Multiple Time Series Forecasting with Dynamic Graph Modeling

Kai Zhao
Aalborg University
kaiz@cs.aau.dk

Chenjuan Guo*
East China Normal University
cjguo@dase.ecnu.edu.cn

Yunyao Cheng
Aalborg University
yunyaoc@cs.aau.dk

Peng Han
UESTC
penghan_study@foxmail.com

Miao Zhang
HIT Shenzhen, Aalborg University
zhangmiao@hit.edu.cn

Bin Yang
East China Normal University
byang@dase.ecnu.edu.cn

ABSTRACT

Multiple time series forecasting plays an essential role in many applications. Solutions based on graph neural network (GNN) that deliver state-of-the-art forecasting performance use the relation graph which can capture historical correlations among time series. However, in real world, it is common that correlations among time series evolve across time, resulting in dynamic relation graph, where the future correlations may be different from those in history. To address this problem, we propose multiple time series forecasting with dynamic graph modeling (MTSF-DG) that is able to learn historical relation graphs and predicting future relation graphs to capture the dynamic correlations. We also propose a causal GNN to extract features from both kinds of relation graphs efficiently. Then we propose a reasoning network to explicitly learn the variant influence from historical timestamps to future timestamps for final forecasting. Extensive experiments on six benchmark datasets show that MTSF-DG consistently outperforms state-of-the-art baselines, and justify our design with dynamic relation graph modeling.

PVLDB Reference Format:
Kai Zhao, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, Bin Yang. Multiple Time Series Forecasting with Dynamic Graph Modeling. PVLDB, 17(4): 753 - 765, 2023.
doi:10.14778/3636218.3636230

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zhkai/MTSF-DG.

1 INTRODUCTION

Many real-world data sensed from cyber-physical systems (CPS) can be modeled as the recordings of time-dependent observations, which form the multiple time series [10, 27, 41]. For example, in power grid there exist multiple electric time series which record the energy consumption of different clients [11], and in transport network there exist multiple traffic time series which record the traffic flows and speeds at different locations across time [5, 26]. Based on the historical observations, forecasting the future observations plays an important role in ensuring effective functionality of CPS for various applications, such as spotting patterns [2, 15, 17, 18] and predicting the future behaviors [3, 29, 44]. Thus, we aim at multiple time series forecasting problem in this paper.

Early methods [20, 34, 48] forecast the future observations by applying machine learning models on the historical observations, which aim to learn the *temporal dependencies*. Some works [23, 43] study time series forecasting in the traffic domain with different kinds of Graph Neural Network (GNN) [14, 19, 51] by learning the correlations among the time series of different locations based on their spatial distance. More recently, a new trend is to employ graph learning [42] to learn a *relation graph* that models correlations among multiple time series without requiring spatial distance to enable time series forecasting. For example, DGSL [31] constructs a relation graph, where time series are considered as nodes, and two time series are connected by an edge if their observations are similar in history.

Figure 1(a) illustrates three time series that record traffic flows in three districts, *e.g.*, a commercial district $\mathbf{X}_1$, a residential district $\mathbf{X}_2$ and an industrial district $\mathbf{X}_3$. On the peak hours of workdays $\mathbf{t}_2$~ $\mathbf{t}_3$, the traffic flows may be more correlated between $\mathbf{X}_2$ and $\mathbf{X}_3$. While on the weekends $\mathbf{t}_6$, the traffic flows may be more correlated between $\mathbf{X}_1$ and $\mathbf{X}_2$. Therefore, multiple historical relation graphs, *e.g.*, G_1, G_2, and G_3, are required to model the dynamic correlations among time series. Similarly, correlations in a future period $\mathbf{t}_f$ are different from those in history, and could be modeled by a future relation graph G_f, as shown in Figure 1(b).

The information of dynamic relation graphs can help multiple time series forecasting, as shown in Table 1 where two traffic datasets [23] are used. Specifically, DCRNN [23] has a static relation graph which is constructed by the distance among locations. AGCRN [1] learns a static relation graph from historical time series data. However, both studies use the same, static relation graph for forecasting at different timestamps. DCRNN-D and AGCRN-D use dynamic relation graphs, which varies across time [13].

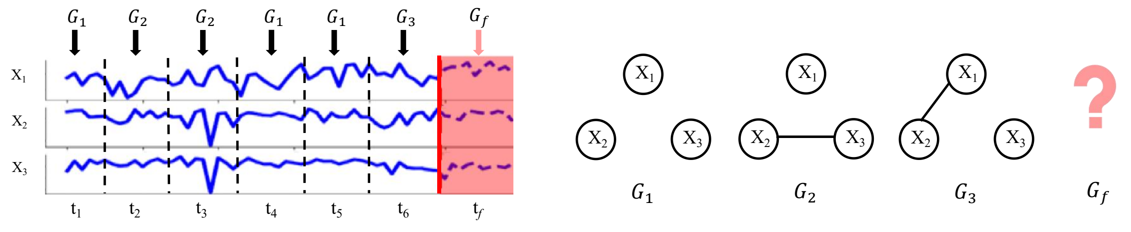
Although the state-of-the-art approaches [42, 45] can learn the relation graph to capture the correlations among multiple time series, they could only use the historical correlations. As the correlations change across time, the single, historical relation graph is insufficient to model the dynamic correlations and hinders the improvement of multiple time series forecasting. However, it is

Table 1: The influence of dynamic relation graphs

Dataset	Metric	DCRNN	DCRNN-D	AGCRN	AGCRN-D
METR-LA	MAE	3.60	**3.55**	3.58	**3.51**
PEMS04		24.70	**21.03**	19.83	**19.51**

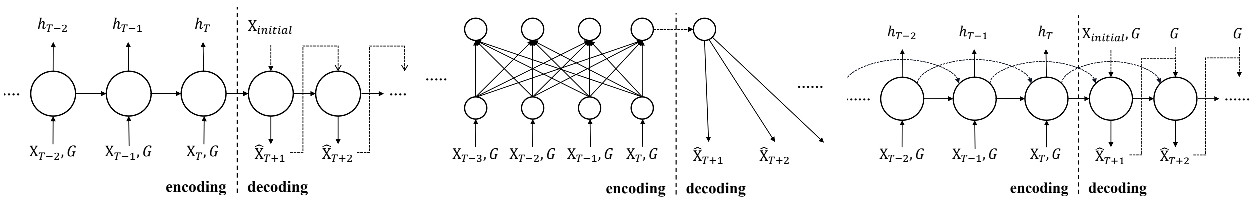
*: Corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 4 ISSN 2150-8097.
doi:10.14778/3636218.3636230



(a) An example of time series for traffic flows at three locations (b) The dynamic relation graphs

Figure 1: Motivations



h_{T-2} h_{T-1} h_T $X_{initial}$
X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding decoding
X_{T-3}, G X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding decoding
h_{T-2} h_{T-1} h_T $X_{initial}, G$ G G
X_{T-2}, G X_{T-1}, G X_T, G $\hat{X}_{T+1}$ $\hat{X}_{T+2}$
encoding decoding

(a) RNN based model (b) Transformer based model (c) Our reasoning network

Figure 2: Model comparison

non-trivial to model the dynamic relation graphs, and to use historical observations to predict the future observations under the changeable relation graphs.

Challenge 1: It is challenging to learn the dynamic relation graphs and extract features from different relation graphs efficiently. Existing studies can construct a relation graph by learning the similarities among multiple time series from the known historical observations. It is possible for ESG [45] to cut the whole historical observations into sub time-windows, and learn a historical relation graph for each historical sub time-window using its historical observations. However, none of the existing methods have demonstrated the ability to learn a future relation graph for a future time-window, as the future observations are unknown. Besides, it is difficult to extract features from different relation graphs efficiently, as the GNN in these existing methods [8, 45] can only deal with one relation graph at one timestamp.

Challenge 2: It is challenging to predict the future observations under the changeable relation graphs. The existing methods use RNN or Transformer [7] based module to learn temporal dependencies. As shown in Figure 2(a) and 2(b), RNN based models only model the influence from one timestamp to the next timestamp explicitly, and Transformer based models only model the influence among historical timestamps. However, one historical timestamp may have different influence on future timestamps with regard to the changeable future relation graphs. The existing methods fail to model such different temporal dependencies explicitly.

Based on the above analysis, in this paper, we propose a novel approach MTSF-DG, multiple time series forecasting with dynamic graph modeling.

For **Challenge 1**, we cut each time series sample into historical and future time-windows, for each of which we build a relation graph distribution. It not only learns the historical relation graph distribution but also predicts the future relation graph distribution, by optimizing the correlation coefficients from an empirical covariance matrix using a memory network. Further, different from traditional GNN, which only models with one graph at a time, we propose a causal GNN, which models the observations with both historical and future relation graphs into a feature vector efficiently.

For **Challenge 2**, we propose a reasoning network with the logical operations and symbolic reasoning procedure to explicitly learn how historical timestamps influence future timestamps. First, we learn feature vectors, say h_{T-2}, h_{T-1} and h_T, as the representations for timestamps $T-2$, $T-1$ and T. Next as shown in Figure 2(c), We use a reasoning procedure, *e.g.,* $h_{T-2} \longrightarrow h_{T-1}$ and $(h_{T-2} \wedge h_{T-1}) \longrightarrow h_T$ being *TRUE* or *FALSE*, to achieve the cognition ability [4, 30] of how one historical timestamp $T-2$ may have different influence on future timestamps $T-1$ and T. In this way, we model the different temporal dependencies explicitly.

To the best of our knowledge, this is the first work that considers to predict future relation graphs, and use both historical and future relation graphs in multiple time series forecasting. And we summarize contributions as follows. First, we propose to model dynamic relation graphs and propose a causal GNN to model the observations with both historical and future relation graphs into feature vectors efficiently. Second, we propose a reasoning network to explicitly learn how historical timestamps have different influence on future timestamps. Third, by evaluating on six real-world benchmark datasets from different domains, we show that our model consistently outperforms the state-of-the-art baselines.

2 PRELIMINARIES

We formalize the multiple time series forecasting problem and introduce reasoning. The notations are summarized in Table 2.

Table 2: Notations

Notation	Explanation
G_t	A relation graph which presents the correlations among time series for timestamp t
v_t^i	The feature vector extracted from time series $\mathbf{X}_t^i$
τ	The max previous time steps used in reasoning
$\star_{G_t}$	The causal graph neural network using graph G_t
c_k	The learnable embedding matrix which denote the relative time interval with the k-th previous time step
h_t^i	The hidden state for i-th time series at timestamp t in the reasoning network

2.1 Problem Definition

Multiple Time Series Forecasting. The multiple time series is represented as $\mathbf{X} \in \mathbb{R}^{N \times L}$, where N is the number of time series and each time series has observations during total L timestamps. We use $\mathbf{X}_t \in \mathbb{R}^N$ to indicate the observations of all time series at timestamp t, use $\mathbf{X}_{t:t+t_\Delta} \in \mathbb{R}^{N \times t_\Delta}$ to indicate the observations of all time series from timestamp t to timestamp $t + t_\Delta$, use $\mathbf{X}^i_{t:t+t_\Delta} \in \mathbb{R}^{t_\Delta}$ to indicate the observations of the i-th time series from timestamp t to timestamp $t + t_\Delta$, and use $\mathbf{X}_t^i \in \mathbb{R}^1$ to indicate the observations of the i-th time series at timestamp t, where $1 \leq i \leq N$ and $1 \leq t, t + t_\Delta \leq L$.

We formulate multiple time series forecasting problem as follows. Given a sub-sequence of historical p timestamps of observations from the multiple time series, *i.e.,* $\mathbf{X}_{T-p+1:T}$, the goal is to predict the values for the q future timestamps, *i.e.,* $\mathbf{X}_{T+1:T+q}$, where $q \geq 1$. Thus, we formulate the multiple time series forecasting problem as finding a mapping function $\mathcal{F}$ as follows:

$$\hat{\mathbf{X}}_{T+1:T+q} = \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}), \tag{1}$$

where θ is the parameters of $\mathcal{F}$, and $\hat{\mathbf{X}}$ denotes the predicted values of multiple time series.

Relation Graph. The latent correlations among time series at timestamp t is represented as a relation graph $G_t = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and each node $TS_i \in \mathcal{V}$ denotes a time series so that $|\mathcal{V}| = N$, $\mathcal{E}$ is the set of edges and each edge $e_{i,j} \in \mathcal{E}$ denotes that time series i and time series j are correlated with each other, and $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. $\mathcal{A}_{ij} = 0$ if $e_{i,j} \notin \mathcal{E}$, $\mathcal{A}_{ij} \neq 0$ if $e_{i,j} \in \mathcal{E}$, and $\mathcal{A}_{ij}$ is the weight that denotes the degree of correlation between time series i and time series j. If $e_{i,j} \in \mathcal{E}$, node i and j are the first-hop neighbors for each other.

2.2 Reasoning

In this paper, we use *propositional logic*, which has basic operations, *i.e.,* conjunction (*AND*, $\wedge$), negation (*NOT*, $\neg$) and material implication ($\rightarrow$), to explicitly learn how historical timestamps have different influence on future timestamps. For time series forecasting: Each hidden state is a vector variable, such as h_t which represents the states of multiple time series at timestamp t, similar to the hidden state used in RNN. The operation over hidden states is called an *expression*, such as $(h_{t-1} \wedge h_t)$ which indicates that multiple time series have had the states h_{t-1} and h_t during two timestamps $t-1$ and t. When the expression has the material implication ($\rightarrow$) operation, it is called a *Horn clause*. The reasoning result of $(h_{t-2} \wedge h_{t-1}) \rightarrow h_t$ is *TRUE* can show that the historical states h_{t-2} and h_{t-1} have significant influence on the future state h_t; otherwise the reasoning result is *FALSE*, the historical states h_{t-2} and h_{t-1} have little influence on the future state h_t.

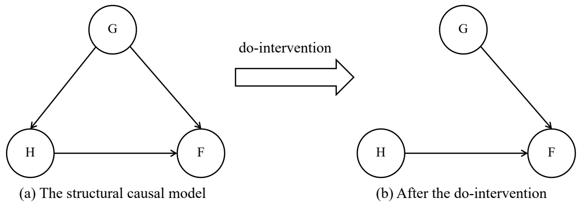

G
H
F
do-intervention
G
H
F
(a) The structural causal model
(b) After the do-intervention

Figure 3: The do-intervention for node H

3 THE FOUNDATION

In this section, we theoretically analyze the limitation of existing methods and point out the importance of predicting the future relation graph distribution for dynamic graph modeling.

Specifically, we use a random variable **H** to denote a sequence of historical observations happened in the past p timestamps. Random variable **F** denotes a sequence of future observations will happen in the future q timestamps. Random variable **G** denotes dynamic relation graphs at different timestamps. Then, we can model the relationship among **H**, **F**, and **G** using a structural causal model [25] as shown in Figure 3(a), where an arrow indicates that there exists some influence. For example, the historical observations **H** influence the future observations **F**. Meanwhile, the dynamic relation graphs **G** influence the historical observations **H** and also the future observations **F**.

Existing methods extract the features from historical observations to forecast future observations. Following the probability theory, their forecasting process is learning the likelihood of conditional probability $P(\mathbf{F}|\mathbf{H})$ as:

$$\hat{\mathbf{X}}_{T+1:T+q} = \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}) = \underset{\mathbf{F}}{argmax}\, P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}). \tag{2}$$

By applying the Bayes rule, we can see that the dynamic relation graphs will influence the forecasting results of these existing methods. By the Bayes rule we can get:

$$P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}) = \sum_{t \in [T-p+1, T+q]} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}), \tag{3}$$

where G_t is the possible relation graphs at different timestamps. As the historical observations $\mathbf{X}_{T-p+1:T}$ are influenced by the historical relation graphs G_t where $t \in [T-p+1, T]$, we have:

$$\begin{aligned} P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) \neq 0, \quad \forall t \in [T-p+1, T], \\ P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) = 0, \quad \forall t \notin [T-p+1, T]. \end{aligned} \tag{4}$$

Then we can get $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ as:

$$\sum_{t \in [T-p+1, T]} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}). \tag{5}$$

From Equation (5), we can see that $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ used in existing methods which only learn from the historical relation graphs could lead to unsatisfactory forecasting performance once the distribution of future relation graphs, *i.e.,* G_t where $t \in [T+1, T+q]$, is changed and different from these in history.

To address this problem, we want to predict the future observations **F** based on the causal causes **H** directly. We apply the do-intervention [25] as shown in Figure 3. We remove the arrow from **G** to **H** (the do-intervention) following Pearl's back-door criterion [38, 47], which enables us to learn the causal effect from **H** to **F** and from **G** to **F** for predicting the future observations. By this, our time series forecasting is:

$$P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T})) = \sum_{t \in [T-p+1, T+q]} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t). \tag{6}$$

Following Equation (6), to forecast the future observations without the bias caused by the historical relation graphs, we should learn the probability distribution of relation graphs for both the historical and future timestamps, which will be presented in Section 4.1.1, and then combine the historical observations with all possible relation graphs to predict the future observations.

4 METHODOLOGY

As shown in Figure 4, we first present the overall framework of our method, which is based on the encoder-decoder architecture and consists of four main components, *i.e.,* the embedding layer, the causal graph layer, the reasoning network and the projection layer.

The encoder takes as inputs the historical observations $\mathbf{X}_t$, with $t \in [T-p+1, T]$, and outputs hidden state vectors h_t recurrently, which is passed to the decoder to predict future hidden states $h_{t'}$, with $t' \in [T+1, T+q]$, and then to project into the future observations $\hat{\mathbf{X}}_{t'}$.

Firstly, the embedding layer maps the original inputs, *i.e.,* the historical observations of each time series $\mathbf{X}_t$ to the high-dimensional representation vectors v_t, with $t \in [T-p+1, T]$, which aim to extract the feature for the observation of each time series at each timestamp.

To address challenge 1, the causal graph layer contains a dynamic relation graph learning module and a causal graph neural network. Specifically, the former learns a historical relation graph distribution $P_{G_{HT}}$ to capture the correlations among time series in the p historical timestamps, and predicts a future relation graph distribution $P_{G_{FT}}$ to capture the possible correlations among observations in the q future timestamps. For each timestamp $t \in [T-p+1, T+q]$, the causal GNN samples a graph from $P_{G_{HT}}$ and $P_{G_{FT}}$, respectively, and combines them with the feature representation v_t into a hidden state o_t, such that o_t contains not only features of observations but also features of their historical and future correlations.

To address challenge 2, the reasoning network learns and outputs a feature vector h_t for every timestamp $t \in [T-p+1, T+q]$, using the current hidden state o_t and the previous τ feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$. The reasoning network is to identify in which way the past timestamps have different influence on the future timestamps.

The projection layer outputs the forecasting observations $\hat{\mathbf{X}}_{t'}$, based on the reasoned feature vector $h_{t'}$, with $t' \in [T+1, T+q]$.

4.1 The Causal Graph Layer

We first introduce our probabilistic model which can present the distribution of dynamic relation graphs. It can not only learn the historical relation graphs but also predict the future relation graphs with a memory network. Lastly, we introduce our causal GNN, which combines the representation vector v_t at timestamp t with the dynamic relation graphs into a hidden state vector o_t efficiently.

4.1.1 Modeling the dynamic relation graphs. Based on the analysis in Section 3, we model the dynamic relation graphs by learning a probability distribution of relation graphs. As the historical and future relation graph distributions may be different, we model them separately.

Given the current timestamp T, we denote the probability distribution of relation graphs at **the historical time-window**, which ranges from timestamp $T-p+1$ to T, as $G \sim P_{G_{HT}}$, where G is a possible relation graph at timestamp $t \in [T-p+1, T]$. Similarly, we denote the probability distribution of relation graphs at **the future time-window**, which ranges from timestamp $T+1$ to $T+q$, as $G \sim P_{G_{FT}}$, where G is a possible relation graph at timestamp $t' \in [T+1, T+q]$.

In this work, we apply the parameterized Bernoulli distribution [31] to model the probability distribution of relation graphs, *i.e.,* $P_{G_{HT}}$ and $P_{G_{FT}}$. Specifically, $P_{G_{HT}}$ is defined as follows, for any two of the i-th and j-th time series:

$$P(\mathcal{A}_{ij} = 1) = P_h(i,j), \ P(\mathcal{A}_{ij} = 0) = 1 - P_h(i,j), \tag{7}$$

where $\mathcal{A}$ is the adjacency matrix for a relation graph G. Specifically, $P(\mathcal{A}_{ij} = 1)$ is the probability of the i-th and j-th time series being correlated with each other, $P(\mathcal{A}_{ij} = 0)$ is the probability of the i-th and j-th time series not correlated with each other, and $0 \leq P_h(i,j) \leq 1$ is the parameter for Bernoulli distribution, which models the correlation strength between the i-th and j-th time series in the historical time-window and needs be learned. Similarly, $P_{G_{FT}}$ is defined as follows:

$$P(\mathcal{A}_{ij} = 1) = P_f(i,j), \ P(\mathcal{A}_{ij} = 0) = 1 - P_f(i,j). \tag{8}$$

In the following parts, we introduce how to construct the probability distribution of relation graphs for the historical time-window and predict the probability distribution of relation graphs for the future time-window, by learning the parameters $P_h(i,j)$ and $P_f(i,j)$ from the correlation coefficients among the observations of multiple time series, which capture the degree to which two time series vary together.

Probability distribution of historical relation graphs. For the observations $\mathbf{X}_{T-p+1:T}$ in the historical time-window, we construct a historical empirical covariance matrix $S_h \in \mathbb{R}^{N \times N}$, to capture the degree to which two time series vary together, as follows:

$$S_h = \frac{1}{p}(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})^\top, \tag{9}$$

where $\overline{\mathbf{X}}_{T-p+1:T}$ denotes the mean value for each time series across these p timestamps, and $\top$ is the matrix transpose operation. Then the normalized historical correlation coefficient matrix $\rho_h \in \mathbb{R}^{N \times N}$ can be obtained by:

$$\rho_h(i,j) = \frac{S_h(i,j)}{\sqrt{S_h(i,i) S_h(j,j)}}, \ \ for\ 1 \leq i,j \leq N, \tag{10}$$

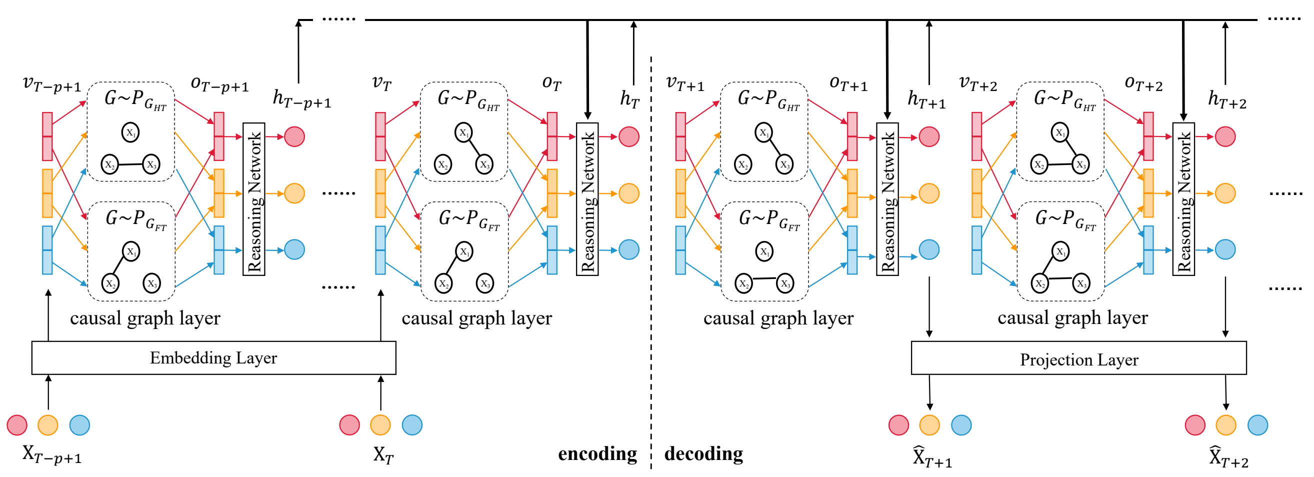


Figure 4: The overall framework

where element $\rho_h(i,j) \in [-1,1]$ is the correlation coefficient between the i-th and j-th time series. If $\rho_h(i,j)$ is closer to 1 (or -1), the positive (or negative) correlation between i-th and j-th time series are more significant, and if $\rho_h(i,j) = 0$, the i-th and j-th time series are linearly independent with each other. Next, based on $\rho_h(i,j)$, we can learn the parameters $P_h(i,j)$. The intuition is as follows. The the bigger the correlation coefficient $|\rho_h(i,j)|$ between the i-th and j-th time series is, the more possible that the i-th and j-th time series are correlated with each other, so that the probability $P_h(i,j)$ is proportional to $|\rho_h(i,j)|$. Thus, we obtain the probability distribution of relation graphs at the historical time-window, *i.e.,* $P_{G_{HT}}$, as follows:

$$P(\mathcal{A}_{ij}=1) = \begin{cases} P_h(i,j) = |\rho_h(i,j)| & if\ |\rho_h(i,j)| \geq \delta \\ 0 & if\ |\rho_h(i,j)| < \delta \end{cases},$$
$$P(\mathcal{A}_{ij}=0) = \begin{cases} 1 - P_h(i,j) = 1 - |\rho_h(i,j)| & if\ |\rho_h(i,j)| \geq \delta \\ 1 & if\ |\rho_h(i,j)| < \delta \end{cases}, \tag{11}$$

where threshold $0 \leq \delta \leq 1$ is a hyper-parameter which controls the sparsity of the relation graph. If the correlation coefficient between the i-th and j-th time series is smaller than δ, we set the i-th and j-th time series as not correlated.

Probability distribution of future relation graphs. After getting the probability distribution for the historical time-window, where observations are **available**, we proceed to predict the probability distribution for the future time-window. We use the features $E_T \in \mathbb{R}^{N \times d^c}$, which are extracted from the historical observations of multiple time series $\mathbf{X}_{T-p+1:T}$, to predict the normalized correlation coefficient matrix $\hat{\rho}_f \in \mathbb{R}^{N \times N}$ for the future time-window, and thus we learn the parameter $P_f(i,j)$ for the future relation graphs which is proportional to $|\hat{\rho}_f|$.

We could only utilize the local features E_T, which are extracted from this current historical time-window from $T-p+1$ to T, to predict the future correlation coefficient matrix $\hat{\rho}_f$. However, it prevents an accurate prediction of the future distribution, as the correlations in the future time-window may be different from those in this historical time-window [13]. Therefore, we use an additional memory unit $E \in \mathbb{R}^{N \times d^c}$ to preserve the global features, as shown in Figure 5. The basic idea is that we use a memory unit to record the correlations that have appeared among multiple time series across all history, *i.e.,* including timestamps long before this time-window $[T-p+1, T]$, which can help to predict the relation graphs for this future time-window $[T+1, T+q]$, as correlations that occurred a long time ago may recur in the future.

Then, by querying the memory unit E with the representation matrix E_T, we learn an outcome feature matrix $E' \in \mathbb{R}^{N \times d^c}$, to predict the future correlation coefficient matrix $\hat{\rho}_f$. Thus, we are able to utilize both local and global features. Last, we update the memory unit E by adding the correlations newly learned from the outcome E'.

Specifically, we map the historical observations of each time series $\mathbf{X}^i_{T-p+1:T}$ to a high-dimensional representation vector $m^i_T \in \mathbb{R}^{d^c}$:

$$m^i_T = \sigma(W_c \mathbf{X}^i_{T-p+1:T}), \tag{12}$$

where σ is an activation function, and $W_c \in \mathbb{R}^{d^c \times p}$ is the parameters to extract the features for predicting the future correlations. We call m^i_T a local view, as it is time-dependent and is extracted from observations in the historical time-window. Thus, the local representation matrix for all time series are $E_T = (m^1_T, m^2_T, \cdots, m^N_T) \in \mathbb{R}^{N \times d^c}$.

Then, we use a learnable embedding vector $m^i \in \mathbb{R}^{d^c}$ to present each time series i across all timestamps. Thus, the memory unit for all time series are $E = (m^1, m^2, \cdots, m^N) \in \mathbb{R}^{N \times d^c}$, and the recorded correlations $\rho(i,j)$ between time series i and time series j can be obtained by the inner-product similarity between m^i and m^j as follows:

$$\rho(i,j) = m^i \cdot m^j. \tag{13}$$

Next, to predict the correlation coefficient matrix for the future time-window, we use the local representation matrix E_T as query to extract the outcome feature matrix $E' \in \mathbb{R}^{N \times d^c}$ from the memory

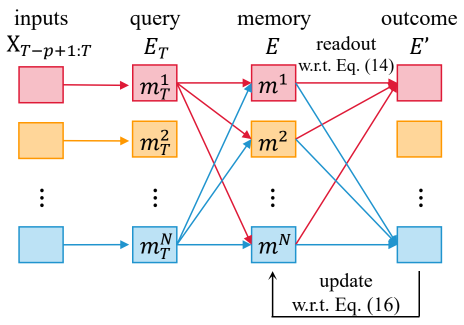


Figure 5: The memory network

unit E as follows:

$$E' = readout(\mathbf{Q}, \mathbf{K}, E) = \varphi(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d^c}})E, \quad \mathbf{Q} = E_t W_{\mathbf{Q}},\ \mathbf{K} = E\, W_{\mathbf{K}}, \tag{14}$$

where $W_{\mathbf{Q}}, W_{\mathbf{K}} \in \mathbb{R}^{d^c \times d^c}$ are the parameters for extracting local and global features, $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times d^c}$ are the learned query and key to readout the features from the memory unit E, φ is the *softmax* function, and $E' \in \mathbb{R}^{N \times d^c}$ is the outcome feature matrix which has extracted both local and global features. Thus, we predict the correlation coefficient matrix for the future time-window, by using the inner-product similarity between the i-th element and j-th element of E' as follows:

$$\hat{\rho}_f(i,j) = E'(i) \cdot E'(j). \tag{15}$$

Last, we update the memory unit E as follows:

$$E = \beta E \,+\, (1-\beta)E', \tag{16}$$

where $0 < \beta < 1$ is a hyper-parameter which controls the percentage of existing correlations kept in the memory unit.

The memory network can be optimized by minimizing the mean square error loss function as follows:

$$\mathcal{L}_{graph} = \frac{1}{N \times N} \sum_{1 \le i,j \le N} \left(\hat{\rho}_f(i,j) - \rho_f(i,j)\right)^2, \tag{17}$$

where:

$$S_f = \frac{1}{p}(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})^\top, \quad \rho_f(i,j) = \frac{S_f(i,j)}{\sqrt{S_f(i,i)S_f(j,j)}}, \quad for\ 1 \le i,j \le N. \tag{18}$$

Similar to Equation (11), we can obtain the probability distribution of relation graphs at the future time-window, denoted as $P_{G_{FT}}$.

4.1.2 Causal graph neural network. Based on Equation (6), we should combine the observations together with the possible historical and future relation graphs to predict the future observations. Thus, at each timestamp t, with $t \in [T-p+1, T]$, we sample a possible relation graph according to the probability distribution $P_{G_{HT}}$ and $P_{G_{FT}}$, respectively. We denote the sampled relation graphs as G_{ht} and G_{ft}, and their adjacency matrix as $\mathcal{A}_{ht}$ and $\mathcal{A}_{ft}$. Then, we propose a causal GNN to extract feature from the sampled historical and future relation graphs, and transfer feature vector v_t, which contains information of the observations, into the hidden state vector o_t. By this, we efficiently integrate the historical observations with the possible historical and future relation graphs.

As the dynamic relation graphs can contain many possible relation graphs, the sampled relation graphs G_{ht} and G_{ft} may be different across time. It is difficult for existing GNN methods [19, 36] to learn with dynamic relation graphs, as they need to learn a set of different parameters $W_{t,k} \in \mathbb{R}^{d^e \times d^e}$, for different graphs at different timestamps as shown in Eq. (19):

$$o = \sum_{k=0}^{K} \left\{ \left(\mathcal{A}_t\right)^k v\, W_{t,k} \right\},\ for\ t \in [T-p+1, T+q], \tag{19}$$

where K is a hyper-parameter which controls the max hop neighbors used in the graph convolution,

Therefore, we need to propose a causal GNN to deal with this problem. Firstly, using the feature vector v_t^i from the embedding layer, we learn a feature vector $v_{ht}^i = W_h v_t^i \in \mathbb{R}^{\frac{d^e}{2}}$ and a feature vector $v_{ft}^i = W_f v_t^i \in \mathbb{R}^{\frac{d^e}{2}}$ for timestamp t, which are then used to combine the historical and future relation graphs, respectively. $W_h, W_f \in \mathbb{R}^{\frac{d^e}{2} \times d^e}$ are the learnable parameters which capture the features for historical and future relation graphs, respectively.

Then, We use the v_{ht} and v_{ft} as the input features for GNN on the historical relation graph G_{ht} and the future relation graph G_{ft}, respectively. To be specific, the causal GNN on relation graph G_{ht} with input feature v_{ht}, *i.e.*, $\star_{G_{ht}}(v_{ht})$, is as follows:

$$o_{ht} = \star_{G_{ht}}(v_{ht}) = \sum_{k=0}^{K} \frac{1}{k+1} \left\{ \left(\widetilde{\mathcal{A}_{ht}}\right)^k v_{ht}\, W_k \right\}, \tag{20}$$

where $\widetilde{\mathcal{A}_{ht}}$ is the adjacency matrix normalized by the diagonal degree matrix D:

$$D_{i,i} = \sum_{1 \le j \le N} \mathcal{A}_{ht}(i,j), \quad \widetilde{\mathcal{A}_{ht}} = D^{-1}\mathcal{A}_{ht}, \tag{21}$$

and $W_k \in \mathbb{R}^{\frac{d^e}{2} \times \frac{d^e}{2}}$, with $k \in [0, K]$, are the learnable parameters which extract the feature from k-th hop neighbors into the output $o_{ht} \in \mathbb{R}^{N \times \frac{d^e}{2}}$. The same as Equation (20), we can get $o_{ft} = \star_{G_{ft}}(v_{ft})$. And finally, the hidden state vector $o_t \in \mathbb{R}^{N \times d^e}$ for all time series, which combine the information of the historical observations with both historical and future relation graphs, is given by $o_t = (o_{ht} \| o_{ft})$.

Thus, instead of learning a set of parameters $W_{t,k}$ for different relation graphs at different timestamps $t \in [T-p+1, T+q]$ with existing GNN, our causal GNN can use v_{ht} and v_{ft} to deal with the dynamics across time, and learn unified neural network parameters W_h, W_f and W_k to extract features from any relation graphs. There are two benefits: (1) It is difficult to train a set of parameters $W_{t,k}$, which can be seen in Section 5.3. (2) Our causal GNN is more efficient regarding time and space complexity, which can be seen in Section 5.5.

4.2 The Reasoning Network

Until now we have got the hidden state o_t, which only contains the features at each timestamp t separately. Now, we propose a

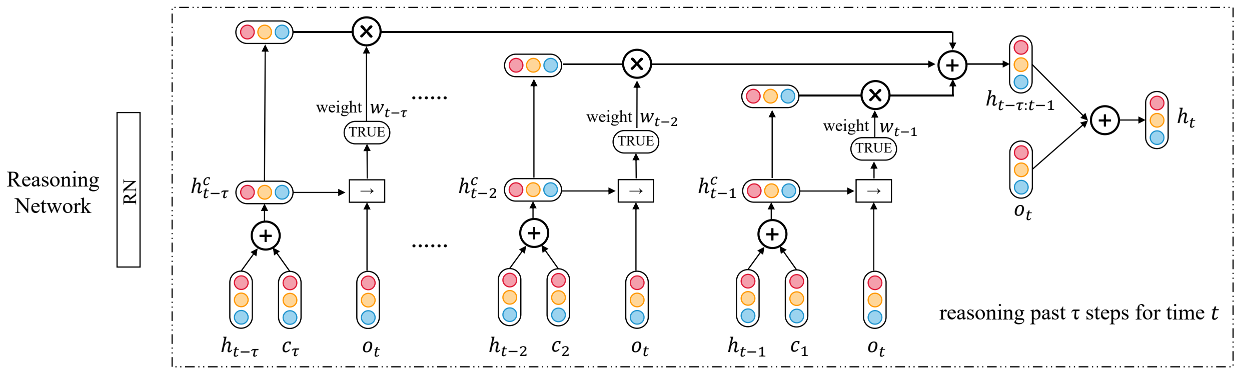


Figure 6: The reasoning network will output the hidden state for multiple time series at each timestamp t using the previous states from past τ steps.

reasoning network as shown in Figure 2(c), which learns a feature vector h_t from not only the current hidden state o_t but also its previous τ timestamps feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, for each timestamp t recurrently. Specifically, h_t contains information of in which way its past τ timestamps influence the current timestamp t, where $1 \leq \tau \leq p$ is a hyper-parameter to controls the previous timestamps used in the reasoning network which make a trade off between efficiency and effectiveness. As this is a recurrent progress, when reasoning for the timestamp t to get h_t, we have obtained h_{t-k} in previous timestamps, with $1 \leq k \leq \tau$.

As shown in Figure 6, the reasoning network firstly combines each feature vector h_{t-k} with a positional embedding c_k, which learns the feature to present a relative time interval from each past timestamp $t-k$ to the current timestamp t, into a feature vector h^c_{t-k}. Then, the reasoning network identifies the importance of the previous timestamp $t-k$ on the current timestamp t by evaluating the horn clause, *i.e.*, $h^c_{t-k} \rightarrow o_t$, and outputs the evaluating result as its importance weight w_{t-k}. Last, the reasoning network obtains the feature vector $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t, which is used for forecasting the observations, by integrating all previous feature vectors h^c_{t-k} with their importance weights w_{t-k} as follows:

$$h_t = \sum_{1 \leq k \leq \tau} w_{t-k} \times h^c_{t-k} + o_t, \tag{22}$$

such that h_t contains the feature of how multiple time series get into the current hidden state o_t based on the condition of previous feature vectors $(h_{t-1}, h_{t-2}, \cdots, h_{t-\tau})$.

Now we present the procedure of our reasoning network in more detail. Firstly, the fact is that under the previous feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$ all together, the multiple time series got into hidden state o_t for $t \leq T$. Therefore, we can have that $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \rightarrow o_t$ is *TRUE* and $(h_{t-\tau} \wedge \cdots \wedge h_{t-2} \wedge h_{t-1}) \rightarrow \neg o_t$ is *FALSE*, where the conjunction operation $\wedge$ joins two feature vectors that multiple time series had in previous timestamps, $\rightarrow o_t = TRUE$ means that all the previous feature vectors will result in the current hidden state o_t, $\neg o_t$ denotes the opposite of the hidden state o_t, and $\rightarrow \neg o_t = FALSE$ means that under the previous feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$, the multiple time series will not get into hidden state $\neg o_t$.

In order to capture the sequential information of the time dependent previous feature vectors, we introduce a total of τ learnable matrices $c_1, c_2, \cdots, c_\tau \in \mathbb{R}^{N \times d^e}$, which are used as the positional embedding [35]. Specifically, each c_k will learn to model the relative time interval from the past timestamp $t-k$ to the current timestamp t. Thus, we can get the time-aware feature vectors:

$$h^c_{t-k} = h_{t-k} + c_k, \ for\ 1 \leq k \leq \tau, \tag{23}$$

and model the historical sequential information by:

$$\begin{aligned} &(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t \ \ is\ TRUE, \\ &(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow \neg o_t \ \ is\ FALSE \end{aligned} \tag{24}$$

where $t \leq T$.

Then, by evaluating whether each horn clause

$$h^c_{t-k} \rightarrow o_t \tag{25}$$

is *TRUE*, we can measure whether the previous feature vector h^c_{t-k} results in the current hidden state o_t. To be specific, if $h^c_{t-k} \rightarrow o_t$ is *TRUE*, we can get that previous feature vector h^c_{t-k} is the cause for the multiple time series to get into current state o_t. On the other hand, if $h^c_{t-k} \rightarrow o_t$ is *FALSE*, we can get that previous feature vector h^c_{t-k} is not the cause for the multiple time series to get into current state o_t.

Now, we model Equations (24) and (25) using our neural reasoning network, which achieves the above logic operations, *i.e.*, $\neg$, $\wedge$ and $\rightarrow$, by neural logic modules. First, We use h_a, h_b and h_c to denote any feature vectors, such as h^c_{t-k}, and each logic operation is calculated by a neural logic module with fully connected layers as follows:

$$\begin{aligned} \neg h_a &= -h_a \\ h_a \wedge h_b \wedge \cdots \wedge h_c &= \sigma\big((h_a \odot h_b \odot \cdots \odot h_c) W_a\big) \\ h_a \rightarrow h_b &= \sigma\big((h_a || h_b) W_i\big) \end{aligned} \tag{26}$$

where $h_a, h_b \in \mathbb{R}^{N \times d^e}$ denote the feature vectors for calculation, $W_a \in \mathbb{R}^{d^e \times d^e}$ and $W_i \in \mathbb{R}^{2d^e \times d^e}$ are the learnable network parameters to achieve the logic operations for $\wedge$ and $\rightarrow$, and $\odot$ is the Hadamard product which multiply matrix on element-wise.

Thus, all logical expressions, such as Equation (24), can be calculated by the neural modules using Equations (26), step by step.

Taking Equation (24), $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$, as an example, we can calculate it as follows:

$$\begin{aligned}\mathbf{Exp}_1 &= (h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \\ &= \sigma\big((h^c_{t-\tau} \odot \cdots \odot h^c_{t-2} \odot h^c_{t-1})W_a\big) \\ \mathbf{Exp}_+ &= \mathbf{Exp}_1 \rightarrow o_t = \sigma\big((\mathbf{Exp}_1 || o_t)W_i\big)\end{aligned} \tag{27}$$

where $\mathbf{Exp}_+ \in \mathbb{R}^{N \times d^e}$ is the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow o_t$. In the same way, we can get the final outcome of logical expression $(h^c_{t-\tau} \wedge \cdots \wedge h^c_{t-2} \wedge h^c_{t-1}) \rightarrow \neg o_t$ as $\mathbf{Exp}_- \in \mathbb{R}^{N \times d^e}$.

Then, another neural module, denoted as $isT()$, is used to evaluate whether an expression $\mathbf{Exp}_a$ is *TRUE* or *FALSE* by:

$$isT(\mathbf{Exp}_a) = sigmoid(\mathbf{Exp}_a\, W_r), \tag{28}$$

where $\mathbf{Exp}_a \in \mathbb{R}^{N \times d^e}$ denotes the outcome of a logical expression, such as $\mathbf{Exp}_+$, for the *TRUE/FALSE* evaluation, $W_r \in \mathbb{R}^{d^e \times 1}$ is the learnable parameter to evaluate the logical expression, and $isT(\mathbf{Exp}_a)$ is the evaluation result. The *sigmoid* function makes sure that the evaluation result is between 0 and 1, where $isT(\mathbf{Exp}_a) = 0$ means that the logical expression is *FALSE* and $isT(\mathbf{Exp}_a) = 1$ means that the logical expression is *TRUE*.

Thus, to satisfy the truth denotes by Equation (24), we have the logical regularization, which is achieved by minimizing the loss function as below:

$$\mathcal{L}_{reg} = \{1 - isT(\mathbf{Exp}_+)\} + isT(\mathbf{Exp}_-) \tag{29}$$

where $isT(\mathbf{Exp}_+)$ being 1 denotes that with the previous feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$ the multiple time series actually got into hidden state o_t, and $isT(\mathbf{Exp}_-)$ being 0 denotes that the multiple time series did not get into hidden state $\neg o_t$.

Next, we can measure whether the previous feature vector h^c_{t-k} results in the current hidden state o_t, by:

$$\mathbf{Exp}_{t-k} = (h^c_{t-k} \rightarrow o_t) = (h^c_{t-k} || o_t)W_i, \tag{30}$$

$$w_{t-k} = isT(\mathbf{Exp}_{t-k}), \tag{31}$$

where w_{t-k} is the importance weight. The more the w_{t-k} is close to 1, the more possible the previous feature vector h^c_{t-k} is the cause for the multiple time series to get into current state o_t.

Lastly, the reasoning network get the feature vector $h_t \in \mathbb{R}^{N \times d^e}$ for the current timestamp t by integrating all previous feature vectors h^c_{t-k} with their importance weights w_{t-k} using Equation (22).

In this way, the reasoning network is able to explicitly learn how historical timestamps have different influence on future timestamps with different horn clauses.

4.3 The Projection Layer

We use the zero matrix $v_{T+1}, o_{T+1} \in \mathbb{R}^{N \times d^e}$ to present the initial hidden states of the multiple time series for the first future timestamp $T+1$, which denote the beginning of forecasting on the future observations. Then, after the reasoning network outputs the feature vector h_{T+1} based on the initial hidden state o_{T+1} and the past τ feature vectors $(h_{t-\tau+1}, \cdots, h_{t-1}, h_t)$, the projection layer outputs the forecasting observations $\hat{\mathbf{X}}_{T+1} \in \mathbb{R}^{N \times 1}$ as follows:

$$\hat{\mathbf{X}}_{T+1} = h_{T+1} W_p, \tag{32}$$

Table 3: The statistics of datasets

Dataset	N	L	Split	p	q
METR-LA	207	34,272	7:1:2	12	12
PEMS-BAY	325	52,116	7:1:2	12	12
PEMS04	307	16,992	6:2:2	12	12
PEMS08	170	17,856	6:2:2	12	12
Solar-Energy	137	52,560	6:2:2	168	1
Electricity	321	26,304	6:2:2	168	1

where $W_p \in \mathbb{R}^{d^e \times 1}$ is the network parameter mapping the feature vectors to the observations of time series. Next, the new feature vector v_{T+2} for next timestamp $T+2$ is obtained from the predicted observations $\hat{\mathbf{X}}_{T+1}$ as follows:

$$v_{T+2} = \hat{\mathbf{X}}_{T+1} W_n, \tag{33}$$

where $W_n \in \mathbb{R}^{1 \times d^e}$ is the learnable parameter to extract feature from the predicted observations. In this way, we can predict the observations for all the q future timestamps, $\hat{\mathbf{X}}_{T+1:T+q}$, recurrently.

4.4 The Objective Function

We use the objective function to enable model learning with gradient descent. Take the mean absolute error (MAE) as example:

$$\mathcal{L}_{MAE} = \frac{1}{N \times q} \left\|\hat{\mathbf{X}}_{T+1:T+q} - \mathbf{X}_{T+1:T+q}\right\|^1, \tag{34}$$

where $\|M\|^1 = \sum_{i,j} |M_{i,j}|$.

Overall, the memory network is optimized by minimizing Eq. (17), and the embedding layer, the causal GNN, the reasoning network and the projection layer is optimized by minimizing Eq. (35):

$$\mathcal{L} = \mathcal{L}_{MAE} + \lambda \mathcal{L}_{reg}, \tag{35}$$

where λ is a hyper-parameter controls the importance of logical regularization.

5 EXPERIMENTS

In this section, we empirically evaluate MTSF-DG on real-world benchmark datasets to justify our model. We introduce the multiple time series forecasting datasets, evaluation metrics and the competitor baselines first. Then we present the main results, and analyze our model with more details and ablation studies.

5.1 Experimental Settings

5.1.1 Datasets. We use a series of benchmark datasets from traffic and energy domains to evaluate the performance of multiple time series forecasting:

- METR-LA and PEMS-BAY: Both datasets are traffic speed time series datasets, released by Li et al. [23]. The METR-LA dataset contains the traffic speed measured by 207 sensors on the highways of Los Angeles County ranging from Mar. 2012 to Jun. 2012. The PEMS-BAY dataset contains the traffic speed measured by 325 sensors in the Bay Area ranging from Jan. 2017 to May 2017.
- PEMS04 and PEMS08: Both datasets are traffic flow time series collected from the Caltrans Performance Measurement System (PEMS), released by Bai et al. [1]. The PEMS04 dataset contains the traffic flow measured by 307 sensors in the San Francisco Bay

Table 4: Accuracy of traffic domain forecasting

Data	q	Metric	DCRNN	GWave	AGCRN	MTGNN	STFGNN	Cformer	FEDFormer	MSDR	ESG	MTSF-DG
METR -LA	3rd	MAE	2.77	2.69	2.83	2.69	2.70	2.69	2.89	2.71	2.68	**2.62**
		RMSE	5.38	5.15	5.45	5.18	5.35	5.17	5.51	5.18	5.15	**5.11**
		MAPE	7.30%	6.90%	7.56%	6.86%	7.21%	6.88%	7.63%	7.08%	6.93	**6.78%**
	6th	MAE	3.15	3.07	3.20	3.05	3.10	3.05	3.27	3.09	3.06	**2.98**
		RMSE	6.45	6.22	6.55	6.17	6.36	6.18	6.56	6.33	6.19	**6.13**
		MAPE	8.80%	8.37%	8.79%	8.19%	8.60%	8.21%	8.87%	8.57%	8.20%	**8.14%**
	12th	MAE	3.60	3.53	3.58	3.49	3.51	3.48	3.69	3.50	3.49	**3.39**
		RMSE	7.60	7.37	7.41	7.23	7.46	7.27	7.66	7.33	7.23	**7.16**
		MAPE	10.50%	10.01%	10.13%	9.87%	10.05%	9.86%	10.44%	9.98%	9.96%	**9.58%**
PEMS -BAY	3rd	MAE	1.38	1.30	1.35	1.32	1.34	1.31	1.47	1.32	1.31	**1.28**
		RMSE	2.95	2.74	2.83	2.79	2.81	2.75	3.02	2.84	2.74	**2.67**
		MAPE	2.90%	2.73%	2.87%	2.77%	2.84%	2.72%	2.96%	2.77%	2.76%	**2.63%**
	6th	MAE	1.74	1.63	1.69	1.65	1.66	1.63	1.81	1.64	1.63	**1.57**
		RMSE	3.97	3.70	3.81	3.74	3.76	3.69	4.01	3.78	3.71	**3.63**
		MAPE	3.90%	3.67%	3.84%	3.69%	3.83%	3.66%	3.99%	3.68%	3.69%	**3.56%**
	12th	MAE	2.07	1.95	1.96	1.94	1.98	1.93	2.06	1.94	1.92	**1.85**
		RMSE	4.74	4.52	4.52	4.49	4.52	4.45	4.78	4.51	4.42	**4.35**
		MAPE	4.90%	4.63%	4.67%	4.53%	4.73%	4.49%	4.88%	4.55%	4.52%	**4.40%**
PEMS 04	1~12	MAE	24.70	19.16	19.83	19.32	19.83	19.50	23.48	19.29	19.47	**18.67**
		RMSE	38.12	30.46	32.26	31.57	31.88	32.00	37.27	31.54	31.66	**30.17**
		MAPE	17.12%	13.26%	12.97%	13.52%	13.02%	13.07%	15.44%	12.89%	13.30%	**12.64%**
PEMS 08	1~12	MAE	17.86	15.13	15.95	15.71	16.64	15.88	17.24	15.11	15.70	**14.80**
		RMSE	27.83	24.07	25.22	24.62	26.22	25.07	26.93	24.42	24.81	**23.68**
		MAPE	11.45%	10.10%	10.09%	10.03%	10.60%	10.17%	11.21%	9.93%	10.07%	**9.54%**

Area ranging from Jan. 2018 to Feb. 2018. The PEMS08 dataset contains the traffic flow measured by 170 sensors in the San Bernardino Area ranging from Jul. 2016 to Aug. 2016.

- Solar-Energy: The Solar-Energy dataset contains the solar power production records collected from 137 PV plants in the Alabama State in 2007, released by Lai et al. [20].
- Electricity: The Electricity energy dataset contains the electricity consumption records collected from 321 clients from 2012 to 2014, released by Lai et al. [20].

The detailed statistics of these datasets are shown in Table 3, where N is the number of time series and L is the total number of timestamps. We follow the same train-validation-test splits as in the original papers [1, 20, 23], as shown in the "Split" column. More details about datasets can be seen in the Appendix.

5.1.2 *Evaluation Metrics.* Following the evaluation methodology in existing works [1, 20, 23], we use mean absolute error (MAE), root mean squared error (RMSE), mean absolute percentage error (MAPE) to evaluate the accuracy of multi-step forecasting, and use Root Relative Squared Error (RRSE) and Empirical Correlation Coefficient (CORR) to measure the accuracy of single-step forecasting. For MAE, RMSE, MAPE, and RRSE, lower values indicate higher accuracy, while larger CORR values indicate higher accuracy.

5.1.3 *Baselines.* We compare MTSF-DG with baseline methods summarized as follows:

- **Methods without relation graphs.** VAR-MLP: It is an autoregressive model using multilayer perception (MLP) [48]. GP: It uses Gaussian Process to model time series [34]. LSTNet: It combines convolutional neural network (CNN) with RNN to learn temporal dependencies [20]. TPA: It is a naive Transformer model [32]. FEDFormer: It uses frequency-enhanced Transformer to extract trend and periodic features [33]. Crossformer: It is the state-of-art Transformer based model, which uses a cross-dimension attention to learn the historical correlations among time series, without learning graphs [49].
- **Methods with a pre-defined graph.** DCRNN: It proposes diffusion graph convolutions to extract spatial dependencies [23]. GWave: It proposes 1D dilated CNN and combines with diffusion graph convolutions [43]. AGCRN: It proposes adaptive recurrent graph convolution network [1]. MSDR: It proposes attention based graph convolutions and multi-step RNN [24].
- **Methods learn relation graphs.** MTGNN: It learns a static relation graph that models similarities among multiple time series, and uses graph convolutions and CNN for forecasting [42]. DGTS: It constructs a static relation graph based on the Euler distances among multiple time series, and uses recurrent graph convolutions for forecasting [31]. STFGNN: It constructs a static relation graph based on the Dynamic Time Warping similarities [16] among multiple time series [22]. ESG: It cuts the historical observations into sub time-windows, learns a historical relation graph for each time-window separately, and use RNN for forecasting [45].

We report results from the original papers if baselines conduct experiments on the dataset with the same setting. For the rest, we have carefully tuned the hyper-parameters based on the recommendations from their original papers.

Table 5: Accuracy of energy domain forecasting

Dataset		Solar-Energy		Electricity	
		q		q	
Method	Metric	3rd	12th	3rd	12th
VAR-MLP	RRSE	0.1922	0.4244	0.1393	0.1557
	CORR	0.9829	0.9058	0.8708	0.8192
GP	RRSE	0.2259	0.5200	0.1500	0.1621
	CORR	0.9751	0.8518	0.8670	0.8394
LSTNet	RRSE	0.1843	0.3254	0.0864	0.1007
	CORR	0.9843	0.9467	0.9283	0.9077
TPA	RRSE	0.1803	0.3234	0.0823	0.0964
	CORR	0.9850	0.9487	0.9439	0.9250
MTGNN	RRSE	0.1778	0.3109	0.0745	0.0916
	CORR	0.9852	0.9509	0.9474	0.9278
DGTS	RRSE	0.1791	0.3144	0.0767	0.0925
	CORR	0.9852	0.9501	0.9470	0.9275
Crossformer	RRSE	0.1772	0.3089	0.0741	0.0905
	CORR	0.9859	0.9511	0.9474	0.9291
FEDFormer	RRSE	0.1788	0.3141	0.0769	0.0924
	CORR	0.9852	0.9498	0.9465	0.9280
ESG	RRSE	0.1708	0.3073	0.0718	0.0898
	CORR	0.9865	0.9519	0.9494	0.9321
MTSF-DG	RRSE	**0.1692**	**0.3025**	**0.0701**	**0.0882**
	CORR	**0.9874**	**0.9533**	**0.9502**	**0.9339**

Table 6: Ablation studies

Dataset	PEMS04			PEMS08		
Method	MAE	RMSE	MAPE	MAE	RMSE	MAPE
MTSF-DG	**18.67**	**30.17**	**12.64%**	**14.80**	**23.68**	**9.54%**
w/o memory	18.68	30.27	12.79%	14.82	23.69	9.58%
w/o causal GNN	18.72	30.23	12.76%	14.85	23.73	9.59%
G_{ht} only	18.98	30.59	12.93%	15.07	23.97	9.85%
G_{ft} only	18.93	30.51	12.91%	14.93	23.95	9.77%
w Transformer	18.91	30.44	12.83%	14.98	23.93	9.69%

5.2 Overall Comparison.

Tables 4 and 5 present the accuracy of MTSF-DG and the baselines on all datasets. We randomly repeat each method 5 times and report the average result. We use bold to highlight the best accuracy, which significantly outperforms the underline second best accuracy.

Key observations are as follows. Firstly, MTSF-DG consistently outperforms the state-of-the-art baseline methods on all datasets. It demonstrates that MTSF-DG is able to learn the dynamic correlations among multiple time series and use them to improve the forecasting performance.

Secondly, from Table 5 we observe that the MTGNN, DGTS, ESG and MTSF-DG methods, which can learn the relation graph(s) for multiple time series, perform better when comparing to VAR-MLP, GP, LSTNet, TPA and FEDFormer, which cannot capture the relations among multiple time series. Crossformer learns the correlations among time series with Cross-Transformer, which also performs better when comparing to VAR-MLP, GP, LSTNet, TPA and FEDFormer. It demonstrates that capturing the relations among multiple time series is important for multiple time series forecasting.

Table 7: Parameter sensitivity

Dataset	PEMS04			PEMS08		
τ	MAE	RMSE	MAPE	MAE	RMSE	MAPE
2	19.83	31.21	12.91%	15.63	25.12	10.00%
3	19.20	30.77	12.73%	15.03	24.23	9.76%
4	18.67	30.17	12.64%	14.80	23.68	9.54%
5	18.69	30.22	12.65%	14.79	23.73	9.52%
12	18.72	30.25	12.69%	14.82	23.76	9.60%

Table 8: Runtime and total parameters used

Dataset	PEMS04		PEMS08	
Method	Runtime (s/epoch)	Parameters (K)	Runtime (s/epoch)	Parameters (K)
DCRNN	226	371	159	371
DGTS	262	381	179	378
MSDR	618	1174	438	1174
ESG	378	1205	255	1205
MTSF-DG	217	803	151	799
w/o causal GNN	298	1270	194	1205

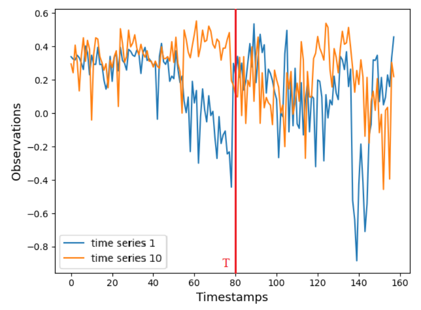
Thirdly, we observe that our MTSF-DG method is also superior when comparing to the other graph based methods, which use a single relation graph or only learn historical relation graphs, to enhance the forecasting accuracy. This is due to the fact that the baselines cannot capture the dynamic correlations among multiple time series which may change across time and be different in the future, where different future relation graphs may influence the future observations differently. A single relation graph or the historical relation graph will bias the forecasting. GWave, MTGNN, STFGNN, MSDR and ESG can only have the second best accuracy on some datasets. There does not exist a single baseline method that consistently outperforms others, which suggests that a single relation graph or the historical relation graph is insufficient for multiple time series forecasting. In contrast, our MTSF-DG can learn the historical and future correlations dynamically, and consistently outperforms baseline methods.

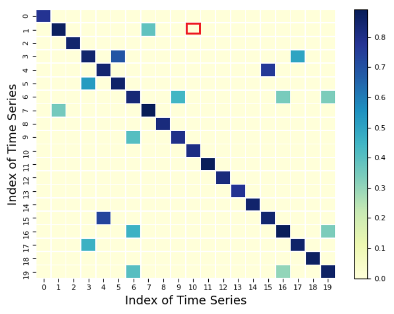
Lastly, MTSF-DG achieves the best accuracy compared to Transformer based models. This suggests that our reasoning network is also good at learning temporal dependencies by explicitly learning how historical timestamps have different influence on future timestamps. This enables MTSF-DG to get more high performance compared to TPA, FEDFormer and Crossformer.

5.3 Ablation Studies.

We conduct ablation studies to validate the effectiveness of our key components that contribute to the improvements. In particular, we compare MTSF-DG with the following variants:

- w/o memory network: This variant does not use the memory network for predicting the future relation graph distribution. It directly uses the local features E_T.
- w/o causal GNN: This variant does not use the causal GNN. It use the existing GNN on the sampled historical relation graph and future relation graph at each timestamp with Eq. (19), and learn a set of parameters $W_{t,k}$.


(a) Dynamic correlations between time series 1 and 10


(b) Heatmap for $P_{G_{HT}}$ before T

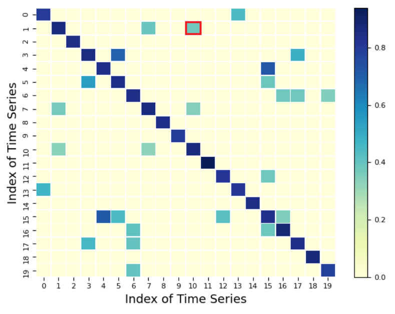

(c) Heatmap for $P_{G_{FT}}$ after T

Figure 7: Case study

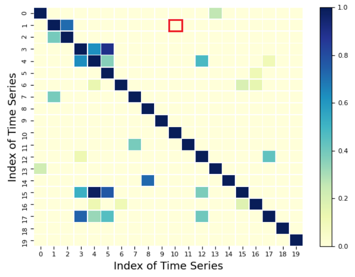


Figure 8: The static graph from DGTS

- G_{ht} only: This variant does not use causal graph layer. It applies the existing GNN [43] on the sampled historical relation graph G_{ht} only.
- G_{ft} only: This variant does not use causal graph layer. It applies the existing GNN [43] on the sampled future relation graph G_{ft} only.
- w Transformer: This variant does not use reasoning network. It uses Transformer [35] to model with the hidden states.

Table 6 shows the accuracy of different variants on PEMS04 and PEMS08 datasets. For the other datasets, the results show similar trends. From Table 6 we observe that: (1) MTSF-DG achieves better accuracy comparing to its variant w/o memory network. This demonstrates the effectiveness of the proposed memory network for predicting the future relation graph distribution. It can improve the forecasting accuracy by providing more accurate future correlations among multiple time series. (2) The information from different hops are not equally important. The near hop neighbors are more important to present the correlations among time series. (3) The existing GNN which learns a set of parameter will suffer from over-fitting and performs worse. (4) Simply using the existing GNN [23, 43] with a single historical relation graph or future relation graph will lower the performance significantly. This result is consistent with our analyses in Section 3, suggesting that our causal GNN, which can learn with historical relation graph and future relation graph jointly, is more effective in multiple time series forecasting. (4) MTSF-DG with reasoning network outperforms the variant with Transformer, which justifying that the Transformer, which only model the influence among historical timestamps, is insufficient to accurately forecast future observations under changeable future relation graphs. However, our reasoning network is able to do so, as it can explicitly learn how historical timestamps have different influence on future timestamps.

5.4 Parameter Sensitivity

We evaluate the impact of the hyperparameter, *i.e.,* τ, which controls the max previous timestamps used in the reasoning network. The experimental results are shown in Table 7. If we use more previous timestamps in the reasoning network up to 4 timestamps, the MTSF-DG model performs better. The reason is that for the traffic prediction task we need to learn the temporal influence for a long range. When the value of τ changes to 5 and 12, the accuracy results are relatively stable.

5.5 Runtime and Total Parameters Used

For our dynamic graph learning, the time complexity is $O(N^2d)$. For causal GNN, the time complexity is $O(KN^2d)$. For reasoning network, the time complexity is $O((p+q)Nd)$. We also show the overall runtime and the number of total parameters used for different methods in Table 8.

We can see that MTSF-DG is better than these baseline methods regarding the runtime. We can also see that the time and space complexity of MTSF-DG is smaller than MSDR, ESG and the variant w/o causal GNN. Our model is only worse than DCRNN and DGTS regarding the number of total parameters used. This is because DCRNN and DGTS use only one relation graph for all timestamps, and our model need more space to learn the dynamic relation graphs.

5.6 Case Study

To show the superiority of our dynamic graph modeling, we give the case study on METR-LA dataset and visualize in Figure 7. From Figure 7(a), we can see that the time series 1 and 10 are more correlated to each other during the first 40 timestamps and the 80-120 timestamps, and are less correlated during the 40-80 timestamps. At timestamp T = 80, if we use the 12 historical observations to predict the 12 future observations, these dynamic correlations are captured by the probability distribution of historical relation graphs

Table 9: Comparisons between representative methods.

Method	Dynamic relation graphs in encoder	Dynamic relation graphs in decoder
[7, 20, 32–34, 37, 39, 48, 49]	×, only use historical observations	×
[1, 9, 12, 22, 24, 31, 40, 42, 46]	×, only a static relation graph	×
[8, 45]	√, but only historical relation graphs	×
MTSF-DG	√	√

$P_{G_{HT}}$ and the probability distribution of future relation graphs $P_{G_{FT}}$, as shown in Figure 7(b) and Figure 7(c). However, the baseline DGTS can only learn a static, historical relation graph as shown in Figure 8, which cannot capture the correlation between time series 1 and 10 that sometimes occurs. By learning such dynamic relation graphs for historical and future time-window, our MTSF-DG can improve the forecasting performance.

6 RELATED WORK

We review existing works on time series forecasting and graph learning, and summary them in Table 9. We also compare reasoning network with RNN and Transformer based models in Figure 2.

6.1 Time Series Forecasting

Early methods try to utilize the statistical methods, *e.g.,* Auto-Regressive model (AR) [48] and Gaussian Process model (GP) [34], to forecast on time series, which model the future observations as the linear combination of the nearby historical observations, called the temporal dependencies. Some works [32, 37] proposed to utilize RNN [6], TCN [21] or attention network [35] for time series forecasting by modeling dependencies using more historical observations. LSTNet [20] employs 1D CNN and RNN to capture temporal dependencies. N-BEATS [28] uses fully connected MLP and residual blocks to predict the trend and periodicity. Timesnet [39] propose a 2D CNN to capture temporal dependencies and periodic frequency. Triformer [7] proposes a Transformer based time series forecasting model with linear complexity attention. FEDFormer [33] uses frequency-enhanced Transformer to extract trend and periodic features. Crossformer [49] uses a cross-dimension attention to implicitly learn the historical correlations among time series without graphs.

However, these methods only use historical observations to predict future observations, and cannot use a relation graph to capture correlations among time series explicitly.

Then, there have been a lot of GNN based methods for traffic time series forecasting [9, 12, 46]. One kind of correlations among multiple time series can be the spatial distance among different locations, which naturally form a spatial graph. The GNN based methods capture spatial dependencies by aggregating features [36, 50] from the neighbors on the spatial graph. DCRNN [23] proposes diffusion graph convolutions to extract spatial dependencies and use GRU for forecasting. AGCRN proposes adaptive recurrent graph convolution network to capture node-specific features for each time series [1]. Graph WaveNet [43] combines diffusion graph convolutions with gated dilated TCN.

However, these methods need a pre-defined graph to present correlations among time series in advance, and they cannot model the dynamic relation graphs.

6.2 Graph Learning

Most recently, a new trend is to employ graph learning [13] to learn relation graphs that models correlations among multiple time series without requiring spatial distance in advance, to enable universal multiple time series forecasting. MTGNN [42], STFGNN [22] and DGSL [31] construct a static relation graph, where time series are considered as nodes, and two time series are connected by an edge if their observations are similar measured by the Cosine distances, Euler distances or Dynamic Time Warping distances [16]. AutoCTS [40] automatically search from RNN, TCN, GNN and attention network, to build a better deep neural network to learn a static relation graph and predict future observations.

EnhanceNet [8] and ESG [45] cut the historical observations into sub time-windows, and construct a historical relation graph which is used in each time-window separately.

However, EnhanceNet and ESG are incapable of learning the dynamic correlations for the future timestamps. Besides, they only use dynamic historical relation graphs in encoder to extract the invariant temporal dependencies from historical observations, which is used to predict the observations for all future timestamps simultaneously. They fail to use the dynamic relation graphs in decoder and cannot learn the different temporal dependencies. But our MTSF-DG can learn the dynamic correlations for the future timestamps using the memory network, and learn the different temporal dependencies using the reasoning network in both encoder and decoder.

7 CONCLUSION

We present MTSF-DG for multiple time series forecasting. We propose to learn historical relation graphs, and predict future relation graphs to capture the dynamic correlations with the memory network, by optimizing the relation graph distributions from an empirical covariance matrix. Then we propose a causal GNN to extract features from both historical and future relation graphs efficiently. Lastly, we propose a reasoning network to explicitly learn how historical timestamps have different influence on future timestamps with the logical operations and symbolic reasoning procedure, and predict the future observations based on reasoning the future feature vectors. Experiments on six benchmark datasets demonstrate the superiority of our method. In future work, it is of interest to extend MTSF-DG to other time series tasks, such as abnormality detection and prediction.

ACKNOWLEDGMENTS

This work was partially supported by Independent Research Fund Denmark under agreements 8022-00246B and 8048-00038B, and the VILLUM FONDEN under agreement 40567.

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *NeurIPS*. 11465–11475.

[2] David Campos, Tung Kieu, Chenjuan Guo, Feiteng Huang, Kai Zheng, Bin Yang, and Christian S. Jensen. 2022. Unsupervised Time Series Outlier Detection with Diversity-Driven Convolutional Ensembles. *Proc. VLDB Endow.* 15, 3 (2022), 611–623.

[3] David Campos, Miao Zhang, Bin Yang, Tung Kieu, Chenjuan Guo, and Christian S. Jensen. 2023. LightTS: Lightweight Time Series Classification with Adaptive Ensemble Distillation. *Proc. ACM Manag. Data* 1, 2 (2023), 171:1–171:27.

[4] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *WWW*. 1516–1527.

[5] Yunyao Cheng, Peng Chen, Chenjuan Guo, Kai Zhao, Qingsong Wen, Bin Yang, and Christian S. Jensen. 2024. Weakly Guided Adaptation for Robust Time Series Forecasting. *Proc. VLDB Endow.* (2024).

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. 1724–1734.

[7] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. 2022. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting. In *IJCAI*. 1994–2001.

[8] Razvan-Gabriel Cirstea, Tung Kieu, Chenjuan Guo, Bin Yang, and Sinno Jialin Pan. 2021. EnhanceNet: Plugin Neural Networks for Enhancing Correlated Time Series Forecasting. In *ICDE*. 1739–1750.

[9] Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. 2022. Towards Spatio- Temporal Aware Traffic Time Series Forecasting. In *ICDE*. 2900–2913.

[10] Razvan-Gabriel Cirstea, Bin Yang, and Chenjuan Guo. 2019. Graph Attention Recurrent Neural Networks for Correlated Time Series Forecasting.. In *MileTS19@KDD*.

[11] Sayda Elmi and Kian-Lee Tan. 2021. DeepFEC: Energy Consumption Prediction under Real-World Driving Conditions for Smart Cities. In *WWW*. ACM / IW3C2, 1880–1890.

[12] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-Temporal Graph ODE Networks for Traffic Flow Forecasting. In *KDD*. 364–373.

[13] David Hallac, Youngsuk Park, Stephen P. Boyd, and Jure Leskovec. 2017. Network Inference via the Time-Varying Graphical Lasso. In *SIGKDD*. 205–213.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[15] Nicola Jones. 2017. How machine learning could help to improve climate forecasts. *Nature* 548 (2017), 379.

[16] Eamonn J. Keogh and Michael J. Pazzani. 2001. Derivative Dynamic Time Warping. In *SDM*. 1–11.

[17] Tung Kieu, Bin Yang, Chenjuan Guo, Razvan-Gabriel Cirstea, Yan Zhao, Yale Song, and Christian S. Jensen. 2022. Anomaly Detection in Time Series with Robust Variational Quasi-Recurrent Autoencoders. In *ICDE*. 1342–1354.

[18] Tung Kieu, Bin Yang, Chenjuan Guo, Christian S. Jensen, Yan Zhao, Feiteng Huang, and Kai Zheng. 2022. Robust and Explainable Autoencoders for Unsupervised Time Series Outlier Detection. In *ICDE*. 3038–3050.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *SIGIR*. ACM, 95–104.

[21] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. 2017. Temporal Convolutional Networks for Action Segmentation and Detection. In *CVPR*. 1003–1012.

[22] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. In *AAAI*. 4189–4196.

[23] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

[24] Dachuan Liu, Jin Wang, Shuo Shang, and Peng Han. 2022. MSDR: Multi-Step Dependency Relation Networks for Spatial Temporal Forecasting. In *KDD*. ACM, 1042–1050.

[25] Glymour Madelyn, Judea Pearl, and Nicholas P. Jewell. 2016. *Causal inference in statistics: A primer.* John Wiley & Sons.

[26] Hao Miao, Yan Zhao, Chenjuan Guo, Bin Yang, Zheng Kai, Feiteng Huang, Jiandong Xie, and Christian S. Jensen. 2024. A Unified Replay-based Continuous Learning Framework for Spatio-Temporal Prediction on Streaming Data. *ICDE* (2024).

[27] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luís Damas. 2013. Predicting Taxi-Passenger Demand Using Streaming Data. *IEEE TITS* 14, 3 (2013), 1393–1402.

[28] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*.

[29] Simon Aagaard Pedersen, Bin Yang, and Christian S. Jensen. 2020. Anytime Stochastic Routing with Hybrid Learning. *Proc. VLDB Endow.* 13, 9 (2020), 1555–1567.

[30] Meng Qu and Jian Tang. 2019. Probabilistic Logic Neural Networks for Reasoning. In *NeurIPS*. 7710–7720.

[31] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *ICLR*.

[32] Shun-Yao Shih, Fan-Keng Sun, and Hung-Yi Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* 108, 8-9 (2019), 1421–1441.

[33] Zhou Tian, Ma Ziqing, Wen Qingsong, Wang Xue, Sun Liang, and Jin Rong. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *ICML*. 27268–27286.

[34] Felipe A. Tobar, Thang D. Bui, and Richard E. Turner. 2015. Learning Stationary Time Series using Gaussian Processes with Nonparametric Kernels. In *NeurIPS*. 3501–3509.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[36] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR, 2018*.

[37] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. MICN: Multi-scale Local and Global Context Modeling for Long-term Series Forecasting. In *ICLR*.

[38] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. 2020. Visual Commonsense R-CNN. In *CVPR*. 10757–10767.

[39] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *ICLR*.

[40] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S. Jensen. 2022. AutoCTS: Automated Correlated Time Series Forecasting. *Proc. VLDB Endow.* 15, 4 (2022), 971–983.

[41] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. *Proc. ACM Manag. Data* 1, 1 (2023), 97:1–97:26.

[42] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *KDD*. 753–763.

[43] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*. 1907–1913.

[44] Sean Bin Yang, Chenjuan Guo, and Bin Yang. 2022. Context-Aware Path Ranking in Road Networks. *TKDE* 34, 7 (2022), 3153–3168.

[45] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the Evolutionary and Multi-scale Graph Structure for Multivariate Time Series Forecasting. In *KDD*. 2296–2306.

[46] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*. 3634–3640.

[47] Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. 2020. Interventional Few-Shot Learning. In *NeurIPS*.

[48] Guoqiang Peter Zhang. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50 (2003), 159–175.

[49] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *ICLR*.

[50] Kai Zhao, Ting Bai, Bin Wu, Bai Wang, Youjie Zhang, Yuanyu Yang, and Jian-Yun Nie. 2020. Deep adversarial completion for sparse heterogeneous information network embedding. In *WWW*. 508–518.

[51] Kai Zhao, Yukun Zheng, Tao Zhuang, Xiang Li, and Xiaoyi Zeng. 2022. Joint Learning of E-commerce Search and Recommendation with a Unified Graph Neural Network. In *WSDM*. 1461–1469.