Time Series Reasoning Networks with Causality

Submission Id: 468

ABSTRACT

Multiple time series forecasting plays an essential role in many application fields. Solutions based on graph neural network that deliver state-of-the-art forecasting performance use the relation graph which captures historical correlations among time series. However, in real world, it is common that correlations among time series evolve across time, resulting in dynamic relation graphs, where the future correlations may be different from those in history. To address this problem, we propose TSRN-C 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 for forecasting. Then we propose a reasoning network to learn how the multiple time series evolve among feature vectors, and predict the future observations based on reasoning the future feature vectors. Extensive experiments on six benchmark datasets show that TSRN-C consistently outperforms state-of-the-art baselines, and justify our model design which is able to forecast the future observations with dynamic relation graphs.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Neural networks**.

KEYWORDS

time series forecasting, graph neural network, reasoning, causality

ACM Reference Format:

. 2023. Time Series Reasoning Networks with Causality. In *Proceedings of the 2023 International Conference on Management of Data (SIGMOD '23).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Many real-world data sensed from cyber-physical systems (CPS) can be modeled as the recordings of time-dependent observations or measurements of entities, which form the multiple time series [26, 31, 36]. For example, in power grid there exist multiple electric time series which record the energy consumption of different clients across time [9], and in transport network there exist multiple traffic time series which record the traffic flows and speeds at different locations across time [14, 42]. Based on the historical observations, forecasting the future observations of time series plays an important role in ensuring effective functionality of CPS for various application fields, such as spotting patterns and trends [16] and predicting the future behaviors [4, 15, 38]. Thus, we aim at multiple time series forecasting problem in this paper.

SIGMOD '23, June 18–23, 2023, Washington, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

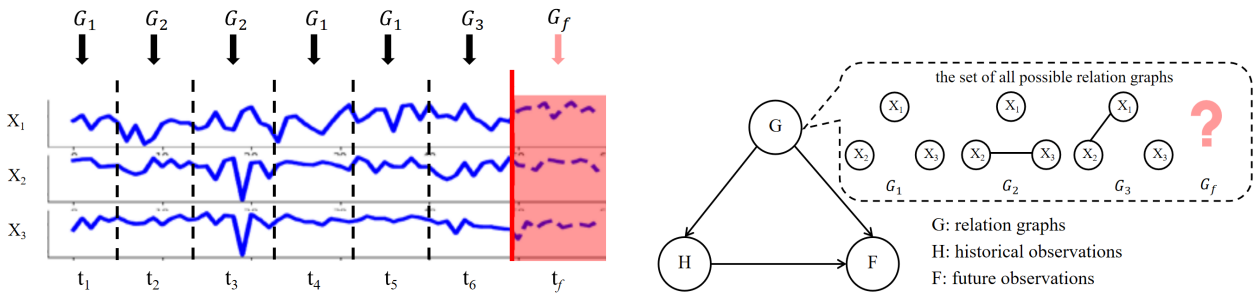
Table 1: The influence of dynamic relation graphs.

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

Early methods [19, 32, 41, 43] forecast the future observations of time series by applying different machine learning models on the historical observations, which aim to learn the *temporal dependencies*. Some works [22, 37] study time series forecasting in the traffic domain with different kinds of Graph Neural Network (GNN) [13, 18, 34] by learning the correlations among the time series of different locations based on their spatial distance in the transport network, which is called the *spatial dependencies*, and use Recurrent Neural Network (RNN) [11] or Transformers [33] to learn temporal dependencies. More recently, a new trend is to employ graph learning [36] to learn a *relation graph* that models correlations among multiple time series without requiring spatial distance in advance to enable time series forecasting. For example, STFGNN [21] and DGSL [30] employ graph learning methods to construct a relation graph, where time series are considered as nodes, and two time series are connected by an edge if they are similar in history. And then these methods predict the future observations with RNN and GNN using the learned relation graph.

Although the state-of-the-art approaches [21, 36] learn a relation graph to capture the correlations among multiple time series, which makes themselves fit well in applications where spatial distances are unavailable, they could only predict the future observations of multiple time series using a single, historical relation graph. However, as the correlations evolve across time, the single, historical relation graph is insufficient to model the dynamic correlations.

Figure 1(a) illustrates three time series that record traffic flows at three locations in different districts, *e.g.,* a commercial district $\mathbf{X}_1$, a residential district $\mathbf{X}_2$ and an industrial district $\mathbf{X}_3$. For example, on the peak hours of workdays $\mathbf{t_2}$~ $\mathbf{t_3}$, the traffic flows may be more correlated between the residential district $\mathbf{X}_2$ and the industrial district $\mathbf{X}_3$. While on the weekends $\mathbf{t_6}$, the traffic flows may be more correlated between the commercial district $\mathbf{X}_1$ and the residential district $\mathbf{X}_2$. 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. Thus, we derive a set of relation graphs to present the dynamic correlations, including both the historical and future relation graphs. We argue that the prediction of future observations are influenced by the set of dynamic relation graphs, as show in Table 1, where two traffic datasets [22] are used. In particular, DCRNN [22] uses a static relation graph which is constructed by the distance among locations for all timestamps, AGCRN [1] learns a static relation graph which models the similarities among multiple time series used for all timestamps, and DCRNN-D and AGCRN-D use a set of



(a) An example of time series for traffic flows at three locations

(b) The structural causal model

Figure 1: Motivations.

relation graphs constructed for each timestamp, which capture the degree to which two time series vary together.

Thus, we could model the relationship among historical observations **H**, future observations **F**, and the dynamic relation graphs **G** using a structural causal model [25] as shown in Figure 1(b), where an arrow indicates that there exists some influence. For example, the historical observations **H** effect the future observations **F**. Meanwhile, the set of relation graphs **G**, which capture the dynamic correlations among multiple time series during different intervals, affect the historical observations **H** and also the future observations **F**, as discussed earlier. Existing approaches only consider the direction from **H** to **F** and **G** to **H**, *i.e.,* the influence from the historical observations to the future observations and the influence from the historical relation graphs to the historical observations, but fail to take the influence from the future relation graphs **G** to future observations **F** into account. However, it is non-trivial to capture the dynamic relation graphs **G**, and to use both dynamic relation graphs **G** and historical observations **H** to predict the future observations **F**.

Challenge 1: It is challenging to model the dynamic relation graphs **G** that evolve across time, and utilize them together with the historical observations **H**. Existing studies [1, 21, 30, 36] fail to learn dynamic relation graphs across time, and only construct a single relation graph that captures static correlations among multiple time series. Specifically, these methods construct a relation graph by learning the similarities among multiple time series from historical observations. It is possible for ESG model [7, 39] to cut the whole historical observations into sub time-windows, learn a historical relation graph for each historical time-window. However, all the existing methods, do not demonstrate the ability to learn the future relation graphs and utilize them to predict future observations **F**, as the future observations are unknown. On the other hand, even if we could learn the future relation graphs, it is difficult for existing methods to model the historical observations together with the historical and future relation graphs, as the GNN in existing methods [1, 7, 36, 39] can only model the historical observations with a single graph, and fail to learn neural networks that could extract features of the historical and future relation graphs both.

Challenge 2: It is challenging to use the dynamic relation graphs to forecast the future observations. The existing methods [1, 22, 36, 37] learn patterns from historical observations with a given or learned static relation graph, and predict future observations by assuming that they follow similar patterns to the historical observations, which is the *perception ability* [3, 29]. However, these methods fail to predict the future observations correctly under the dynamic relation graphs, as the future parts of dynamic relation graphs are uncertain and may be unseen before. Thus, it's impossible to use the perception ability, as did by the existing methods, to derive a new pattern of future observations under unseen future relation graphs. Therefore, it calls for the *cognition ability* [3, 29] that handles the situations that are unseen before, which cannot be done by existing methods.

Based on the above analysis, in this paper, we propose a novel approach: Time Series Reasoning Networks with Causality (TSRN-C) for the multiple time series forecasting task to solve the aforementioned problems and challenges.

For **Challenge 1**, we cut the time series into historical and future time-windows, for each of which we build a relation graph distribution. It not only learns the historical relation graphs but also predicts the future relation graphs by optimizing the correlation coefficients from an empirical covariance matrix. Further, different from traditional GNN, which only models with a static graph, we propose a causal graph neural network, which models the observations of multiple time series with both historical relation graphs and future relation graphs into a feature vector at each timestamp, so as to enable us to use the historical observations **H** together with the dynamic relation graphs **G** to predict future observations **F**.

For **Challenge 2**, we propose a reasoning network with the logical operations, *i.e.,* conjunction (*AND*, $\wedge$), negation (*NOT*, $\neg$) and material implication ($\rightarrow$), and symbolic reasoning procedure to explicitly learn how multiple time series evolve among feature vectors across timestamps. First, we learn feature vectors, say h_a and h_b, as the representations of combining the historical observations with the dynamic relation graphs learned by the causal GNN at timestamps a and b. Next, We use a reasoning procedure, $(h_a \wedge h_b) \rightarrow h_c = TRUE$, to achieve the cognition ability by learning how the multiple time series could get into the current feature h_c under the historical features h_a and h_b. In this way, we learn the evolving process of multiple time series based on features. Thus, instead of learning patterns from historical observations to predict the future observations, whose pattern may be different under the unseen future relation graphs, our model can predict the future observations based on the future features which can be explicitly reasoned even given the unseen future relation graphs.

Table 2: The notations.

Notation	Explanation
N	The number of multiple time series
$\mathbf{X}_t^i$	The value of i-th time series at time step t
$\hat{\mathbf{X}}_t^i$	The predicted value of i-th time series at time step t
G_t	A relation graph denoting the correlations among time series
$\star_{G_t}$	The causal graph nueral network using graph G_t
v_t^i	The feature vector for i-th time series at time step t, which is used for causal GNN
τ	The max previous time steps used in reasoning
c_k	The learnable embedding matrix which denote the relative time interval, used for reasoning with the k-th previous time step
h_t^i	The hidden state for i-th time series at time step t, which is output by reasoning network
$M_{i,j}$	When M is a matrix, $M_{i,j} = M(i, j)$ is the element of M at row i and column j
M^k	When M is a square matrix, M^k is M to the power of k
d^l	The dimension of the hidden vector in layer l
$\|\|$	The concatenation operation

To the best of our knowledge, this is the first work that considers to predict future relation graphs for multiple time series, and use them in multiple time series forecasting problem. And we summarize contributions as follows.

- We propose to build dynamic relation graphs by learning the historical relation graphs and predicting the future relation graphs, and propose a causal GNN to model the observations with both historical and future relation graphs into feature vectors.
- We propose a reasoning network with logical operations to learn how the multiple time series evolve among feature vectors to achieve the cognition ability, and predict the future observations based on reasoning the future feature vectors.
- 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

In this section, we formalize the multiple time series forecasting problem and introduce the basic concepts on reasoning. The important notations used in this paper are summarized in Table 2.

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 $\mathbf{X}_{t:t+t_\Delta}^i \in \mathbb{R}^{t_\Delta}$ 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 $1 \le i \le N$ and $1 \le t, t+t_1 \le L$.

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 $q \ge 1$. 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 relations among time series at timestamp t is represented as a 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 multiple time series evolve among hidden states. For time series forecasting:

- Each hidden state is a variable, such as h_a which represents the observations of multiple time series at one timestamp a.
- A single operation over hidden states is called a *clause*. For example, $(h_a \wedge h_b)$ is a clause, which indicates that multiple time series have had the hidden states h_a and h_b during two timestamps a and b.
- Operations over clauses, such as $(h_a \wedge h_b) \wedge h_c$, is called an *expression*.
- When the expression has the material implication ($\rightarrow$) operation, it is called a *Horn clause*, *e.g.,* $(h_a \wedge h_b) \rightarrow h_c$. And the reasoning result of $(h_a \wedge h_b) \rightarrow h_c$ is *TRUE* can indicate that the historical states h_a and h_b together bring out the future state h_c, where $a < b < c$.

3 METHODOLOGY

As shown in Figure 2, 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 $\hat{\mathbf{X}}_{t'}$.

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 separately.

The causal graph layer contains a dynamic relation graph learning module and a causal graph convolution network. Specifically, the former learns a historical relation graph distribution P_{G_H} to capture the correlations among observations in the p historical

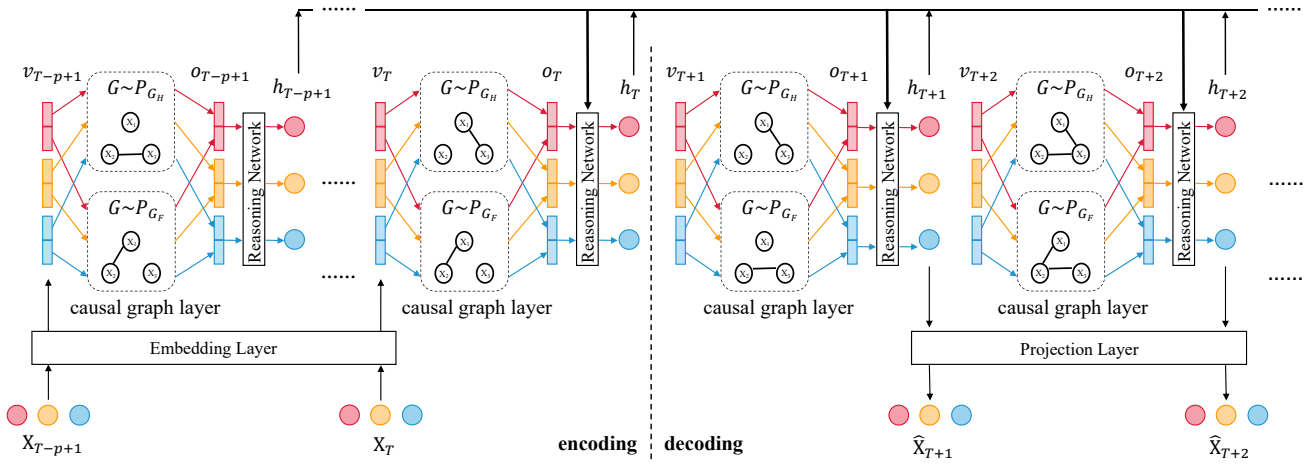


Figure 2: The overall framework.

timestamps, and predicts a future relation graph distribution P_{G_F} to capture the possible correlations among observations in the q future timestamps. For each timestamp t, with $t \in [T-p+1, T+q]$, the latter samples a graph from P_{G_H} and P_{G_F}, 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 possible correlations.

At each timestamp t, with $t \in [T-p+1, T+q]$, the reasoning network learns a feature vector h_t from the current hidden state o_t and the previous τ feature vectors $(h_{t-\tau}, \cdots, h_{t-2}, h_{t-1})$. The purpose of feature vector h_t is to identify in which way its past τ timestamps influence the predication at the current timestamp t, such that the feature vector h_t contains the information of how multiple time series evolve among hidden state sequentially.

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

3.1 The Embedding Layer

The embedding layer maps the original inputs to the high dimensional representations, so as to extract the feature for each time series at each timestamp. More generally, besides the historical observations of multiple time series, *i.e.,* $\mathbf{X}_{T-p+1:T}$, the inputs can also be coupled with other auxiliary attributes, such as the weather and temperature for each time series. The auxiliary attributes for the i-th time series at timestamp t is represented as $\mathbf{a}_t^i \in \mathbb{R}^F$, where F is the total dimensions of the auxiliary attributes. Thus, the embedding layer concatenate the historical observation and the auxiliary attributes $f_t^i = (\mathbf{X}_t^i || \mathbf{a}_t^i)$ as its input, and maps it to the representation vector for each time series at each timestamp by:

$$v_t^i = \sigma(W_e f_t^i), \quad for\ 1 \le i \le N, t-p+1 \le t \le t, \tag{2}$$

where $v_t^i \in \mathbb{R}^{d^e}$ is the learned feature vector, σ is an activation function (*e.g.,* the sigmoid function), and $W_e \in \mathbb{R}^{d^e \times (F+1)}$ is the learnable network parameters which aims to extract the feature for each observation of each time series. Then the representation vectors of all time series, $v_t = (v_t^1, v_t^2, \cdots, v_t^N) \in \mathbb{R}^{N \times d^e}$, are used in the causal graph layer for each timestamp t.

3.2 The Causal Graph Layer

We first theoretically show the foundation of why and how to learn the dynamic relation graphs. Then we introduce our probabilistic model which not only learns the historical relation graphs but also predicts the future relation graphs. Last, 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.

3.2.1 The foundation. Existing approaches [19, 22, 41] extract the features from historical observations to forecast future observations. Following the probability theory, we model the forecasting process by a conditional probability. Specifically, we use a random variable $\mathbf{H}$ to denote the event that a sequence of historical observations happened in the past p timestamps. Random variable $\mathbf{F}$ denotes the event that a sequence of future observations will happen in the future q timestamps. Random variable $\mathbf{G}$ denotes the event that dynamic relation graphs appear. Next, we model the relations between $\mathbf{H}$ and $\mathbf{F}$ by the likelihood of conditional probability $P(\mathbf{F}|\mathbf{H})$, which means the conditional probability distribution of the uncertain future observations under the uncertain historical observations, and derive the forecasting results of these existing methods as:

$$\begin{aligned} \hat{\mathbf{X}}_{T+1:T+q} &= \mathcal{F}_\theta(\mathbf{X}_{T-p+1:T}) \\ &= \underset{\mathbf{F}}{argmax}\, P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}) \end{aligned} \tag{3}$$

By applying the Bayes rule, we can see that the dynamic relation graphs will bias the forecasting results. This is because by the Bayes rule we can get:

$$\begin{aligned} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}) &= P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, \mathbf{G}) P(\mathbf{G}|\mathbf{H} = \mathbf{X}_{T-p+1:T}) \\ &= \sum_{G_t} P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T}, G_t) P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}), \end{aligned} \tag{4}$$

where G_t is any possible relation graph for these multiple time series. As it is certain that the historical observations $\mathbf{X}_{T-p+1:T}$ have been effected by the historical correlations, which are modeled by the set of historical relation graphs G_H (*c.f.* Figure 1, where

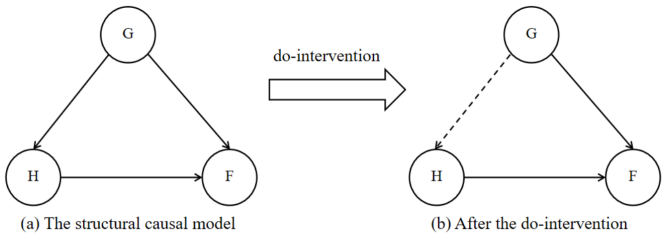


Figure 3: The do-intervention for node H.

$G_H = \{G_1, G_2, G_3\}$), we have:

$$\begin{aligned} &P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) \neq 0, \quad \forall t \in [T-p+1, T], \\ &P(\mathbf{G} = G_t|\mathbf{H} = \mathbf{X}_{T-p+1:T}) = 0, \quad \forall t \notin [T-p+1, T]. \end{aligned} \tag{5}$$

Combining Equations (4) and (5), we can get $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ as:

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

From Equation (6), we can see that $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ can only learn within the recent historical relation graphs, *i.e.,* G_t with $t \in [T-p+1, T]$. Thus, the model is highly biased by these recent relation graphs and fails to learn from the possible future relation graphs, though the correlations among multiple time series change across time and then influence the future observations eventually.

In order to utilize future relation graphs in forecasting future observations, it is necessary to construct some kind of relationship between **H** and **F**, which preserves and considers future relation graphs. However, conditional probability fails here. The problem of modeling a forecasting problem using $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ is that the conditional probability only captures the possibility that future observations will happen when historical observations happen. Therefore, it does not demonstrate the capability to utilize future relation graphs.

Thus, we consider to use a structural causal model [25], as shown in Figure 3(a), to model the causal relationship among historical observations **H**, future observations **F**, dynamic relation graphs **G** that capture the historical and future correlations among multiple time series. This provides future relation graphs an opportunity to contribute in forecasting future observations. In the structural causal graph, an arrow from a node to another indicates a causal relationship and means that a random variable is the cause and will effect another. For example, as future observations could happen only until historical observations have occurred, there is an arrow from node **H** to node **F**, which can be reflected by $P(\mathbf{F}|\mathbf{H})$. The dynamic correlations among multiple time series modeled by a set of relation graphs are the cause of the development of observations (*c.f.* Figure 1(a)). Thus, the arrow from **G** to **H** means that the historical correlations result in the historical observations, which can be reflected by $P(\mathbf{H}|\mathbf{G})$, and the arrow from **G** to **F** means that the future correlations result in the future observations, which can be reflected by $P(\mathbf{F}|\mathbf{G})$.

However, as $P(\mathbf{F}|\mathbf{H}$ still exists in the structural causal model, the issue with bias is not solved. Therefore, in order to avoid the bias in the conditional probability $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$ that only the recent historical dynamic graphs are utilized, we introduce a *do-intervention* event for node **H** in our structural causal model as shown in Figure 3(b), where we no long model the influences from the historical correlations to the historical observations by removing the arrow from node **G** to node **H**. The do-intervention event means that the historical observations $\mathbf{X}_{T-p+1:T}$ have already occurred under the condition of historical correlations among multiple time series. As shown in Figure 3(a), the influence from the historical correlations among multiple time series to the historical observations has already happened before we use the historical observations to predict the future observations. This influence is certain and is contained by the historical observations $\mathbf{X}_{T-p+1:T}$, so we can do intervention for node **H**.

Thus, instead of modeling the probability $P(\mathbf{F}|\mathbf{H} = \mathbf{X}_{T-p+1:T})$, we model $P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T})$ as the probability that future observations **F** will happen given the condition that it is certain for the historical observations $\mathbf{X}_{T-p+1:T}$ to occur. Thus, we have:

$$\begin{aligned} &P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T})) \\ &= P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T}), \mathbf{G}) P(\mathbf{G}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T})) \\ &= P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T}), \mathbf{G}) P(\mathbf{G}) \\ &= \sum_{G_t} P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T}), G_t) P(\mathbf{G} = G_t), \end{aligned} \tag{7}$$

where $P(\mathbf{G}|do(\mathbf{H} = \mathbf{X}_{t-p+1:t})) = P(\mathbf{G})$ because the random variable **G** is independent to the event $do(\mathbf{H} = \mathbf{X}_{T-p+1:T})$. Thus, we are ready to utilize historical and future relation graphs, in order to forecast future observations.

Following Equation (7), to forecast the future observations of multiple time series using the historical observations without the bias caused by the historical part of dynamic relation graphs, we should firstly learn a probability distribution of relation graphs at the historical and future time-windows, respectively, as presented in Section 3.2.2, and then model the influence by combining the historical observations with the each possible relation graph for predicting the future observations, *i.e.,* $P(\mathbf{F}|do(\mathbf{H} = \mathbf{X}_{T-p+1:T}), G_t)$, as presented in Section 3.2.3.

3.2.2 Learn the dynamic relation graphs. Based on the above analysis, we construct 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 $G_t \sim P_{G_H}$, where G_t is a possible relation graph at timestamp t, with $t \in [T-p+1, T]$, and denote the probability distribution of relation graphs at the future time-window, which ranges from timestamp $T+1$ to $T+q$, as $G_{t'} \sim P_{G_F}$, where $G_{t'}$ is a possible relation graph at timestamp t', with $t' \in [T+1, T+q]$.

In this work, we use the parameterized Bernoulli distribution to model the probability distribution of relation graphs for the historical and future time-windows. Specifically, P_{G_H} is defined as follows for any two of the i-th and j-th time series:

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

where $\mathcal{A}$ is the adjacency matrix for a relation graph G_t. 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_F} is defined as follows:

$$P(\mathcal{A}_{ij} = 1) = P_f(i,j),\ P(\mathcal{A}_{ij} = 0) = 1 - P_f(i,j). \tag{9}$$

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:

$$S_h = \frac{1}{p}(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})(\mathbf{X}_{T-p+1:T} - \overline{\mathbf{X}}_{T-p+1:T})^\top, \tag{10}$$

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:

$$\rho_h(i,j) = \frac{S_h(i,j)}{\sqrt{S_h(i,i)S_h(j,j)}},\ \ for\ 1 \leq i,j \leq N, \tag{11}$$

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_H}, as follows:

$$P(\mathcal{A}_{ij} = 1) = \begin{cases} P_h(i,j) = |\rho_h(i,j)| & if\ |\rho_h(i,j)| \geq \delta \\ 0 & if\ |\rho_h(i,j)| < \delta \end{cases},$$
$$P(\mathcal{A}_{ij} = 0) = \begin{cases} 1 - P_h(i,j) = 1 - |\rho_h(i,j)| & if\ |\rho_h(i,j)| \geq \delta \\ 1 & if\ |\rho_h(i,j)| < \delta \end{cases}, \tag{12}$$

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 at the future time-window. We use 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 $|\hat{\rho}_f|$.

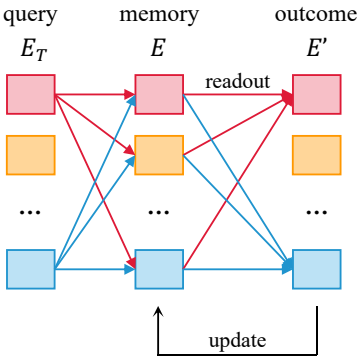


Figure 4: The memory network.

As shown in Figure 4, we propose a memory network to predict the probability distribution for the future time-window. The matrix $E_T \in \mathbb{R}^{N\times d^c}$ contains a representations for each time series extracted from the observations in the historical time-window. We could only utilizes the local features that are extracted from the historical time-window with p timestamps to predict the future correlation coefficient matrix $\hat{\rho}_f$ with E_T. However, it prevents an accurate prediction of the future distribution, as the correlations in the future time-window may be different from those in the historical time-window [12]. Therefore, we use an additional memory unit $E \in \mathbb{R}^{N\times d^c}$ to preserve the global features. The basic idea is that we use a memory unit to record the correlations that have appeared among multiple time series across all timestamps, *i.e.,* before the historical window $[T-p+1, T]$, which can help to predict the relation graphs for the future time-window, 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 $m^i_T \in \mathbb{R}^{d^c}$:

$$m^i_T = \sigma(W_c \mathbf{X}^i_{T-p+1:T}), \tag{13}$$

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:

$$\rho(i,j) = m^i \cdot m^j \tag{14}$$

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 unit E as follows:

$$\begin{aligned} E' &= readout(\mathbf{Q},\mathbf{K},E) = \varphi(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d^c}})E, \\ \mathbf{Q} &= E_t W_\mathbf{Q},\ \mathbf{K} = E\, W_\mathbf{K}, \end{aligned} \tag{15}$$

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:

$$\hat{\rho}_f(i,j) = E'(i)\cdot E'(j) \tag{16}$$

Last, we update the memory unit E as follows:

$$E = \beta E + (1-\beta)E' \tag{17}$$

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:

$$\mathcal{L}_{graph} = \frac{1}{N\times N}\sum_{1\le i,j\le N}\Big(\hat{\rho}_f(i,j) - \rho_f(i,j)\Big)^2, \tag{18}$$

where:

$$\begin{aligned} S_f &= \frac{1}{p}(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})(\mathbf{X}_{T+1:T+q} - \overline{\mathbf{X}}_{T+1:T+q})^\top, \\ \rho_f(i,j) &= \frac{S_f(i,j)}{\sqrt{S_f(i,i)S_f(j,j)}},\ \ for\ 1\le i,j\le N \end{aligned} \tag{19}$$

Similar to Equation (12), we can obtain the probability distribution of relation graphs at the future time-window, denoted as P_{G_F}.

3.2.3 Causal graph neural network. Based on Equation (7), we should combine the historical observations together with the possible relation graphs to predict the future observations. Thus, at each timestamp t, with $t \in [T-p+1, T]$, we sample a relation graph according to the probability distribution P_{G_H} and P_{G_F}, respectively. Then, we propose a causal graph neural network to combine the feature vector v_t, which contains information of the historical observations extracted at the embedding layer, with the sampled historical and future relation graphs into the hidden state vector o_t for each timestamp t. Thus, the features from the historical observations are well integrated with the possible historical and future relation graphs, to enable the future observation forecasting.

First, from each feature vector v_t^i, 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 each time series i at timestamp t, which are 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. Next, we get $v_{ht} = (v_{ht}^1, v_{ht}^2, \cdots, v_{ht}^N) \in \mathbb{R}^{N\times\frac{d^e}{2}}$ and $v_{ft} = (v_{ft}^1, v_{ft}^2, \cdots, v_{ft}^N) \in \mathbb{R}^{N\times\frac{d^e}{2}}$ as features for all time series at timestamp t.

For each timestamp t, we denote the sampled relation graphs as G_{ht} and G_{ft}, and their adjacency matrix as $\mathcal{A}_{ht}$ and $\mathcal{A}_{ft}$. 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. As the dynamic relation graphs, which are modeled by the probability distribution P_{G_H} and P_{G_F}, can contain many possible relation graphs, the sampled relation graphs G_{ht} and G_{ft} may be different across time. Thus, it is difficult for existing GNN methods [18, 34] to learn with dynamic relation graphs, as they fail to learn neural network parameters for each of new graphs. Therefore, we propose a causal GNN to deal with the historical and future relation graphs that can change across time. 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:

$$\begin{aligned} o_{ht} &= \star_{G_{ht}}(v_{ht}) \\ &= \sum_{k=0}^{K}\frac{1}{k+1}\Big\{(\widetilde{\mathcal{A}_{ht}})^k\, v_{ht}\, W_{1,k} + (\widetilde{\mathcal{A}_{ht}^\top})^k\, v_{ht}\, W_{2,k}\Big\}, \end{aligned} \tag{20}$$

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

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

and K is a hyper-parameter which controls the max hop neighbors used in causal convolution, the weight $\frac{1}{k+1}$ denotes that the information from near hop neighbors are important than that from far hop neighbors, and $W_{1,k}, W_{2,k} \in \mathbb{R}^{\frac{d^e}{2}\times\frac{d^e}{2}}$, with $k \in [0,K]$, are the learnable parameters which extract the feature from k-th hop neighbors into the output $o_{ht} \in \mathbb{R}^{N\times\frac{d^e}{2}}$. Thus, our causal GNN can learn unified neural network parameters $W_{1,k}, W_{2,k}$ for different relation graphs, as $W_{1,k}, W_{2,k}$ learn to extract features by the hop that used for any graphs.

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 $o_t = (o_{ht}||o_{ft})$.

3.3 The Reasoning Network

Until now we have got the hidden state o_t, which contains the features for the multiple time series at each timestamp t separately. Now, we propose a reasoning network, which learns a feature vector h_t from the current hidden state o_t and 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 how the multiple time series get into the current feature hidden state o_t, by identifying in which way its past τ timestamps influence the predication at the current timestamp t, where τ is a hyper-parameter which controls the max previous timestamps used in the reasoning network. 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 $1 \le k \le \tau$.

As shown in Figure 5, the reasoning network firstly combines each feature vector h_{t-k} with a positional embedding c_k, which

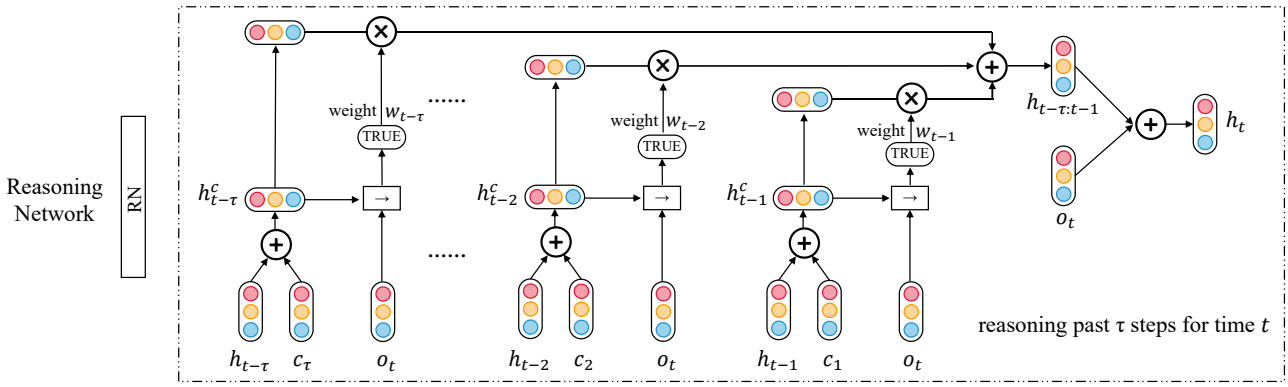


Figure 5: The reasoning network will output the hidden state for multiple time series at each timestamp t using the previous states from past τ steps.

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:

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

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. First, as 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 get into hidden state o_t, 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, the operation $\rightarrow$ denotes whether all the previous feature vectors result in the current hidden state o_t, and $\neg o_t$ denotes not the hidden state o_t, which 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 $\neg o_t$.

However, as the order does not matter in the horn clause, the conjunction operation $\wedge$ cannot model the sequential information, *i.e.,* time-dependent previous feature vectors h_{t-k}. Thus, the horn clause $(h_{t-1} \wedge h_{t-2} \cdots \wedge h_{t-\tau}) \rightarrow o_t$ is also *TRUE*, which means that the multiple time series will also get into hidden state o_t under the previous feature vector $(h_{t-1}, h_{t-2}, \cdots, h_{t-\tau})$ sequentially, which is incorrect.

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. 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 by:

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

and model the time-dependent sequential information by:

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

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 and h_b 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:

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

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:

$$\begin{aligned}\mathbf{Exp}_{\tau} &= h^c_{t-\tau} \wedge h^c_{t-\tau+1} = (h^c_{t-\tau} \odot h^c_{t-\tau+1})W_a \\ \mathbf{Exp}_{\tau-1} &= \mathbf{Exp}_{\tau} \wedge h^c_{t-\tau+2} = (\mathbf{Exp}_{\tau} \odot h^c_{t-\tau+2})W_a \\ &\cdots\cdots \\ \mathbf{Exp}_{2} &= \mathbf{Exp}_{3} \wedge h^c_{t-1} = (\mathbf{Exp}_{3} \odot h^c_{t-1})W_a \\ \mathbf{Exp}_{+} &= \mathbf{Exp}_{2} \rightarrow o_t = (\mathbf{Exp}_2 || o_t)W_i\end{aligned} \tag{27}$$

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 for our reasoning network, which is achieved by minimizing the loss function as below:

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

which denotes that $isT(\mathbf{Exp}_+)$ should be 1 and $isT(\mathbf{Exp}_-)$ should be 0.

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.

Last, 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), such that h_t contains the feature of how multiple time series get into the current hidden state o_t based on the importance of each previous feature vector h_{t-k}.

3.4 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:

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

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.

We use the objective function to evaluate the discrepancy between the forecasting observations and the ground truth, and 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 $\|M\|^1 = \sum_{i,j}|M_{i,j}|$. Finally, our model is optimized by minimizing the loss function:

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

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

4 EXPERIMENTS

In this section, we empirically evaluate TSRN-C 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.

4.1 Experimental Settings

4.1.1 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. [22]. 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 PeMSD4 dataset contains the traffic flow measured by 307 sensors in the San Francisco Bay Area ranging from Jan. 2018 to Feb. 2018. The PeMSD8 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. [19].
- Electricity: The Electricity energy dataset contains the electricity consumption records collected from 321 clients from 2012 to 2014, released by Lai et al. [19].

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, 19, 22], as shown in the "Split" column. The four traffic datasets have pre-defined graphs where each node represents a time series and the adjacency matrix represents the

Table 3: The statistics of datasets.

Dataset	N	L	Split	Input	Output
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

road network distances among time series. The two energy datasets have no pre-defined graphs.

Following existing works [1, 19, 21–23, 36], we adopt a multi-step forecasting setting for the four traffic datasets, where we use 12 historical timestamps to forecast the observations for the next all 12 timestamps. We adopt a single-step forecasting setting for the two energy datasets, where we use 168 historical timestamps to forecast the observations for the next 3rd timestamp and 12th timestamp, respectively.

To enable direct and fair comparisons with existing works [1, 19, 21–23, 36], for METR-LA and PEMS-BAY, we report the accuracy of the forecast on the next 3rd, 6th, and 12th timestamp, respectively; for PEMS04 and PEMS08, we report the average accuracy over the next all 12 timestamps; for Solar-Energy and Electricity, we report the accuracy of the forecast on the next 3rd and 12th timestamp, respectively.

4.1.2 Evaluation Metrics. Following the evaluation methodology in existing works [1, 19, 22], 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.

4.1.3 Baselines. We compare TSRN-C with baseline methods summarized as follows:

- **Methods learn temporal dependencies only.** VAR-MLP: An auto-regressive model with multilayer perception (MLP) [41]. GP: A time series forecasting model with Gaussian Process [32]. LSTNet: A deep neural network model, which combines convolutional neural networks (CNN) with RNN to extract short-term and long-term temporal dependencies [19]. TPA-LSTM: An attention [43] based model [31].
- **Methods learn spatial dependencies with a pre-defined graph.** DCRNN: An auto-regressive model with encoder-decoder architecture, which proposes diffusion graph convolutions to extract spatial dependencies [22]. GWave: It combines diffusion graph convolutions with 1D dilated CNN [37]. AGCRN: It proposes adaptive recurrent graph convolution network [1]. MSDR: An auto-regressive model, which proposes attention based graph convolutions and multi-step RNN to capture long-range temporal dependencies [23].
- **Methods learn relation graph for multiple time series.** MTGNN: It learns a static relation graph that models similarities among multiple time series, and uses graph convolutions and CNN for forecasting [36]. DGTS: It constructs a static relation graph based on the Euler distances among multiple time series, and uses recurrent graph convolutions for forecasting [30]. STFGNN: It constructs a static relation graph based on the Dynamic Time Warping similarities [17] among multiple time series, and fuses the relation graph with the pre-defined road network graph into a fusion graph [21]. 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 [39].

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.

4.1.4 Experimental Details. We conduct grid search on the held-out validation set for each method and dataset to decide its hyper-parameters. Specifically, We optimize with Adam optimizer for a maximum of 200 epochs and use the early stop strategy with patience of 10. The learning rate is tuned from 0.0005, 0.001, 0.0015 and the batch size is tuned from 64, 128. The dimension of hidden state is tuned from 64, 128. The δ, which controls the sparsity of the relation graph, is tuned from 0.1, 0.3, 0.5. The K, which controls the max hop neighbors used in causal convolution, is tuned from 1, 2. The τ, which controls the max previous timestamps used in the reasoning network, is tuned from 3, 6, 12. The β used in the memory unit is set to 0.99, and the λ used in the loss function is set to 0.01. We repeat each experiment 5 times and report the average accuracy result. The code and data are available at https://anonymous.4open.science/r/TSRN-C-BEE8.

4.2 Experimental Results

4.2.1 Overall Comparison. Tables 4 and 5 present the accuracy of TSRN-C and the baselines on all datasets. We use bold to highlight the best accuracy, which significantly outperforms the underline second best accuracy based on paired t-test at the significance level of 0.01.

Key observations are as follows. First, TSRN-C consistently outperforms the state-of-the-art baseline methods on all datasets, under both multi-step forecasting and single-step forecasting settings. It demonstrates that TSRN-C is able to learn the dynamic correlations among multiple time series and use them to improve the forecasting performance.

Second, from Table 5 we observe that the MTGNN, DGTS, ESG and TSRN-C methods, which can learn the relation graph(s) for multiple time series, perform better when comparing to VAR-MLP, GP, LSTNet and TPA-LSTM, which cannot explicitly capture the relations among multiple time series. It demonstrates that explicit learning the relation graph to capture the relations among multiple time series is important for multiple time series forecasting.

Third, from Table 4 we observe that our TSRN-C method is also superior when comparing to the other graph based methods, which use a single relation graph or only learn historical relation graphs for multiple time series, 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 kinds of future correlations may affect on the future observations differently. A

Table 4: Accuracy of multi-step forecasting.

Dataset	timestamp	Metric	DCRNN	GWave	AGCRN	MTGNN	DGTS	STFGNN	MSDR	ESG	TSRN-C
METR-LA	3rd	MAE	2.77	2.69	2.83	2.69	2.75	2.68	2.71	2.68	**2.60**
		RMSE	5.38	5.15	5.45	5.18	5.37	5.17	5.19	5.15	**5.10**
		MAPE	7.30%	6.90%	7.56%	6.86%	7.22%	6.88%	7.08%	6.93	**6.74%**
	6th	MAE	3.15	3.07	3.20	3.05	3.12	3.05	3.09	3.06	**2.97**
		RMSE	6.45	6.22	6.55	6.17	6.39	6.18	6.33	6.19	**6.11**
		MAPE	8.80%	8.37%	8.79%	8.19%	8.61%	8.21%	8.57%	8.20%	**8.12%**
	12th	MAE	3.60	3.53	3.58	3.49	3.55	3.48	3.50	3.49	**3.34**
		RMSE	7.60	7.37	7.41	7.23	7.47	7.27	7.33	7.23	**7.15**
		MAPE	10.50%	10.01%	10.13%	9.87%	10.10%	9.86%	9.98%	9.96%	**9.51%**
PEMS-BAY	3rd	MAE	1.38	1.30	1.35	1.32	1.36	1.31	1.32	1.31	**1.28**
		RMSE	2.95	2.74	2.83	2.79	2.81	2.75	2.84	2.74	**2.65**
		MAPE	2.90%	2.73%	2.87%	2.77%	2.87%	2.72%	2.77%	2.76%	**2.61%**
	6th	MAE	1.74	1.63	1.69	1.65	1.71	1.63	1.64	1.63	**1.57**
		RMSE	3.97	3.70	3.81	3.74	3.86	3.69	3.78	3.71	**3.60**
		MAPE	3.90%	3.67%	3.84%	3.69%	3.85%	3.66%	3.68%	3.69%	**3.51%**
	12th	MAE	2.07	1.95	1.96	1.94	2.00	1.93	1.94	1.92	**1.83**
		RMSE	4.74	4.52	4.52	4.49	4.53	4.45	4.51	4.42	**4.32**
		MAPE	4.90%	4.63%	4.67%	4.53%	4.71%	4.49%	4.55%	4.52%	**4.31%**
PEMS04	1~12	MAE	24.70	19.16	19.83	19.32	19.50	19.83	19.29	19.47	**18.49**
		RMSE	38.12	30.46	32.26	31.57	32.00	31.88	31.54	31.66	**30.13**
		MAPE	17.12%	13.26%	12.97%	13.52%	14.04%	13.02%	12.89%	13.30%	**12.62%**
PEMS08	1~12	MAE	17.86	15.13	15.95	15.71	15.88	16.64	15.11	15.70	**14.74**
		RMSE	27.83	24.07	25.22	24.62	25.07	26.22	24.42	24.81	**23.56**
		MAPE	11.45%	10.10%	10.09%	10.03%	10.17%	10.60%	9.93%	10.07%	**9.38%**

Table 5: Accuracy of single-step forecasting.

Dataset		Solar-Energy		Electricity	
		timestamp		timestamp	
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-LSTM	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
ESG	RRSE	0.1708	0.3073	0.0718	0.0898
	CORR	0.9865	0.9519	0.9494	0.9321
TSRN-C	RRSE	**0.1694**	**0.3022**	**0.0702**	**0.0880**
	CORR	**0.9876**	**0.9537**	**0.9503**	**0.9344**

single relation graph or the historical relation graph will bias the forecast as shown in Section 3.2.1. GWave, MTGNN, STFGNN, MSDR and ESG methods 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 TSRN-C can learn the historical and future correlations dynamically, and consistently outperforms baseline methods.

Fourth, TSRN-C achieves the best accuracy on both short (which use 12 historical timestamps to forecast future) and long (which use 168 historical timestamps to forecast future) datasets. This suggests that our reasoning network, which explicitly learn the evolving process for multiple time series recurrently, is also good at learning both short- and long-term temporal dependencies, which enables TSRN-C to get high-performance in both cases.

4.2.2 Ablation Studies. We conduct ablation studies to validate the effectiveness of our key components that contribute to the improvements.

In particular, we compare TSRN-C with the following variants:

- w/o memory network: This variant does not use the memory network for predicting the future relation graph distribution. It uses the local features E_T directly.
- G_{ht} only: This variant does not use causal GNN. It applies the existing GNN [22, 37] on the sampled historical relation graph G_{ht} only.
- G_{ft} only: This variant does not use causal GNN. It applies the existing GNN on the sampled future relation graph G_{ft} only.
- w/o reasoning network: This variant does not use reasoning network. It uses the GRU [1, 5, 22], a kind of RNN, 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

Table 6: Ablation Studies.

Dataset	PEMS04			PEMS08		
Method	MAE	RMSE	MAPE	MAE	RMSE	MAPE
TSRN-C	**18.49**	**30.13**	**12.62%**	**14.74**	**23.56**	**9.38%**
w/o memory network	18.66	30.24	12.74%	14.83	23.62	9.51%
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/o reasoning network	19.27	30.91	12.87%	15.38	24.53	10.12%

trends. From Table 6 we observe that: (1) TSRN-C 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) Simply using the existing GNN [22, 37] with a single historical relation graph or future relation graph will lower the performance significantly. This result is consistent with our analyses in Section 3.2.1, suggesting that our causal GNN, which can learn with historical relation graph and future relation graph jointly to avoid the bias, is more effective in multiple time series forecasting. (3) TSRN-C with reasoning network significantly outperforms the variant with GRU, which justifying that the RNN is insufficient to accurately forecast future observations under unseen future relation graphs, However, our reasoning network is able to do so, as it explicitly learns the evolving process of multiple time series and predict the future observations based on the future features which can be reasoned even given the unseen future relation graphs.

Experimental results and analyses regarding to parameter sensitivity and runtime are presented in the Appendix at https://anonymous.4open.science/r/TSRN-C-BEE8.

5 RELATED WORK

We review existing works on time series forecasting and relation graph learning.

5.1 Time Series Forecasting

Early methods try to utilize the statistical methods, *e.g.,* Auto-Regressive model (AR) [24, 41] and Gaussian Process model (GP) [32], to forecast on time series, which model the future observations as the linear combination of the nearby historical observations, called the *short-term temporal dependencies*.

Inspired by the ability of deep learning [11], Some works [6, 19, 27, 31, 43] proposed to utilize RNN [5], Temporal CNN [20] or attention network [33] for time series forecasting by modeling dependencies among farther historical observations, called the *long-term temporal dependencies*. LSTNet [19] are the first deep-learning-based method for time series forecasting, which employs CNN to capture correlations among variables and RNN to preserve long-term temporal dependencies. Triformer [6] proposes a triangular attention with linear complexity, which can capture temporal dependencies for very long range. However, these methods cannot capture the correlations among time series explicitly.

Then, there have been a lot of GNN based methods for traffic time series forecasting [2, 4, 10, 14, 28, 40, 42]. 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 from the neighbors on the spatial graph. DCRNN [22] proposes diffusion graph convolutions to extract spatial dependencies and use GRU for forecasting. Graph WaveNet [37] combines diffusion graph convolutions with gated dilated CNN. AGCRN proposes adaptive recurrent graph convolution network to capture node-specific features for each time series [1]. STA [8] propose a spatio-temporal aware unit which generates node-specific and time-varying features. MSDR [23] proposes attention based graph convolutions and multi-step RNN to capture temporal dependencies for very long range. However, these methods need a pre-defined graph to present correlations among time series in advance.

5.2 Relation Graph Learning

Most recently, a new trend is to employ graph learning [12] to learn relation graphs that models correlations among multiple time series without requiring spatial distance in advance to enable multiple time series forecasting. MTGNN [36], STFGNN [21] and DGSL [30] construct a static relation graph, where time series are considered as nodes, and two time series are connected by an edge if they are similar by the Cosine distances, Euler distances or Dynamic Time Warping distances [17]. AutoCTS [35] automatically search from RNN, CNN, GNN and attention network, to build a better deep neural network with the learned static relation graph to forecast. EnhanceNet [7] and ESG [39] cut the historical observations into sub time-windows, and construct a historical relation graph which is used in each time-window separately. After constructing the relation graph for multiple time series, they combine GNN with RNN or CNN for forecasting.

Although existing methods have achieved significant improvements, they are incapable of learning the dynamic correlations among multiple time series for the future, thus they are biased by the historical relation graphs.

6 CONCLUSION

We present TSRN-C for multiple time series forecasting. We propose to learn historical relation graphs and predicting future relation graphs to capture the dynamic correlations among time series, and propose a causal GNN to extract features from the observations with both historical and future relation graphs. Then we propose a reasoning network to learn how the multiple time series evolve among feature vectors, 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 TSRN-C to other time series tasks, such as abnormality detection and prediction.

REFERENCES

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

[2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. In *NeurIPS*.

[3] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *WWW'21*. ACM / IW3C2, 1516–1527.

[4] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. 2020. Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting. In *AAAI*. AAAI Press, 3529–3536.

[5] 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*. ACL, 1724–1734.

[6] 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*. ijcai.org, 1994–2001.

[7] 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*. IEEE, 1739–1750.

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

[9] Sayda Elmi and Kian-Lee Tan. 2021. DeepFEC: Energy Consumption Prediction under Real-World Driving Conditions for Smart Cities. In *WWW'21*. ACM / IW3C2, 1880–1890.

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

[11] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning*. MIT Press.

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

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

[14] Liangzhe Han, Bowen Du, Leilei Sun, Yanjie Fu, Yisheng Lv, and Hui Xiong. 2021. Dynamic and Multi-faceted Spatio-temporal Deep Learning for Traffic Speed Forecasting. In *SIGKDD*. ACM, 547–555.

[15] Jilin Hu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2018. Risk-aware path selection with time-varying, uncertain travel costs: a time series approach. *Proc. VLDB J.* 27, 2 (2018), 179–200.

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

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

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

[19] 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.

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

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

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

[23] Dachuan Liu, Jin Wang, Shuo Shang, and Peng Han. 2022. MSDR: Multi-Step Dependency Relation Networks for Spatial Temporal Forecasting. In *KDD '22*. ACM, 1042–1050.

[24] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. 2016. Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization. In *SIGKDD*. ACM, 1005–1014.

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

[26] 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.

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

[28] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In *SIGKDD*. ACM, 1720–1730.

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

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

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

[36] 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 *SIGKDD*. ACM, 753–763.

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

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

[39] 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 '22*. ACM, 2296–2306.

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

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

[42] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *AAAI*. AAAI Press, 1234–1241.

[43] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*. AAAI Press, 11106–11115.

A APPENDIX

In this section, we provide more details of our implementation and experimental results.

A.1 Implementation Details

All the model training experiments are conducted on a server running Ubuntu 18.04.5 LTS system, with Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz and NVIDIA Quadro RTX 8000 GPU. Baseline models in Python and Pytorch are open available from the original papers, and all the deep learning models are executed with Pytorch 1.2.0.

A.2 Parameter Sensitivity

We evaluate the impact of the key hyperparameter in TSRN-C, *i.e.*, τ, which controls the max previous timestamps used in the reasoning network. The experimental results on PEMS04 and PEMS08 datasets are shown in Table 7. We can see that as we use more previous timestamps in the reasoning network up to 6 timestamps, the TSRN-C model performs better. The reason is that for the traffic prediction task we need to learn the evolution process for a long timestamps range. Then when the value of τ varies from 6 to 12, we can see that the results are relatively stable.

Table 7: Parameter Sensitivity.

Dataset	PEMS04			PEMS08		
τ	MAE	RMSE	MAPE	MAE	RMSE	MAPE
2	19.84	31.22	12.94%	15.63	25.11	10.01%
3	19.31	30.89	12.77%	15.12	24.38	9.94%
4	19.09	30.74	12.78%	15.06	24.18	9.44%
5	18.58	30.31	12.67%	14.87	23.80	9.43%
6	18.49	30.13	12.62%	14.74	23.56	9.38%
12	18.51	30.18	12.63%	14.74	23.61	9.40%