Information Sciences 648 (2023) 119560

Contents lists available at ScienceDirect

Information Sciences

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

Learning and integration of adaptive hybrid graph structures for multivariate time series forecasting

Ting Guo [a,b], Feng Hou [b,*], Yan Pang [a], Xiaoyun Jia [c], Zhongwei Wang [a], Ruili Wang [a,b,*]

[a] School of Logistics and Transportation, Central South University of Forestry and Technology, Changsha, Hunan, China
[b] School of Mathematical and Computational Sciences, Massey University, Auckland, New Zealand
[c] Institute of Governance & School of Politics and Public Administration, Shandong University, Qingdao, Shandong, China

ARTICLE INFO

Keywords:
Multivariate time series forecasting
Global graph
Local graph
Graph structure learning
Information fusion

ABSTRACT

Recent status-of-the-art methods for multivariate time series forecasting can be categorized into graph-based approach and global-local approach. The former approach uses graphs to represent the dependencies among variables and apply graph neural networks to the forecasting problem. The latter approach decomposes the matrix of multivariate time series into global components and local components to capture the shared information across variables. However, both approaches cannot capture the propagation delay of the dependencies among individual variables of a multivariate time series, for example, the congestion at intersection A has delayed effects on the neighboring intersection B. In addition, graph-based forecasting methods cannot capture the shared global tendency across the variables of a multivariate time series; and global-local forecasting methods cannot reflect the nonlinear inter-dependencies among variables of a multivariate time series. In this paper, we propose to combine the advantages of both approaches by integrating **A**daptive **G**lobal-**L**ocal **G**raph Structure Learning with **G**ated **R**ecurrent **U**nits (AGLG-GRU). We learn a global graph to represent the shared information across variables. And we learn dynamic local graphs to capture the local randomness and nonlinear dependencies among variables. We apply diffusion convolution and graph convolution operations to global and dynamic local graphs to integrate the information of graphs and update gated recurrent unit for multivariate time series forecasting. The experimental results on seven representative real-world datasets demonstrate that our approach outperforms various existing methods.

1. Introduction

Precisely forecasting the future states of systems and environment is vital for engineering, management, and decision (policy) making. Thus, time series forecasting, a task of predicting the future values of variables based on historical data, has been studied for over five decades. With the availability of massive amounts of real-time records of multiple variables collected by networked low-cost sensors and devices, time series forecasting has evolved into multivariate time series forecasting. Each variable in a multivariate time series is dependent not only on the past values of its own but also depends on the past or current values of other variables.

* Corresponding authors.
E-mail address: f.hou@massey.ac.nz (F. Hou).

https://doi.org/10.1016/j.ins.2023.119560
Received 22 April 2023; Received in revised form 26 June 2023; Accepted 13 August 2023

Available online 21 August 2023

Multivariate time series forecasting [24] can exploit such complex dependencies between variables and has been applied in electrical grid management, traffic congestion reduction for intelligent transportation systems, and financial services.

Traditional statistical forecasting methods, such as Auto-regressive (AR), Auto-regressive Integrated Moving Average (ARIMA) [45], and Vector Auto-regressive (VAR), are based on the linear dependency assumption between variables, which is not valid for most cases. Moreover, the complexity grows quadratically with the number of variables.

With the success of deep learning in computer vision, speech [31], and natural language processing, the RNN-based Seq2Seq model for language generation is adopted as a popular architecture for multi-step multivariate time series forecasting [15]. To capture the non-linearity of multivariate time series, LSTNet [19] uses convolutional neural networks to encode the short-term patterns of a local variable. Convolutional neural networks are also used by TPA-LSTM [29] to compute attention scores for multiple steps. However, these RNN-based Seq2Seq models can capture seasonality only when the dataset possesses homogeneous seasonal patterns [14]. Moreover, such models focus on the local forecasting of individual time series based on historical data and cannot efficiently capture the pair-wise interdependencies between variables. LSTM-FC [48] use a fully connected neural networks to integrate the predictions of variables, this can capture the pair-wise dependencies to some degree but is incompetent for data in non-Euclidean space [1].

Graph is very effective for modeling the pair-wise dependencies between variables, including the variables in non-Euclidean space. Recently, with the advancement in graph neural networks, many deep learning methods for multivariate time series forecasting have used deep learning on graphs to model the complex dependencies among variables. For example, DCRNN [34] proposes diffusion convolutional recurrent neural network for traffic forecasting; ASTGCN [13] uses attention-based graph convolutional networks for multivariate time series forecasting. For these graph-based multivariate time series forecasting methods, the nodes of a graph represent the variables of a multivariate time series; the edges represent the degree of dependencies between a pair of variables; and the graph is denoted by an adjacency matrix. The graph for multivariate time series forecasting usually is pre-defined or hand-crafted to model the inter-dependencies [34]. Such pre-defined graph cannot express the genuine interdependencies. Thus, researchers propose to learn a static adjacency matrix [38] or dynamic adjacency matrices [20] for multi-step forecasting. The former cannot reflect the dynamic interdependencies and the latter cannot capture the global components of a multivariate time series. Graph-based methods are good at modeling the interdependencies yet cannot identify the common behaviors of all the variables.

Global-local methods are proposed to capture and exploit the common behaviors of all variables of a multivariate time series. The key notion of global-local methods is to identify the shared global pattern of multiple time series and use it to facilitate the forecasting of local time series (variables). The global components of a multivariate time series characterize the general tendencies and dependencies of all the variables of a multivariate time series, for example the seasonality of electricity consumption and the congestion on roads during rush hours. Global-local forecasting models [24], [27] decompose the matrix of multivariate time series into an addition of k latent global components ($k < n, n$ is the number of variables) and a local component; and the future values of each variable are predicted based on these components. These global-local models can capture the similar trends of multiple variables as shared information to accelerate the forecasting of local (individual) time series. However, due to the linear addition form of local and global components, global-local methods cannot efficiently capture the nonlinearity of time series.

The limitations of existing graph-based and global-local forecasting methods can be summarized as follows: (i) Graph-based forecasting methods cannot capture the shared global tendency across the variables of a multivariate time series; (ii) Global-local forecasting methods cannot reflect the nonlinear interdependencies among variables of a multivariate time series; (iii) Existing methods cannot capture the propagation delay between the local variables of a multivariate time series, for example, the congestion at intersection A has delayed effects on the next intersection B.

In this paper, we propose to combine the advantages of graph-based forecasting methods and global-local forecasting methods by learning a global graph and multiple dynamic graphs for multi-step multivariate time series forecasting. The global graph is to capture the global information of a multivariate time series. The dynamic graphs are for capturing duo-local interdependencies and propagation delays between variables of the same multivariate time series. We propose a novel method to integrate and incorporate the global and local information by a joint convolution operation on the global and dynamic local graphs. We introduce a regularization term for model training to increase forecasting precision. We name our method as **A**daptive **G**lobal-**L**ocal **G**raph Structure Learning and Integration with **G**ated **R**ecurrent **U**nits (AGLG-GRU) for multivariate time series forecasting. Our contributions can be summarized as follows:

1. We learn a global graph by sampling from a differentiable probabilistic graphical model to capture the shared information and global dependencies among variables of a multivariate time series.

2. We use a hyper-network to learn dynamic local graphs to capture the local randomness and correlations between pairs of variables; and multi-scale historical data are input to the hyper-network to capture the delayed temporal dependencies of variables.

3. We devise a mechanism to integrate the information of global and dynamic local graphs by a joint graph convolution operation, and the integrated information is used to update the gated recurrent units for multi-step forecasting.

The remainder of this manuscript is organized as follows. In Section 2, related work is reviewed. The proposed methods are described in detail in Section 3. Section 4 presents the results of extensive experiments and the analysis of the results. This paper concludes with conclusions and future work in Section 5.

2. Related work

In recent decades, data-driven methods for multivariate time series forecasting have shown significant improvements over traditional knowledge-driven methods. With the success of deep learning, data-driven methods have predominantly adopted variants of

deep learning methods to improve forecasting performance. Although deep recurrent neural networks (RNN) can be directly used on multivariate time series data [10], simple RNN cannot efficiently model the complex dependencies among variables. Recent methods use graph or decomposing time series data into global and local components to model the dependencies of variables. These methods can be categorized into graph-based methods and global-local methods.

2.1. Graph-based methods for multivariate time series forecasting

Graph-based methods use graphs to represent the complex dependencies between variables and apply graph neural networks (GNN) to the graphs to forecast the future values of variables. Deep learning operations on graphs are usually performed in the spatial domain or spectral domain. Spatial-domain GNNs are usually applied to Euclidean data (e.g., images) with shareable parameters. Spectral-domain GNNs are performed on the eigenvectors of matrices associated with a graph, such as its adjacency matrix or Laplacian matrix. For graph-based multivariate time series forecasting, GNN operations are performed in spectral domain based the adjacency matrix of a graph.

Initially, the adjacency matrices are pre-defined and fixed for the model training. For instance, DCRNN [22] computes the pairwise road network distances between sensors and builds the adjacency matrix using thresholded Gaussian kernel; STGCN [42] also builds the distance-based graph similarly; some other methods use grid-based graph [41]. However, these pre-defined adjacency matrices are built using the prior geo-information, which is not available for some multivariate time series. Moreover, such static graphs cannot reflect the complex dependencies and dynamic properties of multivariate time series. AGC-Seq2Seq [47] proposes to learn a Laplacian matrix during the model training stage, however the Laplacian matrix is still fixed during the inference stage. ASTGCN [13] accompanies the adjacency matrix with a learned spatial attention matrix to dynamically adjust the impacting weights between nodes (variables); however still uses the empirical Laplacian matrix as a mask matrix [12].

Recently, inspired by the research in generic graph structure learning [9], adaptive graph [37], [46], [2], [11], [33] is proposed to learn unseen graph structures automatically from multivariate time series without prior knowledge. Among these studies, some proposed methods directly learn the adjacency matrix as model parameters [38], [13]; some methods use a latent network to extract features for constructing the graph [46], [12]; others calculate node similarities from node embeddings [37], [2]. However, such adaptively learned adjacency matrices are fixed for multi-step forecasting, thus cannot reflect the dynamic evolution of time series. To tackle this issue, some researchers propose to learn a series of adjacency matrices in a recurrent manner [20].

Many deep learning operations are implemented on the pre-defined graphs or learned adaptive graphs to forecast the future values based on the captured dependencies of variables. Graph convolutional networks are widely used for graph-based multivariate time series forecasting [42], [49], [13], [2], [39]. DCRNN [22] models the dynamics of the traffic flow as a diffusion process and proposes the diffusion convolution operation to capture the pair-wise spatial correlations using bidirectional random walks on a directed graph. DCRNN uses an encoder-decoder architecture to capture temporal dependency. Rank influence learning is introduced to the diffusion convolutional networks to adjust the importance of neighboring nodes [16]. A common practice of graph-based forecasting methods is to use the so-called spatial-temporal graph neural networks to model the spatial and temporal correlations separately [22], [42], [16], [49], [39], [46], [13], [2], i.e., they use graph convolutional networks or diffusion convolutional networks to capture the spatial dependencies and use recurrent neural networks to capture the temporal correlations. However, recurrent neural networks employed in these methods cannot capture long-range temporal sequences. Graph waveNet [37], [25] is proposed to handle long sequences. Inspired by the wavenet architecture for audio generation, the spatial-temporal layers in graph wavenet are connected by residual connections and the output of each spatial-temporal layer is skip connected to the output layer. Multi Scale Graph Wavenet [25] model is able to capture the global trends of change of variables with a learnable adjacency matrix.

2.2. Global-local methods for multivariate time series forecasting

Global-local methods [27], [36], [26], [24] assume that multivariate time series are composed of latent global components and local components. The global components are designed to capture the global trends and dependencies among variables of the entire multivariate time series. The local components are considered to handle the randomness of individual variables.

TRMF (Temporal Regularized Matrix Factorization) [43] employs matrix factorization to extract the global components of multivariate time series. Unlike the graph-based approaches, TRMF uses well-studied time series models to describe temporal dependencies explicitly; and designs a new regularization term to incorporate the temporal dependency into the matrix factorization model. TRMF can not only be used for forecasting of time series but also used for missing-value imputation of time series.

DeepGLO [27] extends the TRMF method to non-linear settings by imposing the temporal regularization with a temporal convolution network (TCN). DeepGLO is composed of two components: a matrix factorization model regularized by a TCN (TCN-MF) for global patterns and a temporal convolution network for local patterns of individual time series. TCN-MF extracts basis time series as global components and the predictions from TCN-MF are used as covariates for the local temporal convolution network. DeepGLO can capture the global features from a whole dataset as well as the local patterns of individual time series while both training and prediction.

DFMRE (Deep Factor Models with Random Effects) [36] is a global-local forecasting method based on a global DNN backbone and local probabilistic graphical models. They assume that each time series is determined by a global component (non-random) and a local random component. The global components are a linear combination of k latent global factors modeled by RNNs. Three models are proposed to model the local random components based on three typical local choices: white noise processes, linear dynamical systems (LDS) and Gaussian processes (GPs).

DeepAR [26] is a global-local forecasting method based on autoregressive recurrent neural networks. DeepAR learns a global RNN model from historical data of all variables, while the local components are obtained by probabilistic sampling based on prior distribution.

The aforementioned methods only focus on global patterns for forecasting and ignore the local patterns of the time series.

DeepGate [24] decomposes multivariate time series into the underlying global and local series and predicts the future value of each series based on these global and local series. They adopt a pre-training-based transfer training scheme to maintain global–local decomposition in a joint learning scenario. Unlike other global–local methods which use global factors as latent features, DeepGate explicitly model global and local features in an additive manner to support disentanglement.

3. Methodology

In this section, we first introduce some preliminaries and the formulation of multivariate time series forecasting in SubSection 3.1. We present our method, **A**daptive **G**lobal-**L**ocal **G**raph Structure Learning and Integration with **G**ated **R**ecurrent **U**nits (AGLG-GRU) for multivariate time series forecasting, in SubSection 3.2. Finally, SubSection 3.5 describes how our multivariate forecasting model is trained and regularized.

3.1. Preliminaries and problem formulation

Multivariate time series forecasting is to predict the future values of correlated multi-variables based on the historical data of these variables. The historical data of multivariate time series with auxiliary prior knowledge can be denoted as a tensor $\boldsymbol{X}_{T\times N\times F}$, where T is the length of temporal sequence, N represents the number of different variables (e.g., nodes in sensors network), F is the dimension of features of each node. Concatenating input signals with auxiliary features, we assume the inputs for model learning $\boldsymbol{X}_{T\times N\times F}=\left\{\boldsymbol{X}_{T\times N\times f_1},\boldsymbol{X}_{T\times N\times f_2},\cdots,\boldsymbol{X}_{T\times N\times f_i},\cdots\right\}$ and the first column $\boldsymbol{X}_{T\times N\times f_1}$ is the target value for forecasting. For example, we usually use traffic volume as target for traffic flow forecasting, traffic speed for highway speed forecasting, and daily electricity consumption observations for city electricity forecasting. The primary feature of input signals can be coupled with some auxiliary features $\boldsymbol{X}_{T\times N\times f_i}$, such as holidays information, historical weather observations, road network distance, point of interest (PoI) similarities [4].

For variables with geospatial properties, such as variables of a traffic network, multivariate time series can be denoted as a weighted graph $\mathcal{G}\left(\mathcal{V},\mathcal{E},\boldsymbol{A}^P\right)$. Here, $\mathcal{V}$ is the set of vertices that represent the different sensors (e.g., energy sensors or traffic segments) on networks; $N=|\mathcal{V}|$ denotes the number of nodes (variables) in a graph and $\mathcal{E}$ is a set of connected edges; $\boldsymbol{A}^P$ is a pre-defined adjacency matrix with $\boldsymbol{A}^P_{ij}=c>0$ if $\left(v_i,v_j\right)\in\mathcal{E}$ and $\boldsymbol{A}^P_{ij}=0$ if $\left(v_i,v_j\right)\notin\mathcal{E}$, which is a mathematical structure created by computing distances and up-down relationships between the nodes of a graph.

Multivariate time series forecasting can be formulated as follows: Given the historical T steps data $\boldsymbol{X}_{(t-T):(t-1)}$ of N variables (nodes) before the current time step t and the corresponding graph $\mathcal{G}(\mathcal{V}_N,\mathcal{E},\boldsymbol{A}^P)$, the objective is to learn a function $\boldsymbol{\mathcal{F}}$ that is able to predict the target value of the next S steps $\tilde{\boldsymbol{X}}_{t:(t+S)}$ as represented by Equation (1).

$$\tilde{\boldsymbol{X}}_{t:(t+S)}=\boldsymbol{\mathcal{F}}\left(\boldsymbol{X}_{(t-T):(t-1)},\mathcal{G}(\mathcal{V}_N,\mathcal{E},\boldsymbol{A}^P)\right) \tag{1}$$

where $\tilde{\boldsymbol{X}}_{t:(t+S)}=\left(\tilde{\boldsymbol{X}}_{t,N\times f_1},\tilde{\boldsymbol{X}}_{(t+1),N\times f_1},\cdots,\tilde{\boldsymbol{X}}_{(t+S),N\times f_1}\right)$ for multi-step forecasting ($S>1$).

Multivariate time series forecasting is challenging due to:

- *Complex spatial dependency and underlying global trends of change*. The variables in a multivariate time series are intrinsically dependent with each other, and have similar trends of change. Existing graph-based multivariate time series forecasting methods focus on modeling the spatial dependency but ignore the identification of the global trends of changes of all variables, while global-local forecasting methods do not consider the spatial dependency among variables. Thus, it is necessary to develop a method to capture both the spatial dependency and the global trends of change.
- *Dynamic non-linear temporal dependency and randomness of local variables*. The current value of a variable depends on the history of itself and many other variables, and variables show independent randomness. Existing graph-based methods focus on capturing temporal dependency but ignore the randomness of local variables. Global-local methods ignore the dynamic temporal dependency of variables. Thus, it is essential to consider both the temporal dependency and the randomness of local variables.

3.2. The architecture of AGLG-GRU

As shown in Fig. 1, our multivariate time series forecasting framework AGLG-GRU consists of four components: (1) a global graph structure learning module, (2) a local graph structure learning module, (3) a GRU module for hierarchical weighted feature integration and (4) a GRU-based encoder-predictor module. First, in the global graph structure learning module, a global probabilistic graphical model is employed to extract the global (shared) trends of variables at the whole observation scale. Second, in the local graph structure learning module, a local hyper-network is used to extract the local features of time series and capture the dynamic dependencies among variables at dynamic observation scale. Third, the GRU module for hierarchical weighted feature fusion adaptively merges the global-local feature embedded in the graphs. Finally, the fused feature vector is fed to a GRU-based

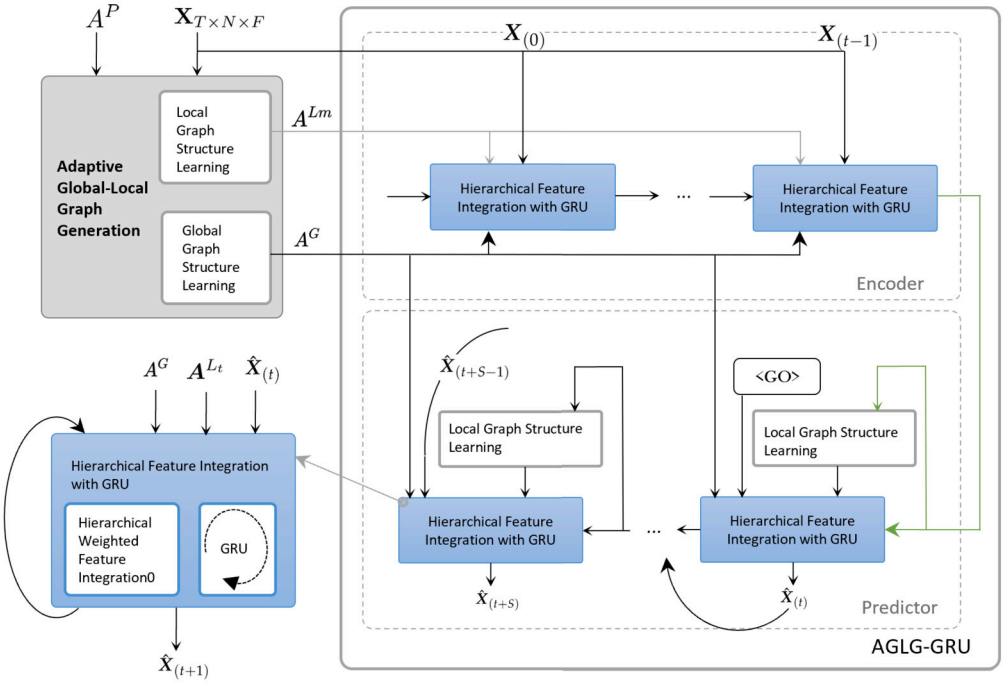


Fig. 1. The overall architecture of AGLG-GRU. The Encoder and Predictor Consist of a module of Hierarchical Feature Integration with GRU. An Adaptive Global-Local Graph Structure Learning Module adaptively generates global and local graphs for the encoder and predictor. The global graph learning module is used to capture the global trends and shared information of variables (nodes). The local graph learning module is used to learn the local randomness of variables and the correlation between variables.

encoder-predictor for single-step and multi-step ahead forecasting. $\langle GO \rangle$ indicates the first of the predictor inputs, and the remaining predictor inputs are derived from the previous outputs.

3.3. Adaptive global-local graph structure learning

In this section, we present our method for generating the global graph $\boldsymbol{A}^G$ and local graphs $\boldsymbol{A}^{L_i}$. The global graph is used to capture the shared global trends of all variables. The local graphs are generated dynamically to capture the local randomness and dynamic dependencies among variables. To improve the sensitivity to random changes in non-periodic time series, we fuse the learned global features with the learned local features in the subsequent stage.

3.3.1. Global graph structure learning

We aim to use a global graph to capture the global trends and shared information of variables. By measuring the global similarity of variables, we can capture the delayed dependencies between a pair of variables. As shown in Fig. 2, the probabilistic graph model for learning the global graph structure is composed of three components: a feature extractor, a link predictor, and a sampling component.

The feature extractor uses a shared convolution kernel to extract the shared features (e.g., periodic fluctuations, seasonality) of all variables from the whole time series and auxiliary data. In particular, we first feed time series $\boldsymbol{X}_{T\times N\times F}$ into two layers of convolutional neural network $\boldsymbol{Conv2d}$ and then use a fully-connected layer $\mathbb{FC}$ to reduce the dimension of node feature in the feature extractor. The generated node vector $\boldsymbol{v}_i$ represents the temporal characteristics of the ith variable. Two variables with similar temporal characteristics will have similar node vectors. The feature of i_{th} node v^i is generated as Equation (2).

$$\boldsymbol{v}_i = \mathbb{FC}\left(\boldsymbol{Conv2d}\left(\boldsymbol{X}_{0:T,i,0:F}\right)\right) \tag{2}$$

As shown in Equation (3), we concatenate a pair of node vectors $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ and implement a fully-connected layer to generate an edge scalar ϵ_{ij} in the link predictor.

$$\epsilon_{ij} = \mathbb{FC}\left(\boldsymbol{v}_i \| \boldsymbol{v}_j\right) \tag{3}$$

As part of the sampling procedure for global graph structure learning, we employ the Gumbel reparameterization trick [28] to determine the parameters of the global adjacency matrix $\boldsymbol{A}^G$ based on sampling variables from discrete distributions. Equation (4) shows the formula for generating $\boldsymbol{A}^G$.

$$\boldsymbol{A}^G_{ij} = \text{sigmoid}\left[\log\left(\epsilon_{ij}/\left(1-\epsilon_{ij}\right)\right) + \left(g^1_{ij} - g^2_{ij}\right)\right] \tag{4}$$

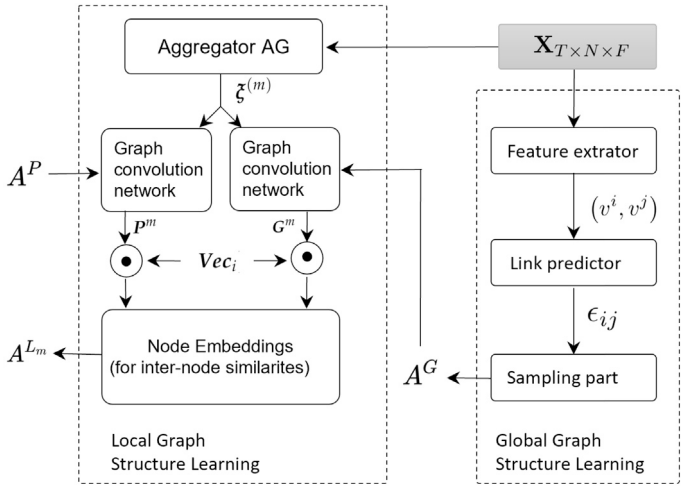


Fig. 2. The framework of adaptive global-local graph structure learning.

where $g_{ij}^1, g_{ij}^2 \sim \text{Gumbel}(0,1)$ for all i, j.

3.3.2. Local graph structure learning

We aim to use local graphs to capture the local randomness of variables and their dynamic correlations. The local randomness and dynamic correlations are more prominent when variables are observed at a smaller observation scale. Thus, we split the given historical data into smaller segments, and learn a series of dynamic local graphs $\left(\boldsymbol{A}^{L_0}, \boldsymbol{A}^{L_1}, \cdots, \boldsymbol{A}^{L_m}, \cdots, \boldsymbol{A}^{L_M}\right)$ from these segments. This process is denoted in the left part of Fig. 2.

Specifically, we first divide the given historical data $\boldsymbol{X}_{T\times N\times F}$ into independent segments along the time dimension, and use an aggregator to aggregate the features of each segment along the time dimension. We implement the aggregator by performing an averaging operation on each segment. This process will generate a sequence of new features $\left[\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(m)}, \ldots, \boldsymbol{\xi}^{(M)}\right]$ as follows:

$$\boldsymbol{\xi}^{(m)} = \mathbb{AVG}\left(\boldsymbol{X}^{((m-1)B+1:mB)}\right) \in R^{N\times F} \tag{5}$$

where M is the total number of segments, B is the size of segments, $\mathbb{AVG}$ indicates an averaging aggregator. According to Equation (1), our model requires T step historical data to predict future time series variables at step S. Therefore, the length of the input in the model is dimensionally consistent with the historical data provided in T steps.

We then use a feature segment $\boldsymbol{\xi}^{(m)}$ to perform graph convolution on the pre-refined graph $\boldsymbol{A}^P$ and the global graph $\boldsymbol{A}^G$ respectively, as follows.

$$\begin{aligned} \boldsymbol{P}^m &= \boldsymbol{Gcovn}\left(\boldsymbol{\xi}^{(m)}, \boldsymbol{A}^P\right) \\ \boldsymbol{G}^m &= \boldsymbol{Gcovn}\left(\boldsymbol{\xi}^{(m)}, \boldsymbol{A}^G\right) \end{aligned} \tag{6}$$

where $\boldsymbol{Gcovn}$ represents the graph convolution with learnable parameters. In case of missing, incomplete or incorrect $\boldsymbol{A}^P$ data in the dataset, we will use $\boldsymbol{A}^G$ instead of $\boldsymbol{A}^P$ or use different weights in the subsequent steps to adjust the importance of $\boldsymbol{A}^P$ data in the feature construction. In the adjacency matrices $\boldsymbol{A}^P$ and $\boldsymbol{A}^G$, there are static distance-based relations between nodes, as well as global features, that can be used in the message-passing process for handling dynamic node status information.

Based on the outputs in Equation (6), we compute a dynamic node embedding $\boldsymbol{E}_i^m$ for each variable. The dynamic node embeddings are computed based on the element-wise multiplication between $\boldsymbol{P}^m$, $\boldsymbol{G}^m$ and the static node embeddings $\boldsymbol{Vec}_i$ as follows.

$$\boldsymbol{E}_i^m = \tanh\left(\varphi_p\left(\boldsymbol{P}_i^m \odot \boldsymbol{Vec}_i\right) + \varphi_g\left(\boldsymbol{G}_i^m \odot \boldsymbol{Vec}_i\right)\right) \tag{7}$$

where φ_p and φ_g are hyper-parameters that control the weights of different components, $\odot$ denotes the element-wise multiplication. By adjusting weights in case of missing, incomplete, or incorrect data in the dataset, we are able to optimize prediction accuracy based on experimental results.

The adjacency matrix of the mth local graph $\boldsymbol{A}^{L_m}$ is derived by computing the pairwise similarity between variables. The elements of $\boldsymbol{A}^{L_m}$ are computed as follows:

$$\boldsymbol{A}_{ij}^{L_m} = \text{ReLU}\left(\tau \boldsymbol{E}_i^m \boldsymbol{E}_j^{m^T}\right) \tag{8}$$

where τ controls the saturation rate of the activation function.

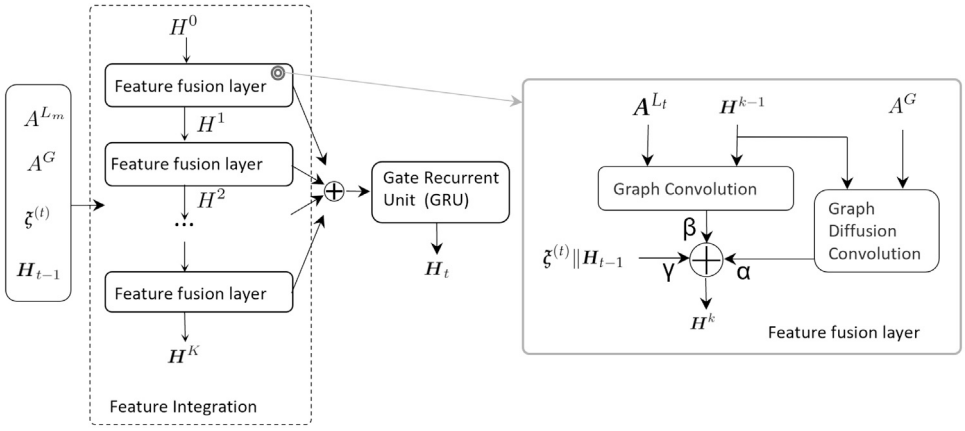


Fig. 3. Structure Of the hierarchical weighted feature fusion GRU with K feature fusion layers.

3.4. Hierarchical feature integration with GRUs

The adaptive global and local graphs capture different aspects of the features of multivariate time series. In this section, we describe how to integrate the information captured by the global graph and a local graph. The details of global-local graph integration using a GRU are shown in Fig. 3. It should be noted that the GRUs in the encoder and predictor (as shown in Fig. 1) are the same except the initialization of the hidden state.

3.4.1. Hierarchical weighted feature integration

As shown in Fig. 3, we recurrently perform K-hop feature fusion to integrate the information captured by the global and local graphs; then we update an GRU with the integrated information. The operations of each hop are illustrated in the right part of Fig. 3. The operations include: (1) a graph diffusion convolution on the global graph $\boldsymbol{A}^G$; (2) a graph convolution on the dynamic local graph $\boldsymbol{A}^{L_t}$; (3) a linear weighted addition operation.

For the global graph $\boldsymbol{A}^G$, we perform the diffusion convolution operation from DCRNN [34] defined using a bidirectional random graph walk. The diffusion convolution operation calculates bidirectional transition matrices that represent inflows and outflows during diffusion. As multi-step diffusion produces homogeneous nodes, complicating parametric training and confusing node features, we only use one-step diffusion convolution. In the k-th feature fusiong layer, our one-step graph diffusion convolution on the global graph is defined in Equation (9).

$$\boldsymbol{H}^{(k-1)} \star \boldsymbol{A}^G = \left[\omega_0 + \omega_1 \left(\boldsymbol{D}_O^{-1}\boldsymbol{A}^G\right) + \omega_2 \left(\boldsymbol{D}_I^{-1}\left(\boldsymbol{A}^G\right)^T\right)\right] \boldsymbol{H}^{(k-1)} \tag{9}$$

where $\boldsymbol{H}^{(k-1)}$ denotes the output of the previous feature fusion layer; $\star$ denotes our one-step graph diffusion convolution; $\boldsymbol{D}_O$ and $\boldsymbol{D}_I$ are the out-degree and in-degree matrices of A^G respectively, $\omega_0, \omega_1, \omega_2$ are model hyperparameters.

Based on the graph diffusion convolution defined in Equation (9), the information propagation step is defined in Equation (10), where the t-th feature segment $\boldsymbol{\xi}^{(t)}$ and hidden state of the previous time step $\boldsymbol{H}_{t-1}$ are concatenated as the input for the K-hop feature fusion.

$$\begin{aligned} \boldsymbol{H}^{(0)} &= \boldsymbol{\xi}^{(t)} \| \boldsymbol{H}_{t-1} \\ \boldsymbol{H}^{(k)} &= \alpha\left(\boldsymbol{H}^{(k-1)} \star \boldsymbol{A}^G\right) + \beta\left(\boldsymbol{A}^{L_t}\boldsymbol{H}^{(k-1)}\right) + \gamma\left(\boldsymbol{\xi}^{(t)} \| \boldsymbol{H}_{t-1}\right) \end{aligned} \tag{10}$$

where $*$ denotes a tensor-tensor multiplication, $\star$ denotes the graph diffusion convolution, α, β and γ are hyper-parameters that control the weights of different components.

We learn different integrated feature representations for updating different gates of GRU. The information selection step Θ_q for the gate q of GRU is defined in Equation (11).

$$\Theta_q = \sum_{j=0}^{K} \boldsymbol{H}^{(k)} \omega_q^j \tag{11}$$

where q denotes that the integrated feature representation is used to update the q gate of GRU, ω_q^j are learnable parameters, K is the depth of information propagation. Finally, the outputs of our weighted feature fusion unit are fed into a gated recurrent unit.

3.4.2. Updating GRUs for the GRU-based encoder-predictor model

The GRUs updating process is defined in the following form, utilizing the hierarchically integrated feature representation.

$$\boldsymbol{H}_t = \boldsymbol{U}_t \odot \boldsymbol{H}_{t-1} + \left(1 - \boldsymbol{U}_t\right) \odot \boldsymbol{C}_t \tag{12}$$

Table 1
Statistics of the seven datasets used for experiments.

Datasets	Sample	Sample Rate [34]	Nodes	Input length	Output length
METR-LA	34272	5 minutes	207	12	12
PEMS04	16992	5 minutes	307	12	12
PEMS08	17856	5 minutes	170	12	12
TRAFFIC	17,544	1 hour	862	168	1
SOLAR-ENERGY	52,560	10 minutes	137	168	1
ELECTRICITY	26,304	1 hour	321	168	1
EXCHANGE-RATE	7588	1 day	8	168	1

$$\boldsymbol{C}_t = \tanh\left(\Theta_c\left[\boldsymbol{\xi}^{(t)} \| \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right)\right] + b_c\right) \tag{13}$$

and the update gate $\boldsymbol{U}_t$ and reset gate $\boldsymbol{R}_t$ can be respectively given by

$$\boldsymbol{R}_t = \sigma\left(\Theta_r\left[\boldsymbol{\xi}^{(t)} \| \boldsymbol{H}_{t-1}\right] + b_r\right) \tag{14}$$

$$\boldsymbol{U}_t = \sigma\left(\Theta_u\left[\boldsymbol{\xi}^{(t)} \| \boldsymbol{H}_{t-1}\right] + b_u\right) \tag{15}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function, $\boldsymbol{R}_t, \boldsymbol{U}_t$ are reset gate and update gate at the current time step t, $\Theta_r, \Theta_u, \Theta_c$ are different K-hop feature fusion layers with learnable parameters for the corresponding graph convolution modules.

As shown in Fig. 1, when our model makes multi-step predictions, it is an sequence-to-sequence model with an encoder and a decoder (predictor). We use the final hidden state of the encoder to initialize the decoder. The decoder will be trained with the teacher forcing mode, i.e., the previous ground truth observations are provided for training. During testing time, the predictions by the model will be used for predicting values of the next time step. To alleviate the performance degradation caused by the discrepancy of distribution between the training and testing mode, we use scheduled sampling [3] during the training. More details about our model training are provided in the next section.

3.5. Training strategy

In our research, we consider the time series forecasting problem as a regression problem. We use the Mean Absolute Error (MAE) between the predicting result $\hat{\boldsymbol{X}}_{t:(t+S-1)}$ and the ground truth data $\boldsymbol{X}_{t:(t+S-1)} = \left(\boldsymbol{X}_{t,N\times f_1}, \boldsymbol{X}_{(t+1),N\times f_1}, \cdots, \boldsymbol{X}_{(t+S-1),N\times f_1}\right) \in R^{S\times N\times f_1}$ as the base training loss. The base training loss $\boldsymbol{L}^t_{\text{base}}$ is defined as Equation (16).

$$\boldsymbol{L}^t_{\text{base}}\left(\hat{\boldsymbol{X}}_{t:(t+S-1)}, \boldsymbol{X}_{t:(t+S-1)}\right) = \frac{1}{S}\sum_{t'=t}^{t+S-1}\left|\hat{\boldsymbol{X}}_{t',N\times f_1} - \boldsymbol{X}_{t',N\times f_1}\right| \tag{16}$$

We also propose a cross-entropy loss $\boldsymbol{L}_{CE}$ between the edge scalar ϵ_{ij} of the learned global graph (Equation (3)) and the pre-defined adjacency matrix values A^P_{ij}. By introducing the pre-defined graph in the loss function, we increase the robustness of our model to noises of the training data. Thus, we can use $\boldsymbol{L}_{CE}$ as a regularization term to alleviate over-fitting. $\boldsymbol{L}_{CE}$ is defined in Equation (17):

$$\boldsymbol{L}_{CE} = \sum_{ij}\left[\boldsymbol{A}^P_{ij}\log\epsilon_{ij} - \left(1-\boldsymbol{A}^P_{ij}\right)\log\left(1-\epsilon_{ij}\right)\right] \tag{17}$$

For our method, the overall training loss is defined as $\boldsymbol{L} = \sum_t \boldsymbol{L}^t_{\text{base}} + \delta\boldsymbol{L}_{CE}$, where $\delta > 0$ is the regularization magnitude. Different weights $\delta > 0$ are applied depending on whether $\boldsymbol{A}^P$ data are missing, incomplete, or incorrect in the dataset. We demonstrate experimentally that prior knowledge $\boldsymbol{A}^P$ can help substantially improve the effectiveness and quality of the learned global graph by incorporating $\boldsymbol{L}_{CE}$ as a regularization term. During training and testing, forecasting accuracy can be significantly improved by learning the appropriate graph.

4. Experiments

We conduct extensive experiments to evaluate our proposed AGLG-GRU model on several real-world datasets for multivariate time series forecasting. The extensive experiments include both single-step and multi-step forecasting. We compare our model with several competitive baselines and conduct ablation studies to assess the effectiveness of key components of our model.

4.1. Datasets

We use seven datasets to evaluate our method. These seven datasets include data from traffic, energy consumption and finance, and they have been widely used by research in multivariate time series forecasting. Thus, we can conveniently compare our method with other methods. The statistics of the seven datasets are listed in Table 1. More details about the datasets are as follows:

- **METR-LA** [18] This dataset on public transportation is derived from sensors installed at intersections and loops in Los Angeles, California [34]. There are 207 sensors in this dataset created based on pairwise distances from a demarcation point. During March 1 through June 30, 2012, this sensor network recorded traffic speeds in miles per hour (mph) at five-minute intervals. The METR-LA dataset also presents adjacency data in a graphical format through a 207×207 matrix representation.
- **PEMS04** [8] This data collection represents actual public transit speeds recorded by the performance measurement system of the California Transportation Agency (CalTrans). PEMS04 includes traffic data from 307 detectors in the Bay Area during January and February 2018.
- **PEMS08** [8] This is a collection of public transit speed data collected by the performance measurement system of the California Transportation Agency. PEMS08 contains 170 detectors, and the time range selected is July to August 2016. The time-frequency and velocity units are identical to the METR-LA data set.
- **TRAFFIC** [21] This traffic dataset is provided by the California Department of Transportation, 862 sensors collected measurements during 2015 and 2016 and reported their findings. These figures represent the highway occupancy rates in the San Francisco Bay region.
- **SOLAR-ENERGY** [21] This is a dataset on solar energy generated by the National Renewable Energy Laboratory based on information received from 137 solar panels in Alabama State in 2007.
- **ELECTRICITY** [21] This is a dataset on electricity based on the information provided by the UCI Machine Learning Repository. It contains information regarding the energy use of 321 clients between 2012 and 2014.
- **EXCHANGE-RATE** [21] This dataset consists of the daily exchange rates for eight nations from 1990 to 2016. The list has eight countries: Australia, Great Britain, Canada, Switzerland, China, Japan, New Zealand, and Singapore.

4.2. Experimental settings

The time granularity of the three data sets that use speed as an informative characteristic is five minutes. For multi-step forecasting, the length of the input sequence is twelve, whereas the size of the target sequence is twelve. For single-step forecasting, the sequence length of the input data is 168 and the sequence length of the output data is one. We adjust the velocity data by zero-mean normalization, and any missing values are filled in by near interpolation. We take the road sensors to be vertex points and use their midpoints to calculate the location of the road segment. The pre-defined adjacency matrix $\boldsymbol{A}^P$ is generated by applying a Gaussian threshold kernel to the Euclidean distance between the two road segments [30].

Hyper-parameter values are optimized via grid search. All databases have a batch size of 64, while AGLG-GRU has a depth of 1 for its gated recurrent unit. Following the convolution layer, the learning rate is set to 0.1, 0.01, and 0.001, while the dropout rate is set to 0.1, 0.2, and 0.3. We adopt a variable number of hidden units from [16,32,64,96,128] since the dimension of the gated recurrent unit hidden state can significantly impact the accuracy of its forecasting. Given the preliminary results, the hidden unit dimension has been set to 64 for METR-LA and PEMS04 and 32 for PEMS08. For the purpose of preventing overfitting, the Adam optimizer uses the early stop mechanism to train the model. The specific values of the hyper-parameters, such as α, β, and hidden cell dimension, are adjusted based on the experimental results with different data sets. We compare the experimental results of some parameters at different values in the subsection 4.7.

Following [19], we divide the four sets of single-step predictive datasets into three sets in chronological order: the training set (60%) the validation set (20%), and the test set (20%) and we train independent models to predict the target future step (horizon) 3, 6, 12, and 24. We divide these multi-step forecasting datasets into three sets in chronological order - training data (70%), validation data (20%), and test data (10%) and our multi-steps forecasting are based on a random optimizer 15 minutes, 30 minutes, 45 minutes, and an hour in advance.

4.3. Evaluation metrics

We use four evaluation metrics to evaluate the performance of single-step and multi-step forecasting, namely Relative Squared Error (RSE) [21], Empirical Correlation Coefficient (CORR) [21], Root Mean Squared Error (RMSE) [8], and Mean Absolute Error (MAE) [18]. For RMSE, MAE, and RSE, lower values are better. For CORR, higher values are better. The formal definition of the evaluation metrics is introduced as Equations (18).

$$\begin{aligned}
&\mathrm{RSE}(x,\hat{x}) = \frac{\sqrt{\sum_{(i,t)\in\Omega_{\mathrm{Test}}}\left(x_{it}-\hat{x}_{it}\right)^2}}{\sqrt{\sum_{(i,t)\in\Omega_{\mathrm{Test}}}\left(x_{it}-\mathrm{mean}(x_i)\right)^2}},\\
&\mathrm{CORR}(x,\hat{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_t\left(x_{it}-\mathrm{mean}(x_i)\right)\left(\hat{x}_{it}-\mathrm{mean}(\hat{x}_i)\right)}{\sqrt{\sum_t\left(x_{it}-\mathrm{mean}(x_i)\right)^2\left(\hat{x}_{it}-\mathrm{mean}(\hat{x}_i)\right)^2}},\\
&\mathrm{RMSE}(x,\hat{x}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{x}_i-x_i\right)^2},\\
&\mathrm{MAE}(x,\hat{x}) = \frac{1}{|n|}\sum_{i=1}^{n}\left|\hat{x}_i-x_i\right|
\end{aligned} \tag{18}$$

where x_i denotes the ground truth and the $\hat{x}_i$ represents the predicted time series.

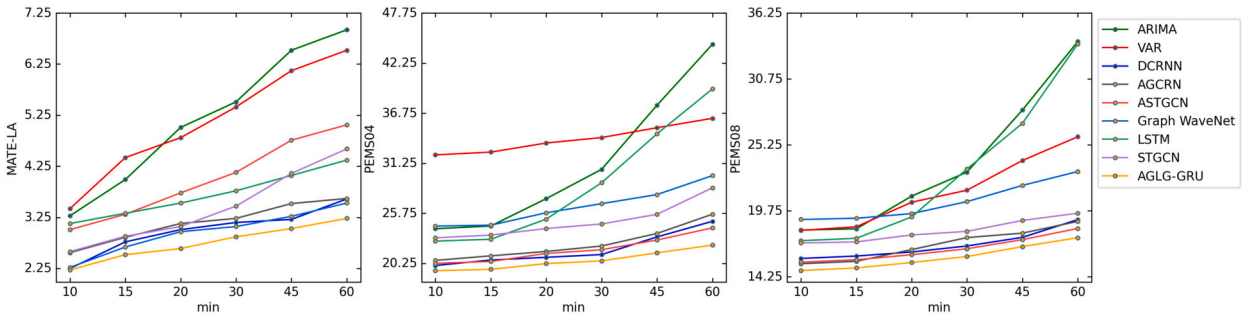


Fig. 4. The multi-step forecasting accuracy of different methods.

4.4. Baselines

We compare our proposed model, AGLG-GRU, with models that reflect recent improvements in multivariate time series forecasting. Baseline methods are summarized as follows:

4.4.1. Single-step forecasting baselines

- AR [5]: A prediction model using auto-regression.
- VAR-MLP [50]: A hybrid model of vector auto-regressive model and multilayer perceptron.
- RNN-GRU [44]: A prediction model based on recurrent neural network with fully connected gated recurrent unit.
- TPA-LSTM [29]: A forecasting method based on recurrent neural network with attention mechanism.
- MTGNN [38]: A graph neural network framework for multivariate time series forecasting. It consists of a graph learning module, a graph convolution module and a temporal convolution module.

4.4.2. Multi-step forecasting baselines

- ARIMA [45]: A widely used time series analysis technique based on the combination of autoregression and moving average.
- VAR [16]: A method based on vector auto-regression.
- FC-LSTM [48]: A method based on recurrent neural networks with fully connected Long Short-term Memory hidden unit.
- Graph WaveNet [37]: A method consists of an adaptive dependency graph learning module and a stacked dilated 1D convolution module.
- STGCN [42]: A method uses spatial-temporal graph convolution network and incorporates graph convolutions with 1D convolutions.
- DCRNN [22]: A method based on diffusion convolution recurrent neural network. It incorporates diffusion graph convolution with recurrent neural network in an encoder-decoder architecture.
- AGCRN [2]: A GNN-based and RNN-based model, which employs adaptive graph learning and integrates gated recurrent unit with graph convolutions.
- ASTGCN [13]: A model uses a devised spatial-temporal attention mechanism to capture the spatial-temporal correlations through calculating the attention matrix.

4.5. Main results

This section presents experimental results of single-step and multi-step forecasting. Table 2 and 3 provide the experimental results for AGLG-GRU, demonstrating that AGLG-GRU achieves state-of-the-art results on the majority of the tasks.

4.5.1. Single-step forecasting performance

In this experiment, we compare AGLG-GRU with other time series prediction models. Table 2 shows the experimental results for the single-step forecasting task. The AGLG-GRU routinely achieves the best results across nearly all data horizons pertaining to solar energy, traffic, and electricity. Particularly for traffic data, AGLG-GRU has demonstrated considerable RSE improvement. As a result of the more natural nature of traffic data, AGLG-GRU improves the results of our model, which aligns better with our assumptions regarding spatio-temporal dependence. The future occupancy rate of a road is inherently affected by its past and its related highways. Due to the reduced graph size and the lack of available exchange-rate training examples, AGLG-GRU does not enhance the exchange-rate data.

4.5.2. Multi-step forecasting performance

Table 3 lists the performances of our model and baselines for predicting 15 minutes, 30 minutes, 45 minutes, and one hour on the METR-LA, PEMS04, and PEMS08 datasets. The results can be summarized as follows:

1) The underlying properties of highly complex traffic flow data are easier to capture using deep learning algorithms than with typical time series methods (such as ARIMA and VAR).

Table 2
Comparative results of single-step forecasting.

Data	Method	3 horizon		6 horizon		12 horizon		24 horizon	
		RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR
Solar-Energy	AR	0.2435	0.971	0.379	0.9263	0.5911	0.8107	0.8699	0.5314
	VAR-MLP	0.1922	0.9829	0.2679	0.9655	0.4244	0.9508	0.6841	0.7149
	RNN-GRU	0.1932	0.9823	0.2628	0.9675	0.4163	0.915	0.4852	0.8823
	TPA-LSTM	0.1803	0.985	0.2347	0.9742	0.3234	0.9487	0.4389	0.9081
	MTGNN	0.1778	0.9852	0.2348	0.9726	0.3109	0.9509	0.427	0.9031
AGLG-GRU		**0.1762**	**0.9842**	**0.2302**	**0.9682**	**0.3021**	**0.9532**	**0.413**	**0.9084**
Traffic	AR	0.5991	0.7752	0.6218	0.7568	0.6252	0.7544	0.63	0.7519
	VAR-MLP	0.5582	0.8245	0.6579	0.7695	0.6023	0.7929	0.6146	0.7891
	RNN-GRU	0.5358	0.8511	0.5522	0.8405	0.5562	0.8345	0.5633	0.83
	TPA-LSTM	0.4487	0.8812	0.4658	0.8717	0.4641	0.8717	0.4765	0.8625
	MTGNN	0.4162	0.8963	0.4754	0.8667	0.4461	0.8794	0.4535	0.8810
AGLG-GRU		**0.4173**	**0.8958**	**0.4722**	**0.8541**	**0.4427**	**0.8755**	**0.4526**	**0.8842**
Electricity	AR	0.0995	0.8845	0.1035	0.8632	0.105	0.8591	0.1054	0.8595
	VAR-MLP	0.1393	0.8708	0.1620	0.8389	0.1557	0.8192	0.1274	0.8679
	RNN-GRU	0.1102	0.8597	0.1144	0.8623	0.1183	0.8472	0.1295	0.8651
	TPA-LSTM	0.0823	0.9439	0.0916	0.9337	0.0964	0.925	0.1006	0.9133
	MTGNN	0.0745	0.9474	0.0878	0.9316	0.0916	0.9278	0.0953	0.9234
AGLG-GRU		**0.0738**	**0.9434**	**0.0864**	**0.9302**	**0.0912**	**0.9283**	**0.0947**	**0.9274**
Exchange-Rate	AR	0.0228	0.9734	0.0279	0.9656	0.0353	0.9526	0.0445	0.9357
	VAR-MLP	0.0265	0.8609	0.0394	0.8725	0.0407	0.8280	0.0578	0.7675
	RNN-GRU	0.0192	0.9786	0.0264	0.9712	0.0408	0.9531	0.0626	0.9223
	TPA-LSTM	0.0174	0.979	0.0241	0.9709	0.0341	0.9564	0.0444	0.9381
	MTGNN	0.0194	0.9786	0.0259	0.9708	0.0349	0.9551	0.0456	0.9372
AGLG-GRU		**0.0191**	**0.9792**	**0.0233**	**0.9701**	**0.0328**	**0.9548**	**0.0449**	**0.9372**

Table 3
Comparison with baselines on multi-step traffic flow forecasting.

Data	Method	15 min		30 min		45 min		60 min	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
METR-LA	ARIMA	3.9	8.21	5.15	10.45	5.82	11.83	6.92	13.23
	VAR	4.42	7.89	5.41	9.13	6.13	9.76	6.52	10.11
	LSTM	3.44	6.30	3.77	7.23	4.08	7.97	4.37	8.69
	Graph WaveNet	2.67	5.51	3.07	6.22	3.32	6.77	3.53	7.37
	STGCN	2.88	5.74	3.47	7.24	4.12	8.76	4.59	9.43
	DCRNN	2.77	5.38	3.15	6.45	3.43	7.14	3.61	7.62
	AGCRN	2.86	6.27	3.23	6.58	3.34	7.05	3.62	7.51
	ASTGCN	3.31	7.62	4.13	9.77	4.54	10.08	5.06	11.66
AGLG-GRU		**2.74**	**4.87**	**3.1265**	**5.710**	**3.36**	**6.27**	**3.706**	**6.784**
PEMS04	ARIMA	24.36	43.47	30.58	61.30	37.62	66.43	44.31	71.02
	VAR	32.47	49.84	34.07	52.06	35.15	53.61	36.18	55.05
	LSTM	22.90	37.10	29.11	44.98	34.38	51.48	39.41	60.84
	Graph WaveNet	24.45	35.45	26.81	37.29	27.81	39.82	29.89	42.63
	STGCN	23.36	36.04	24.58	38.01	25.64	40.15	28.55	42.53
	DCRNN	20.62	32.11	21.22	34.33	23.16	36.11	24.87	37.12
	AGCRN	21.08	33.23	22.16	34.89	23.54	35.73	25.65	36.79
	ASTGCN	20.46	31.83	21.75	33.76	22.82	35.34	24.15	36.93
AGLG-GRU		**19.60**	**29.85**	**20.53**	**30.13**	**21.41**	**31.32**	**22.26**	**32.54**
PEMS08	ARIMA	18.25	29.58	22.98	39.61	28.16	50.03	33.87	60.15
	VAR	18.42	26.96	21.47	31.37	23.95	34.83	25.93	37.41
	LSTM	17.46	28.14	23.24	37.05	27.05	43.37	33.67	50.31
	Graph WaveNet	19.13	31.05	20.52	33.94	21.87	34.19	23.03	34.23
	STGCN	17.16	26.04	18.04	27.48	18.95	28.86	19.55	29.93
	DCRNN	15.98	23.67	16.82	26.36	17.54	26.49	19.02	28.67
	AGCRN	15.54	23.34	17.52	27.09	17.89	26.77	18.87	27.74
	ASTGCN	15.67	23.76	16.59	25.29	17.35	26.43	18.28	28.05
AGLG-GRU		**14.99**	**22.24**	**15.94**	**23.81**	**16.77**	**25.13**	**17.51**	**26.28**

Table 4
Summary of models for ablation studies.

Model Name	Description of the model
AGLG-GRU	The full model of **A**daptive **G**lobal-**L**ocal **G**raph structure learning with **G**ated **R**ecurrent **U**nits (AGLG-GRU)
AGLG-GRU-ndi	The graph diffusion convolution operation is replaced by graph convolution.
AGLG-GRU-ndg	The dynamic local graph structure learning module is removed. The local adjacency matrices are replaced by an identity matrix.
AGLG-GRU-nsg	The static global graph structure learning module is removed. The global adjacency matrix is replaced by an identity matrix

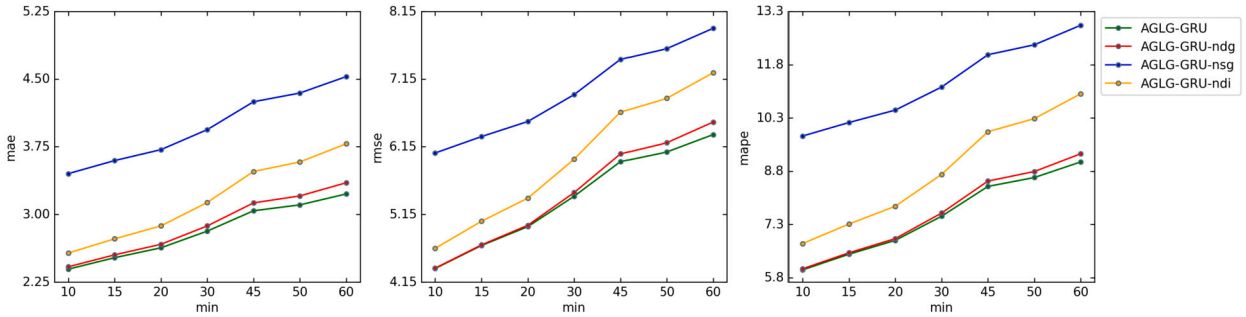


Fig. 5. Ablation study on METR-LA.

2) Graph-based methods allow the application of deep learning models to non-Euclidean geometric structures by considering both the correlation and topology of graph structures. Graph-based models (such as Graph WaveNet, STGCN, AGCRN, ASTGCN, and our model) outperform classic deep learning approaches (Long Short-term Memory and DCRNN). Consequently, the experimental results indicate that non-euclidean data should be taken into account when estimating traffic flow.

3) Our AGLG-GRU achieves the best results for the four forecasting intervals. According to Fig. 4, the forecasting accuracy varies with the length of the interval for each model. As only the temporal correlation of the time series is modeled, the VAR and ARIMA models are advantageous for short-term time series forecasting. With the forecasting interval increases, the performances deteriorate dramatically. With the addition of spatial correlation to DCRNN and Long Short-term Memory, the forecasting accuracy gradually decreases as time intervals increase. The non-Euclidean geometry of the road network plays a significant role in the accuracy of some road network models, such as Graph WaveNet, STGCN, AGCRN, ASTGCN, and our own. Experiments demonstrate that non-Euclidean data is essential for predicting traffic flows, and the AGLG-GRU model outperforms all other models on all three datasets.

By combining the global spatial characteristics of the network with the dynamic temporal characteristics of the data, our AGLG-GRU model achieves a higher forecasting accuracy than other baseline techniques during the experiment. When the forecasting interval is longer than 30 minutes, the advantages of our model are even more evident. The model produces better results for three main reasons.

- The global structure learning module uses a more efficient mapping of static spatial-temporal features of the time series as input to the forecasting model by learning the probabilistic properties of the graph.
- The local structure learning modules modify the weights of the graph feature vectors across batches of data, capturing the dynamic relevance of the data.
- Hierarchically using gated recurrent unit improved by bivariate diffusion of graph random walks, the model captures global and local spatial features of multivariate time series in a targeted manner.

A longer-term forecasting with more precision will be advantageous for practical applications since this will provide road managers and travelers sufficient time to plan and prepare for travel. Through dynamically capturing the hierarchical spatial-temporal features of traffic flow data, we have enhanced the performance of long-term forecasting of traffic networks with complex traffic topologies.

4.6. Ablation study

We use the METR-LA data to conduct ablation research in order to evaluate the effectiveness of key components of our proposed model. As shown in Table 4, we test the full model of AGLG-GRU with three other models that have certain component removed or replaced. Following [38], we repeat each experiment ten times with 50 epochs per repetition and calculate the MAE, RMSE, and MAPE on the validation set. The average values of the three metrics are given in Fig. 5.

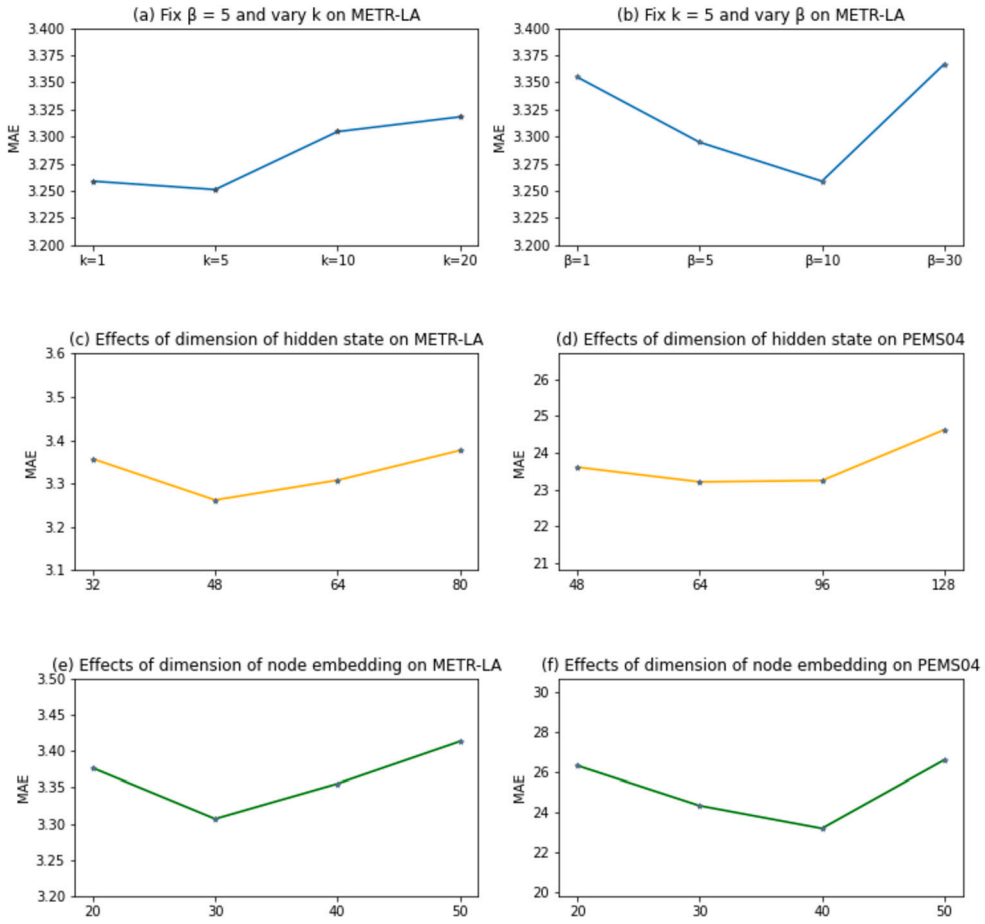


Fig. 6. Parameter Study on METR-LA and PEMS04.

The introduction of a global graph structure learning module significantly improves the results as it enables the improvement of input quality by learning the probabilistic properties from space features. The effect of the diffusion convolution mechanism is evident as well: it confirms that bidirectional graph random walks are effective in capturing global information in diffusion convolution layers. Lastly, our local graph structure learning modules prove to be effective in terms of improving long-term traffic flow forecasting (horizon 6 and 12). It enables our model to hierarchically capture dynamic parameters step by step as the level of learning difficulty increases.

4.7. Hyper-parameter study

This section investigates the influence of hyper-parameters using a parameter analysis of the AGLG-GRU model. Each experiment is conducted ten times, with the MAE on the test set acting as a comparison indicator. During each experiment, only one parameter is examined, while all the other variables are held constant. Fig. 6 displays the experimental results of hyper-parameter study on the datasets METR-LA and PEMS04 for the 30-minute traffic flow forecasting. The hyper-parameters β and K range are set between 1 and 30. The dynamic local graph convolution module can be simplified by setting β to 0, where the dynamic local graph convolution is eliminated. Fig. 6-(a) and 6-(b) depicts the experimental findings for the MAE indicator. It is evident from the curves that decreasing K or raising β increases forecasting accuracy.

Node embedding dimensions range from 20 to 50, and hidden state dimensions range from 32 to 128. Fig. 6-(c) and 6-(d) demonstrates that by increasing the dimension of the embedded node in the dynamic graph and the hidden state of the gated recurrent unit, the representational capability of the AGLG-GRU will be strengthened, while reducing the MAE loss. We find from the experimental results that the forecasting accuracy initially increases as the hidden unit dimension increases, but declines with time in METR-LA. According to the results shown in Fig. 6-(e), it appears that the AGLG-GRU model is insensitive to the concealed state of the large size nodes in PEMS04. Due to the significantly more complex traffic conditions of the PEMS04 dataset, embeddings of nodes must have a greater dimension to hold sufficient information. Fig. 6-(f) illustrates the excellent performance of the AGLG-GRU on the PEMS04 dataset with 40 embedded nodes. The increase in the hidden cell dimension or the number of hidden cells makes the model more complicated and hence captures a wider range of data attributes. However, overfitting arises when these variables

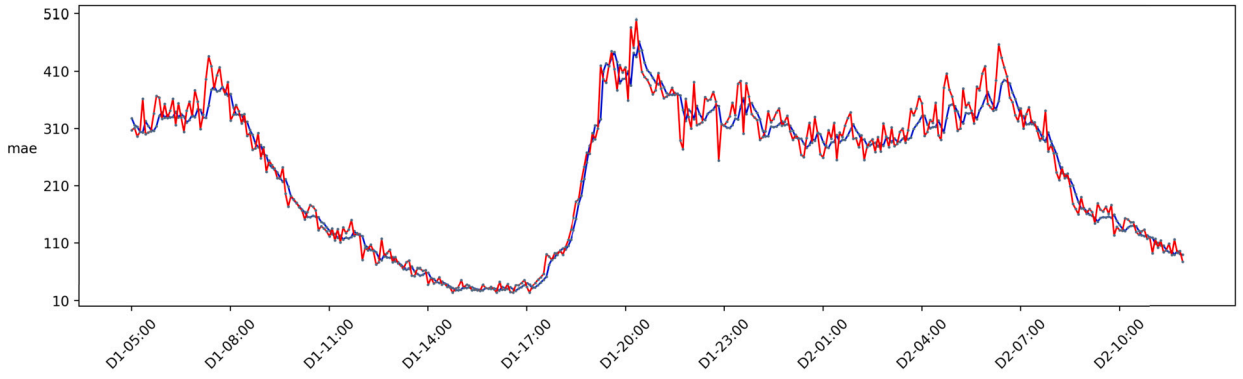


Fig. 7. Data Visualization on PEMS04 for node 45.

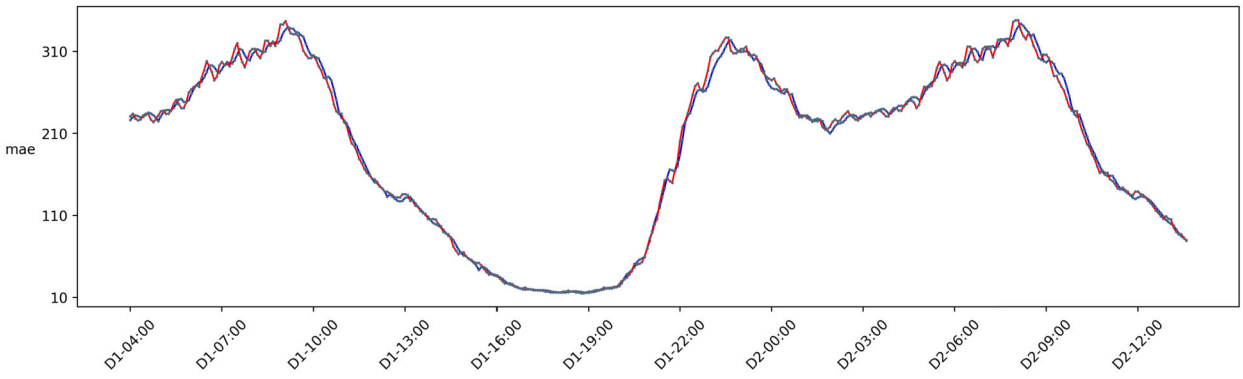


Fig. 8. Data Visualization on PEMS04 for node 125.

take on more than a certain value, and the ideal values for the hidden state dimension and node embedding for the METR-LA dataset are around 64 and 30 respectively. Therefore, it is vital for this study to determine the correct hyper-parameter values based on individual data sets, as the selection of each hyper-parameter value will have a significant impact on the final performance of the model.

4.8. Model interpretation

For a deeper understanding of the model, we compare the predicted speed values with the ground-truth data of some nodes in the PEMS04 traffic network. As depicted in the illustration of comparison of data of node 45 in Fig. 7, our AGLG-GRU model can predict traffic peaks and trends with a high degree of accuracy and works well when simulating rapid changes in traffic speeds. Fig. 8 illustrates the comparison between predicted data and ground-truth data for node 125. This demonstrates that by utilizing our AGLG-GRU model, we can obtain accurate and smooth forecasting at nodes with moderate fluctuations in traffic speed.

We design a hierarchy experiment to demonstrate the effectiveness of the model in acquiring dynamic features by training several batches of test data and examining the weight matrix hierarchically. Three separate sets of data spanning different observation scales were fed into the trained model from the METR-LA test set. Fig. 9 depicts the heat map of the dynamic adjacency matrix obtained through the encoder and predictor structures at the scale (0,1). The weights of the i^{th} rows and j^{th} columns in the heatmap represent the influence of $node_i$ on $node_j$. As shown in Fig. 9, some rows and columns have significantly higher weights than others, indicating that the nodes corresponding to the high-weight rows and columns have more influence than others during that period of time.

To test the effectiveness of our proposed dynamic local graphs, we compare the differences in the attributes of the same node in the dynamic adjacency matrix at different observation scales. In our experiment, we select the node 75 in METR-LA dataset for case study. Some subsets of METR-LA data collected from 8:00 am to 10:00 am and 4:00 pm to 6:00 pm were fed into a trained dynamic local graph learning module to generate dynamic local graphs.

Fig. 10 distinguishes the ten most influential nodes for node 75 at two different observation scales, where the red dots represent node 75, the green dots represent the ten most influential nodes between 8:00 and 10:00 am, and the yellow dots represent the ten most influential nodes between 4:00 and 6:00 pm. Since the relationships between neighboring nodes in our model are dynamically acquired at different observation scales, our model allows for the dynamic discovery of spatial-temporal dependencies in the time series and goes beyond the limitations of a fixed adjacency matrix to perform more efficiently.

We calculate the importance of METR-LA data nodes at different observation scales and plot the top ten nodes in terms of importance on Fig. 11. The following Equation (19) is used to calculate node importance:

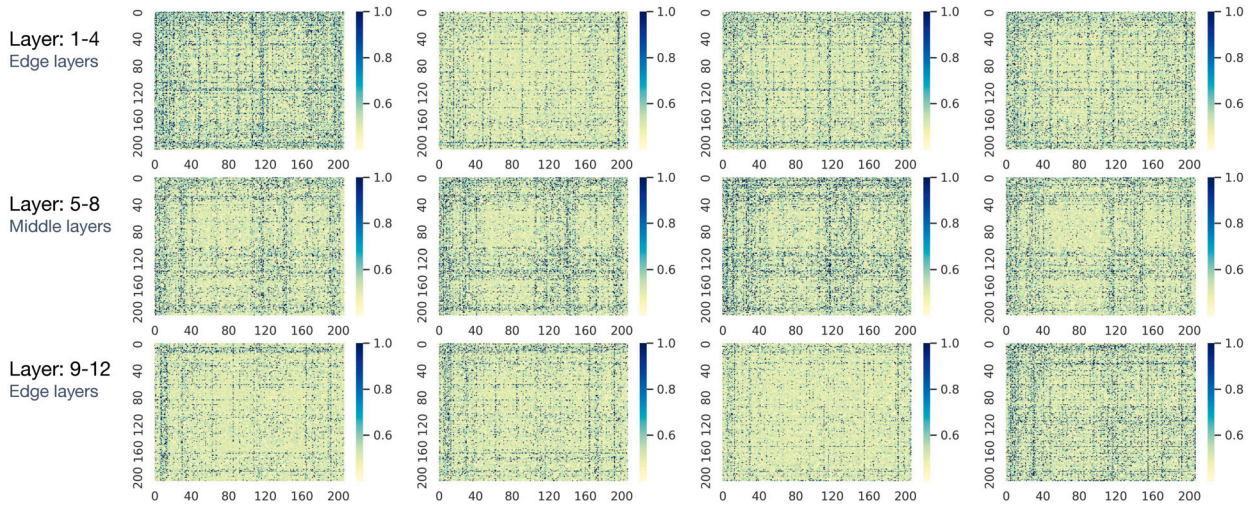


Fig. 9. Heatmaps of the dynamic adjacency matrix for each encoder layer.

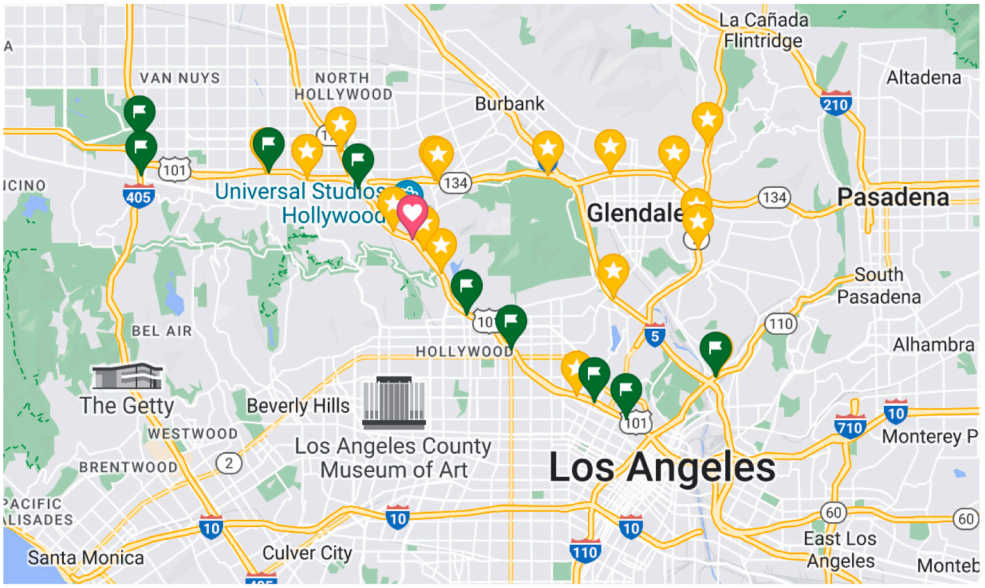


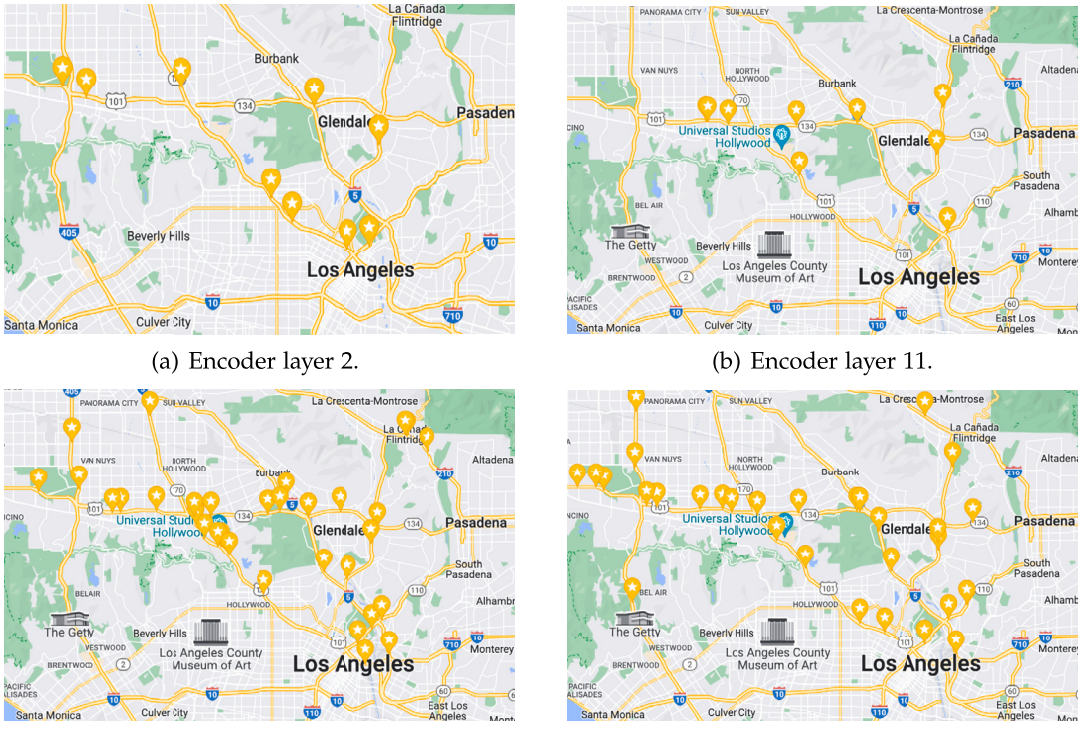
Fig. 10. Top 10 nodes affecting node 75 on METR-LA at various times.

$$\boldsymbol{IM}_q = \sum \boldsymbol{A}_{iq}^G + \sum \boldsymbol{A}_{qj}^G \tag{19}$$

where $\boldsymbol{IM}_q$ represents the importance of the node q, and $\boldsymbol{A}^G$ represents the dynamic adjacency matrix. A sampling of data from the METR-LA dataset was taken between 8:00 AM and 10:00 AM. Fig. 11 displays the distribution of the more influential nodes over the several network layers of the encoder. Most of the prominent nodes are located near junctions, which is a characteristic feature of traffic networks.

4.9. Model computational complexity analysis

Table 5 presents a comparison of computational complexity, measured in terms of FLOP (Floating Point Operations) counts. It is evident that our method exhibits higher FLOP counts compared to other methods. This is primarily attributed to the graph structure learning process. The inclusion of graph structure learning introduces a considerable number of FLOPs since it entails learning the correlation features of the data's graph structure. It should be noted that although our method may introduce additional complexity, they possess the potential to offer superior accuracy and capture intricate dependencies, global trends and local randomness within time series data.



(a) Encoder layer 2.

(b) Encoder layer 11.

(c) Encoder layer 7.

(d) Encoder layer 8.

Fig. 11. The top 10 nodes as evaluated at different encoder levels in terms of influence.

Table 5
A comparison of computational complexity on METR-LA.

Method	Flops
DCRNN	1.0e07
AGCRN	1.0e07
LSTM	8.1e08
ASTGCN	2.3e09
AGLG-GRU	**4.5e09**

5. Conclusion

In this paper, we propose AGLG-GRU, a novel multivariate time series forecasting method that combines the advantages of graph-based methods and global-local methods. Our AGLG-GRU model consists of four modules: a global graph structure learning module, a local graph structure learning module, a GRU module for hierarchical weighted feature integration, and a GRU-based encoder-predictor module for multi-step forecasting. The learned global graph captures the global trends and shared information of variables. The learned local graphs capture the local randomness of variables and their dynamic correlations. The GRU module fuses the information captured by the global graph and local graphs to update the gates for encoding input features and forecasting multi-step future values. Experiments demonstrate that our approach improves multivariate time series forecasting accuracy, particularly for long-term forecasting.

In the future work, we will investigate to combine our method with deep stochastic configuration networks (SCN) [35,7]. Robust SCN [40] has shown advantages in processing data with noises and outliers, and federated SCN [6] is capable of analyzing distributed large-scale data. We will explore using robust SCN to learn adaptive graphs from data with noises and outliers. We expect to be able to process distributed multivariate time series data by using federated SCN. We will also try the latest methods of graph stream summarization [17], preference modeling in next location prediction [23], and graph structure learning-based knowledge tracing [32].

CRediT authorship contribution statement

Ting Guo: Conceptualization, Methodology, Software, Writing – original draft. **Feng Hou:** Conceptualization, Writing – original draft. **Yan Pang:** Funding acquisition, Investigation. **Xiaoyun Jia:** Writing – review & editing. **Zhongwei Wang:** Project administration, Validation. **Ruili Wang:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the datasets are publicly available

Acknowledgement

This study was supported by the Key Research and Development Project of Hunan Province (2022GK2025); Hunan Key Laboratory of Intelligent Logistics Technology, China (No. 2019TP1015).

References

[1] N.A. Asif, Y. Sarker, R.K. Chakrabortty, M.J. Ryan, M.H. Ahamed, D.K. Saha, F.R. Badal, S.K. Das, M.F. Ali, S.I. Moyeen, et al., Graph neural network: a comprehensive review on non-Euclidean space, IEEE Access 9 (2021) 60588–60606.
[2] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, Adv. Neural Inf. Process. Syst. 33 (2020) 17804–17815.
[3] S. Bengio, O. Vinyals, N. Jaitly, N. Shazeer, Scheduled sampling for sequence prediction with recurrent neural networks, Adv. Neural Inf. Process. Syst. 28 (2015) 1–10.
[4] D. Cao, K. Zeng, J. Wang, P.K. Sharma, X. Ma, Y. Liu, S. Zhou, Bert-based deep spatial-temporal network for taxi demand prediction, IEEE Trans. Intell. Transp. Syst. 23 (7) (2021) 9442–9454.
[5] J. Che, J. Wang, Short-term electricity prices forecasting based on support vector regression and auto-regressive integrated moving average modeling, Energy Convers. Manag. 51 (10) (2010) 1911–1917.
[6] W. Dai, L. Ji, D. Wang, Federated stochastic configuration networks for distributed data analytics, Inf. Sci. 614 (2022) 51–70.
[7] M.J. Felicetti, D. Wang, Deep stochastic configuration networks with optimised model and hyper-parameters, Inf. Sci. 600 (2022) 431–441.
[8] S. Feng, J. Huang, Q. Shen, Q. Shi, Z. Shi, A hybrid model integrating local and global spatial correlation for traffic prediction, IEEE Access 10 (2021) 2170–2181.
[9] L. Franceschi, M. Niepert, M. Pontil, X. He, Learning discrete structures for graph neural networks, in: International Conference on Machine Learning, 2019, pp. 1972–1982.
[10] R. Fu, Z. Zhang, L. Li, Using LSTM and GRU neural network methods for traffic flow prediction, in: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE, 2016, pp. 324–328.
[11] X. Geng, X. He, L. Xu, J. Yu, Graph correlated attention recurrent neural network for multivariate time series forecasting, Inf. Sci. 606 (2022) 126–142.
[12] K. Guo, Y. Hu, Z. Qian, Y. Sun, J. Gao, B. Yin, Dynamic graph convolution network for traffic forecasting based on latent network of Laplace matrix estimation, IEEE Trans. Intell. Transp. Syst. 23 (2) (2020) 1009–1018.
[13] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 922–929.
[14] H. Hewamalage, C. Bergmeir, K. Bandara, Recurrent neural networks for time series forecasting: current status and future directions, Int. J. Forecast. 37 (1) (2021) 388–427.
[15] J. Hong, J. Park, S. Park, StackDA: a stacked dual attention neural network for multivariate time-series forecasting, IEEE Access 9 (2021) 145955–145967.
[16] Y. Huang, Y. Weng, S. Yu, X. Chen, Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting, in: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2019, pp. 678–685.
[17] Y. Jia, Z. Gu, Z. Jiang, C. Gao, J. Yang, Persistent Graph Stream Summarization for Real-Time Graph Analytics, World Wide Web, 2023, pp. 1–21.
[18] X. Kong, W. Xing, X. Wei, P. Bao, J. Zhang, W. Lu, STGAT: spatial-temporal graph attention networks for traffic flow forecasting, IEEE Access 8 (2020) 134363–134372.
[19] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long- and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.
[20] F. Li, J. Feng, H. Yan, G. Jin, F. Yang, F. Sun, D. Jin, Y. Li, Dynamic graph convolutional recurrent network for traffic prediction: benchmark and solution, ACM Trans. Knowl. Discov. Data 17 (1) (2023) 1–21.
[21] S. Li, H. Huang, W. Lu, A neural networks based method for multivariate time-series forecasting, IEEE Access 9 (2021) 63915–63924.
[22] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, in: International Conference on Learning Representations, 2018, pp. 1–16.
[23] J. Liu, Y. Chen, X. Huang, J. Li, G. Min, GNN-based long and short term preference modeling for next-location prediction, Inf. Sci. 629 (2023) 1–14.
[24] J. Park, C. Park, J. Choi, S. Park, DeepGate: global-local decomposition for multivariate time series modeling, Inf. Sci. 590 (2022) 158–178.
[25] N. Rathore, P. Rathore, A. Basak, S.H. Nistala, V. Runkana, Multi scale graph wavenet for wind speed forecasting, in: 2021 IEEE International Conference on Big Data (Big Data), IEEE, 2021, pp. 4047–4053.
[26] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.
[27] R. Sen, H.-F. Yu, I.S. Dhillon, Think globally, act locally: a deep neural network approach to high-dimensional time series forecasting, Adv. Neural Inf. Process. Syst. 32 (2019) 1–10.
[28] C. Shang, J. Chen, Discrete graph structure learning for forecasting multiple time series, in: Proceedings of International Conference on Learning Representations, 2021, pp. 1–11.

[29] S.-Y. Shih, F.-K. Sun, H.-y. Lee, Temporal pattern attention for multivariate time series forecasting, Mach. Learn. 108 (8) (2019) 1421–1441.
[30] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. 30 (3) (2013) 83–98.
[31] S. Singh, R. Wang, F. Hou, Improved meta learning for low resource speech recognition, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 4798–4802.
[32] X. Song, J. Li, T. Cai, S. Yang, T. Yang, C. Liu, A survey on deep learning based knowledge tracing, Knowl.-Based Syst. 258 (2022) 110036.
[33] A. Sriramulu, N. Fourrier, C. Bergmeir, Adaptive dependency learning graph neural networks, Inf. Sci. (2023) 700–714.
[34] K. Tamil Selvi, R. Thamilselvan, S. Mohana Saranya, Diffusion Convolution Recurrent Neural Network–a Comprehensive Survey, Materials Science and Engineering Conference Series, 2021, p. 012119.
[35] D. Wang, M. Li, Stochastic configuration networks: fundamentals and algorithms, IEEE Trans. Cybern. 47 (10) (2017) 3466–3479.
[36] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, T. Januschowski, Deep factors for forecasting, in: International Conference on Machine Learning, 2019, pp. 6607–6617.
[37] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph waveNet for deep spatial-temporal graph modeling, in: The 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 1907–1913.
[38] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: multivariate time series forecasting with graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 753–763.
[39] G. Xiao, R. Wang, C. Zhang, A. Ni, Demand prediction for a public bike sharing program based on spatio-temporal graph convolutional networks, Multimed. Tools Appl. 80 (15) (2021) 22907–22925.
[40] A. Yan, J. Guo, D. Wang, Robust stochastic configuration networks for industrial data modelling with student's t mixture distribution, Inf. Sci. 607 (2022) 493–505.
[41] H. Yao, X. Tang, H. Wei, G. Zheng, Z. Li, Revisiting spatial-temporal similarity: a deep learning framework for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 5668–5675.
[42] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640.
[43] H.-F. Yu, N. Rao, I.S. Dhillon, Temporal regularized matrix factorization for high-dimensional time series prediction, Adv. Neural Inf. Process. Syst. 29 (2016) 1–10.
[44] Z. Zainuddin, P.A. E.A., M. Hasan, Predicting machine failure using recurrent neural network-gated recurrent unit (RNN-GRU) through time series data, Bull. Electr. Eng. Inform. 10 (2) (2021) 870–878.
[45] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.
[46] Q. Zhang, J. Chang, G. Meng, S. Xiang, C. Pan, Spatio-temporal graph structure learning for traffic forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 1177–1185.
[47] Z. Zhang, M. Li, X. Lin, Y. Wang, F. He, Multistep speed prediction on traffic networks: a deep learning approach considering spatio-temporal dependencies, Transp. Res., Part C, Emerg. Technol. 105 (2019) 297–322.
[48] J. Zhao, F. Deng, Y. Cai, J. Chen, Long short-term memory-fully connected (LSTM-FC) neural network for PM2.5 concentration prediction, Chemosphere 220 (2019) 486–492.
[49] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T-GCN: a temporal graph convolutional network for traffic prediction, IEEE Trans. Intell. Transp. Syst. 21 (9) (2019) 3848–3858.
[50] S. Zhao, S. Lin, Y. Li, J. Xu, Y. Wang, Urban traffic flow forecasting based on memory time-series network, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2020, pp. 1–6.