Modeling Complex Spatial Patterns with Temporal Features via Heterogeneous Graph Embedding Networks

Yida Huang[1*], **Haoyan Xu**[1*†], **Ziheng Duan**[1*]
Anni Ren[2], **Jie Feng**[1], **Qianru Zhang**[3], **Xiaoqian Wang**[4†]
[1]Zhejiang University
[2]Nanjing University of Aeronautics and Astronautics
[3]Harbin Institute of Technology
[4]Purdue University
stevenhuang@zju.edu.cn, haoyanxu@zju.edu.cn, duanziheng@zju.edu.cn
anniren.edu@gmail.com, zjucse_fj@zju.edu.cn, irenechzhang@gmail.com, joy.xqwang@gmail.com

Abstract

Multivariate time series (MTS) forecasting is an important problem in many fields. Accurate forecasting results can effectively help decision-making. Variables in MTS have rich relations among each other and the value of each variable in MTS depends on its historical values and other variables. These rich relations can be static and predictable or dynamic and latent. Existing methods do not incorporate these rich relational information into modeling or only model certain relation among MTS variables. To jointly model rich relations among variables and temporal dependencies within the time series, we propose a novel end-to-end deep learning model in this paper, termed Multivariate Time Series Forecasting via Heterogeneous Graph Neural Networks (MTHetGNN). To characterize rich relations among variables, a relation embedding module is introduced in our model, where each variable is regarded as a graph node and each type of edge represents a specific static/dynamic relationship among variables to model the latent dependency among variables. In addition, convolutional neural network (CNN) filters with different perception scales are used for time series feature extraction, which is used to generate the feature of each node. Finally, heterogeneous graph neural networks are adopted to handle the complex structural information generated by temporal embedding module and relation embedding module. Three benchmark datasets from the real world are used to evaluate the proposed MTHetGNN and the comprehensive experiments show that MTHetGNN achieves state-of-the-art results in MTS forecasting task.

Introduction

Multivariate time series (MTS) are generally composed of multiple single-dimensional time series of the same object, such as the observation of the same object by multiple sensors, the traffic flow of each block in the same area, or the exchange rate information of different countries. People are very interested in analyzing historical time series to get predictions about future trends. MTS have the following two major characteristics: (1) Each single-dimensional time series has internal temporal dependency pattern; (2) MTS have rich spatial relations among different variables. Therefore, the modeling of multi-dimensional time series should consider these two sources of information and provide probabilistic explanations to make the predictions more reasonable.

Regarding each time series in MTS as a variable, interdependency among variables is useful information to exploit. Each variable in MTS depends on its historical values and other variables. For example, the activity of sun shows periodic pattern in historic observations. The future traffic flow of a certain street is easier to be predicted by introducing the traffic information of neighboring areas, while the impact from the area farther away is relatively slight. If such priori relation information can be considered, it is more conducive to the interaction among variables with dependency which can be an effective guideline for forecasting. Apart from these priori interdependency information which is available and helpful for MTS forecasting, there also exits relations among variables that are unknown or changing over time, implicitly exhibited. However, existing methods cannot exploit latent and rich interdependencies among variables efficiently and effectively.

Over the years, researchers have adopted different techniques and assumptions to model MTS. Many machine learning methods have been applied for MTS forecasting. Autoregressive integrated moving average model (ARIMA) is a popular machine learning model which can be applied flexibly to various types of time series with a high computational cost. VAR extends the autoregressive (AR) model to multivariate time series, thus it cannot integrate the relations among time series in the model. Classical MTS forecasting models consider statistic information of historical measurement and make the prediction. When deep learning methods are adopted for MTS forecasting, recurrent neural networks are often used. Due to the lack of ability to model long-term time series, GRU cell and highway structure are applied, boosting the performance of RNN model. Further deep learning model LSTNet and MLCNN, both consider the long-term dependency and short-term variance of time

*Equal contribution with order determined by rolling the dice.
†Correspondence to Haoyan Xu and Xiaoqian Wang.

series, but they take the assumption that the time series variables have the same effect to each other, thus they cannot model the pairwise dependencies among variables explicitly. Recently, researches found it promising to model multivariate time series using graph neural networks (Xu et al. 2020; Wu et al. 2020), which is a novel and effective way as the authors suggest. In graph theory, a graph is a structure containing a set of objects in which objects may in some way "related". Objects are called *vertices* or *nodes* and the relations between nodes are called *edges*. Time series variables can be considered as nodes while the interrelations among them as edges. The information of MTS is stored in this graph structure and is then processed by the following graph neural networks. Graphs with featured nodes and weighted edges contain rich structural information, thus modeling MTS as graph is promising and reasoning. However, TEGNN, MT-GNN can only reveal one type of relation, lacking the ability to model both static and dynamic relations in time series. The classical machine learning methods and mentioned deep learning methods can not fully explore the abundant implicit relations among time series.

In this work, a novel framework, termed ***M***ultivariate ***T***ime Series Forecasting with ***Het***erogeneous ***G***raph ***N***eural ***N***etwork (MTHetGNN) is proposed and applied for MTS forecasting tasks. MTHetGNN embeds each relation/interdependency into each graph structure and fuses all graph structures with temporal features. Relation embedding module considers both static, prior and dynamic, latent relations among variables, characterizing the global relations (such as similarity and causality) and local dynamic relations among time series respectively. In addition, convolution neural networks (CNN) are used for temporal feature extraction. Finally, heterogeneous graph neural networks take the output of the former modules and tackle the complex structural embedding of graph structure generated by MTS for forecasting task. Thus our major contributions are:

- We first propose a heterogeneous graph network based framework that are compatible to take full advantage of rich relations among variables of MTS.
- We construct a relation embedding module to explore the relations among time series in both dynamic and static approaches.
- We conduct extensive experiments on MTS benchmark datasets. The experimental results validate that the performance of the proposed method is better than state-of-the-art models.

Related Work

Graph Network Embedding

Nowadays, neural networks have been employed for representing graph structured data, such as social networks and knowledge bases. Extending the word2vec(Mikolov et al. 2013), an unsupervised algorithm DeepWalk(Perozzi, Al-Rfou, and Skiena 2014) is designed to learn representations of nodes in graph based on random walk. Unsupervised network embedding algorithms LINE (Tang et al. 2015) and node2vec (Grover and Leskovec 2016) are popular after that. Apart from that, Convolution Neural Network and Recurrent Neural Network are also defined and employed on graph data. Originated from Graph Signal Processing (Ortega et al. 2018), classical convolutions are extended to spectral domain, which is space and time consuming. Further research (Defferrard, Bresson, and Vandergheynst 2016) approximate the spectral convolution using K-hops polynomials, reducing the time complexity effectively. Finally, GCN (Kipf and Welling 2016), a scalable approach chose the convolutional architecture via a localized approximation with Chebyshef Polynomial, which is an efficient variant and can operate on graphs directly. However, these methods can only implement on undirected graphs. Previously in form of recurrent neural networks, Graph Neural Networks (GNNs) are proposed to operate on directed graphs.

Heterogeneous Network Embedding

Conventional methods for dealing with heterogeneous networks usually start with the extraction of typed structural features, aiming to pursue meaningful vector representations for each node for downstream applications. However, this task needs to consider structural information composed of multiple types of nodes and edges, which is challenging. Many methods of dealing with heterogeneous networks involve the concept of *meta-structure* (Dong et al. 2020). For example, metapath2vec (Dong, Chawla, and Swami 2017) uses a path composed of nodes obtained from random walks guided by metapaths, and considers heterogeneous semantic information to model the context of nodes. HAN (Wang et al. 2019) uses metapath to model higher-order similarities (not directly using first-order neighbors), and uses the attention mechanism to learn different weights for different neighbors. Similarly, HetGNN (Zhang et al. 2019) uses LSTM to aggregate node neighbors under a certain relationship and update the node representations.

Preliminary

Task Formulation

In this work, we explore the task of multivariate time series forecasting. Formally, given a time series $X_i = \{x_{i1}, x_{i2}, ..., x_{iT}\}$, where $x_{it} \in R^n$ is the observation with n variables at time stamp t from the i^{th} sample. T is the number of time stamps contained in a sample. The task is to predict the future value x_{t+h} where h denotes the horizon ahead of the current time stamp. Considering all samples $\mathcal{X} = \{X_1, X_2, ..., X_s\}$ where s is the number of samples, and the ground truth forecasting value $\mathcal{Y} = \{Y_1, Y_2, ..., Y_s\}$, we aims to model the mapping from $\mathcal{X}$ to $\mathcal{Y}$ using objective function.

Heterogeneous Graph network

Formally, a heterogeneous graph network is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The heterogeneous network information are represented with type mapping function $\tau : V \to T_v$ and $\phi(e) : E \to T_e$ (Dong et al. 2020). T_v and T_e denote the sets of vertex types and edge types, where $|T_v| + |T_e| > 2$. When modeling multivariate time series as a graph G, the i^{th} time series is regarded as node V_i in G belonging to the same node type T_v (Xu et al. 2020). We aim to use heterogeneous graph neural networks to embed static relation T_e^s and dynamic relation T_e^d shown between time series. Thus the multivariate time series could be represented as $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{|T_e|}\}$ from a graph perspective considering heterogeneity where $\mathcal{G}_i$ denotes graph with type T_v^i.

The Framework

Model Architecture

We first narrate our proposed model MTHetGNN, i.e., *M*ultivariate *T*ime Series Forecasting via *Het*erogenous *G*raph *N*eural *N*etworks in detail, which is a framework for modeling multivariate time series from a graph perspective with compatible modules. An overview of MTHetGNN is illustrated in Figure 1. MTHetGNN contains three components: *Temporal Embedding Module*, *Relation Embedding Module* and *Heterogeneous Graph Embedding Module*. To capture temporal features from time series, we adopt CNN with multi receptive fields in temporal embedding module, which could be replaced by methods like RNN and its variants. The relation embedding module captures different internal static and dynamic relations among variables in MTS. Taking the above two modules' output, heterogeneous graph embedding models can exploit rich spatial dependencies in graph structures to model heterogeneity in time series. The three modules jointly learn in an end-to-end fashion to exploit and fuse priori information, dynamic latent relations and temporal features. The following sections detail the three modules we design.

Temporal Embedding Module

This module aims to capture temporal features by applying multiple CNN filters. As shown in Figure 1, CNN filters with different receptive fields are applied on multivariate time series, thus features under different periods are extracted from raw signals.

Time series may have multiple meaningful periods with possible temporal patterns. For example, similar trends of traffic flow in a certain district are observed through week days. This kind of temporal patterns are unknown when designing the network structure. Thus the strategy of using filters with multiple sizes from convolution neural networks are applied. The CNN kernel sizes should be carefully chosen to capture fine-grained features shown in different temporal periods. Follow the concept of *inception* suggested in (Szegedy et al. 2015), we use p CNN filters with kernel sizes $(1 \times k_i)(i = 1, 2, 3, ..., p)$ to scan through input time series x to capture features at multiple time scales. Here set of convolution filters with kernel sizes of $[1 \times 3, 1 \times 5, 1 \times 7]$ are used to capture features at multiple time scales.

Relation Embedding Module

The relation embedding module learns graph adjacency matrix to model the internal relations among time series.

We model implicit relations in MTS variables using both static and dynamic strategies. From the static perspective, we use correlation coefficient and transfer entropy to model static linear relationships and implicit causality among variables. From the dynamic perspective, we adopt dynamic graph learning concept and learn the graph structure adaptively, modeling time-varying graph structure. By using the above three strategies, varies adjacency matrices are generated and then fed into heterogeneous graph neural networks to interpret the relations between graph nodes in both static and dynamic way.

Recall that for all samples $\mathcal{X} = \{X_1, X_2, ..., X_s\}$, the similarity adjacency matrix $A^{sim} \in \mathcal{R}^{n \times n}$ is generated by:

$$A^{sim} = Similarity(\mathcal{X}), \tag{1}$$

where $Similarity$ is a distance metric which measures the pairwise similarity scores between time series. Existed work to measure distance include *Euclidean Distance*, *Landmark Similarity* and *Dynamic Time Warping (DTW)*, etc. Here we adopt *correlation* and *coefficient* to measure the global correlation among time series, offering a priori knowledge of overal linear relation. Thus element in A^{sim} is generated by:

$$A_{ij}^{sim} = \frac{Cov(X_i, X_j)}{\sqrt{D(X_i)}\sqrt{D(X_j)}}, \tag{2}$$

where $Cov(X_i, X_j)$ is the covariance between X_i and X_j, $D(X_i)$ and $D(X_j)$ is the variance of time series X_i and X_j respectively.

The causality adjacency matrix $A^{cas} \in \mathcal{R}^{n \times n}$ is generated by:

$$A^{cas} = Causality(\mathcal{X}), \tag{3}$$

where $Causality$ is a metric to measure causality between time series. Various efforts have been made to measure causal inference among variables, such as Grager, etc. Here we use Transfer Entropy (TE) to process non-stationary time series, a pseudo-regression relationship will be produced in which pairwise time series be considered causal if they have an overall trend caused by common factors. The causality mentioned here is not strict, but the value is helpful for predicting. Given graph variables X and Y, the transfer entropy of variable A to B is defined as:

$$T_{B \to A} = H(A_{t+1}|A_t) - H(A_{t+1}|A_t, B_t), \tag{4}$$

in which the concept of conditional entropy is used. Let m_t represent the value of variable m at time t, and $m_t^{(k)} = [m_t, m_{t-1}, ..., m_{t-k+1}]$. $H(M_{t+1}|M_t)$ is the conditional entropy representing the information amount of M_{t+1} under the condition that the variable M_t is known:

$$H(M_{t+1}|M_t) = \sum p\left(m_{t+1}, m_t^{(k)}\right) \log_2 p\left(m_{t+1}|m_t^{(k)}\right). \tag{5}$$

By calculate transfer entropy between time series, the element in causality adjacency matrix A^{cas} is calculated as:

$$A_{ij}^{cas} = T_{X_i \to X_j} - T_{X_j \to X_i}. \tag{6}$$

The third strategy adopts the concept of dynamic evolving networks(Skarding, Gabrys, and Musial 2020). In a certain period of time, the time series are persist to establish a relatively stable graph network, and node properties like

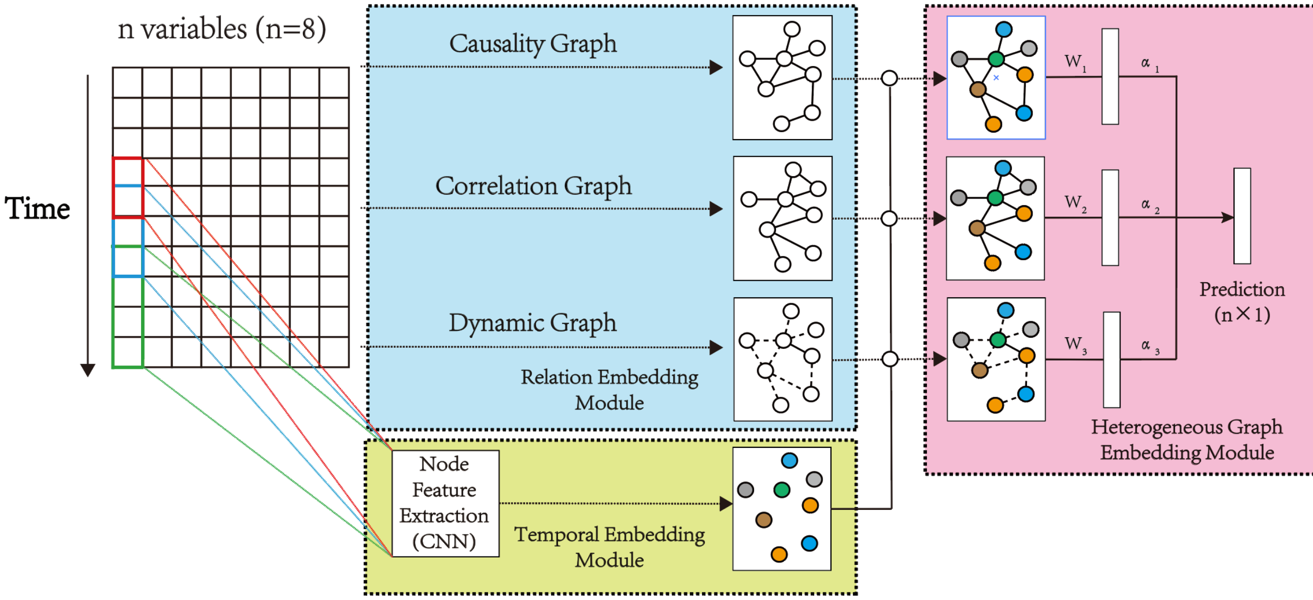

Figure 1: The general architecture of our model MTHetGNN. MTHetGNN contains three modules and they jointly learn in an end-to-end fashion. Temporal embedding module captures temporal features as node features. Relation embedding module captures static, prior and dynamic, latent spatial relations among variables. Heterogeneous graph embedding module exploits and fuses rich spatial patterns with temporal features for better forecasting.

node degree can be updated in traning process. We propose a dynamic relation embedding strategy, which learns the adjacency matrix $A^{DA} \in \mathcal{R}^{n\times n}$ adaptively to model latent relations in the given time series sample X_i, denoted as:

$$A^{dy} = Evolve(X_i). \tag{7}$$

Given the input time series $X_k = x_1, x_2, ..., x_n \in R^{n\times T}$ from the k_{th} sample with length T, where x_i, x_j denote the i^{th}, j^{th} time series. We first calculate the distance matrix D between sampled time series:

$$D_{ij} = \frac{exp(-\sigma(distance(x_i, x_j)))}{\sum_{p=0}^{n}\sigma(exp(-\sigma(distance(x_i, x_p)))}, \tag{8}$$

,where $distance$ denotes the distance metric. Then the dynamic adjacency matrix $A^{dy} \in \mathcal{R}^{n\times n}$ can be calculated as:

$$A^{dy} = \sigma(DW), \tag{9}$$

where W is learnable model parameters, σ is an activation function.

Normalization is applied to the output of each strategy respectively to form three normalized adjacency matrix. What's more, to improve training efficiency, reduce the effect of noise, amplify the effective relations and make the model more robust, threshold is set to make all the adjacency matrices sparse:

$$A^r_{ij} = \begin{cases} A^r_{ij} & A^r_{ij} >= threshold \\ 0 & A^r_{ij} < threshold \end{cases} \tag{10}$$

Heterogeneous Graph Embedding Module

This module could be viewed as a graph based aggregation method, which fuses temporal features and spatial relations between time series to get forecasting results.

Our model is primarily motivated by rGCNs (Schlichtkrull et al. 2018) which learns an aggregation function that is representation-invariant and can operate on large-scale relational data. We adopt the idea of fusing node embeddings of each heterogeneous graph with attention mechanism. We propose the following propagation function:

$$H^{(l+1)} = \sigma\left(H^{(l)}W_0^l + \sum_{r\in\mathcal{R}} softmax(\alpha_r)\hat{A}_r H^{(l)} W_r^{(l)}\right) \tag{11}$$

where $H^{(l)}$ is the matrix of node embedding in the l^{th} layer, $H^{(0)} = X$; $\alpha_{(r)}$ is the weight coefficient of each heterogeneous graph, and $softmax(\alpha_r) = \frac{exp(\alpha_r)}{\sum_{i=1}^{|R|} exp(\alpha_i)}$. $W_o^{(l)}$ and $W_r^{(l)}$ are layer-specific weight matrix. σ is a nonlinear activation function, usually being $Relu$.

We use attention mechanism to draw global dependencies between input time series and output forecasting results. Additive attention and dot-product attention are two commonly used attention functions. Additive attention function uses a feedforward neural network with a hidden layer. The input layer is a horizontal splicing of two vectors, and the activation function of the output layer is sigmoid to indicate the correlation between the two vectors. Matrix operations are supported in dot-product attention, which makes the calculation faster and space saving. Inspired by scaled dot-product attention mechanism (Vaswani et al. 2017), we can extend the use of the dot-product operation to compute the attention coefficients between heterogeneous graph neural network.

Noting that the aggregation function of graph structure used in this module is identical to GCN (Kipf and Welling 2016), which uses the 1^{th} order Chebyshev polynomials to approximation of spectral convolution, assuming only the direct neighbor of a node is reachable at each layer. (As GCN faces oversmoothing problem, the information from farther neighbor nodes may not be sufficiently integrated.) Other form of graph information propagation methods could be considered to replace the GCN layer. For example, GAT (Vaswani et al. 2017) layer considers the importance of different neighbor nodes with attention mechanism. MixHop (Abu-El-Haija et al. 2019) layer fuses information from different hops of localized spectral filters, keeping a balance between local and neighborhood information. We leave this for future work.

Objective Function

Objective function using $\mathcal{L}_2$ loss is used in many forecasting tasks:

$$minimize(\mathcal{L}_2) = \frac{1}{k}\sum_{i}^{k}\sum_{j}^{n}(y_{ij} - \hat{y}_{ij})^2, \qquad (12)$$

where k is the training size and n is the variables in time series. $\hat{y}$ is the prediction and y is the ground truth.

And researchers have found that objective function using $\mathcal{L}_1$ loss has a stable gradient for different inputs, which can reduce the impact of outliers while avoiding gradient explosions:

$$minimize(\mathcal{L}_1) = \frac{1}{k}\sum_{i}^{k}\sum_{j}^{n}|y_{ij} - \hat{y}_{ij}| \qquad (13)$$

We use the Adam optimizer in our traning, and decide which objective function to use by the performance shown on validation set.

Experiments

We conduct experiments on MTHetGNN model on 3 benchmark datasets and compare the performance of MTHetGNN with 6 baseline methods for multivariate time series forecasting tasks. Our codes will be available online soon[1].

Data

We use three benchmark datasets which are commonly used in multivariate time series forecasting. Datasets information are details as following:

- Traffic[2]: The traffic highway occupancy rates measured by 862 sensors in San Francisco Bay Area from 2015 to 2016 by California Department of Transportation.
- Solar-Energy[3]: Continuous collected Solar energy data from the National Renewable Energy Laboratory, which contains the solar energy output collected from 137 photovoltaic power plants in Alabama in 2007.
- Exchange-Rate: The exchange rate data containing the daily exchange rates from eight countries, including UK, Japan, New Zealand, Canada, Switzerland, Singapore, Australia and China, ranging from 1990 to 2016.

Methods for Comparison

We evaluate the performance of MTHetGNN model with other five baseline models on multivariate time series forecasting task. The overview of baseline methods are summarized as bellow:

- VAR-MLP: A machine learning model mentioned in (Zhang 2003) which is the combination of Multilayer Perception(MLP) and Autoregressive model.
- RNN-GRU: A Recurrent Neural Network adopting GRU cell (Dey and Salemt 2017).
- LSTNet (Lai et al. 2018): A deep learning method, which uses Convolution Neural Network and Recurrent Neural Network to discover both short and long term patterns for time series.
- MLCNN (Cheng, Huang, and Zheng 2019): A deep neural network which fuses forecasting information of different future time.
- MTGNN (Wu et al. 2020): A graph neural network designed for multivariate time series forecasting.
- TEGNN (Xu et al. 2020): A novel deep learning framework to tackle forecasting problem of graph structure generated by multivariate time series considering causal relevancy.

Metrics

The following conventional evaluation metrics are used to evaluate all methods: *Relative Squared Error (rse)*, *Relative Absolute Error (rae)*, and *Empirical Correlation Coefficient (CORR)*. For *rse* and *rae*, lower value is better, while for *corr* higher is better.

Experimental Details

On three bench mark datasets, data are split into training set, validation set and testing set in a ratio of $6:2:2$, then we use the model with the best performance based on *rse*, *rae* and *corr* metrics on validation set for testing. We conduct grid search strategy over adjustable hyperparameters for all methods. The windowsize T for all methods are the same under the same setting. For LSTNet, the hidden CNN and RNN layer is chosen from $\{20, 50, 100, 200\}$, the length of recurrent-skip is set to 24. For MTGNN, the mixhop propagation depth is set to 2, the activation saturation rate of graph learning layer is set to 3. For RNN, the hidden RNN layer is chosen from $\{10, 20, 50, 100\}$, the dropout

[1]Our codes will be released as well upon the acceptance of this paper.

[2]http://pems.dot.ca.gov

[3]http://www.nrel.gov/grid/solar-power-data.html

rate is chosen from $\{0.1, 0.2, 0.3\}$. For TEGNN and MTHetGNN, the hidden graph convolutional networks is chosen from $\{5, 10, 15, ..., 100\}$. The Adam algorithm is used to optimize the parameters for MTHetGNN. More detailed parameter settings are illustrated in our code.

Effectiveness

Table 1 summarizes the evaluation results of MTHetGNN and other baseline methods on three benchmark datasets under different settings. Following the settings of LSTNet (Lai et al. 2018), we test the model performance on forecasting future values $\{X_{t+3}, X_{t+6}, X_{t+12}, X_{t+24}\}$, which means future value from 3 to 24 days over the Exchange-Rate data, 30 to 240 minutes over the Solar-Energy data, and 3 to 24 hours over the Traffic data, s thus $horizon$ are set to $\{3, 6, 12, 24\}$ for three benchmark datasets respectively. As shown in table 1, the best results under 4 different $horizon$ with 3 evaluation metrics are set bold, among which MTHetGNN has records set bold.

Clearly, the proposed MTHetGNN model consistently shows state-of-the-art results on all datasets. TEGNN, MTGNN and MTHetGNN use graph structure to model time series, the strong representing ability of graph neural networks make these three models behave better than other baseline methods to some extent. It is noteworthy that MTHetGNN outperforms the strong graph-based baseline TEGNN, especially on datasets containing plenty variables, indicating the strong information aggregation capabilities heterogeneous graph networks shows under the same neural network depth. This is partly because TEGNN model captures the causality of multivariate time series while MTHetGNN focus on heterogeneity. Measuring transfer entropy on the whole time series makes the measurement of causality more accurate. However, macroscopic observations will filter out the fluctuations in a single time series segment, thus transfer entropy matrix cannot fully represent the relationship between variables in different time segments. Considering this, MTHetGNN not only takes the static relations among time series into account, but also considers the dynamic correlations shown in a shorter time segment, fully exploiting the heterogeneity of time series. Detailed analysis are shown in following sections.

Efficiency

MTHetGNN has three graph embedding paths, which increases the complexity of the model to a certain extent. But the node feature matrix generated by temporal embedding module is shared by all the three paths. And the adjacency matrix A^{TE} and A^{CO} can be calculated in the offline modeling stage, which has little effect on the model complexity. In order to verify the time complexity of the MTHetGNN model, we record the testing time of the MTHetGNN model and other methods on the Exchange-Rate dataset. As shown in Figure 2, the MTHetGNN model mines time series multi relations while having relatively high computing efficiency.

Ablation Study

In this subsection, we conduct ablation studies on Exchange-Rate dataset to understand the contributions of heterogeneous graph network in MTHetGNN model. There are two main types of settings, type1, 2, 3 removes the heterogeneous graph part and use one relation extracting way respectively, type4 replaces the attention part with the average operation. The detailed setting of each variant model is as followed:

- type1: Only the Transfer Entropy matrix is used to integrate neighbor information in each layer.
- type2: Only the Correlation Coefficient matrix is used to integrate neighbor information in each layer.
- type3: Only the Dynamic matrix is used to integrate neighbor information in each layer.
- type4: The MTHetGNN model without attention component, in which the adjacency matrix obtained by three strategies are averaged to a single matrix.

The results are shown in figure 3. We notice that MTHet-

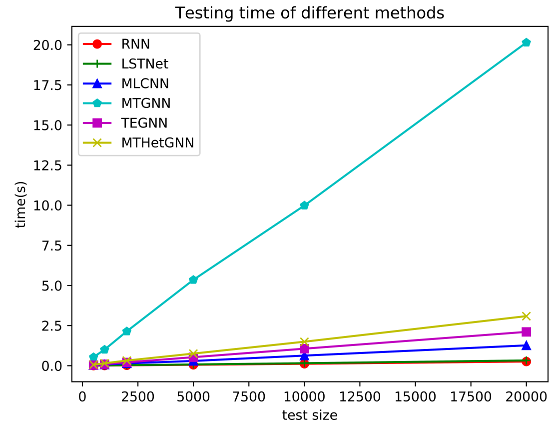

Figure 2: Running time for testing process for all methods on Exchange-Rate dataset when horizon is 3.

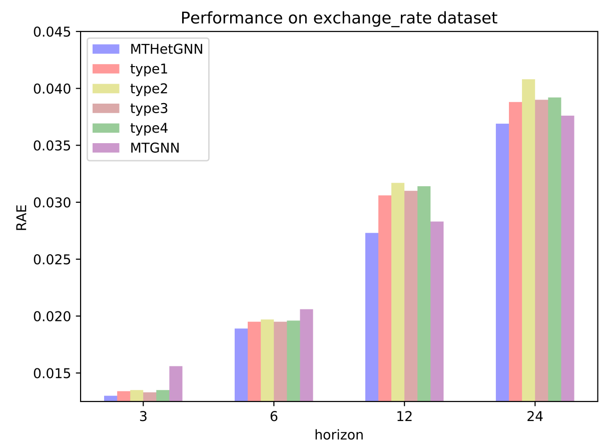

Figure 3: Performance of MTHetGNN and four variants on Exchange-Rate dataset after training 100 epochs. The experiment settings are the same for these methods.

Table 1: MTS forecasting results measured by RSE/RAE/CORR score over three datasets.

Dataset		Exchange-Rate				Solar				Traffic			
Methods	Metrics	horizon 3	horizon 6	horizon 12	horizon 24	horizon 3	horizon 6	horizon 12	horizon 24	horizon 3	horizon 6	horizon 12	horizon 24
	RSE	0.0186	0.0262	0.0370	0.0505	0.1932	0.2721	0.4307	0.8216	0.5513	0.6155	0.6240	0.6107
	RAE	0.0141	0.0208	0.0299	0.0427	0.0995	0.1484	0.2372	0.4810	0.3909	0.4066	0.4177	0.4032
VAR	CORR	0.9674	0.9590	0.9407	0.9085	0.9819	0.9544	0.9010	0.7723	0.8213	0.7826	0.7750	0.7858
	RSE	0.0200	0.0262	0.0366	0.0527	0.1909	0.2686	0.4270	0.4938	0.5200	0.5201	0.5320	0.5428
	RAE	0.0157	0.0209	0.0298	0.0442	0.0946	0.1432	0.2302	0.2849	0.3625	0.3708	0.3669	0.3844
RNN-GRU	CORR	0.9772	0.9688	0.9534	0.9272	0.9832	0.9660	0.9112	0.8808	0.8436	0.8459	0.8316	0.8232
	RSE	0.0216	0.0277	0.0359	0.0482	0.1940	0.2755	0.4332	0.4901	0.4769	0.4890	0.5110	0.5037
	RAE	0.0171	0.0226	0.0295	0.0404	0.0999	0.1510	0.2413	0.2997	0.3161	0.3291	0.3435	0.3441
LSTNET	CORR	0.9749	0.9678	0.9534	0.9353	0.9825	0.9633	0.9065	0.8673	0.8730	0.8657	0.8534	0.8537
	RSE	0.0172	0.0449	0.0519	0.0438	0.1794	0.2983	0.3673	0.5191	0.4924	0.4992	0.5214	0.5353
	RAE	0.0129	0.0334	0.0422	0.0375	0.0844	0.1342	0.1873	0.3131	0.3376	0.3243	0.3766	0.3825
MLCNN	CORR	0.9780	0.9610	0.9550	0.9407	0.9814	0.9642	0.9210	0.8513	0.8629	0.8416	0.8320	0.8255
	RSE	0.0194	0.0253	0.0345	0.0447	0.1767	0.2342	0.3088	0.4352	0.4178	0.4774	0.4461	0.4535
	RAE	0.0156	0.0206	0.0283	0.0376	0.0837	0.1171	0.1627	0.2563	0.2435	0.2670	0.2739	0.2651
MTGNN	CORR	0.9782	0.9711	0.9564	0.9370	0.9852	0.9727	0.9511	0.8931	0.8960	0.8665	0.8794	0.8810
	RSE	0.0178	0.0245	0.0363	0.0449	0.1824	0.2612	0.3289	0.4733	0.4421	0.4433	0.4508	0.4692
	RAE	0.0135	0.0195	0.0306	0.0388	0.0851	0.1312	0.1766	0.2821	0.2651	0.2616	0.2740	0.2855
TEGNN	CORR	0.9815	0.9732	0.9566	0.9352	0.9847	0.9676	0.9379	0.8854	0.8853	0.8820	0.8743	0.8671
	RSE	**0.0173**	**0.0238**	**0.0327**	**0.0430**	**0.1668**	**0.2175**	**0.2872**	**0.3862**	**0.4142**	**0.4303**	**0.4376**	**0.4500**
	RAE	**0.0132**	**0.0190**	**0.0266**	**0.0361**	**0.0788**	**0.1111**	**0.1514**	**0.2217**	**0.2349**	**0.2490**	**0.2592**	**0.2661**
MTHetGNN	CORR	**0.9824**	**0.9746**	**0.9604**	**0.9415**	**0.9872**	**0.9772**	**0.9583**	**0.9210**	**0.8975**	**0.8887**	**0.8828**	**0.8776**

GNN can model the time series trend more precisely than each variant model, which indicates the effectiveness of both heterogeneous network embedding and attention mechanism in modeling MTS. As is shown, using heterogeneous graph instead of each relation graph raises the *rse* metric of $5.1\%, 7.2\%, 4.5\%$ respectively for type1, 2, 3. It is not surprising given the motivation of using heterogeneous graph. Type1, 2, 3 each only considers relation of the time series in one perspective, while MTHetGNN adopts the concept of heterogeneous graph and fuses the relations in both static and dynamic way. The difference in results between MTHetGNN and type4 indicates the effectiveness of attention mechanism as mentioned in the former section. By adopting attention mechanism, MTHetGNN can treat each relation graph with different weight, in accordance with the importance of each type of relation altered in training process.

Meanwhile, we change the network parameters of the heterogeneous graph network and test the performance of the MTHetGNN model under different parameter settings on Exchange-rate dataset. Figure 4 shows that the MTHetGNN model does not rely on specific parameters, being relatively not sensitive to parameter changes, showing its effectiveness and stability.

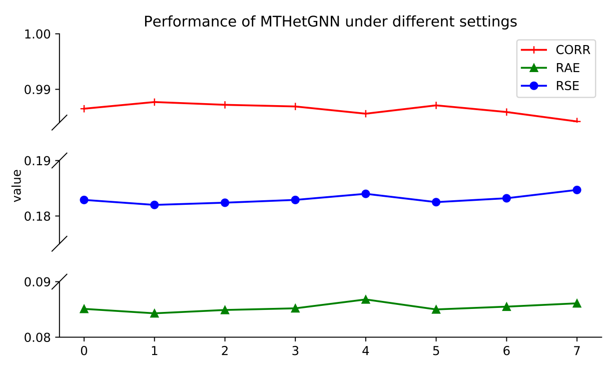

Figure 4: Performance of MTHetGNN under different settings of heterogeneous graph neural networks, the hidden size of GNN layers is varying while other hyperparameters remain the same.

Conclusion

In this paper, we propose a novel heterogeneous graph embedding network based framework(MTHetGNN) for multivariate time series forecasting. MTHetGNN can exploit and fuse rich spatial relation information and temporal features generated by MTS. Experiments on three real-world datasets show that our model outperforms 6 baselines in terms of three metrics.

There are several directions to go for the future work: (1) It is promising to explore more effective dynamic graph embedding updating strategies to learn complex latent spatial dependencies among variables; (2) Heterogeneous graph neural models with stronger representation ability can be incorporated into our framework to characterize rich spatial relations and temporal features, so as to improve prediction accuracy and make MTHetGNN more robust; (3) Our method and other current deep learning methods can only model spatial and temporal patterns within MTS slices (window size) and cannot model the relation between MTS Slices. It's interesting to explore whether the idea of dynamic graph recurrent neural networks can be adopted in MTS forecasting.

References

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Steeg, G. V.; and Galstyan, A. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing.

Cheng, J.; Huang, K.; and Zheng, Z. 2019. Towards Better Forecasting by Fusing Near and Distant Future Visions.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, 3844–3852.

Dey, R.; and Salemt, F. M. 2017. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, 1597–1600. IEEE.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 135–144.

Dong, Y.; Hu, Z.; Wang, K.; Sun, Y.; and Tang, J. 2020. Heterogeneous Network Representation Learning. IJCAI.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Ortega, A.; Frossard, P.; Kovačević, J.; Moura, J. M.; and Vandergheynst, P. 2018. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE* 106(5): 808–828.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Skarding, J.; Gabrys, B.; and Musial, K. 2020. Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey. *arXiv preprint arXiv:2005.07496* .

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 1067–1077.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*, 2022–2032.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks.

Xu, H.; Huang, Y.; Duan, Z.; Wang, X.; Feng, J.; and Song, P. 2020. Multivariate Time Series Forecasting with Transfer Entropy Graph.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 793–803.

Zhang, G. P. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50: 159–175.