share the same hidden layer. ′ indicates the inclusion of a residual-like structure

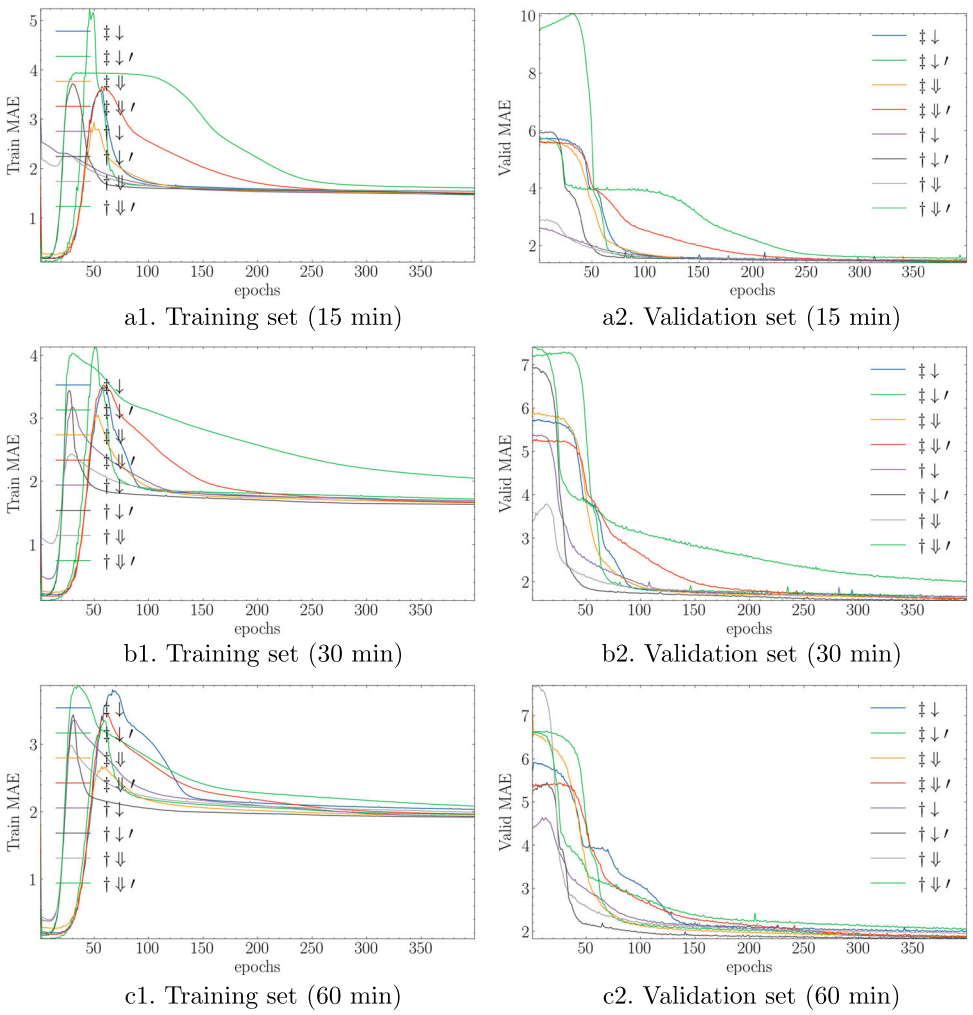


Fig. 6 The decreasing curve of Loss value for different time lengths predicted by iGoRNN model with different Integrator

model with a single-layer stacking structure. Nevertheless, it has shown a relatively good performance. We expect to conduct more in-depth analysis on more complex datasets in the future, including exploring more efficient ways of information aggregation and the combined effect of the number of model stacking layers on prediction results and time consumption.

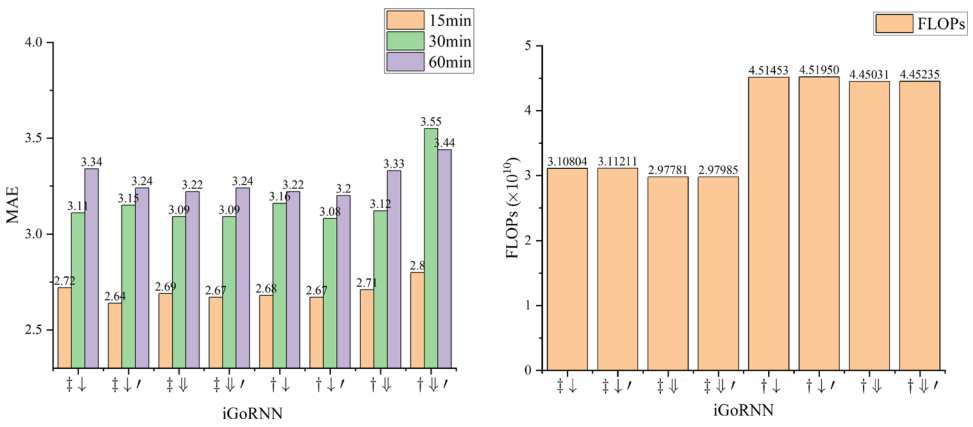


Fig. 7 Performance Analysis of the iGoRNN Model Using Different Integrators. Sub-Fig.a represents the optimal MAE loss value of the model for different prediction lengths, and sub-Fig.b represents the number of floating point operations for each model

5 Conclusion

Multivariate time series analysis has applications in economics, sociology, meteorology, environmental science, engineering, etc. Previous authors have proposed many time series models to meet the needs of various sectors of society. These models have a core concept of the mining as much quality information as possible from historical observations to achieve accurate forecasting. For univariate time series, the models can only achieve precise forecasting by thoroughly learning the hidden periodic statement of the time series. However, for multivariate time series, in addition to mining the frequent patterns from the time series itself, exploiting the potential dependencies between multiple time series will also positively affect the final prediction results. To fully exploit the spatial dependencies among multivariate time series, this paper designs an integrated model iGoRNN. The model assumes no single graph network can mine all the underlying correlations among multivariate time series. Hence, the iGoRNN model utilizes multiple graph operators to capture quality information from the series more comprehensively from different perspectives. The network as a whole also adopts a recurrent structure based on an encoder-decoder so that the model can aggregate temporal features while capturing spatial features instead of fragmenting the intersectionality of spatiotemporal features. Finally, this paper conducts comparison experiments with the baseline model using the publicly available METR-LA dataset and PeMS-BAY dataset. The experimental results show that the iGoRNN network is better than the other models in terms of prediction accuracy and can be competent for multivariate time series prediction tasks. Meanwhile, this paper gives seven other Integrator modules to complement the prediction tasks in different domains.

Funding This research was funded by the National Natural Science Foundation of China and General Project Fund In The Field of Equipment Development Department, grant number No.61901079, No.61403110308. The APC was funded by Dalian University.

Data Availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration

Conflicts of interest The authors declare no conflict of interest.

References

1. Chen Y, Tong C, Ge Y, Lan T (2021) Fault detection based on auto-regressive extreme learning machine for nonlinear dynamic processes. Appl Soft Comput 106:107319. https://doi.org/10.1016/j.asoc.2021.107319
2. Finesso L, Spreij P (2019) Approximation of nonnegative systems by moving averages of fixed order. Automatica 107:1–8. https://doi.org/10.1016/j.automatica.2019.05.007
3. Caliwag A, Lim W (2021) Optimal least square vector autoregressive moving average for battery state of charge estimation and forecasting. ICT Express 7(4):493–496. https://doi.org/10.1016/j.icte.2021.03.008
4. Yao L, Ma R, Wang H (2021) Baidu index-based forecast of daily tourist arrivals through rescaled range analysis, support vector regression, and autoregressive integrated moving average. Alexandria Eng J 60(1):365–372. https://doi.org/10.1016/j.aej.2020.08.037
5. Yang Y, Ling S (2017) Self-weighted LAD-based inference for heavy-tailed threshold autoregressive models. J Econ 197(2):368–381. https://doi.org/10.1016/j.jeconom.2016.11.009
6. Chang F, Huang H, Chan AHS, Man SS, Gong Y, Zhou H (2022) Capturing long-memory properties in road fatality rate series by an autoregressive fractionally integrated moving average model with generalized autoregressive conditional heteroscedasticity: a case study of florida, the united states, 1975–2018. J Saf Res 81:216–224. https://doi.org/10.1016/j.jsr.2022.02.013
7. Khan NA, Ali S, Choi K (2022) An efficient and accurate multisensor if estimator based on doa information and order of fractional fourier transform. Entropy 24(4):452
8. Liu X, Dong X, Zhang L, Chen J, Wang C (2023) Least squares support vector regression for complex censored data. Artif Intell Med 136:102497. https://doi.org/10.1016/j.artmed.2023.102497
9. Gao W, Xu F, Zhou Z-H (2022) Towards convergence rate analysis of random forests for classification. Artif Intell 313:103788. https://doi.org/10.1016/j.artint.2022.103788
10. Turska E, Jurga S, Piskorski J (2021) Mood disorder detection in adolescents by classification trees, random forests and xgboost in presence of missing data. Entropy 23(9):1210
11. Lee S, Park J, Kim N, Lee T, Quagliato L (2023) Extreme gradient boosting-inspired process optimization algorithm for manufacturing engineering applications. Mater Des 226:111625. https://doi.org/10.1016/j.matdes.2023.111625
12. Yu Y, Si X, Hu C, Zhang J (2019) A review of recurrent neural networks: lstm cells and network architectures. Neural Comput 31(7):1235–1270
13. Lombardo E, Rabe M, Xiong Y, Nierer L, Cusumano D, Placidi L, Boldrini L, Corradini S, Niyazi M, Reiner M, Belka C, Kurz C, Riboldi M, Landry G (2023) Evaluation of real-time tumor contour prediction using LSTM networks for MR-guided radiotherapy. Radiother Oncol 182:109555. https://doi.org/10.1016/j.radonc.2023.109555
14. Mou L, Zhao P, Xie H, Chen Y (2019) T-lstm: a long short-term memory neural network enhanced by temporal information for traffic flow prediction. Ieee Access 7:98053–98060
15. Wen T Keyes R (2019) Time series anomaly detection using convolutional neural networks and transfer learning. arXiv:1905.13628
16. Wang S, Ren P, Takyi-Aninakwa P, Jin S, Fernandez C (2022) A critical review of improved deep convolutional neural network for multi-timescale state prediction of lithium-ion batteries. Energies 15(14):5053. https://doi.org/10.3390/en15145053
17. Xie Y, Sun W, Ren M, Chen S, Huang Z, Pan X (2023) Stacking ensemble learning models for daily runoff prediction using 1d and 2d CNNs. Expert Syst Appl 217:119469. https://doi.org/10.1016/j.eswa.2022.119469
18. Li R, Gao R, Suganthan PN (2023) A decomposition-based hybrid ensemble CNN framework for driver fatigue recognition. Inf Sci 624:833–848. https://doi.org/10.1016/j.ins.2022.12.088
19. Al-Battal AF, Lerman IR, Nguyen TQ (2023) Multi-path decoder u-net: a weakly trained real-time segmentation network for object detection and localization in ultrasound scans. Comput Med Imag-

ing Graph 102205. https://doi.org/10.1016/j.compmedimag.2023.102205
20. Li Wang Y, Lai Y-Z, Li Q-Q, Huang S-T (2022) RU-net: an improved u-net placenta segmentation network based on ResNet. Comput Methods Prog Biomed 227:107206. https://doi.org/10.1016/j.cmpb.2022.107206
21. Gao J, Wu J, Zhang X, Li Y, Han C, Guo C (2022) Partition and learned clustering with joined-training: active learning of GNNs on large-scale graph. Knowl-Based Syst 258:110050. https://doi.org/10.1016/j.knosys.2022.110050
22. Shi M, Tang Y, Zhu X, Huang Y, Wilson D, Zhuang Y, Liu J (2022) Genetic-GNN: evolutionary architecture search for graph neural networks. Knowl-Based Syst 247:108752. https://doi.org/10.1016/j.knosys.2022.108752
23. Liu K, Gao L, Khan NM, Qi L, Guan L (2021) Integrating vertex and edge features with graph convolutional networks for skeleton-based action recognition. Neurocomputing 466:190–201. https://doi.org/10.1016/j.neucom.2021.09.034
24. Sun Y, Huang H, Yun X, Yang B, Dong K (2022) Triplet attention multiple spacetime-semantic graph convolutional network for skeleton-based action recognition. Appl Intell 52(1):113–126
25. Liu F, Zheng J, Zheng L, Chen C (2020) Combining attention-based bidirectional gated recurrent neural network and two-dimensional convolutional neural network for document-level sentiment classification. Neurocomputing 371:39–50. https://doi.org/10.1016/j.neucom.2019.09.012
26. Wu Z Pan S Long G Jiang J Chang X Zhang C (2020) Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 753–763
27. Song C, Lin Y, Guo S, Wan H (2020) Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting. Proc AAAI Conf Artif Intell 34(01):914–921. https://doi.org/10.1609/aaai.v34i01.5438
28. Tao H, Duan Q (2023) An adaptive frame selection network with enhanced dilated convolution for video smoke recognition. Expert Syst Appl 215:119371. https://doi.org/10.1016/j.eswa.2022.119371
29. Salehi A, Balasubramanian M (2023) DDCNet: deep dilated convolutional neural network for dense prediction. Neurocomputing 523:116–129. https://doi.org/10.1016/j.neucom.2022.12.024
30. Liu S, Wang A, Deng X, Yang C (2022) MGNN: a multiscale grouped convolutional neural network for efficient atrial fibrillation detection. Comput Biol Med 148:105863. https://doi.org/10.1016/j.compbiomed.2022.105863
31. Lu Y, Jiang M, Wei L, Zhang J, Wang Z, Wei B, Xia L (2021) Automated arrhythmia classification using depthwise separable convolutional neural network with focal loss. Biomed Signal Proc Control 69:102843. https://doi.org/10.1016/j.bspc.2021.102843
32. Gan C, Wang L, Zhang Z, Wang Z (2020) Sparse attention based separable dilated convolutional neural network for targeted sentiment analysis. Knowl-Based Syst 188:104827. https://doi.org/10.1016/j.knosys.2019.06.035
33. Bas E, Egrioglu E, Aladag CH, Yolcu U (2015) Fuzzy-time-series network used to forecast linear and nonlinear time series. Appl Intell 43(2):343–355. https://doi.org/10.1007/s10489-015-0647-0
34. Li Y Yu R Shahabi C Liu Y (2017) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv:1707.01926
35. Zhang J Shi X Xie J Ma H King I Yeung D-Y (2018) Gaan: gated attention networks for learning on large and spatiotemporal graphs. arXiv:1803.07294
36. Man J, Dong H, Yang X, Meng Z, Jia L, Qin Y, Xin G (2022) GCG: graph convolutional network and gated recurrent unit method for high-speed train axle temperature forecasting. Mech Syst Signal Process 163:108102. https://doi.org/10.1016/j.ymssp.2021.108102
37. Liu Y, Song Z, Xu X, Rafique W, Zhang X, Shen J, Khosravi MR, Qi L (2022) Bidirectional gru networks-based next poi category prediction for healthcare. Int J Intell Syst 37(7):4020–4040
38. Wu F Souza A Zhang T Fifty C Yu T Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on machine learning, PMLR, pp 6861–6871
39. Schlichtkrull M Kipf TN Bloem P Berg Rvd Titov I Welling M (2018) Modeling relational data with graph convolutional networks. In: European semantic web conference, Springer, pp 593–607
40. Jiang Y, Lin H, Li Y, Rong Y, Cheng H, Huang X (2023) Exploiting node-feature bipartite graph in graph convolutional networks. Inf Sci 628:409–423. https://doi.org/10.1016/j.ins.2023.01.107
41. Atwood J Towsley D (2016) Diffusion-convolutional neural networks. Adv Neural Inf Process Syst 29
42. Mallick T, Balaprakash P, Rask E, Macfarlane J (2020) Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting. Transp Res Rec: J Transp Res Board 2674(9):473–488. https://doi.org/10.1177/0361198120930010
43. Shuman DI, Narang SK, Frossard P, Ortega A, Vandergheynst P (2013) The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Proc Mag 30(3):83–98
44. Wang Y, Wang H, Lu W, Yan Y (2023) HyGGE: hyperbolic graph attention network for reasoning over knowledge graphs. Inf Sci 630:190–205. https://doi.org/10.1016/j.ins.2023.02.050
45. Zhang X, Zhang C, Guo J, Peng C, Niu Z, Wu X (2023) Graph attention network with dynamic representation of relations for knowledge graph completion. Expert Syst Appl 219:119616. https://doi.org/10.1016/j.eswa.2023.119616
46. Vaswani A Shazeer N Parmar N Uszkoreit J Jones L Gomez AN Kaiser Ł Polosukhin I (2017) Attention is all you need. Advances in neural information processing systems 30
47. Chen Y, Xiong Q, Guo Y (2022) Session-based recommendation: learning multi-dimension interests via a multi-head attention graph neural network. Appl Soft Comput 131:109744. https://doi.org/10.1016/j.asoc.2022.109744
48. Zeng P, Hu G, Zhou X, Li S, Liu P, Liu S (2022) Muformer: a long sequence time-series forecasting model based on modified multi-head attention. Knowl-Based Syst 254:109584. https://doi.org/10.1016/j.knosys.2022.109584
49. Williams BM, Hoel LA (2003) Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results. J Transp Eng 129(6):664–672. https://doi.org/10.1061/(asce)0733-947x(2003)129:6(664)
50. Wu C-H, Ho J-M, Lee D-T (2004) Travel-time prediction with support vector regression. IEEE Trans Intell Transp Syst 5(4):276–281
51. Sutskever I Vinyals O Le QV (2014) Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst 27
52. Yu B Yin H Zhu Z (2017) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv:1709.04875
53. Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. Proceedings of the AAAI conference on artificial intelligence 33:922–929
54. Zheng C, Fan X, Wang C, Qi J (2020) Gman: a graph multi-attention network for traffic prediction. Proceedings of the AAAI conference on artificial intelligence 34:1234–1241