Multi-Scale Adaptive Graph Neural Network for Multivariate Time Series Forecasting

Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang

***Abstract*—Multivariate time series (MTS) forecasting plays an important role in the automation and optimization of intelligent applications. It is a challenging task, as we need to consider both complex intra-variable dependencies and inter-variable dependencies. Existing works only learn temporal patterns with the help of single inter-variable dependencies. However, there are multi-scale temporal patterns in many real-world MTS. Single inter-variable dependencies make the model prefer to learn one type of prominent and shared temporal patterns. In this article, we propose a multi-scale adaptive graph neural network (MAGNN) to address the above issue. MAGNN exploits a multi-scale pyramid network to preserve the underlying temporal dependencies at different time scales. Since the inter-variable dependencies may be different under distinct time scales, an adaptive graph learning module is designed to infer the scale-specific inter-variable dependencies without pre-defined priors. Given the multi-scale feature representations and scale-specific inter-variable dependencies, a multi-scale temporal graph neural network is introduced to jointly model intra-variable dependencies and inter-variable dependencies. After that, we develop a scale-wise fusion module to effectively promote the collaboration across different time scales, and automatically capture the importance of contributed temporal patterns. Experiments on six real-world datasets demonstrate that MAGNN outperforms the state-of-the-art methods across various settings.**

***Index Terms*—Multivariate time series forecasting, multi-scale modeling, graph neural network, graph learning.**

I. Introduction

MULTIVARIATE time series (MTS) are ubiquitous in various real-world scenarios, e.g., the traffic flows in a city, the stock prices in a stock market, and the household power consumption in a city block [1]. MTS forecasting, which aims at forecasting the future trends based on a group of historical observed time series, has been widely studied in recent years. It is of great importance in a wide range of applications, e.g., a better driving route can be planned in advance based on the forecasted traffic flows, and an investment strategy can be designed with the forecasting of the near-future stock market [2], [3], [4], [5].

Making accurate MTS forecasting is a challenging task, as both intra-variable dependencies (i.e., the temporal dependencies within one time series) and inter-variable dependencies (i.e., the forecasting values of a single variable are affected by other variables) need to be considered jointly. To solve this problem, traditional methods [6], [7], [8], e.g., vector auto-regression (VAR), temporal regularized matrix factorization (TRMF), vector auto-regression moving average (VARMA), and gaussian process (GP), often rely on the strict stationary assumption and cannot capture the non-linear dependencies among variables. Deep neural networks have shown superiority on modeling non-stationary and non-linear dependencies. Particularly, two variants of recurrent neural network (RNNs) [9], namely the long-short term memory (LSTM) and the gated recurrent unit (GRU), and temporal convolutional networks (TCNs) [10] have significantly achieved impressive performance in time series modeling. To capture both long-term and short-term temporal dependencies, existing works [3], [11], [12], [13], [14] introduce several strategies, e.g., skip-connection, attention mechanism, and memory-based network. These works focus on modeling temporal dependencies, and process the MTS input as vectors and assume that the forecasting values of a single variable are affected by all other variables, which is unreasonable and hard to meet in realistic applications. For example, the traffic flows of a street are largely affected by its neighboring streets, while the impact from distant streets is relatively small. Thus, it is crucial to model the pairwise inter-variable dependencies explicitly.

Graph is an abstract data type representing relations between nodes. Graph neural networks (GNNs) [15], [16], which can effectively capture nodes' high-level representations while exploiting pairwise dependencies, have been considered as a promising way to handle graph data. MTS forecasting can be considered from the perspective of graph modeling. The variables in MTS can be regarded as the nodes in a graph, while the pairwise inter-variable dependencies as edges. Recently, several works [17], [18], [19] exploit GNNs to model MTS taking advantage of the rich structural information (i.e., featured nodes and weighted edges) of a graph. These works stack GNN and temporal convolution modules to learn temporal patterns, and have achieved promising results. Nevertheless, there are still two important aspects neglected in above works.

First, existing works only consider temporal dependencies on a single time scale, which may not properly reflect the variations in many real-world scenarios. In fact, the temporal patterns

Manuscript received 14 January 2022; revised 27 February 2023; accepted 8 April 2023. Date of publication 19 April 2023; date of current version 15 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0505000 and in part by Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies. Recommended for acceptance by X. Zhu. *(Ling Chen and Donghui Chen are co-first authors.) (Corresponding author: Ling Chen.)*

Ling Chen, Donghui Chen, Zongjiang Shang, and Binqing Wu are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: lingchen@cs.zju.edu.cn; chendonghui@zju.edu.cn; zongjiangshang@zju.edu.cn; binqingwu@cs.zju.edu.cn).

Cen Zheng, Bo Wen, and Wei Zhang are with the Alibaba Group, Hangzhou 311100, China (e-mail: mingyan.zc@alibaba-inc.com; wenbo.wb@alibaba-inc.com; zwei@alibaba-inc.com).

Digital Object Identifier 10.1109/TKDE.2023.3268199

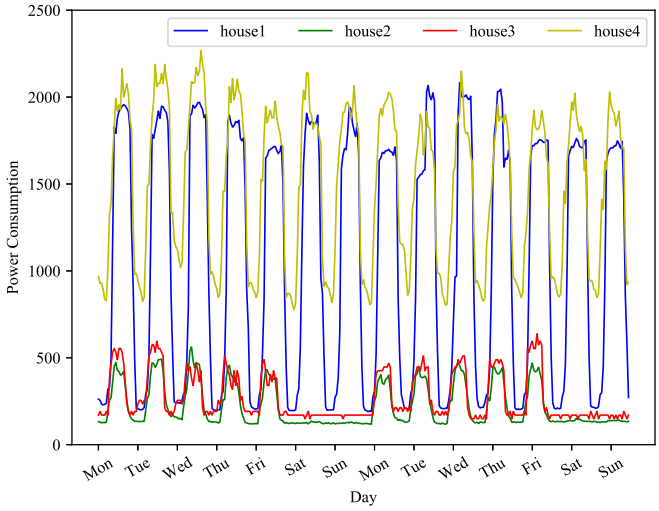


Fig. 1. The power consumptions of 4 households within two weeks (from Monday 00:00 to Sunday 24:00). Households 1 and 4 have both daily and weekly repeating patterns, while households 2 and 3 have weekly repeating patterns.

hidden in real-world MTS are much more complicated, including daily, weekly, monthly, and other specific periodic patterns. For example, Fig. 1 shows the power consumptions of 4 households within two weeks. There exists a mixture of short-term and long-term repeating patterns (i.e., daily and weekly). These multi-scale temporal patterns provide abundant information to model MTS. Furthermore, if the temporal patterns are learned from different time scales separately, and are then straightforwardly concatenated to obtain the final representation, the model is failed to capture cross-scale relationships and cannot focus on contributed temporal patterns. Thus, an accurate MTS forecasting model should learn a feature representation that can comprehensively reflect all kinds of multi-scale temporal patterns.

Second, existing works learn a shared adjacent matrix to represent the rich inter-variable dependencies, which makes the models be biased to learn one type of prominent and shared temporal patterns. In fact, different kinds of temporal patterns are often affected by different inter-variable dependencies, and we should distinguish the inter-variable dependencies when modeling distinct temporal patterns. For example, when modeling the short-term patterns of the power consumptions of a household, it might be essential to pay more attention to the power consumptions of its neighbors. Because the dynamics of short-term patterns are often affected by a common event, e.g., a transmission line fault decreases the power consumptions of a street block, and a sudden cold weather increases the power consumptions. When modeling the long-term patterns of the power consumptions of a household, it might be essential to pay more attention to the households that have similar living habits, e.g., working and sleeping hours, as these households would have similar daily and weekly temporal patterns. Therefore, the complicated inter-variable dependencies need to be fully considered when modeling these multi-scale temporal patterns.

In this paper, we propose a general framework termed Multi-scale Adaptive Graph Neural Network (MAGNN) for MTS forecasting to address above issues. Specifically, we introduce a multi-scale pyramid network to decompose the time series with different time scales in a hierarchical way. Then, an adaptive graph learning module is designed to automatically infer the scale-specific graph structures in the end-to-end framework, which can fully explore the abundant and implicit inter-variable dependencies under different time scales. After that, a multi-scale temporal graph neural network is incorporated into the framework to model intra-variable dependencies and inter-variable dependencies at each time scale. Finally, a scale-wise fusion module is designed to automatically consider the importance of scale-specific representations and capture the cross-scale correlations. In summary, our contributions are as follows:

- Propose MAGNN, which learns a temporal representation that can comprehensively reflect both multi-scale temporal patterns and the scale-specific inter-variable dependencies.
- Design an adaptive graph learning module to explore the abundant and implicit inter-variable dependencies under different time scales, and a scale-wise fusion module to promote the collaboration across these scale-specific temporal representations and automatically capture the importance of contributed temporal patterns.
- Conduct extensive experiments on six real-world MTS benchmark datasets. The experiment results demonstrate that the performance of our method is better than that of the state-of-the-art methods.

The remainder of this paper is organized as follows: Sections II and III give a survey of related work and preliminaries. Section IV describes the proposed MAGNN method. Section V presents the experimental results and Section VI concludes the paper.

II. Related Work

We briefly review the related work from two aspects: the MTS forecasting and graph learning for MTS.

A. MTS Forecasting

The problem of time series forecasting has been studied for decades. One of the most prominent traditional methods used for time series forecasting is the auto-regressive integrated moving average (ARIMA) model, because of its statistical properties and the flexibility on integrating several linear models, including auto-regression (AR), moving average, and auto-regressive moving average. However, limited by the high computational complexity, ARIMA is infeasible to model MTS. Vector auto-regression (VAR) and vector auto-regression moving average (VARMA) are the extension of AR and ARIMA, respectively, that can model MTS. Gaussian process (GP) [6] is a Bayesian method to model distributions over a continuous domain of functions. GP can be used as a prior over the function space in Bayesian inference and has been applied to MTS forecasting. However, these works often rely on the strict stationary assumption and cannot capture the non-linear dependencies among variables.

Recently, deep learning-based methods have shown superior capability on capturing non-stationary and non-linear dependencies. Most of existing works rely on LSTM and GRU to capture temporal dependencies [20]. Compared with RNN-based approaches, dilated 1D convolutions [18], [21] are able to handle long-range sequences. However, the dilation rates of dilated 1D convolutions may cause the loss of local information, which brings in negative effects on modeling short-term dependencies. Some other efforts exploit TCNs and self-attention mechanism [22] to model long time series efficiently. To capture both long-term and short-term temporal dependencies, LSTNet [3] introduces the convolutional neural network to capture short-term temporal dependencies, and a recurrent-skip layer that can exploit the long-term periodic property hidden in time series. TPA-LSTM [13] utilizes an attention mechanism, which enables the model to extract important temporal patterns and focus on different time steps for different variables. MTNet [12] exploits the memory component and attention mechanism to effectively capture long-term temporal dependencies and periodic patterns. However, these works assume that each variable affects all other variables equally, which is unreasonable and hard to meet in realistic applications.

B. Graph Learning for MTS

Graph neural networks (GNNs) [15], [16], which can model the interaction between nodes through weighted edges, have received increasing attention. Recently, there are many works using GNNs to capture inter-variable dependencies in the area of MTS modeling. One of the challenges of the GNNs-based MTS forecasting is to obtain a well-defined graph structure as the inter-variable dependencies. To solve this problem, existing methods can be roughly divided into three major categories: prior-knowledge-based, rule-based, and learning-based methods.

Prior-knowledge-based methods [23], [24], [25], [26] often exploit the extra information (e.g., road networks, physical structures, and extra feature matrices) in their specific scenarios. For example, in traffic flow forecasting [23], [25], the graph structure can be constructed by the connections of road networks. If there is a connected road between two nodes, an edge is constructed in the graph structure, as the traffic flow at the upstream node will affect the traffic flow at the downstream node. In skeleton-based action recognition [26], the graph structure can be constructed by the physical structure of the human body, e.g., the multiple joints on the same arm are linked by the human skeleton, and edges can be constructed between these joints. In the ride-hailing demand forecasting [24], multiple different graph structures are constructed from different views: the proximity of spatial distance, the connection of urban road network, and the similarity of region functionality. However, these methods require domain knowledge to design a graph structure, which is difficult to transfer between different scenarios.

Rule-based methods [19], [27], [28], [29], as non-parametric methods, provide a data-driven manner to construct the graph structure. These methods usually include causal discovery (e.g., Granger causality and additive noise model) [27], [29], entropy-based methods (e.g., transfer entropy and relative entropy) [19], similarity-based methods (e.g., Pearson correlation, mutual information, DTW distance, and edit distance) [28]. For example, Huang et al. [29] used Granger causality to construct a causal graph. Xu et al. [19] calculated the pairwise transfer entropy between variables, which is regarded as the adjacency matrix of the graph structure. He et al. [28] exploited dynamic time warping (DTW) algorithm, which is competent to capture the pattern similarities between two time series. However, these methods are non-parameterized methods and have limited flexibility, which can only learn a kind of specific inter-variable dependency.

Learning-based methods [17], [18], [30], [31], [32], [33], [34], [35] introduce a parameterized module to learn pairwise inter-variable dependencies automatically. Kipf et al. [32] first introduced a neural relational inference model, which uses the original time series as input and exploits the variational inference to learn a graph structure. Subsequently, Webb et al. [34] proposed a decomposition-based neural relational inference model to learn multiple types of graph structure. Graber et al. [31] proposed a neural relational inference model that achieves different graph structures at each time step. The attention-based learning methods use the attention mechanism to learn the pairwise inter-variable dependencies. For traffic flow forecasting, Tang et al. [33] used a graph attention module to learn graph structure. Zheng et al. [35] used spatial attention mechanism to learn the correlation of traffic flow at different nodes. In addition, several works achieve this more directly, i.e., randomly intializing the representation of each node, and calculating the pairwise similarity of these nodes. The representations of the nodes can be optimized to obtain the most suitable value for the current data distribution. For example, Wu et al. [18] exploited a graph learning module to learn inter-variable dependencies, and modelled MTS using the GNNs and dilated convolution networks. Bai et al. [17] introduced a data adaptive graph generation module to infer the inter-variable dependencies and a node adaptive parameter learning module to capture node-specific features. However, existing works only learn single inter-variable dependencies, making the models biased to learn one type of prominent and shared temporal patterns among MTS.

III. Preliminaries

A. Problem Formulation

Problem Statement: In this paper, we focus on MTS forecasting. Formally, given a sequence of observed time series signals $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t \in \mathbb{R}^{N\times 1}$ denotes the values at time step t, N is the variable dimension, and $\boldsymbol{x}_{t,i}$ denotes the value of the ith variable at time step t, MTS forecasting aims at forecasting the future values $\widehat{\boldsymbol{x}}_{T+h} \in \mathbb{R}^{N\times 1}$ at time step $T+h$, where h denotes the look-ahead horizon. The problem can be formulated as:

$$\widehat{\boldsymbol{x}}_{T+h} = \mathcal{F}\left(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T; \theta\right), \tag{1}$$

where $\mathcal{F}$ is the mapping function and θ denotes all learnable parameters.

Then, we give several definitions regarding MTS forecasting.

Definition 1. MTS to Graph: A graph is defined as $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ denotes the node set and $|\boldsymbol{V}| = N$. $\boldsymbol{E}$ is the edge set. Given the MTS $\boldsymbol{X} \in \mathbb{R}^{N\times T}$, the ith variable is regarded

as the ith node $v_i \in \boldsymbol{V}$, the values of $\{\boldsymbol{x}_{1,i}, \boldsymbol{x}_{2,i}, \ldots, \boldsymbol{x}_{T,i}\}$ are the features of v_i, and each edge $(v_i, v_j) \in \boldsymbol{E}$ indicates there is an inter-variable dependency between v_i and v_j.

Definition 2. Weighted Adjacency Matrix: The weighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ of a graph is a type of mathematical representation to store the weights of the edges, where $\boldsymbol{A}_{i,j} > 0$, if $(v_i, v_j) \in \boldsymbol{E}$, and $\boldsymbol{A}_{i,j} = 0$, if $(v_i, v_j) \notin \boldsymbol{E}$.

To pure MTS data without any prior knowledge, the weighted adjacency matrices of multiple graphs need to be learned to represent the abundant and implicit inter-variable dependencies. Accordingly, the formulation of MTS forecasting can be modified as:

$$\widehat{\boldsymbol{x}}_{T+h} = \mathcal{F}\left(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T; \boldsymbol{G}; \theta\right), \tag{2}$$

where $\boldsymbol{G} = \{G^1, G^2, \ldots, G^K\}$ represents the set of graphs that can be utilized by GNNs for MTS forecasting.

B. Graph Neural Networks

Graph neural networks (GNNs) [15], [16] are a type of deep neural network applied to graphs. Graphs can be irregular, a graph may have a variable size of unordered nodes, and nodes from a graph may have different numbers of neighbors. GNNs can be easy to compute in the graph domain, which can overcome the limitation of CNNs.

GNNs can be divided into two categories based on the implementation philosophy: spectral-based and spatial-based methods [15]. Spectral-based methods define the graph convolution by introducing a filter from the perspective of graph signal processing. The graph convolution operation can be interpreted as removing noise from the graph signal. Space-based methods define the graph convolution through information propagation, which aggregates the representation of a central node and the representations of its neighbors to get the updated representation for the node.

We briefly describe the graph convolution operation applied in our method, which can be defined as:

$$\boldsymbol{x} *_G \theta = \sigma\left(\theta\left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\right)\boldsymbol{x}\right), \tag{3}$$

where $G = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A})$ is a graph with a weighted adjacency matrix, $\boldsymbol{x}$ is the representations of nodes, σ is an activation function, θ is the learnable parameter matrix, $\widetilde{\boldsymbol{A}} = \boldsymbol{I}_n + \boldsymbol{A}$ is the adjacency matrix with self-connection, $\widetilde{\boldsymbol{D}}$ is the diagonal degree matrix of $\widetilde{\boldsymbol{A}}$, and $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$. By stacking the graph convolution operation multiple layers, we can aggregate the information of multi-order neighbors.

Multi-scale GNNs [36], [37], [38], named hierarchical GNNs alternatively, usually construct coarse-grained graphs based on the fine-grained graph hierarchically. MAGNN is concerned about scales in the temporal dimension, which is very different from general multi-scale GNNs that focus on scales in the spatial dimension. MAGNN introduces a multi-scale pyramid network to transform raw time series into feature representations from smaller scale to larger scale, on which it learns scale-specific graphs with the same size for each scale and utilizes basic GNNs as defined in (3) for each graph.

IV. Methodology

A. Framework

Fig. 2 illustrates the framework of MAGNN, which consists of four main parts: a) a multi-scale pyramid network to preserve the underlying temporal hierarchy at different time scales; b) an adaptive graph learning module to automatically infer inter-variable dependencies; c) a multi-scale temporal graph neural network to capture all kinds of scale-specific temporal patterns; d) a scale-wise fusion module to effectively promote the collaboration across different time scales.

B. Multi-Scale Pyramid Network

A multi-scale pyramid network is designed to preserve the underlying temporal dependencies at different time scales. Following the pyramid structure, it applies multiple pyramid layers to hierarchically transform raw time series into feature representations from smaller scale to larger scale. Such multi-scale structure gives us the opportunity to observe raw time series in different time scales. Specifically, the smaller scale feature representations can retain more fine-grained details, while the larger scale feature representations can capture the slow-varying trends.

Multi-scale pyramid network generates multi-scale feature representations through pyramid layers. Each pyramid layer takes the outputs of a preceding pyramid layer as the input and generates the feature representations of a larger scale as the output. Specifically, given the input MTS $\boldsymbol{X} \in \mathbb{R}^{N \times T}$, the multi-scale pyramid network generates feature representations of K scales, and the kth scale feature representation is denoted as $\boldsymbol{X}^k \in \mathbb{R}^{N \times \frac{T}{2^{k-1}} \times c^k}$, where N is the variable dimension, $\frac{T}{2^{k-1}}$ is the sequence length in the kth scale, and c^k is the channel size of the kth scale.

A pyramid layer takes convolutional neural networks to capture local patterns in the time dimension. Following the design philosophy of image processing, different pyramid layers employ different kernel sizes. The beginning convolution kernel has larger filter, and the size is slowly decreased at each pyramid layer, which can control the receptive field size and maintain the sequence characteristics of large scale time series. For example, the kernel sizes can be set as 1×7, 1×6, and 1×3 at each pyramid layer, and the stride size of convolution is set to 2 to increase the time scale. Formally,

$$\boldsymbol{X}^k_{\text{rec}} = ReLU(\boldsymbol{W}^k_{\text{rec}} \otimes \boldsymbol{X}^{k-1} + \boldsymbol{b}^k_{\text{rec}}), \tag{4}$$

where $\otimes$ denotes convolution operator, $\boldsymbol{W}^k_*$ and $\boldsymbol{b}^k_*$ denote the convolution kernel and bias vector in the kth pyramid layer, respectively. However, different pyramid layers are expected to preserve the underlying temporal dependencies at different time scales. The flexibility of using only one convolutional neural network is limited, as the granularities of the temporal dependencies captured in the feature representations at two consecutive pyramid layers are highly sensitive to the hyperparameter settings (i.e., kernel size and stride size). To alleviate this issue, following the existing works in image processing [39], [40], we introduce another convolutional neural network with kernel size

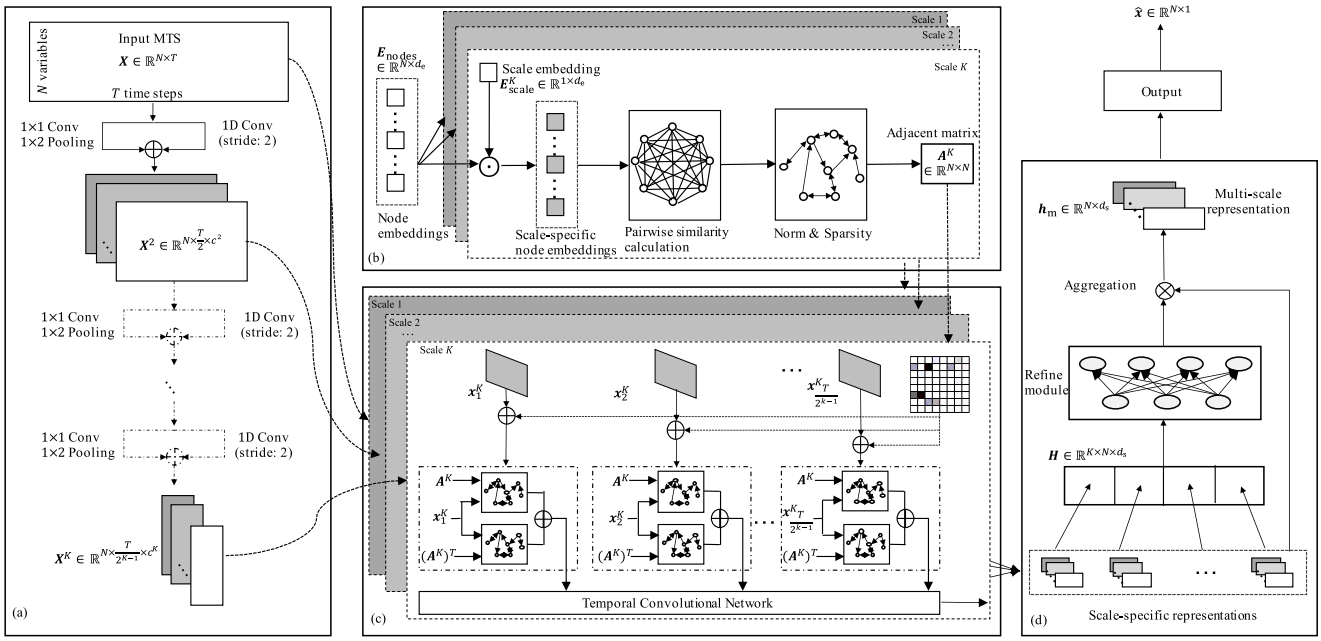


Fig. 2. The Multi-scale Adaptive Graph Neural Network (MAGNN) framework, which consists of four main parts: (a) The two parallel convolutional neural networks and point-wise additions at each layer transform feature representations from smaller scale to larger scale hierarchically. (b) An adaptive graph learning module takes node embeddings and scale embeddings as inputs and outputs the scale-specific adjacency matrices. (c) Each scale-specific feature representation and adjacency matrix are fed into a temporal GNN to obtain scale-specific representations. (d) Scale-specific representations are weighted fused to capture the contributed temporal patterns. The final multi-scale representation is fed into the output module including two convolutional neural networks to obtain the predicted values.

1×1 and a 1×2 pooling layer, which is a parallel structure with the original convolutional neural network, formally,

$$\boldsymbol{X}^k_{\text{norm}} = Pooling\left(ReLU(\boldsymbol{W}^k_{\text{norm}} \otimes \boldsymbol{X}^{k-1} + \boldsymbol{b}^k_{\text{norm}})\right). \quad (5)$$

Then, a point-wise addition is utilized to the outputs of these two convolutional neural networks at each scale:

$$\boldsymbol{X}^k = \boldsymbol{X}^k_{\text{rec}} + \boldsymbol{X}^k_{\text{norm}}. \quad (6)$$

After that, the learned multi-scale feature representations are flexible and comprehensive to preserve various kinds of temporal dependencies. During the process of feature representation learning, to avoid the interaction between the variables of MTS, the convolutional operations are performed on the time dimension, and the variable dimension is fixed, i.e., the kernels are shared between the variable dimension at each pyramid layer.

C. Adaptive Graph Learning

The adaptive graph learning module automatically generates adjacency matrices to represent the inter-variable dependencies among MTS. Existing learning-based methods [15], [16], [41] only learn a shared adjacency matrix, which is useful to learn the most prominent inter-variable dependencies among MTS in many problems, and can significantly reduce the number of parameters and avoid the overfitting problem. However, the inter-variable dependencies may be different under different time scales. The shared adjacency matrix makes the models biased to learn one type of prominent and shared temporal patterns. Therefore, it is essential to learn multiple scale-specific adjacency matrices.

However, directly learning a unique adjacent matrix for each scale will introduce too many parameters and make the model hard to train, especially when the number of nodes is large [42]. To solve this problem, we propose an adaptive graph learning (AGL) module that has K scale-specific layers. Inspired by the matrix factorization, AGL has two kinds of parameters: 1) node embeddings $\boldsymbol{E}_{\text{nodes}} \in \mathbb{R}^{N \times d_e}$ shared between all scales, where d_e is the embedding dimension and $d_e \ll N$; 2) scale embeddings $\boldsymbol{E}_{\text{scale}} \in \mathbb{R}^{K \times d_e}$. For the kth scale-specific layer, scale-specific node embeddings $\boldsymbol{E}^k_{\text{spec}}$ are obtained by the point-wise multiplication of the kth scale embedding $\boldsymbol{E}^k_{\text{scale}} \in \mathbb{R}^{1 \times d_e}$ and node embeddings $\boldsymbol{E}_{\text{nodes}}$:

$$\boldsymbol{E}^k_{\text{spec}} = \boldsymbol{E}_{\text{nodes}} \odot \boldsymbol{E}^k_{\text{scale}}. \quad (7)$$

By such a design, the number of parameters is limited, while $\boldsymbol{E}^k_{\text{spec}}$ contains both the shared node information and the scale-specific information. Then, similar to calculating the node proximities by a similarity function, we calculate pairwise node similarities as follows:

$$\begin{aligned} \boldsymbol{M}^k_1 &= [tanh(\boldsymbol{E}^k_{\text{spec}}\theta^k)]^T, \\ \boldsymbol{M}^k_2 &= tanh(\boldsymbol{E}^k_{\text{spec}}\varphi^k), \\ \boldsymbol{A}^k_{\text{full}} &= ReLU(\boldsymbol{M}^k_1\boldsymbol{M}^k_2 - (\boldsymbol{M}^k_2)^T(\boldsymbol{M}^k_1)^T), \end{aligned} \quad (8)$$

where $\theta^k \in \mathbb{R}^{1\times 1}$ and $\varphi^k \in \mathbb{R}^{1\times 1}$ are learnable parameters to obtain the receiver and sender features of nodes from $\boldsymbol{E}^k_{\text{spec}}$[18],

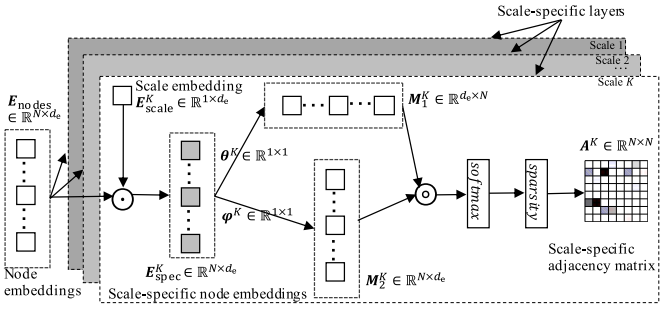


Fig. 3. The detailed architecture of the adaptive graph learning module.

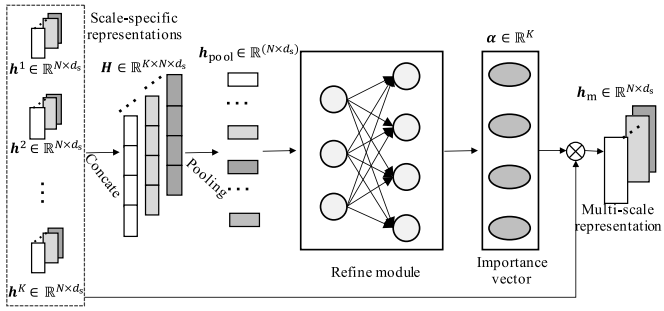


Fig. 4. The detailed architecture of the scale-wise fusion module.

[43], i.e., $\boldsymbol{M}_1^k$ and $\boldsymbol{M}_2^k$, respectively. The activation function $tanh$ is used to normalize the input values to $[-1, 1]$. The values of $\boldsymbol{A}_{\text{full}}^k \in \mathbb{R}^{N\times N}$ are then normalized to $[0, 1]$ through the activation function $ReLU$, which are used as the soft edges among the nodes. To reduce the computation cost of the graph convolution, reduce the impact of noise, and make the model more robust, we introduce a strategy to make $\boldsymbol{A}_{\text{full}}^k$ sparse:

$$\boldsymbol{A}^k = Sparse\left(Softmax(\boldsymbol{A}_{\text{full}}^k)\right), \tag{9}$$

where $\boldsymbol{A}^k \in \mathbb{R}^{N\times N}$ is the final adjacent matrix of the kth layer, $Softmax$ function is used to achieve normalization, and $Sparse$ function is defined as:

$$\boldsymbol{A}_{ij}^k = \begin{cases} \boldsymbol{A}_{ij}^k, & \boldsymbol{A}_{ij}^k \in TopK(\boldsymbol{A}_{i*}^k, \tau) \\ 0, & \boldsymbol{A}_{ij}^k \notin TopK(\boldsymbol{A}_{i*}^k, \tau) \end{cases}, \tag{10}$$

where τ is the threshold of $TopK$ function and denotes the max number of neighbors of a node. The overall architecture of the AGL module is shown in Fig. 3. Finally, we can obtain the scale-specific adjacent matrices $\{\boldsymbol{A}^1, \ldots, \boldsymbol{A}^k, \ldots, \boldsymbol{A}^K\}$.

D. Multi-Scale Temporal Graph Neural Network

Given the multi-scale feature representations $\{\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k, \ldots, \boldsymbol{X}^K\}$ generated from the multi-scale pyramid network, and the scale-specific adjacent matrices $\{\boldsymbol{A}^1, \ldots, \boldsymbol{A}^k, \ldots, \boldsymbol{A}^K\}$ generated from the AGL module, a multi-scale temporal graph neural network (MTG) is proposed to capture scale-specific temporal patterns across time steps and variables.

Existing works [17], [23] integrate the GRU and the GNN, which replaces the MLP in the GRU with the GNN to learn inter-variable dependencies. However, the RNN-based solutions often suffer from the gradient vanishing and exploding problems, and adopt the step-by-step strategy for recurrent layers, which makes the model training inefficient, especially when the time series is long enough [44]. Temporal convolutional networks (TCNs) have shown superiority on modeling temporal patterns. Thus, we propose a solution that combines the GNNs and temporal convolution layers, i.e., replacing the GRU with temporal convolution layers.

Specifically, the MTG consists of K temporal graph neural networks, each of which combines the TCNs and the GNN to capture scale-specific temporal patterns. For the kth scale, we first split $\boldsymbol{X}^k$ at time dimension and obtain $\{\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_t^k, \ldots, \boldsymbol{x}_{\frac{T}{2^{k-1}}}^k\}(\boldsymbol{x}_t^k \in \mathbb{R}^{N\times c^k})$. Similar with [18], [43], we introduce $\boldsymbol{A}^k$ and the transpose of $\boldsymbol{A}^k$ (i.e., $(\boldsymbol{A}^k)^T$), and exploit two GNNs to capture both incoming information and outgoing information. Then, the results of GNNs are added:

$$\widetilde{\boldsymbol{h}}_t^k = GNN_{\text{in}}^k(\boldsymbol{x}_t^k, \boldsymbol{A}^k, \boldsymbol{W}_{\text{in}}^k) + GNN_{\text{out}}^k(\boldsymbol{x}_t^k, (\boldsymbol{A}^k)^T, \boldsymbol{W}_{\text{out}}^k), \tag{11}$$

where $\boldsymbol{W}_*^k$ denotes the trainable parameters of GNNs in the kth scale. Then, we can obtain all the outputs $\{\widetilde{\boldsymbol{h}}_1^k, \ldots, \widetilde{\boldsymbol{h}}_t^k, \ldots, \widetilde{\boldsymbol{h}}_{\frac{T}{2^k}}^k\}$, which are fed into a temporal convolution layer to obtain the scale-specific representations $\boldsymbol{h}^k$:

$$\boldsymbol{h}^k = TCN^k\left(\left[\widetilde{\boldsymbol{h}}_1^k, \ldots, \widetilde{\boldsymbol{h}}_t^k, \ldots, \widetilde{\boldsymbol{h}}_{\frac{T}{2^k}}^k\right], \boldsymbol{W}_{\text{tcn}}^k\right), \tag{12}$$

where $\boldsymbol{W}_{\text{tcn}}^k$ denotes the trainable parameters in the kth temporal convolution layer.

We can see the advantages of exploiting MTG: 1) it can capture scale-specific temporal patterns across time steps and variables; 2) the graph convolution operator enables the model to explicitly consider the inter-variable dependencies.

E. Scale-Wise Fusion

All the scale-specific representations $\{\boldsymbol{h}^1, \ldots, \boldsymbol{h}^k, \ldots, \boldsymbol{h}^K\}$ can comprehensively reflect all kinds of temporal patterns, where $\boldsymbol{h}^k \in \mathbb{R}^{N\times d_s}$, and d_s denotes the output dimension of TCNs. To obtain the final multi-scale representation, the intuitive solution is to directly concatenate these scale-specific representations or aggregate these representations by a global pooling layer. However, this solution treats each scale-specific representation equally and ignores the difference in contribution to the final forecasting results. For example, the small scale representations are more important for short-term forecasting, while the large scale representations are more important for long-term forecasting. Thus, we propose a scale-wise fusion module to learn a robust multi-scale representation from these scale-specific representations, which can consider the importance of scale-specific temporal patterns and capture the cross-scale correlations.

Fig. 4 shows the overall architecture of the scale-wise fusion module. Given the scale-specific representations

$\{\boldsymbol{h}^1, \ldots, \boldsymbol{h}^k, \ldots, \boldsymbol{h}^K\}$, we first concatenate these representations to obtain the multi-scale matrix $\boldsymbol{H} \in \mathbb{R}^{K \times N \times d_s}$:

$$\boldsymbol{H} = Concat(\boldsymbol{h}^1, \ldots, \boldsymbol{h}^k, \ldots, \boldsymbol{h}^K), \tag{13}$$

where $Concat$ denotes the concatenation operation. Then, we exploit an average pooling layer on the scale dimension:

$$\boldsymbol{h}_{\text{pool}} = \frac{\sum_{k=1}^{K} \boldsymbol{H}^k}{K}, \tag{14}$$

where $\boldsymbol{h}_{\text{pool}} \in \mathbb{R}^{1 \times N \times d_s}$. Then, we flat $\boldsymbol{h}_{\text{pool}}$ and fed it into a refining module that consists of two full connected layers to compact the fine-grained information across different time scales:

$$\begin{aligned} \boldsymbol{\alpha}_1 &= ReLU\left(\boldsymbol{W}_1 \boldsymbol{h}_{\text{pool}} + \boldsymbol{b}_1\right), \\ \boldsymbol{\alpha} &= Sigmoid\left(\boldsymbol{W}_2 \boldsymbol{\alpha}_1 + \boldsymbol{b}_2\right), \end{aligned} \tag{15}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are weight matrices. $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are bias vectors. The $sigmoid$ activation function is used in the second layer. $\boldsymbol{\alpha} \in \mathbb{R}^K$ is defined as the importance score vector that represents the importance of different scale-specific representations. Finally, a weighted aggregation layer is exploited to combine the scale-specific representations:

$$\boldsymbol{h}_{\text{m}} = ReLU\left(\sum_{k=1}^{K} \boldsymbol{\alpha}[k] \times \boldsymbol{h}^k\right), \tag{16}$$

where $\boldsymbol{h}_{\text{m}}$ is the final multi-scale representation.

F. Output Module & Objection Function

The output module includes a convolutional neural network with $1 \times d_s$ kernel size to transform $\boldsymbol{h}_{\text{m}} \in \mathbb{R}^{N \times d_s}$ into the desired output dimension, and a followed convolutional neural network with 1×1 kernel size to obtain the predicted values $\widehat{\boldsymbol{x}} \in \mathbb{R}^{N \times 1}$.

The objective function can be formulated as:

$$\mathcal{L} = \frac{1}{\mathcal{T}_{\text{train}}} \sum_{i=1}^{\mathcal{T}_{\text{train}}} \sum_{j=1}^{N} \left(\widehat{\boldsymbol{x}}_{i,j} - \boldsymbol{x}_{i,j}\right)^2, \tag{17}$$

where $\mathcal{T}_{\text{train}}$ is the number of training samples, and N is the number of variables. $\widehat{\boldsymbol{x}}_{i,j}$ and $\boldsymbol{x}_{i,j}$ are the predicted value and ground-truth of the jth variable in the ith sample, respectively.

G. Complexity Analysis

The time complexity of MAGNN consists of the main four modules. For the multi-scale pyramid network, the time complexity of the kth scale is $\Theta(N \times \frac{T}{2^{k-1}} \times c_{\text{in}} \times c_{\text{out}})$ and the overall time complexity is $\Theta(N \times T \times c_{\text{in}} \times c_{\text{out}})$, where N is the variable dimension, T is the input sequence length, c_{in} and c_{out} are the numbers of input channels and output channels, respectively. Since c_{in} and c_{out} are regarded as constants, the time complexity of the multi-scale pyramid network is $\Theta(N \times T)$. For the AGL module, the time complexity is $\Theta(K \times N \times d_{\text{e}}^2 + K \times N^2 \times d_{\text{e}})$, where K is the number of scales and d_{e} is the dimension of node or scale embedding. The first half part denotes the point-wise multiplication between node embeddings and scale embeddings. The latter part denotes the pairwise similarity calculation. Since d_{e} is regarded as a constant, the time complexity of the AGL module is $\Theta(K \times N^2)$. For the MTG module, the time complexity is $\Theta(K(m \times d_1 + N \times d_{\text{in}} \times d_{\text{s}}))$, where m denotes the number of edges. d_{in} and d_{s} denote the input dimension and output dimension, respectively. This result comes from the message passing and information aggregation of GNN. Regarding d_1, d_{in}, and d_{s} as constants, the time complexity of the MTG module is $\Theta(K(m+N))$. For the scale-wise fusion module, the time complexity is $\Theta(N \times d_{\text{s}} \times d_1 + d_1 \times K)$, where d_1 is the output dimension of the first full connected layer. Since d_{s} and d_1 are regarded as constants, the time complexity of the scale-wise fusion module is $\Theta(N+K)$.

TABLE I
DATASET STATISTICS

Datasets	# Samples	# Variables	Sample rate
Solar-Energy	52560	137	10 minutes
Traffic	17544	862	1 hour
Electricity	26304	321	1 hour
Exchange-Rate	7588	8	1 day
Nasdaq	40560	82	1 minute
METR-LA	34272	207	5 minutes

V. EXPERIMENTS

A. Datasets and Settings

Datasets: To evaluate the performance of MAGNN, we conduct experiments on six public benchmark datasets: Solar-Energy, Traffic, Electricity, Exchange-Rate, Nasdaq, and METR-LA. Table I gives the summarized dataset statistics, and the details about the six public benchmark datasets are given as follows:

- Solar-Energy: This dataset contains the collected solar power from the National Renewable Energy Laboratory, which is sampled every 10 minutes from 137 PV plants in Alabama State in 2007.
- Traffic: This dataset contains the road occupancy rates (between 0 and 1) from the California Department of Transportation, which are hourly aggregated from 862 sensors in San Francisco Bay Area from 2015 to 2016.
- Electricity: This dataset contains the electricity consumptions from the UCI Machine Learning Repository, which are hourly aggregated from 321 clients from 2012 to 2014.
- Exchange-Rate: This dataset contains the exchange rates of eight countries, which are sampled daily from 1990 to 2016.
- Nasdaq: This dataset contains the stock prices of 82 corporations, which are sampled per minute from July 2016 to December 2016.
- METR-LA: This dataset contains the average traffic speeds from Los Angeles County, which are 5-minute aggregated from 207 loop detectors on the highways from March 2012 to June 2012.

Following existing works [3], [13], [18], the six datasets are split into the training set (60%), validation set (20%), and test set (20%) in chronological order.

TABLE II
SETTINGS OF NNI

	Parameters	Choise
Search spaces	Channel size	$\{8, 16, 32, 64\}$
	Dropout rate	$\{0.1, 0.5\}$, uniform
	# neighbors (Exchange-Rate dataset)	$\{5, 6, 7, 8\}$
	# neighbors (the other datasets)	$\{20, 30, 40, 50\}$
Configures	Max trial number	15
	Optimization algorithm	Tree-structured Parzen Estimator
	Early stopping strategy	Curvefitting

Experimental Settings: MAGNN is implemented in Python with PyTorch 1.7.1 and trained with one GPU (NVIDIA RTX 3090), and the source code is released on GitHub.[1] For experimental settings, unlike existing works that conduct grid search over all tunable hyper-parameters, we exploit Neural Network Intelligence (NNI)[2] toolkit to automatically search the best hyper-parameters, which can greatly reduce computation costs. The search spaces of hyper-parameters and the configures of NNI are given in Table II. Following existing works [3], [18], the input window size T is set to 168. The learning rate is set to 0.001. Adam optimizer is used and all trainable parameters can be optimized through back-propagation. For all datasets, the number of scales is 4. The kernel size of CNNs in the multi-scale pyramid network is set to 1×7, 1×6, and 1×3 from the first layer to the final layer of the pyramid network, and the stride size is set to 2 for all CNNs. We set horizon $h = \{3, 6, 12, 24\}$, respectively, which means the forecasting horizons are set from 3 to 24 minutes for Nasdaq dataset, from 15 to 120 minutes for METR-LA dataset, from 30 to 240 minutes for Solar-Energy dataset, from 3 to 24 hours for Traffic and Electricity datasets, and from 3 to 24 days for Exchange-Rate dataset. The larger the forecasting horizon is, the harder the forecasting is.

Evaluation Metrics: Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) are exploited as evaluation metrics, which are defined as:

$$\text{RSE} = \frac{\sqrt{\sum_{i=1}^{\mathcal{T}_{\text{test}}} \sum_{j=1}^{N} \left(\hat{\boldsymbol{x}}_{i,j} - \boldsymbol{x}_{i,j}\right)^2}}{\sqrt{\sum_{i=1}^{\mathcal{T}_{\text{test}}} \sum_{j=1}^{N} \left(\boldsymbol{x}_{i,j} - \text{mean}(\boldsymbol{x})\right)^2}}, \tag{18}$$

$$\text{CORR} = \frac{1}{\mathcal{T}_{\text{test}}} \sum_{j=1}^{N} \frac{\sum_{i=1}^{\mathcal{T}_{\text{test}}} (\boldsymbol{x}_{i,j} - \text{mean}(\boldsymbol{x}_{*,j}))(\hat{\boldsymbol{x}}_{i,j} - \text{mean}(\hat{\boldsymbol{x}}_{*,j}))}{\sqrt{\sum_{i=1}^{\mathcal{T}_{\text{test}}} (\boldsymbol{x}_{i,j} - \text{mean}(\boldsymbol{x}_{*,j}))^2 (\hat{\boldsymbol{x}}_{i,j} - \text{mean}(\hat{\boldsymbol{x}}_{*,j}))^2}}, \tag{19}$$

where $\mathcal{T}_{\text{test}}$ is the total time steps used for test. For RSE, a lower value is better, while for CORR, a higher value is better.

B. Methods for Comparison

The methods in our comparative evaluation and the search spaces of their key hyper-parameters are as follows.

Conventional methods:

- AR: It stands for the auto-regressive model. The number of lags is chosen from $\{2^0, 2^2, 2^4, 2^6\}$.
- TRMF [7]: It stands for the auto-regressive model using temporal regularized matrix factorization. The hidden dimension size of latent temporal embedding and the regularization coefficient λ are chosen from $\{2^2, 2^3, \ldots, 2^6\}$ and $\{0.1, 1, 10\}$, respectively.
- GP [6]: It stands for the Gaussian process time series model. The RBF kernel bandwidth σ and the noise level α are chosen from $\{2^{-10}, 2^{-8}, \ldots, 2^{10}\}$.
- VAR-MLP [45]: It stands for a hybrid model that combines auto-regressive model (VAR) and multilayer perception (MLP). The size of dense layers is chosen from $\{32, 50, 100\}$.
- RNN-GRU [9]: It stands for the RNN using GRU cell for time series forecasting. The hidden dimension size of RNN layers is chosen from $\{32, 50, 100\}$.

Attentive recurrent methods:

- LSTNet [3]: It introduces the CNNs to capture short-term temporal dependencies, and a recurrent-skip layer to capture long-term periodic patterns. The hidden dimension sizes of recurrent layers, convolutional layers, and recurrent-skip layers are chosen from $\{32, 50, 100\}$, $\{32, 50, 100\}$, and $\{20, 50, 100\}$, respectively.
- MTNet [12]: It exploits the memory component and attention mechanism to capture long-term temporal dependencies and periodic patterns. The hidden dimension sizes of GRU and convolutional layers are chosen from $\{32, 50, 100\}$.
- TPA-LSTM [13]: It utilizes an attention mechanism to extract important temporal patterns from different time steps and different variables. The hidden dimension sizes of recurrent and convolutional layers are chosen from $\{32, 50, 100\}$.

MTS modeling with graph learning:

- Graph WaveNet [21]: It utilizes graph convolutions and dilated 1D convolutions to model spatial-temporal relations. The hidden dimension size of node embedding is chosen from $\{1, 3, 5, 10, 15, 20, 30\}$.
- AGCRN [17]: It exploits adaptive graph convolutional recurrent network to infer the inter-variable dependencies. The hidden dimension size of node embedding is chosen from $\{1, 3, 5, 10, 15, 20, 30\}$.
- MTHetGNN [46]: It utilizes heterogeneous graph embedding module to characterize complex relations among variables. The hidden dimension size of graph convolutional layers is chosen from $\{5, 10, 15, 20, 50, 100, 200\}$.
- MTGNN [18]: It uses a graph learning module to learn inter-variable dependencies, and models MTS using GNN

[1][Online]. Available: https://github.com/shangzongjiang/MAGNN
[2][Online]. Available: https://nni.readthedocs.io/en/latest/

TABLE III
RESULTS SUMMARY (IN TERMS OF RSE) OF ALL METHODS ON SIX DATASETS

Datasets	Horizons	AR	TRMF	VAR-MLP	GP	RNN-GRU	LSTNet	MTNet	TPA-LSTM	AGCRN	Graph WaveNet	MTHetGNN	MTGNN	MAGNN
Solar-Energy	3	0.2435	0.2473	0.1922	0.2259	0.1932	0.1843	0.1847	0.1803	0.1840	0.1773	0.1838	0.1778	**0.1771**
	6	0.3790	0.3470	0.2679	0.3286	0.2628	0.2559	0.2398	0.2347	0.2432	**0.2279**	0.2600	0.2348	0.2361
	12	0.5911	0.5597	0.4244	0.5200	0.4163	0.3254	0.3251	0.3234	0.3185	0.3068	0.3169	0.3109	**0.3015**
	24	0.8699	0.9005	0.6841	0.7973	0.4852	0.4643	0.4285	0.4389	0.4141	0.4206	0.4231	0.4270	**0.4108**
Traffic	3	0.5991	0.6708	0.5582	0.6082	0.5358	0.4777	0.4764	0.4487	0.4379	0.4484	0.4826	0.4162	**0.4097**
	6	0.6218	0.6261	0.6579	0.6772	0.5522	0.4893	0.4855	0.4658	0.4635	0.4689	0.5198	0.4754	**0.4555**
	12	0.6252	0.5956	0.6023	0.6406	0.5562	0.4950	0.4877	0.4641	0.4694	0.4725	0.5147	0.4461	**0.4423**
	24	0.6300	0.6442	0.6146	0.5995	0.5633	0.4973	0.5023	0.4765	0.4707	0.4741	0.5250	0.4535	**0.4434**
Electricity	3	0.0995	0.1802	0.1393	0.1500	0.1102	0.0864	0.0840	0.0823	0.0766	0.0746	0.0749	**0.0745**	**0.0745**
	6	0.1035	0.2039	0.1620	0.1907	0.1144	0.0931	0.0901	0.0916	0.0894	0.0922	0.0892	0.0878	**0.0876**
	12	0.1050	0.2186	0.1557	0.1621	0.1183	0.1007	0.0934	0.0964	0.0921	0.0909	0.0959	0.0916	**0.0908**
	24	0.1054	0.3656	0.1274	0.1273	0.1295	0.1007	0.0969	0.1006	0.0967	0.0962	0.0969	**0.0953**	0.0963
Exchange-Rate	3	0.0228	0.0351	0.0265	0.0239	0.0192	0.0226	0.0212	**0.0174**	0.0269	0.0251	0.0198	0.0194	0.0183
	6	0.0279	0.0875	0.0394	0.0272	0.0264	0.0280	0.0258	**0.0241**	0.0331	0.0300	0.0259	0.0259	0.0246
	12	0.0353	0.0494	0.0407	0.0394	0.0408	0.0356	0.0347	**0.0341**	0.0374	0.0381	0.0345	0.0349	0.0343
	24	0.0445	0.0563	0.0578	0.0580	0.0626	0.0449	**0.0442**	0.0444	0.0476	0.0486	0.0451	0.0456	0.0474
Nasdaq	3	0.0028	0.0020	0.0021	0.0016	0.0014	0.0018	0.0014	0.0012	0.0012	0.0018	0.0016	0.0015	**0.0010**
	6	0.0029	0.0023	0.0025	0.0024	0.0019	0.0019	0.0019	0.0013	0.0018	0.0022	0.0026	0.0018	**0.0011**
	12	0.0031	0.0021	0.0027	0.0022	0.0022	0.0021	0.0022	**0.0018**	0.0022	0.0023	0.0020	0.0020	**0.0018**
	24	0.0033	0.0026	0.0028	0.0027	0.0024	0.0025	0.0024	0.0024	0.0023	0.0026	0.0030	0.0026	**0.0020**
METR-LA	3	0.6153	0.5266	0.4240	0.4273	0.4338	0.4259	0.4274	0.4238	0.4218	0.4206	0.4217	**0.4201**	0.4202
	6	0.6734	0.7907	0.5468	0.5426	0.6865	0.5384	0.5426	0.5371	0.5485	0.5420	0.5448	0.5422	**0.5345**
	12	0.7579	0.7896	0.6854	0.6752	0.7183	0.6881	0.6752	0.6758	0.6744	0.6746	0.6805	**0.6642**	0.6728
	24	0.8706	0.8490	0.8635	0.8230	0.8310	0.8202	0.8230	0.8292	0.8277	0.8054	0.8324	0.8146	**0.8044**

The best results are *bolded*, and the second best results are underlined.

TABLE IV
RESULTS SUMMARY (IN TERMS OF CORR) OF ALL METHODS ON SIX DATASETS

Datasets	Horizons	AR	TRMF	VAR-MLP	GP	RNN-GRU	LSTNet	MTNet	TPA-LSTM	AGCRN	Graph WaveNet	MTHetGNN	MTGNN	MAGNN
Solar-Energy	3	0.9710	0.9703	0.9829	0.9751	0.9823	0.9843	0.9840	0.9850	0.9841	0.9846	0.9845	0.9852	**0.9853**
	6	0.9263	0.9418	0.9655	0.9448	0.9675	0.9690	0.9723	0.9742	0.9708	**0.9743**	0.9681	0.9726	0.9724
	12	0.8107	0.8475	0.9058	0.8518	0.9150	0.9467	0.9462	0.9487	0.9487	0.9527	0.9486	0.9509	**0.9539**
	24	0.5314	0.5598	0.7149	0.5971	0.8823	0.8870	0.9013	0.9081	0.9087	0.9055	0.9031	0.9031	**0.9097**
Traffic	3	0.7752	0.6964	0.8245	0.7831	0.8511	0.8721	0.8728	0.8812	0.8850	0.8801	0.8643	0.8963	**0.8992**
	6	0.7568	0.7430	0.7695	0.7406	0.8405	0.8690	0.8681	0.8717	0.8670	0.8674	0.8452	0.8667	**0.8753**
	12	0.7544	0.7748	0.7929	0.7671	0.8345	0.8614	0.8644	0.8717	0.8679	0.8646	0.8744	0.8794	**0.8815**
	24	0.7519	0.7278	0.7891	0.7909	0.8300	0.8588	0.8570	0.8629	0.8664	0.8646	0.8418	0.8810	**0.8813**
Electricity	3	0.8845	0.8538	0.8708	0.8670	0.8597	0.9283	0.9319	0.9439	0.9408	0.9459	0.9456	0.9474	**0.9476**
	6	0.8632	0.8424	0.8389	0.8334	0.8623	0.9135	0.9226	**0.9337**	0.9309	0.9310	0.9307	0.9316	0.9323
	12	0.8591	0.8304	0.8192	0.8394	0.8472	0.9077	0.9165	0.9250	0.9222	0.9267	0.8783	0.9278	**0.9282**
	24	0.8595	0.7471	0.8679	0.8818	0.8651	0.9119	0.9147	0.9133	0.9183	0.9226	0.8782	**0.9234**	0.9217
Exchange-Rate	3	0.9734	0.9142	0.8609	0.8713	0.9786	0.9735	0.9767	**0.9790**	0.9717	0.9740	0.9769	0.9786	0.9778
	6	0.9656	0.8123	0.8725	0.8193	**0.9712**	0.9658	0.9703	0.9709	0.9615	0.9640	0.9701	0.9708	**0.9712**
	12	0.9526	0.8993	0.8280	0.8484	0.9531	0.9511	0.9561	**0.9564**	0.9531	0.9510	0.9539	0.9551	0.9557
	24	0.9357	0.8678	0.7675	0.8278	0.9223	0.9354	**0.9388**	0.9381	0.9334	0.9294	0.9360	0.9372	0.9339
Nasdaq	3	0.5055	0.7768	0.8855	0.8960	0.9566	0.9882	0.9851	0.9745	0.9878	0.9953	0.9919	0.9912	**0.9975**
	6	0.5316	0.7744	0.8937	0.8940	0.9536	0.9865	0.9840	0.9707	0.9877	0.9943	0.9897	0.9876	**0.9951**
	12	0.4109	0.7562	0.8970	0.8805	0.9446	0.9827	0.9804	0.9705	0.9816	0.9840	0.9849	0.9834	**0.9864**
	24	0.4427	0.7234	0.8830	0.8837	0.9371	0.9751	0.9837	0.9627	0.9701	0.9834	0.9799	0.9754	**0.9846**
METR-LA	3	0.8317	0.8522	0.8963	0.8342	0.8964	0.8965	0.8963	0.8663	0.8978	0.8985	0.8972	**0.8992**	0.8987
	6	0.7723	0.6216	0.8281	0.8311	0.8004	0.8219	0.8230	0.7955	0.8245	0.8292	0.8312	0.8296	**0.8328**
	12	0.6770	0.6172	0.7147	0.7153	0.7031	0.7093	0.7156	0.6898	0.6859	0.6760	0.7215	0.7279	**0.7289**
	24	0.5326	0.4998	0.5038	0.5366	0.5589	0.5790	0.5798	0.5519	0.5781	0.5846	**0.5859**	0.5855	**0.5859**

The best results are *bolded*, and the second best results are underlined.

and dilated convolution. The number of neighbors for each node is chosen from $\{5, 6, 7, 8, 15, 30\}$.

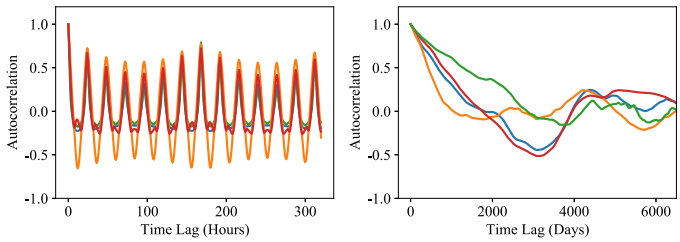
- MAGNN: It is our proposed method.

On Solar-Energy, Electricity, Traffic, and Exchange-Rate datasets, most baselines (AR, TRMF, VAR-MLP, GP, RNN-GRU, LSTNet, MTNet, TPA-LSTM, and MTGNN) have been compared in the existing literature [3], [12], [13], [18]. Thus, we directly adopt the experimental results in literature. For the results of AGCRN, Graph WaveNet, and MTHetGNN on these four datasets and the results of all baselines on Nasdaq and METR-LA datasets, we use the code released in the original papers and tune the key hyper-parameters according to the validation error by NNI toolkit.

C. Main Results

Tables III and IV report the evaluation results of all the methods on the six datasets, and the following tendencies can be discerned:

1) Our method (MAGNN) achieves the state-of-the-art results on these datasets. Particularly, on Traffic and Nasdaq datasets, MAGNN outperforms existing methods on all the horizons and all the metrics. The reason might be that the traffic and stock data are very suitable for our assumption, as there are multi-scale temporal dependencies and complicated inter-variable dependencies. However, on Exchange-Rate dataset, MAGNN obtains slightly worse performance than existing methods. To explore the reasons, Fig. 5 shows the autocorrelation graphs of sampled variables on Traffic and Exchange-Rate datasets. For Traffic



(a) Traffic dataset (b) Exchange-Rate dataset

Fig. 5. The autocorrelation graphs of four sampled variables.

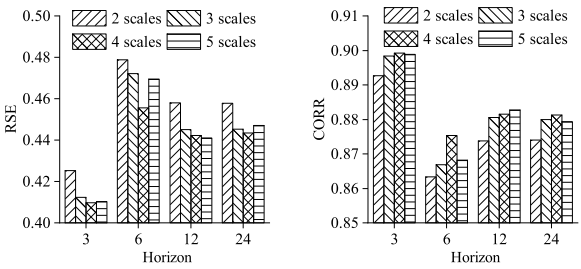


Fig. 6. The results of MAGNN under different numbers of scales.

dataset, we can clearly observe the daily and weekly patterns. In contrast, for Exchange-Rate dataset, we can hardly see the multi-scale temporal dependencies. These observations provide empirical guidance for the success of using MAGNN in modeling MTS.

2) Traditional methods (AR, TRMF, and GP) get worse results than deep learning methods, as they cannot capture the non-stationary and non-linear dependencies.

3) Deep learning methods (VAR-MLP, RNN-GRU, LSTNet, MTNet, and TPA-LSTM) do not explicitly model the pairwise inter-variable dependencies. Thus, they get worse performance than AGCRN, Graph WaveNet, MTHetGNN, MTGNN, and MAGNN on most datasets. However, for Exchange-Rate dataset, TPA-LSTM and MTNet outperform most graph-based methods. Specifically, for the RSE evaluation metric, TPA-LSTM achieves the best performance at horizons 3, 6, and 12, and MTNet performs best at horizon 24. One possible explanation for such a phenomenon is that Exchange-Rate dataset only has 7588 samples, leading to the underfitting of graph-based methods that have much more parameters than TPA-LSTM and MTNet.

4) AGCRN, Graph WaveNet, MTHetGNN, and MTGNN are the state-of-the-art methods that use graph learning modules to learn inter-variable dependencies. However, they fail to consider multi-scale inter-variable dependencies and get worse performance than MAGNN in most cases, e.g., MAGNN outperforms MTGNN in 19 out of 24 cases (6 datasets $\times$ 4 horizons) in terms of both RSE and CORR, and exceeds Graph WaveNet in 22 out of 24 cases in terms of both metrics. In contrast, MAGNN learns a temporal representation that can comprehensively reflect both multi-scale temporal patterns and the scale-specific inter-variable dependencies.

D. Effect of Multi-Scale Modeling

To investigate the effect of multi-scale modeling, we evaluate the performance of MAGNN with different numbers of scales (i.e., 2 scales, 3 scales, 4 scales, and 5 scales). Fig. 6 shows the results of MAGNN under different numbers of scales on Traffic dataset. We can observe that when the number of scales increases from 2 to 4, the performance of MAGNN is significantly improved. This is because MAGNN can capture more diversified short-term and long-term patterns. When the number of scales increases up to 5, the performance of MAGNN has not improved, which might be because the number of scales is already meet the needs of the task, and excessive parameters are prone to overfitting.

E. Effect of Multi-Scale Feature Extraction

To investigate the effect of multi-scale feature extraction, we conduct ablation study by carefully designing the following variant.

- MAGNN-dila: It extracts multi-scale features by dilated 1D convolutions.

The results of MAGNN-dila and MAGNN on Electricity dataset are shown in Table V. We can see that MAGNN achieves better performance than MAGNN-dila. The reason is that the dilation rates of dilated 1D convolutions may cause the loss of local information, bringing negative effects on modeling short-term dependencies.

F. Effect of the Parallel CNNs in the Multi-Scale Pyramid Network

To demonstrate the effect of the parallel CNNs in the multi-scale pyramid network, we conduct ablation study by carefully designing the following variant.

- MAGNN-w/o parallel CNNs: It removes the convolutional neural network with kernel size 1×1 and a 1×2 pooling layer from pyramid layers.

The results presented in Table VI show that MAGNN achieves the best performance in all cases on Traffic dataset, indicating the effectiveness of the parallel CNNs. The possible reason for these results is that using the parallel CNNs could make the extracted multi-scale features more stable.

G. Effect of Adaptive Graph Learning

To demonstrate the effect of adaptive graph learning, we conduct ablation study by carefully designing the following four variants.

- MAGNN-dy: For the kth scale at time step t, the graph learning module takes the dynamic feature representation $\boldsymbol{x}_t^k$ as input rather than the static node embeddings. Thus, the graphs are different at different time steps.
- MAGNN-full: It removes the sparsity strategy and obtains the multi-scale full-connected adjacent matrices.
- MAGNN-one: It only learns one shared adjacent matrix to describe the inter-variable dependencies of multi-scale feature representations.
- MAGNN-sym: It uses the symmetric adjacency matrix obtained by $\boldsymbol{M}_1^k$ and its transpose with one GNN rather than the asymmetric adjacency matrix $\boldsymbol{A}^k$ with two GNNs.

The results of these methods on Solar-Energy, Electricity, and Exchange-Rate datasets are shown in Table VII, and the following tendencies can be discerned:

1) MAGNN achieves the best performance in most cases, which indicates the superiority of our learned scale-specific adjacency matrices. Specifically, MAGNN performs better than MAGNN-full, MAGNN-one, and MAGNN-sym, showing the effectiveness of the sparsity strategy, the multiple scale-specific graphs, and the asymmetric adjacency matrix, respectively.

TABLE V
THE RESULTS OF DIFFERENT MULTI-SCALE FEATURE EXTRACTION METHODS

Methods	3		6		12		24	
	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR
MAGNN-dila	0.0751	0.9423	**0.0864**	**0.9323**	**0.0899**	0.9237	0.0969	0.9176
MAGNN	**0.0745**	**0.9476**	0.0876	**0.9323**	0.0908	**0.9282**	**0.0963**	**0.9217**

TABLE VI
THE RESULTS OF DIFFERENT MULTI-SCALE PYRAMID NETWORKS

Methods	3		6		12		24	
	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR
MAGNN-w/o parallel CNNs	0.4407	0.8833	0.4567	0.8737	0.4688	0.8693	0.4739	0.8644
MAGNN	**0.4097**	**0.8992**	**0.4555**	**0.8753**	**0.4423**	**0.8815**	**0.4434**	**0.8813**

TABLE VII
THE RESULTS OF DIFFERENT GRAPH LEARNING METHODS

Methods		Solar-Energy				Electricity				Exchange-Rate			
		3	6	12	24	3	6	12	24	3	6	12	24
MAGNN-dy	RSE	**0.1766**	0.2372	0.3138	**0.4086**	0.0766	0.0864	0.0939	0.0987	0.0258	0.0310	0.0373	0.0460
	CORR	**0.9854**	0.9721	0.9504	**0.9109**	0.9401	0.9310	0.9191	0.9166	0.9719	0.9647	0.9534	**0.9373**
MAGNN-full	RSE	0.1772	0.2398	0.3068	0.4246	0.0749	**0.0854**	0.0909	0.0976	0.0255	0.0284	0.0389	0.0485
	CORR	0.9853	0.9716	0.9522	0.9032	0.9441	0.9278	0.9236	0.9186	0.9727	0.9682	0.9514	0.9369
MAGNN-one	RSE	0.1769	0.2377	0.3085	0.4257	0.0776	0.0873	0.0948	0.0981	0.0277	0.0305	0.0378	0.0475
	CORR	0.9853	0.9720	0.9516	0.9021	0.9386	0.9287	0.9190	0.9142	0.9697	0.9644	0.9527	0.9348
MGNN-sym	RSE	0.1819	0.2423	0.3139	0.4099	0.0773	0.0866	0.0928	0.0979	0.0226	0.0311	0.0372	**0.0454**
	CORR	0.9844	0.9711	0.9501	0.9105	0.9385	0.9296	0.9227	0.9183	0.9759	0.9651	0.9529	0.9366
MAGNN	RSE	0.1771	**0.2361**	**0.3015**	0.4108	**0.0745**	0.0876	**0.0908**	**0.0963**	**0.0183**	**0.0246**	**0.0343**	0.0474
	CORR	0.9853	**0.9724**	**0.9539**	0.9097	**0.9476**	**0.9323**	**0.9282**	**0.9217**	**0.9778**	**0.9712**	**0.9557**	0.9339

TABLE VIII
THE RESULTS OF DIFFERENT MULTI-SCALE TEMPORAL GRAPH NEURAL NETWORKS

Methods		Solar-Energy				Electricity				Exchange-Rate			
		3	6	12	24	3	6	12	24	3	6	12	24
MAGNN-one GNN	RSE	0.1810	0.2461	0.3104	0.4271	0.0777	0.0897	0.0925	0.0976	0.0256	0.0297	0.0373	0.0476
	CORR	0.9847	0.9702	0.9515	0.9034	0.9432	0.9221	0.9228	0.9184	0.9720	0.9665	0.9525	0.9332
MAGNN	RSE	**0.1771**	**0.2361**	**0.3015**	**0.4108**	**0.0745**	**0.0876**	**0.0908**	**0.0963**	**0.0183**	**0.0246**	**0.0343**	**0.0474**
	CORR	**0.9853**	**0.9724**	**0.9539**	**0.9097**	**0.9476**	**0.9323**	**0.9282**	**0.9217**	**0.9778**	**0.9712**	**0.9557**	**0.9339**

TABLE IX
THE RESULTS OF DIFFERENT FUSION METHODS

Methods		Solar-Energy				Electricity				Exchange-Rate			
		3	6	12	24	3	6	12	24	3	6	12	24
MAGNN-con	RSE	0.1813	0.2452	0.3117	0.4407	0.0754	**0.0852**	0.0921	0.0977	0.0254	0.0313	0.0391	**0.0460**
	CORR	0.9847	0.9703	0.9513	0.8954	0.9428	0.9298	0.9221	0.9168	0.9726	0.9643	0.9529	**0.9373**
MAGNN-pooling	RSE	0.1855	0.2496	0.3256	0.4297	0.0764	0.0867	0.0963	0.0979	0.0241	0.0311	0.0370	0.0475
	CORR	0.9839	0.9692	0.9457	0.9013	0.9420	0.9280	0.9193	0.9168	0.9745	0.9647	0.9537	0.9350
MAGNN-att	RSE	0.1817	0.2410	0.3174	0.4368	0.0772	0.0868	0.0925	0.0979	0.0270	0.0304	0.0387	0.0462
	CORR	0.9845	0.9714	0.9495	0.8957	0.9423	0.9303	0.9231	0.9190	0.9706	0.9662	0.9527	0.9356
MAGNN	RSE	**0.1771**	**0.2361**	**0.3015**	**0.4108**	**0.0745**	0.0876	**0.0908**	**0.0963**	**0.0183**	**0.0246**	**0.0343**	0.0474
	CORR	**0.9853**	**0.9724**	**0.9539**	**0.9097**	**0.9476**	**0.9323**	**0.9282**	**0.9217**	**0.9778**	**0.9712**	**0.9557**	0.9339

TABLE X
THE COMPUTATION COSTS OF DIFFERENT METHODS

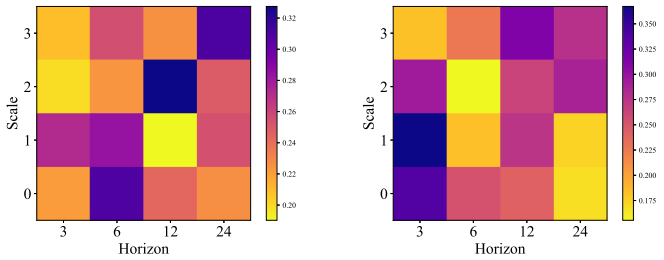
Methods	# Parameters	Training time/epoch	Total training time	RSE	CORR
LSTNet	**71613**	**34.11s**	**0.94h**	0.4777	0.8721
TPA-LSTM	379051	313.41s	8.71h	0.4487	0.8812
MTGNN	337345	349.57s	4.86h	0.4162	0.8963
MAGNN	163325	111.89s	1.55h	**0.4097**	**0.8992**

2) MAGNN-dy shows competitive performance on Solar-Energy and Exchange-Rate datasets. The results imply the potential of learning dynamic adjacency matrices to model time-varying inter-variate dependencies for forecasting. However, the dramatic fluctuation of adjacency matrices makes MAGNN-dy difficult to maintain stable and excellent performance for all horizons on all datasets.

H. Effect of Multi-Scale Temporal Graph Neural Network

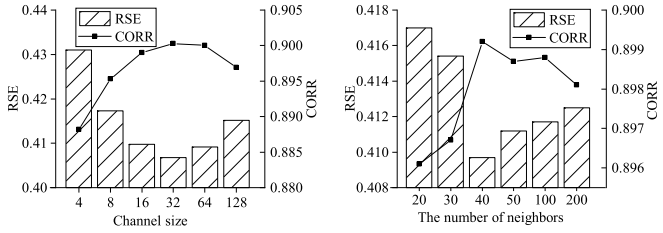
To demonstrate the effect of two GNNs in the multi-scale temporal graph neural network, we conduct ablation study by carefully designing the following variant.

- MAGNN-one GNN: It only applies one GNN on the learned asymmetric adjacency matrix.



(a) Traffic dataset (b) Solar-Energy dataset

Fig. 7. Visualization of the weights of MAGNN for Traffic and Solar-Energy datasets. Models are trained with 4 scales (Y-axis) and the forecasting horizon are 3, 6, 12, and 24 (X-axis).



(a) Traffic dataset (b) Solar-Energy dataset

Fig. 8. The effects of hyper-parameters.

The results presented in Table VIII show that MAGNN achieves the best performance in all cases on Solar-Energy, Electricity, and Exchange-Rate datasets, indicating the effectiveness of two GNNs. The possible reason for these results is that using two GNNs could exploit more hidden complementary information than using one GNN.

I. Effect of Scale-Wise Fusion

To demonstrate the effect of scale-wise fusion, we conduct ablation study by carefully designing the following three variants.

- MAGNN-con: It removes the scale-wise fusion module and directly concatenates these scale-specific representations.
- MAGNN-pooling: It removes the scale-wise fusion module and aggregates these scale-specific representations by a global pooling layer.
- MAGNN-att: It replaces the simple concatenation operation in (13) with attention-based aggregation.

The results of these methods on Solar-Energy, Electricity, and Exchange-Rate datasets are shown in Table IX. We can see that, MAGNN achieves the best performance in most cases. The results imply that MAGNN learns a robust multi-scale representation from these scale-specific representations, as our scale-wise fusion can consider the importance of scale-specific temporal patterns and capture the cross-scale correlations.

To investigate the effect of different scales, we visualize the weights of temporal representations of different scales for the different forecasting horizons on Traffic and Solar-Energy datasets. The visual results are shown in Fig. 7, which indicate that the representations of small scales are more important for short-term forecasting while those of large scales play more essential roles for long-term forecasting.

J. Parameter Study

We study the two important parameters (i.e., convolutional channel size and the number of neighbors), which could influence the performance of MAGNN. Fig. 8(a) shows the results of MAGNN on Traffic dataset by varying convolutional channel size from 4 to 128. The best performance can be obtained when convolutional channel size is 32. It might be that a small convolutional channel size limits the expressive ability of MAGNN, and a large convolutional channel size would make the model hard to train. Fig. 8(b) shows the results of MAGNN on Traffic dataset by varying the number of neighbors from 20 to 200. The best performance can be obtained when the number of neighbors is 40. The reason may be that a small number of neighbors limits the ability to exploit inter-variable dependencies, and a large number of neighbors would introduce noises.

K. Computation Cost

To evaluate the computation cost, we compare the parameter numbers, training time, and forecasting performances of MAGNN, LSTNet, TPA-LSTM, and MTGNN on Traffic dataset in Table X. LSTNet has least parameter number and runs fastest in these methods. But it gets worst forecasting results. Compared with TPA-LSTM and MTGNN, MAGNN runs fastest and gets best forecasting results. Overall, comprehensively considering the significant forecasting performance improvement and the computation cost, MAGNN demonstrates the superiority over existing methods.

VI. Conclusions and Future Work

In this article, we propose a multi-scale adaptive graph neural network (MAGNN) for MTS forecasting. By exploiting a multi-scale pyramid network to model temporal hierarchy, an adaptive graph learning module to automatically infer inter-variable dependencies, a multi-scale temporal graph neural network to model intra-variable and inter-variable dependencies, and a scale-wise fusion module to promote the collaboration across different time scales, MAGNN outperforms the state-of-the-art methods on six datasets. With the theoretical analysis and experimental verification, we believe that MAGNN can capture multi-scale temporal patterns and complicated inter-variable dependencies for accurate MTS forecasting.

In the future, it is of interest to extend this work in the following three aspects: First, we will design a method to learn dynamic adjacency matrices at different time steps, and introduce a regularizer to constrain the dramatic fluctuation of adjacent matrices. Second, we will design a neural architecture search framework to automatically capture both inter-variable dependencies and intra-variable dependencies. Third, we will further develop a graph matching-based AGL module by evaluating the structural and semantic similarities of multi-scale graphs, which can reduce time complexity and enhance scalability.

REFERENCES

[1] A. K. Palit and D. Popovic, *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications*. Berlin, Germany: Springer, 2006.

[2] D. Cao et al., "Spectral temporal graph neural network for multivariate time-series forecasting," in *Proc. 33rd Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 17766–17778.

[3] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 95–104.

[4] D. Chen, L. Chen, Y. Zhang, B. Wen, and C. Yang, "A multiscale interactive recurrent network for time-series forecasting," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 8793–8803, Sep. 2022.

[5] Z. Pan et al., "Spatio-temporal meta learning for urban traffic prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1462–1476, Mar. 2022.

[6] R. Frigola, "Bayesian time series learning with Gaussian processes," PhD thesis, Dept. Eng., Univ. Cambridge, Cambridge, U.K., 2015.

[7] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proc. 29th Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 847–855.

[8] B. Abraham and J. Ledolter, *Statistical Methods for Forecasting*. Hoboken, NJ, USA: Wiley, 2005.

[9] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Automat.*, 2016, pp. 324–328.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv: 1803.01271*.

[11] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng, "Multi-range attentive bicomponent graph convolutional network for traffic forecasting," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 3529–3536.

[12] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," 2018, *arXiv: 1809.02105*.

[13] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Mach. Learn.*, vol. 108, no. 8, pp. 1421–1441, 2019.

[14] L. Chen, W. Chen, B. Wu, Y. Zhang, B. Wen, and C. Yang, "Learning from multiple time series: A deep disentangled approach to diversified time series forecasting," 2021, *arXiv:2111.04942*.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[16] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 32, pp. 57–81, 2020.

[17] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Proc. 33rd Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 17804–17815.

[18] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 753–763.

[19] H. Xu, Y. Huang, Z. Duan, J. Feng, and P. Song, "Multivariate time series forecasting based on causal inference with transfer entropy and graph neural network," 2020, *arXiv: 2005.01185*.

[20] D. Roy, S. Srivastava, A. Kusupati, P. Jain, M. Varma, and A. Arora, "One size does not fit all: Multi-scale, cascaded RNNs for radar classification," *ACM Trans. Sensor Netw.*, vol. 17, no. 2, pp. 1–27, 2021.

[21] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.

[22] S. Li et al., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. 33rd Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5243–5253.

[23] C. Chen et al., "Gated residual recurrent graph neural networks for traffic prediction," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 485–492.

[24] X. Geng et al., "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 3656–3663.

[25] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1720–1730.

[26] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *Proc. IEEE/CVF 14th Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12026–12035.

[27] C. Amornbunchornvej, E. Zheleva, and T. Berger-Wolf, "Variable-lag Granger causality and transfer entropy for time series analysis," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 4, pp. 1–30, 2021.

[28] K. He, X. Chen, Q. Wu, S. Yu, and Z. Zhou, "Graph attention spatial-temporal network with collaborative global-local learning for citywide mobile traffic prediction," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1244–1256, Apr. 2022.

[29] H. Huang, C. Xu, and S. Yoo, "Bi-directional causal graph learning through weight-sharing and low-rank neural network," in *Proc. IEEE 19th Int. Conf. Data Mining*, 2019, pp. 319–328.

[30] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5415–5428, Nov. 2022.

[31] C. Graber and R. Loh, "Unsupervised discovery of dynamic neural circuits," in *Proc. 33rd Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–5.

[32] T. N. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. S. Zemel, "Neural relational inference for interacting systems," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2688–2697.

[33] C. Tang, J. Sun, Y. Sun, M. Peng, and N. Gan, "A general traffic flow prediction approach based on spatial-temporal graph attention," *IEEE Access*, vol. 8, pp. 153731–153741, 2020.

[34] E. Webb, B. Day, H. Andres-Terre, and P. Lió, "Factorised neural relational inference for multi-interaction systems," 2019, *arXiv: 1905.08721*.

[35] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 1234–1241.

[36] Y. Shen, W. Dai, C. Li, J. Zou, and H. Xiong, "Multi-scale graph convolutional network with spectral graph wavelet frame," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 7, pp. 595–610, 2021.

[37] E. Chien et al., "Node feature extraction by self-supervised multi-scale neighborhood prediction," in *Proc. 28th Int. Conf. Learn. Representations*, 2021, pp. 1–20.

[38] P. Zhou, Z. Wu, G. Wen, K. Tang, and J. Ma, "Multi-scale graph classification with shared graph neural network," *World Wide Web*, vol. 26, pp. 949–966, 2023.

[39] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.

[40] P. Savarese and M. Maire, "Learning implicitly recurrent CNNs through parameter sharing," in *Proc. 25th Int. Conf. Learn. Representations*, 2018, pp. 1–15.

[41] T. Li, K. Zhang, S. Shen, B. Liu, Q. Liu, and Z. Li, "Image co-saliency detection and instance co-segmentation using attention graph clustering based graph convolutional network," *IEEE Trans. Multimedia*, vol. 24, pp. 492–505, 2022.

[42] R. Hu, Z. Deng, and X. Zhu, "Multi-scale graph fusion for co-saliency detection," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 7789–7796.

[43] R.-G. Cirstea, C. Guo, and B. Yang, "Graph attention recurrent neural networks for correlated time series forecasting," in *Proc. SIGKDD Workshop Mining Learn. Time Ser.*, 2019, pp. 1–6.

[44] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond exploding and vanishing gradients: Analysing RNN training using attractors and smoothness," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2370–2380.

[45] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[46] Y. Wang, Z. Duan, Y. Huang, H. Xu, J. Feng, and A. Ren, "MTHetGNN: A heterogeneous graph embedding framework for multivariate time series forecasting," *Pattern Recognit. Lett.*, vol. 153, pp. 151–158, 2022.

Ling Chen received the BS and PhD degrees in computer science from Zhejiang University, China, in 1999 and 2004, respectively. He is currently a professor with the College of Computer Science and Technology, Zhejiang University, China. His research interests include ubiquitous computing and data mining.

Donghui Chen received the PhD degree in computer science from Zhejiang University, China, in 2021. His research interests include time series representation learning and prediction.

Zongjiang Shang received the MS degree in electronic information from Northwestern Polytechnical University, China, in 2020. He is currently working toward the PhD degree with the College of Computer Science and Technology, Zhejiang University, China. His research interests include time series representation learning and prediction.

Binqing Wu received the BEng degree in computer science from Southwest Jiangtong University, China, in 2020. She is currently working toward the PhD degree with the College of Computer Science and Technology, Zhejiang University, China. Her research interests include urban computing and data mining.

Cen Zheng received the MS degree in computer science from Shanghai Jiao Tong University, China, in 2011. He is currently a staff engineer with Alibaba Group. His research interests include distributed storage and database.

Bo Wen received the BS degree in computer science from Jishou University, China, in 2012. He is currently a software engineer with Alibaba Group. His research interests include time-series analytics and time-series oriented database.

Wei Zhang received the PhD degree from UCSB. He is currently a principal engineer and leads the NoSQL-Database Team, Alibaba Group. His research interests include storage, database, and AI.