Graph Convolutional Networks for Traffic Forecasting with Missing Values

Jingwei Zuo[1*], Karine Zeitouni[1], Yehia Taher[1] and Sandra Garcia-Rodriguez[2]

[1*]DAVID Lab, UVSQ, Université Paris-Saclay, Versailles, France.
[2]Data Analysis and Systems Intelligence Laboratory, CEA, LIST, Gif Sur Yvette, France.

*Corresponding author(s). E-mail(s): jingwei.zuo@uvsq.fr;
Contributing authors: karine.zeitouni@uvsq.fr;
yehia.taher@uvsq.fr; sandra.garciarodriguez@cea.fr;

Abstract

Traffic forecasting has attracted widespread attention recently. In reality, the traffic data usually contains missing values due to sensor or communication errors. The Spatio-temporal feature in traffic data brings more challenges for processing such missing values, for which the classic techniques (e.g., data imputations) are limited: 1) in temporal axis, the values can be randomly or consecutively missing; 2) in spatial axis, the missing values can happen on one single sensor or on multiple sensors simultaneously. Recent models powered by Graph Neural Networks achieved satisfying performance on traffic forecasting tasks. However, few of them are applicable to such a complex missing-value context. To this end, we propose GCN-M, a Graph Convolutional Network model with the ability to handle the complex missing values in the Spatio-temporal context. Particularly, we jointly model the missing value processing and traffic forecasting tasks, considering both local Spatio-temporal features and global historical patterns in an attention-based memory network. We propose as well a dynamic graph learning module based on the learned local-global features. The experimental results on real-life datasets show the reliability of our proposed method.

Keywords: Traffic Forecasting, Missing Values, Graph Convolutional Networks, Memory Networks

1 Introduction

Traffic forecasting has played a critical role in intelligent transportation systems, which helps the transportation department better manage and control traffic congestion. Generally represented by geo-located Multivariate Time Series (MTS), traffic data not only shows the typical characteristics of MTS, i.e., temporal dependency, but also integrates the spatial information of the traffic network, i.e., the spatial dependency between the sensor traffic nodes over the road network.

In recent years, by leveraging the spatial-temporal patterns in traffic data, many deep learning models based on recurrent neural network (RNN) [15], temporal convolutional network (TCN) [21], graph convolutional networks (GCN) [14], etc., have been applied in traffic forecasting task and achieved the state-of-the-art performance. They all have a strong assumption that the data is complete or has been well-preprocessed [25]. However, since the traffic data are generally collected from the geo-located sensors, sensor failures or communication errors will result in missing values in the collected data, which will deteriorate the performance of the forecasting model. We should remark that the missing measures are usually marked as *zero* in traffic data [14], which should be distinguished from the non-missing measures but with *zero* values. A typical example comes to the traffic flow data: no vehicles are detected during the night, then the traffic measures are marked as *zero* instead of being considered as missing.

The missing values can either be ignored in the learning model when calculating the loss function [19] or be considered before or during the training process. Apparently, ignoring the missing values hinders the model from benefiting from the rich data information for better performance, especially when the missing ratio is high. When considering the missing values in traffic data, most work [5] conduct data imputation during the preprocessing step, then import the completed data into the training step, i.e., *two-step processing*. Recent work [3, 6, 7, 17, 18] tend to jointly consider the missing values and the forecasting modeling during the training step (i.e., *one-step processing*) and declared a better performance than the two-step processing. However, the above-mentioned work suffer from three major issues. First, the missing and zero values are usually considered indifferent, leading to unnecessary, even harmful data imputations, thus contradicting the raw data information. Second, most of the work [3, 6, 17, 18] considers missing values from the temporal aspect, ignoring the rich information from the spatial perspective. Third, they are generally designed for processing the missing values in some basic scenarios, such as random missing or temporal block missing, but lack of power for the complex scenarios as shown in Fig. 1. In the real-world, the missing values in traffic data occur on both long-range (e.g., *device power-off*) and short-range (e.g. *device errors*) settings, on partial (e.g., *local sensor errors*) and entire transportation network (e.g., *control center errors*). The complex scenarios require a holistic approach for handling various types of missing values together.

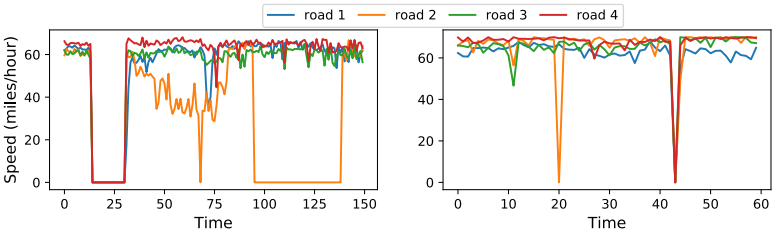


Fig. 1: Missing measures of traffic speed data from METR-LA dataset [15]. *left*) long-range missing on entire network (i.e., Spatio-temporal block) and partial network (i.e., temporal block); *right*) short-range random missing on partial network (i.e., temporal values) and entire network (i.e., Spatio-temporal vectors).

Therefore, to handle both the Spatio-temporal patterns and complex missing-value scenarios in traffic data, we propose GCN-M, **G**raph **C**onvolutional **N**etworks for Traffic Forecasting with **M**issing Values. The graph neural network-based structure allows jointly modeling the Spatio-temporal patterns and the missing values in one-step processing. We construct local statistic features from spatial and temporal perspectives for handling short-range missing values, which is further enhanced by a memory module to extract global historical features for processing long-range missing blocks. The combined local-global features allow not only identifying the missing measures from the inherent zero values but also enriching the traffic embeddings, thus generating dynamic traffic graphs to model the dynamic spatial interactions between traffic nodes. The missing values on a partial and entire network can then be considered from spatial and temporal perspectives.

We summarize the paper's main contributions as follows:

- **Complex missing value modeling**: We study the complex scenario where missing traffic values occur on both short & long ranges and on partial & entire transportation network.
- **Spatio-temporal memory module**: We propose a memory module that can be used by GCN-M to learn both local Spatio-temporal features and global historical patterns in traffic data for handling the complex missing values.
- **Dynamic graph modeling**: We propose a dynamic graph convolution module that models the dynamic spatial interactions. The dynamic graph is characterized by the learned local-global features at each timestamp, which not only offset the missing values' impact but also help learn the graph.
- **Joint model optimization**: We jointly model the Spatio-temporal patterns and missing values in one-step processing, which allows processing missing values specifically for traffic forecasting tasks, thus bringing better model performance than two-step processing.
- **Extensive experiments on real-life data**: The experiments are carried out on two real-life traffic datasets. We provide detailed evaluations with

12 baselines, which show the effectiveness of GCN-M over the state-of-the-art.

The rest of this paper starts with a review of the most related work. Then, we formulate the problems of the paper. Later, we present in detail our proposal GCN-M, followed by the experiments on real-life datasets and the conclusion.

2 Related Works

2.1 Graph Convolutional Networks for Traffic Forecasting

Graph Convolutional Network (GCN) is a special kind of Convolutional Neural Network (CNN) generalized for graph-structured data. Most of the GCN-related work focuses on graph representation, which learns node embedding by integrating the features from the node's local neighbors based on the given graph structure, i.e., adjacency matrix. The traffic data shows strong dependencies between the spatial nodes, for which GCN can be naturally suitable. Various work [15, 19, 22, 25] empowered by GCN achieved remarkable performance when doing traffic forecasting tasks. However, they either rely on spatial and temporal completion of the data or calculate loss function for non-zero entries, i.e., only calculate the loss on entries that contain valid sensor readings, which may introduce derivations when modeling the Spatio-temporal relations between the sensor nodes. In other words, missing values may hinder the traffic graph learning [14], especially for dynamic graph learning [8] where non-missing measures are required to characterize the dynamic graph at each timestamp.

2.2 Missing value processing

The simplest solution of processing missing values in MTS would be data imputation such as statistic imputation (e.g., mean, median), EM-based imputation, K-nearest neighborhood, and matrix factorization. It's generally believed that those methods fail to model temporal dynamics of time series [17]. In other words, they are not applicable for handling long-range missing values. Recent generative models [24] show reliable performance for long-range time series imputation. However, isolating the imputation model from the forecasting model leads to a two-step processing and may generate sub-optimal results [5]. To handle this issue, recent studies [3, 17] jointly model the missing values and forecasting task in one-step processing. For instance, GRU-D [3] considers the nearby temporal statistical features to do imputations inside GRUs, whereas LSTM-I [6] infers missing values at current time step from preceding LSTM cell states and hidden states, SGMN [7] improved the state transition process via a Graph Markov Process. Limited to short-period missing context, those methods are further enhanced by LGnet [17] with the global temporal dynamics to handle the long-range missing issue, and by LSTM-M [18] with multi-scale modeling to better explore historical information. However, the above-mentioned models handle missing values with a focus on the

temporal aspect without considering the complex Spatio-temporal features in traffic data. Specifically, the strong spatial connections between the sensor nodes should provide us with more information to handle the missing values. Moreover, the one-step processing models are generally designed for single-step forecasting without considering the multi-step settings. Table 1 shows the method comparison for traffic forecasting with missing values.

Table 1: Existing methods for Traffic Forecasting with missing values

	Imputation-based	GRU-D	LSTM-I	LSTM-M	LGnet	SGMN	**GCN-M**
Short-range missing	✓	✓	✓	✓	✓	✓	✓
Long-range missing	-	-	-	✓	✓	-	✓
Multi-scale modeling	-	-	-	✓	-	-	✓
Spatial modeling	-	-	-	-	-	✓	✓
One-step processing	-	✓	✓	✓	✓	✓	✓
Multi-step forecasting	✓	-	-	-	✓	-	✓

3 Problem Formulation

We aim to predict the traffic data in the future by leveraging historical traffic data. The traffic data can be represented as multivariate time series on the traffic network. Let the traffic network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, ..., v_N\}$ is a set of N traffic sensor nodes and $\mathcal{E} = \{e_1, ..., e_E\}$ is a set of E edges connecting the nodes. Each node contains F features representing traffic flow, speed, occupancy, etc. We use $\mathcal{X}=\{\mathbf{X}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times F\times\tau}$ to denote all the feature values of all the nodes over τ time slices, $\mathbf{X}_t = (\mathbf{x}_t^1, ..., \mathbf{x}_t^N) \in \mathcal{R}^{N\times F}$ denotes the observations at time t, where $\mathbf{x}_t^i \in \mathcal{R}^F$ is the i-th variable of X_t. We define a mask sequence $\mathcal{M}=\{\mathbf{M}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times F\times\tau}$, $\mathbf{M}_t = (\mathbf{m}_t^1, ..., \mathbf{m}_t^N) \in \mathcal{R}^{N\times F}$, $\mathbf{m}_t^i \in \{0, 1\}^F$ denotes the features' missing status for the i-th variable. To simplify, we adopt $x_t^i \in \mathcal{R}$ and $m_t^i \in \mathcal{R}$ to denote respectively the observation and mask value of *one single feature* for the i-th variable of $\mathbf{X}_t$. We take $m_t^i = 0$ if x_t^i is missing, otherwise $m_t^i = 1$.

We aim to build a model f, which can take an incomplete traffic sequence $\{\mathcal{X}, \mathcal{M}\}$ and the traffic network $\mathcal{G}$ as input, to predict the traffic data for the next T_p time steps $\mathbf{Y} = \{y_{\tau+1}, ..., y_{\tau+T_p}\} \in \mathcal{R}^{N\times T_p}$.

4 Proposal: GCN-M

The traffic data are collected under complex urban conditions. Apart from the Spatio-temporal patterns in the traffic data, we also consider the scenarios of complex missing values. We design a solution that models the local Spatio-temporal features and global historical patterns in a dynamic manner. The complex missing values are considered when building the forecasting model, i.e., one-step processing.

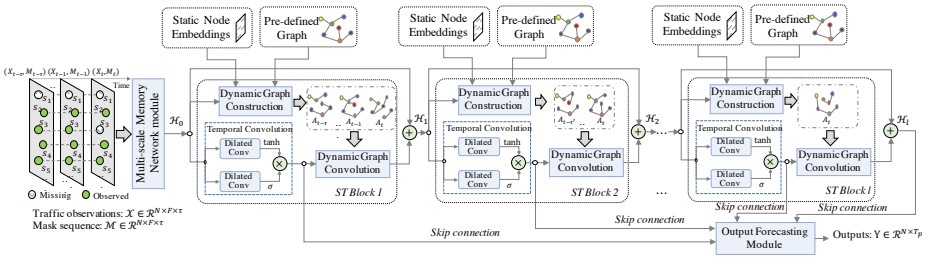


Fig. 2: Main architecture of GCN-M

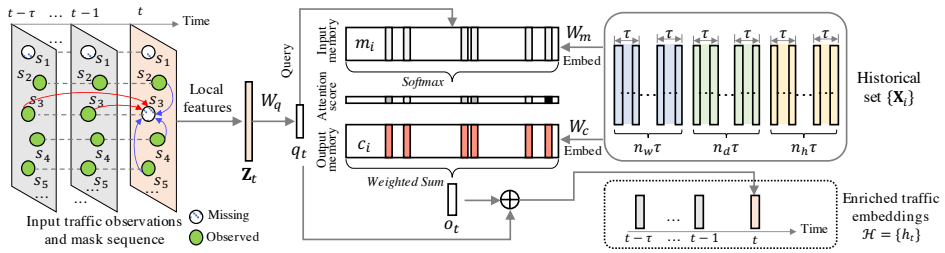


Fig. 3: Memory module enriches traffic embeddings with multi-scale global features

4.1 Model Architecture

The global structure of GCN-M is shown in Fig. 2, integrating a Multi-scale Memory Network module, an Output Forecasting module, and l Spatio-Temporal (ST) blocks. Each ST block integrates three key components: Temporal Convolution, Dynamic Graph Construction, and Dynamic Graph Convolution. The input traffic observations $\mathcal{X} \in \mathcal{R}^{N\times F\times\tau}$ and the mask sequence $\mathcal{M} \in \mathcal{R}^{N\times F\times\tau}$ are fed into the multi-scale memory network to extract the local statistic features and global historical patterns thus enriching the traffic embeddings. On the one hand, the enriched embeddings $\mathcal{H}_i$ on each ST block are used to mark the dynamic traffic status, thus generating dynamic graphs by combining both static node embeddings and predefined graph information. On the other hand, the learned dynamic graphs are combined with the temporal convolution module via a dynamic graph convolution to capture temporal and spatial dependencies in the traffic embeddings. We adopt residual connections between the input and output of each ST block to avoid the gradient vanishing problem. The output forecasting module takes the skip connections on the output of the final ST block and the hidden states after each temporal convolution for final prediction.

4.2 Multi-scale Memory Network

To extract the local statistic features and global historical patterns then form an enriched embedding, we adopt the concept of memory network, which was firstly proposed in [20] with primary application in Question-Answer (QA) systems. As shown in Fig. 3, the main idea of our memory network is to learn from historical memory components which conserve the long-range multi-scale patterns, i.e., recent, daily-periodic, and weekly-periodic dependencies. The scale range depends on the data characteristics. Specifically, we first extract local Spatio-temporal features as keys to query the memory components; the weighted historical long-range patterns will be cooperated with the local statistic features to eliminate the side effect from the missing values. Then, the local-global features will be output as the enriched traffic embeddings.

4.2.1 Local Spatio-temporal features

We first extract the Spatio-temporal features using the contextual information from observed parts of the time series. Unlike prior studies [3], we consider both temporal and spatial aspects for generating the following statistic features of every timestamp:

Empirical Temporal Mean: The mean of previous observations reflects the recent traffic state and serves as a contextual knowledge of x_t^i. Therefore, for a missing value $x_t^i \in \mathcal{R}$, we construct its temporal mean using L observed samples x_*^i before time t:

$$\bar{x}_t^i = \sum_{l=t-L}^{t-1} m_l^i x_l^i / \sum_{l=t-L}^{t-1} m_l^i \tag{1}$$

Last Temporal Observation: we adopt the assumption in [3] that any missing value inherits more or less the information from the last non-missing observation, in other words, the temporal neighbor stays close to the current missing value. We use $\dot{x}_t^i$ to denote the last temporal observation of x_t^i, their temporal distance is defined as $\dot{\delta}_t^i$.

Empirical Spatial Mean: Another contextual knowledge of x_t^i is from the nearby nodes, which reflects the current local traffic situation. For each missing value x_t^i, we construct its empirical spatial mean using S observed samples x_t^* nearby the sensor node i:

$$\bar{\bar{x}}_t^i = \sum_{s=1}^{S} m_t^s x_t^s / \sum_{s=1}^{S} m_i^s \tag{2}$$

Nearest Spatial Observation: typically, the state of a graph node remains relatively similar to its neighbors, especially in a traffic graph where the nearby nodes share similar traffic situation. We define $\ddot{x}_t^i$ as the nearest spatial observation of x_t^i, their spatial distance is denoted as $\ddot{\delta}_t^i$.

Generally, when $\dot{\delta}_t^i$ or $\ddot{\delta}_t^i$ is smaller, we tend to trust $\dot{x}_t^i$ or $\ddot{x}_t^i$ more. When the spatial/temporal distance becomes larger, the spatial/temporal mean would

be more representative. Under this assumption, we model the temporal and spatial decay rate γ as

$$\gamma_t(\dot{\delta}_t^i) = exp\{-max(0, w^i\dot{\delta}_t^i + b^i\} \tag{3}$$

$$\gamma_s(\ddot{\delta}_t^i) = exp\{-max(0, w_t\ddot{\delta}_t^i + b_t\} \tag{4}$$

where w^i, w_t, b^i and b_t are model parameters that we train jointly with other parameters of the traffic forecasting model. We chose the exponentiated negative rectifier [3] so that the decay rates γ_t and γ_s decrease monotonically in the range between 0 and 1. Considering the trainable decays, our proposed model incorporates the spatial/temporal estimations to define the local features of x_t^i:

$$z_t^i = m_t^i x_t^i + (1 - m_t^i)(\gamma_t\dot{x}_t^i + \gamma_s\ddot{x}_t^i + (1 - \gamma_t)\bar{x}_t^i + (1 - \gamma_s)\bar{\bar{x}}_t^i) \tag{5}$$

Therefore, for $\mathbf{X}_t \in \mathcal{R}^{N\times F}$, we can get its local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$.

4.2.2 Multi-scale Memory Construction

The global historical patterns play a critical role in building an enriched traffic embedding. The historical observations in multiple scales (e.g., hourly, daily, weekly) can be embedded into memory as complement information for the local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$. The main idea is to adopt local features to query similar historical patterns in the memory and output a weighted feature representation for the current timestamp. In this manner, the enriched multi-scale historical and local features allow not only eliminating the side effect of missing values but also improving the current feature embeddings. At time t, the query q_t of $\mathbf{X}_t$ can be embedded from the local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$:

$$q_t = \mathbf{Z}_t W_q + b_q \in \mathcal{R}^{N\times d} \tag{6}$$

where $W_q \in \mathcal{R}^{F\times d}$, $b_q \in \mathcal{R}^{N\times d}$ are parameters, d is the embedding dimension.

The input memory components are the temporal segments of multiple scales:

- The recent (e.g., hourly) segment is: $X_h = \{\mathbf{X}_i\}_{i=t-\tau}^{t-1} \in \mathcal{R}^{N\times F\times n_h\tau}$, with n_h recent periods (e.g., hours) before t, each period contains τ observations.
- The daily-periodic segment is: $X_d = \{\mathbf{X}_i\} \in \mathcal{R}^{N\times F\times n_d\tau}$ with $i \in [t - n_dT_d - \tau/2 : t - n_dT_d + \tau/2] \parallel [t - (n_d - 1)T_d - \tau/2 : t - (n_d - 1)T_d + \tau/2] \parallel ... \parallel [t - T_d - \tau/2 : t - T_d + \tau/2]$, we store τ samples around time t for each of the past n_d days. T_d denotes the sample number during one day, $\parallel$ indicates the concatenation operation.
- The weekly-periodic segment is: $X_w = \{\mathbf{X}_i\} \in \mathcal{R}^{N\times F\times n_w\tau}$ with $i \in [t - n_wT_w - \tau/2 : t - n_wT_w + \tau/2] \parallel [t - (n_w - 1)T_w - \tau/2 : t - (n_w - 1)T_w + \tau/2] \parallel ... \parallel [t - T_w - \tau/2 : t - T_w + \tau/2]$, we store τ samples around time t for each of the past n_w weeks. T_w denotes the sample number during one week, $\parallel$ indicates the concatenation operation.

The input set of $\{\mathbf{X}_i\} = [X_h \| X_d \| X_w] \in \mathcal{R}^{N\times F\times(n_d+n_w+n_h)\tau}$ are embedded into the input memory vectors $\{m_i\}$ and output memory vectors $\{c_i\}$:

$$m_i = \mathbf{X}_i W_m + b_m \in \mathcal{R}^{N\times d} \tag{7}$$

$$c_i = \mathbf{X}_i W_c + b_c \in \mathcal{R}^{N \times d} \tag{8}$$

where $W_m, W_c \in \mathcal{R}^{F \times d}$, $b_m, b_c \in \mathcal{R}^{N \times d}$ are parameters.

In the embedding space, we compute the attention score between the query q_t and each memory m_i by taking the inner product followed by a softmax:

$$p_{t,i} = Softmax(q_t^T m_i) \tag{9}$$

The attention score represents the similarity of each historical observation to the query. Any pattern with a higher attention score is more similar to the context of targeting missing value. As shown in Fig. 3, the response vector from memory is then a sum over the output memory vectors, weighted by the attention score from the input:

$$o_t = \sum_{i=1}^{(n_d+n_w+n_h)\tau} c_i p_{t,i} \in \mathcal{R}^{N \times d} \tag{10}$$

We can finally integrate both local Spatio-temporal and global multi-scale features and output the enriched traffic embeddings:

$$h_t = (q_t \| o_t) W_h + b_h \in \mathcal{R}^{N \times d} \tag{11}$$

where $W_h \in \mathcal{R}^{2d \times d}$, $b_h \in \mathcal{R}^d$ are parameters, and $\|$ denotes the concatenation operation. Therefore, for input $\mathcal{X} = \{\mathbf{X}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N \times F \times \tau}$, we can get its enriched traffic embeddings $\mathcal{H} = \{h_t\}_{t=1}^{\tau} \in \mathcal{R}^{N \times d \times \tau}$.

4.3 Dynamic Graph Construction

A predefined graph is usually constructed with the distance or the connectivity between the spatial nodes. However, recent studies [14, 19, 22] show that the cross-region dependence does exist for those nodes which are not physically connected but share similar patterns. Learning dynamic graphs should probably show better performance than learning static graphs or adopting the predefined graphs. Considering the missing values in traffic data, instead of using the raw traffic observations to mark the dynamic traffic status [10, 14], we construct dynamic graphs (i.e., adjacency matrix) with the enriched traffic embeddings $\mathcal{H}_i$ at each ST block, which integrates both local and global multi-scale patterns at each time step, allowing capturing the spatial relationship between traffic nodes robustly. As shown in Fig. 4, the main idea here is to generate dynamic filters from the predefined graphs $\mathcal{G}$ and the traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N \times d \times \tau_i}$ (τ_i is the sequence length at the i-th ST block), which are applied on the static randomly initialized node embeddings to construct dynamic adjacency matrix. In more detail, the core steps in Fig. 4 are illustrated as follows:

[Dynamic Filter Generation] Given $\mathcal{H}_i = \{h_t\} \in \mathcal{R}^{N \times d \times \tau_i}$, the traffic embedding h_t at time t is firstly combined with the predefined adjacency matrix $A_{\mathcal{G}} \in \mathcal{R}^{N \times N}$ to generate dynamic graph filters via a diffusion convolution layer as proposed in [15]:

$$\mathcal{F}_t = \sum_{k=0}^{K} P_k h_t W_k \in \mathcal{R}^{N \times d} \tag{12}$$

where K denotes the diffusion step, $P_k = A_{\mathcal{G}} / rowsum(A_{\mathcal{G}})$ represents the power series of the transition matrix [21], and $W_k \in \mathcal{R}^{d \times d}$ is the model parameter matrix.

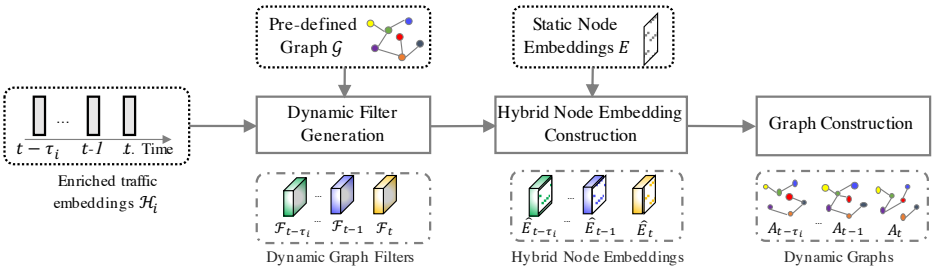


Fig. 4: Dynamic Graph Construction from the enriched traffic embeddings

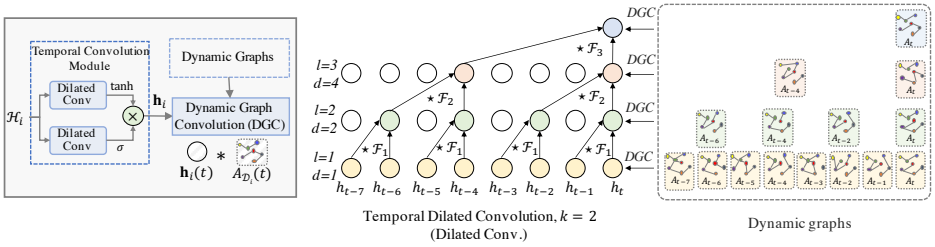


Fig. 5: Temporal Convolution module with Dynamic Graph Convolution

[**Hybrid Node Embedding Construction**] Considering both the source and target traffic node, we initialise two random node embeddings $E^1, E^2 \in \mathcal{R}^{N\times d}$, representing the static node features [19] which are not reflected in the observations but learnable during training. Thus, two dynamic filters are applied over the static node embeddings:

$$\begin{aligned}\hat{E}_t^1 &= tanh(\alpha(\mathcal{F}_t^1 \odot E^1)) \in \mathcal{R}^{N\times d}\\ \hat{E}_t^2 &= tanh(\alpha(\mathcal{F}_t^1 \odot E^2)) \in \mathcal{R}^{N\times d}\end{aligned} \tag{13}$$

where $\odot$ denotes the Hadamard product [21]. $\hat{E}_t^1$ and $\hat{E}_t^1$ are hybrid node embeddings combining both static and dynamic settings of the traffic data.
[**Graph Construction**] Following [22], we extract uni-directional relationships between traffic nodes. The dynamic adjacency matrix is constructed from the hybrid embeddings:

$$A_t = ReLU(tanh(\alpha(\hat{E}_t^1 \hat{E}_t^{2^T} - \hat{E}_t^2 \hat{E}_t^{1^T}))) \in \mathcal{R}^{N\times N} \tag{14}$$

Therefore, we can construct the dynamic graphs $A_{\mathcal{D}_i} = \{A_t\} \in \mathcal{R}^{N\times N\times\tau_i}$ for the enriched traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N\times d\times\tau_i}$ at the i-th ST block.

4.4 Temporal Convolution Module

The temporal convolution network (TCN) [13] consists of multiple dilated convolution layers, which allows extracting high-level temporal trends. Compared to RNN-based approaches, dilated causal convolution networks are capable of handling long-range sequences in a parallel manner. The output of the last layer is a representation that captures temporal dynamics in history. As shown

in Fig. 5, considering the temporal dynamics in traffic data, we adopt the temporal convolution module [21] with the consideration of the gating mechanism over the enriched traffic embeddings $\mathcal{H}_i$. One dilated convolution block is followed by a tangent hyperbolic activation function to output the temporal features. The other block is followed by a sigmoid activation function as a gate to determine the ratio of information that can pass to the next module.

Given the enriched traffic embeddings $\mathcal{H}_i = \{h_t\} \in \mathcal{R}^{N \times d \times \tau_i}$, a filter $\mathcal{F} \in \mathcal{R}^{1 \times \mathrm{K}}$, K is the temporal filter size, $\mathrm{K} = 2$ by default. The dilated causal convolution operation of $\mathcal{H}_i$ with $\mathcal{F}$ at time t is represented as:

$$\mathcal{H}_i \star \mathcal{F}_i(t) = \sum_{s=0}^{\mathrm{K}} \mathcal{F}_i(s)\mathcal{H}_i(t - \mathbf{d} \times s) \in \mathcal{R}^{N \times d \times \tau_{i+1}} \tag{15}$$

where $\star$ is the convolution operator, $\mathbf{d}$ is the dilation factor, d is the embedding dimension size, τ_{i+1} is the new sequence length after the convolution operation, which equals to one on the last layer. Fig. 5 shows a three-layer dilated convolution block with $\mathrm{K} = 2$, $\mathbf{d} \in [1, 2, 4]$. Considering the gating mechanism, we define the output of the temporal convolution module:

$$\mathbf{h}_i = tanh(W_{\mathcal{F}^1} \star \mathcal{H}_i) \odot \sigma(W_{\mathcal{F}^2} \star \mathcal{H}_i) \in \mathcal{R}^{N \times d \times \tau_{i+1}} \tag{16}$$

where $W_{\mathcal{F}^1}$, $W_{\mathcal{F}^2}$ are learnable parameters of convolution filters, $\odot$ denotes the element-wise multiplication operator, $\sigma(\cdot)$ is the sigmoid function.

A classic temporal convolution module stacks the temporal features at each time step t. Therefore, the upper layer contains richer information than the lower layer. The gating mechanism allows filtering the temporal features on the lower layers by weighting features on different time steps without considering the spatial node interactions at each time step. Moreover, the spatial interactions in traffic data always show a dynamic nature [22]. To this end, the gating mechanism from a dynamic spatial aspect is envisaged to better capture the Spatio-temporal patterns.

4.5 Dynamic Graph Convolution

Spatial interactions between the traffic nodes could be used to improve traffic forecasting performance. The dynamic spatial interaction leads to considering a dynamic version of graph convolution to conduct it on different graphs at different timestamps. Different from previous work [14] which uses raw traffic observations to mark the dynamic traffic status, we adopt the enriched traffic embeddings, which consider the missing-value issues to generate robust dynamic graphs.

As shown in Fig. 5, we apply the dynamic graph convolution on $\mathbf{h}_i$, i.e., the output of the temporal convolution module, to further select the features at each time step from the spatial perspective. As mentioned in Section 4.3, the dynamic graphs $A_{\mathcal{D}_i} \in \mathcal{R}^{N \times N \times \tau_i}$ are generated from the enriched traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N \times d \times \tau_i}$ at the i-th ST block. A_t reflects the spatial relationships between nodes at time t. The temporal features $\mathbf{h}_i(t)$ aggregate spatial information according to the adjacency matrix A_t. Inspired by DCRNN [15], we consider the traffic situation as the diffusion procedure on graph. The graph

convolution will generate the aggregated spatial information at each time step:

$$\mathcal{H}'_i(t) = \sum_{k=0}^{K} \left(A_{\mathcal{D}_i}(t)\right)^k \mathbf{h}_i(t) W_k \in \mathcal{R}^{N \times d} \tag{17}$$

where K denotes the diffusion step, W_k is the learnable parameter matrix. We adopt the residual connection [11] between the input and output of each ST block to avoid the gradient vanishing issue in model's training. Therefore, the input of the $(i+1)^{th}$ ST block is defined as:

$$\mathcal{H}_{i+1}(t) = \mathcal{H}_i(t) + \mathcal{H}'_i(t) \tag{18}$$

4.6 Output Forecasting Module

The outputs $\mathbf{h}_i \in \mathcal{R}^{N \times d \times \tau_{i+1}}$ of the middle temporal convolution modules and $\mathcal{H}_l \in \mathcal{R}^{N \times d \times 1}$ of the last ST block are considered for the final prediction, which represent the hidden states at various Spatio-temporal levels. We add skip connection on each of the hidden states which are essentially $1 \times \tau_{i+1}$ standard convolutions, τ_{i+1} denotes the sequence length at the output of the i-th ST block. the concatenated output features are defined as follows:

$$O = (\mathbf{h}_0 W_s^0 + b_s^0) \| ... \| (\mathbf{h}_i W_s^i + b_s^i) \| ... \| (\mathbf{h}_{l-1} W_s^{l-1} + b_s^{l-1}) \, \| (\mathcal{H}_l W_s^l + b_s^l) \tag{19}$$

where $O \in \mathcal{R}^{N \times ld}$, W_s^i, b_s^i are learnable parameters for the convolutions. Two fully-connected layers are added to project the concatenated features into the desired output dimension:

$$\hat{\mathbf{Y}} = W_{fc}^2 (W_{fc}^1 O + b_{fc}^1) + b_{fc}^2 \in \mathcal{R}^{N \times T_p} \tag{20}$$

where W_{fc}^1, W_{fc}^2, b_{fc}^1, b_{fc}^2 are learnable parameters for the fully-connected layers, N is the node number, T_p denotes the forecasting steps.

Given the ground truth $\mathbf{Y} \in \mathcal{R}^{N \times T_p}$ and the predictions $\hat{\mathbf{Y}} \in \mathcal{R}^{N \times T_p}$, we use mean absolute error (MAE) as our model's loss function for training:

$$L = \frac{1}{N T_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n| \tag{21}$$

5 Experiments

In this section, we demonstrate the effectiveness of GCN-M [1] with real-life traffic datasets. The experiments were designed to answer the following research questions (RQs):

- RQ 1 *Performance on complete datasets:* How well our model performs on traffic datasets without missing values?
- RQ 2 *Complex scenarios of missing values:* How successful is our model in forecasting traffic data considering the complex missing values scenarios?
- RQ 3 *Dynamic graph modeling:* How our method performs on dynamic graph modeling considering the missing values?

[1] The source code is publicly available in https://github.com/GCN-M/GCN-M

5.1 Experimental settings

[**Datasets**] We base our experiments on the public traffic datasets: PEMS-BAY and METR-LA released by [15], which are widely used in the literature. PEMS-BAY records six months of traffic speed on 325 sensors in the Bay Area. METR-LA records four months of traffic flow on 207 sensors on the highways of Los Angeles County. Both datasets contain some zero/missing values, though PEMS-BAY has been pre-processed by the domain experts from the data provider [1] to interpolate most of the missing values. Following [15], the datasets are split with 70% for training, 10% for validation and 20% for testing. We use recent $\tau = 12$ timestamps as input to predict the next T_p timestamp. Considering that the missing values are marked as zeros, we scale the input by dividing with the max speed of the training set instead of applying Z-score normalization, which avoids changing the zero values and removing the missing markers. Table 2 shows the summary statistics of the datasets.

Table 2: Summary statistics of PEMS-BAY and METR-LA

Data	#Nodes	#Edges	Length	Sample Rate	Observations	Zero ratio
PEMS-BAY	325	2369	52 116	5 mins	16 937 179	0.0031%
METR-LA	207	1515	34 272	5 mins	6 519 002	8.11%

[**Evaluation metrics**] The forecasting accuracy of all tested models is evaluated by three metrics, including mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE).

$$
\begin{aligned}
MAE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n| \\
RMSE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \sqrt{\frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n|^2} \\
MAPE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\frac{\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n}{\mathbf{Y}_t^n}|
\end{aligned}
\tag{22}
$$

where N denotes the node numbers, T_p represents the forecasting steps.
[**Execution and Parameter Settings**] The proposed model is implemented by PyTorch 1.6.0 and is trained using the Adam optimizer with a learning rate of 0.001. All the models are tested on a single Tesla V100 GPU of 32 Go memory. In the multi-scale memory module, L, S are set to 12 and 5. n_h, n_d, n_w are all set to 2. We apply four ST blocks in which the Temporal Convolution module contains two dilated layers with dilation factor $\mathbf{d} \in [1, 2]$. The embedding dimension d is set to 32.

5.2 Baseline Approaches

We only compare with the baseline models whose source code is publicly available. We follow the default parameter settings described in each paper for

training. According to the strategy for handling missing values, the baseline models can be organized into two categories:

1. Jointly model the missing values and forecasting task, i.e., one-step processing
   - GRU [4]: Gated Recurrent Unit (GRU) can be considered as a basic structure for traffic forecasting.
   - GRU-I [3]: A variation of GRU, which infers the missing values with the predictions from previous steps.
   - GRU-D [3]: Based on GRU, GRU-D incorporates the missing patterns, including the masking information and time intervals between missing and observed values, to help improve the prediction performance.
   - LSTM-I [7]: Based on LSTM, LSTM-I is similar to GRU-I for inferring the missing values.
   - LSTM-M [18]: Based on LSTM, LSTM-M is designed for traffic forecasting on data with short-period and long-period missing values.
   - SGMN [7]: Based on the graph Markov process, SGMN incorporates a spectral graph convolution to do traffic forecasting on data with random missing values.
2. Ignore the missing values when optimizing the model:
   - DCRNN [15]: Based on the predefined graphs, DCRNN integrates GRU with dual directional diffusion convolution.
   - STGCN [25]: Based on the predefined graphs, STGCN combines graph convolution into 1D convolutions.
   - Graph WaveNet [21]: Graph WaveNet learns an adaptive graph and integrates diffusion graph convolutions with temporal convolutions.
   - MTGNN [22]: MTGNN learns an adaptive graph and integrates mix-hop propagation layers in the graph convolution module. Moreover, it designed the dilated inception layers in temporal convolutions.
   - AGCRN [2]: AGCRN learns an adaptive graph and integrates with recurrent graph convolutions with node adaptive parameter learning.
   - GTS [16]: GTS learns a probabilistic graph which is combined with the recurrent graph convolutions to do traffic forecasting.

   Note: In practice, any model can ignore the missing values in their optimization process. We list here some classic models and the most recent models designed specifically for traffic forecasting.

5.3 RQ 1: Performance on complete datasets

Recently, a lot of traffic forecasting models [12] have been proposed, achieving remarkable performance on the benchmark datasets PEMS-BAY and METR-LA. Our objective is not to beat all the models in terms of forecasting accuracy but to validate our proposal for jointly modeling missing values and forecasting. Therefore, it's essential to know how GCN-M performs in a primary setting, i.e., on the original datasets without or with a few missing values. We pick three classic models (DCRNN [15], STGCN [25] and Graph WaveNet [21]) and

three most recent models (MTGNN [22], AGCRN [2] and GTS [16]), which focus on the Spatio-temporal modeling of traffic data and generally ignore the missing values when training the model. We consider as well the group of works [3, 7, 18] which are specifically designed for modeling the missing values in the forecasting model, i.e., one-step processing models.

Table 3: Performance comparison on **complete** PEMS-BAY dataset

PEMS-BAY	Horizon=1 (5 mins)			Horizon=3 (15 mins)			Horizon=6 (30 mins)			Horizon=12 (60 mins)		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	0.96	1.63	1.81%	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	0.98	1.84	1.98%	1.44	2.88	3.16%	1.85	3.82	4.20%	2.21	4.52	5.09%
GraphWaveNet	0.91	**1.56**	1.72%	**1.31**	2.75	**2.73%**	1.65	3.75	3.74%	1.99	4.62	4.78%
MTGNN	**0.87**	1.57	**1.70%**	1.33	2.80	2.80%	1.65	3.75	3.70%	1.95	4.49	4.56%
AGCRN	0.95	1.81	1.94%	1.37	2.92	2.94%	1.69	3.87	3.82%	1.99	4.61	4.62%
GTS	0.91	1.64	1.77%	1.32	2.80	2.75%	1.63	3.74	**3.63%**	**1.90**	**4.40**	**4.44%**
GRU	1.29	2.46	2.54%	1.89	3.53	3.98%	2.27	4.24	5.02%	2.65	4.90	5.92%
GRU-I	1.30	2.57	2.57%	1.89	3.52	3.99%	2.26	4.22	4.99%	2.62	4.89	5.87%
GRU-D	5.40	9.25	13.83%	5.34	9.25	13.76%	5.42	9.26	13.85%	5.41	9.27	13.85%
LSTM-I	1.71	2.69	2.80%	1.97	3.45	4.08%	2.57	5.52	5.62%	2.74	5.00	6.21%
LSTM-M	1.35	2.31	2.71%	1.87	3.39	3.95%	2.33	4.33	5.17%	3.45	8.32	7.29%
SGMN	0.98	1.85	1.88%	1.63	3.40	3.32%	2.29	4.91	4.88%	3.31	6.86	7.32%
GCN-M (ours)	0.91	1.57	1.75%	1.33	**2.72**	2.76%	**1.62**	**3.64**	3.64%	1.95	**4.40**	4.61%

Table 4: Performance comparison on **complete** METR-LA dataset

METR-LA	Horizon=1 (5 mins)			Horizon=3 (15 mins)			Horizon=6 (30 mins)			Horizon=12 (60 mins)		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	2.45	4.21	5.99%	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
STGCN	2.58	4.32	6.22%	3.04	5.48	8.00%	3.60	6.51	9.97%	4.21	7.37	11.61%
GraphWaveNet	2.41	4.29	5.93%	**2.68**	**5.14**	6.87%	3.06	6.14	8.23%	3.52	7.25	**9.77%**
MTGNN	**2.24**	3.92	**5.39%**	**2.68**	5.16	**6.86%**	**3.05**	6.16	8.19%	**3.50**	**7.24**	9.83%
AGCRN	2.41	4.27	6.08%	2.86	5.54	7.66%	3.22	6.55	8.92%	3.58	7.45	10.24%
GTS	2.32	4.15	6.12%	2.72	5.42	7.11%	3.11	6.47	**7.49%**	3.52	7.49	10.07%
GRU	2.83	4.56	6.78%	3.48	5.80	9.02%	3.97	6.74	10.72%	4.65	7.86	13.00%
GRU-I	2.80	4.52	6.70%	3.49	5.83	9.05%	3.97	6.74	10.75%	4.60	7.88	12.80%
GRU-D	7.46	11.82	24.55%	7.43	11.85	24.62%	7.45	11.84	24.62%	7.47	11.86	24.68%
LSTM-I	2.86	4.57	6.77%	3.57	5.88	9.05%	4.10	6.85	10.94%	4.78	8.13	13.34%
LSTM-M	3.15	5.58	7.03%	3.46	5.74	8.75%	4.08	6.86	10.89%	4.63	7.83	12.92%
SGMN	3.11	6.02	7.01%	4.23	8.54	9.89%	5.46	10.88	13.01%	7.37	13.78	17.81%
GCN-M (ours)	2.34	**3.89**	5.88%	2.74	5.21	6.94%	3.12	6.18	8.25%	3.54	**7.12**	10.01%

Table 3 and 4 show the performance comparison on the complete PEMS-BAY and METR-LA datasets, respectively. It should be noted that the original datasets already contain missing values (0.0031% missed in PEMS-BAY, 8.11% missed in METR-LA). We train the models for single-step (horizon=1) and multi-step (horizon =3,6,12) forecasting. We report the evaluation errors on each horizon step. We observe from the results that no model achieved evident better performance than the others. However, the first group of works (e.g., DCRNN) performs better than the one-step processing models, which is not surprising as they incorporate the advanced graph models (e.g., *mix-hop propagation* [22]) and training techniques (e.g., *curriculum learning* [22]) to improve the Spatio-temporal forecasting performance. Among the one-step processing models, surprisingly, GRU-D [3] shows much worse performance than the others, which is probably due to the fact that it has been designed for health care

applications, whose data have a more stable status than the dynamic traffic data. LSTM-M [18] and SGMN [7], that are designed for traffic forecasting with missing values, show relative good performance in PEMS-BAY especially on single-step forecasting. However, they did not show a clear advantage to the first group of works. The one-step processing models are generally designed for single-step forecasting; their performance gap with the first group of works becomes larger under a multi-step forecasting setting. Even though GCN-M belongs to the one-step processing models, its performance remains close to the first group of works. Moreover, the advanced graph models and training techniques in recent work [22] can be considered to improve the performance of GCN-M further.

5.4 RQ 2: Complex scenarios of missing values

In this section, we demonstrate the power of GCN-M in handling complex scenarios of missing values for the purpose of traffic forecasting.

As mentioned previously in Figure 1, there are several scenarios of missing values in real-life traffic datasets (e.g., METR-LA): short-range or long-range missing; partial or entire network missing. The results in Table 3 and 4 did not show the superiority of GCN-M over other models on the original datasets with a low missing rate. Therefore, to test the model's capability of handling complex missing values, we design three scenarios with various missing rates (10%, 20%, and 40%) and remove the observations from the datasets accordingly. We use $\hat{X} \in \mathcal{R}^{n \times f \times t}$ to represent each of the observations to be removed from $\mathcal{X} \in \mathcal{R}^{N \times F \times \tau}$. Then, we design the scenarios of:

- Short-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t = 1$
- Long-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t = \tau$
- Mix-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t \in [1, ..., \tau]$

In Table 5 and 6, we show the performance comparison on the PEMS-BAY and METR-LA datasets under various missing value scenarios. We highlight the best results among the one-step processing models (underlined values) and all the models (bold values). Globally, GCN-M shows the best performance under all the settings compared to the one-step processing models, in which the graph-based model SGMN [7] performs much worse than other one-step processing models under long-range and mix-range missing settings, indicating that it applies only on simple missing scenarios, i.e., short-range random missing. GCN-M does not always show superiority compared with the first group of works, especially in the short-range missing scenario, where MTGNN and GTS usually show good performances. Besides, MTGNN typically performs better than GCN-M when the missing rate is low (10%), except under the mix-range missing scenario of PEMS-BAY. We can draw an interesting conclusion from this observation: a robust Spatio-temporal forecasting model can offset the impact of the missing values to some extent, as it allows exploring the information thoroughly from the observed measures. GCN-M becomes the best forecasting model when the missing rate gets higher, as the missing

Table 5: Performance comparison on the **incomplete** PEMS-BAY dataset with various random settings on missing values. The results show a one-hour (12-step) average of the forecasting errors. The underlined values represent the best results among the one-step processing models, the bold values represent the best results among all the models.

		Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	Models	MAE	RMSE	MAPE	MAE	RMSE	MAPEy	MAE	RMSE	MAPE
Short-range missing	DCRNN	1.76	3.94	3.94%	1.82	3.96	4.01%	1.85	4.26	4.04%
	STGCN	1.82	4.11	4.25%	1.91	4.18	4.41%	1.97	4.33	4.42%
	GraphWaveNet	1.69	3.79	3.81%	1.74	3.75	3.75%	1.79	3.87	3.90%
	MTGNN	**1.58**	**3.42**	**3.33%**	1.72	3.78	3.83%	1.83	4.03	3.94%
	AGCRN	1.65	3.81	3.78%	1.66	3.81	3.79%	1.72	3.96	3.95%
	GTS	1.65	3.86	3.74%	1.65	3.86	3.76%	**1.69**	3.92	**3.86%**
	GRU	2.60	4.64	5.75%	2.67	4.78	5.90%	2.86	5.10	6.37%
	GRU-I	2.29	4.28	5.06%	2.31	4.31	5.09%	2.41	4.47	5.38%
	GRU-D	5.38	9.29	13.84%	5.46	9.36	13.96%	7.20	11.58	16.91%
	LSTM-I	2.35	4.33	5.22%	2.82	6.63	6.05%	3.06	7.47	6.56%
	LSTM-M	2.47	4.55	5.50%	2.56	4.70	5.74%	3.34	7.09	7.68%
	SGMN	2.32	4.96	4.94%	2.34	5.01	5.00%	2.45	5.20	5.23%
	GCN-M (ours)	1.62	3.67	3.60%	**1.63**	**3.73**	**3.68%**	1.75	**3.81**	3.90%
Long-range missing	DCRNN	1.83	4.07	4.22%	1.96	4.22	4.42%	2.07	4.45	4.67%
	STGCN	1.92	4.22	4.42%	2.03	4.37	4.72%	2.14	4.52	4.76%
	GraphWaveNet	1.74	3.96	4.03%	1.87	4.09	4.18%	1.94	4.21	4.33%
	MTGNN	**1.65**	**3.68**	**3.72%**	1.89	4.01	4.17%	2.01	4.42	4.61%
	AGCRN	1.72	3.78	3.94%	1.84	4.11	4.13%	1.90	4.18	4.31%
	GTS	1.68	3.86	3.91%	1.78	4.12	4.97%	1.88	4.17	4.22%
	GRU	2.93	5.12	6.32%	3.06	5.31	6.63%	3.35	5.78	7.03%
	GRU-I	2.52	4.51	5.33%	2.53	4.57	5.73%	2.71	4.82	5.51%
	GRU-D	9.33	14.51	22.31%	9.89	13.94	22.86%	11.07	15.88	23.13%
	LSTM-I	2.65	4.65	5.88%	3.13	6.35	6.82%	3.62	9.53	7.12%
	LSTM-M	3.93	7.25	9.17%	5.45	10.06	13.67%	5.57	10.12	14.59%
	SGMN	8.86	12.57	14.54%	11.45	14.56	18.31%	14.62	17.23	23.13%
	GCN-M (ours)	1.70	3.75	3.74%	**1.73**	**3.88**	**3.92%**	**1.79**	**4.07**	**4.14%**
Mix-range missing	DCRNN	1.81	4.01	4.15%	1.91	4.16	4.31%	2.02	4.36	4.52%
	STGCN	1.85	4.13	4.21%	1.98	4.31	4.56%	2.11	4.43	4.68%
	GraphWaveNet	1.72	3.92	3.96%	1.83	4.06	4.14%	1.89	4.11	4.21%
	MTGNN	1.69	3.77	3.78%	1.86	4.03	4.11%	1.98	4.32	4.44%
	AGCRN	1.67	3.85	3.88%	1.72	3.95	3.99%	1.80	4.10	4.13%
	GTS	1.70	3.96	3.92%	1.75	3.98	3.89%	1.79	4.09	4.09%
	GRU	2.71	4.88	6.03%	2.82	5.08	6.28%	3.05	5.43	6.82%
	GRU-I	2.31	4.30	5.11%	2.34	4.39	5.18%	2.40	4.50	5.37%
	GRU-D	8.90	13.71	20.03%	9.46	14.50	21.04%	10.21	15.19	22.44%
	LSTM-I	2.46	4.51	5.49%	2.75	5.85	6.02%	3.39	9.15	6.88%
	LSTM-M	3.86	7.06	8.93%	5.19	9.71	13.15%	5.27	9.74	13.29%
	SGMN	7.41	10.91	13.47%	9.95	13.49	17.56%	13.10	16.96	22.58%
	GCN-M (ours)	**1.65**	**3.67**	**3.69%**	**1.66**	**3.72**	**3.62%**	**1.69**	**3.79**	**3.83%**

values become a more critical factor that impacts the forecasting model than Spatio-temporal pattern modeling.

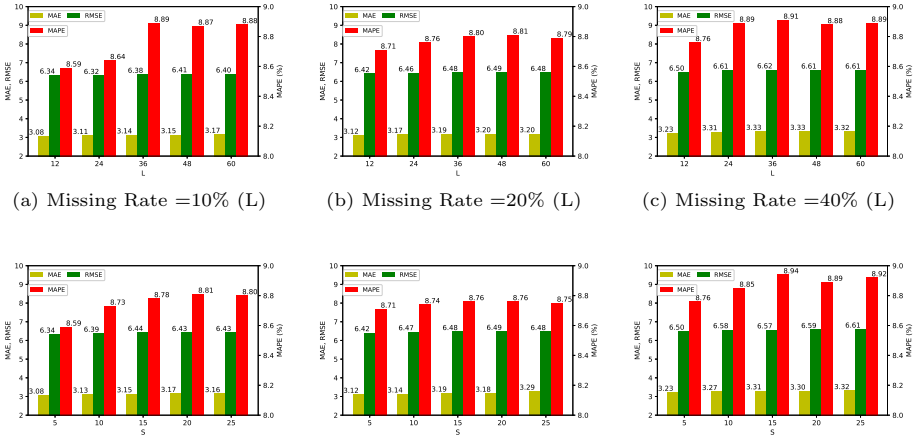
Compared to the short-range missing scenario, GCN-M shows a more robust performance under long-range and mix-range missing scenarios, where the recent temporal values and the nearby nodes' values are not always observed. The multi-scale memory block in GCN-M allows enriching the traffic embedding at each timestamp, thus making the model robust in the two complex scenarios. The memory block searches for the periodic global patterns from historical data and the valuable local features from nearby nodes or recent observations at each timestamp. When nearby nodes values are unobserved, GCN-M favors more on recent observations and vice versa. The memory module with the periodic historical patterns can distinguish the inherent zero

Table 6: Performance comparison on the **incomplete** METR-LA dataset with various random settings on missing values. The results show a one-hour (12-step) average of the forecasting errors. The underlined values represent the best results among the one-step processing models, the bold values represent the best results among all the models.

		Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	Models	MAE	RMSE	MAPE	MAE	RMSE	MAPEy	MAE	RMSE	MAPE
Short-range missing	DCRNN	3.31	6.61	9.47%	3.44	6.80	9.57%	3.50	6.90	9.68%
	STGCN	3.53	7.08	9.73%	3.59	7.25	10.25%	3.66	7.45	10.41%
	GraphWaveNet	3.28	6.60	9.11%	3.36	6.74	9.50%	3.45	6.81	9.57%
	MTGNN	**2.98**	**6.03**	**8.35%**	3.19	**6.44**	8.69%	3.26	6.59	9.07%
	AGCRN	3.19	6.47	8.81%	3.24	6.60	9.01%	3.25	6.61	9.19%
	GTS	3.08	6.40	8.59%	**3.14**	6.52	**7.58%**	**3.12**	6.56	**8.61%**
	GRU	4.20	7.09	11.27%	4.27	7.16	11.42%	4.45	7.41	11.94%
	GRU-I	4.02	6.83	10.89%	4.03	6.88	10.83%	4.09	6.91	10.88%
	GRU-D	7.50	11.87	24.69%	7.45	11.86	24.66%	7.53	11.91	24.76%
	LSTM-I	4.12	6.89	11.04%	4.18	6.98	11.10%	4.21	7.08	11.26%
	LSTM-M	4.10	6.92	10.91%	4.15	6.98	11.03%	4.26	7.18	11.44%
	SGMN	5.54	10.93	13.17%	5.61	10.99	13.35%	5.81	11.18	13.83%
	GCN-M (ours)	3.17	6.33	8.72%	3.23	6.47	8.99%	3.26	**6.35**	8.98%
Long-range missing	DCRNN	3.46	6.78	9.62%	3.54	6.96	9.75%	3.62	7.02	9.89%
	STGCN	3.71	7.20	9.91%	3.76	7.39	10.42%	3.88	7.66	10.67%
	GraphWaveNet	3.43	6.64	9.07%	3.57	6.92	9.62%	3.61	7.03	10.71%
	MTGNN	3.19	**6.32**	**8.48%**	3.39	6.85	9.21%	3.50	6.95	9.74%
	AGCRN	3.31	6.54	8.94%	3.33	6.68	8.98%	3.33	6.78	9.45%
	GTS	3.25	6.61	8.93%	3.29	6.74	8.85%	3.46	6.86	9.37%
	GRU	4.37	7.28	12.54%	4.47	7.44	11.72%	4.76	7.81	12.71%
	GRU-I	4.20	6.91	11.78%	4.09	6.97	15.42%	4.21	7.03	11.43%
	GRU-D	7.59	11.94	24.72%	7.82	12.46	25.67%	7.96	12.45	26.12%
	LSTM-I	4.20	7.09	11.08%	4.25	7.12	11.32%	4.36	7.32	11.56%
	LSTM-M	4.53	7.47	11.88%	5.21	8.84	15.34%	6.08	10.02	18.13%
	SGMN	9.47	14.30	20.72%	11.49	16.01	24.55%	13.97	18.24	29.10%
	GCN-M (ours)	**3.18**	6.39	8.71%	**3.23**	**6.56**	**8.78%**	**3.27**	**6.68**	**9.12%**
Mix-range missing	DCRNN	3.33	6.69	9.53%	3.47	6.85	9.64%	3.56	6.95	9.78%
	STGCN	3.56	7.12	9.81%	3.64	7.28	10.33%	3.73	7.51	10.62%
	GraphWaveNet	3.28	6.51	9.02%	3.43	6.78	9.52%	3.51	6.94	9.62%
	MTGNN	**3.04**	**6.18**	**7.84%**	3.14	6.72	9.07%	3.44	6.82	9.12%
	AGCRN	3.19	6.49	8.77%	3.21	6.56	8.95%	3.26	6.65	8.98%
	GTS	3.12	6.51	8.61%	3.22	6.61	8.84%	3.34	6.72	8.86%
	GRU	4.30	7.14	11.47%	4.35	7.31	11.68%	4.65	7.72	12.58%
	GRU-I	4.05	6.83	10.94%	4.01	6.86	10.83%	4.11	6.97	10.98%
	GRU-D	7.53	11.89	24.74%	7.71	12.32	25.43%	7.89	12.34	25.63%
	LSTM-I	4.15	6.94	11.06%	4.19	7.01	11.18%	4.30	7.18	11.35%
	LSTM-M	4.40	7.38	11.91%	5.14	8.77	14.92%	6.02	9.92	17.88%
	SGMN	9.33	14.16	20.47%	11.42	15.87	24.40%	13.84	18.13	28.97%
	GCN-M (ours)	3.08	6.34	8.59%	**3.12**	**6.42**	**8.71%**	**3.23**	**6.50**	**8.76%**

values from the missing values, as the zero values usually show periodicity while missing values show contingency [1]. The current node readings combined with the historical patterns will eliminate the effect from missing values but conserve that of zero values.

In Figure 6, we show the effects of the memory module's parameters L and S on the model's performance. The two parameters represent the searching range of the local temporal and spatial features, respectively. We report the model's evaluation errors with various missing rates. From the results in Figure 6, we observe that when the searching range becomes more extensive, the model's performance decreases more. This can be explained by the fact that the mean value of a larger space and the less recent observations will lead to a weaker information dependency with the current timestamp, thus affecting



(a) Missing Rate =10% (L) (b) Missing Rate =20% (L) (c) Missing Rate =40% (L)

(d) Missing Rate =10% (S) (e) Missing Rate =20% (S) (f) Missing Rate =40% (S)

Fig. 6: Parameters effects: we report the model errors of GCN-M on METR-LA dataset considering *a,b,c) L* observed samples before current timestamp for constructing the *Empirical Temporal Mean* in equation 1; *d,e,f) S* observed samples nearby current node for constructing the *Empirical Spatial Mean* in equation 2.

the information enrichment of the traffic embedding. In real-life datasets, we can set the parameters from a small value, such as considering local features during the last one hour (L=12) with five nearest sensor nodes (S=5).

5.5 RQ 3: Dynamic Graph Modeling

In the dynamic traffic system, the spatial dependency can be considered as a dynamic system status, which evolves along time [10]. The traffic observations at each timestamp are always adopted to characterize the dynamic traffic status and help learn the dynamic graphs [14]. However, due to the missed observations, the traffic status at certain timestamps can not be characterized, thus affecting the dynamic graph learning process.

This issue can be handled by the enriched traffic embeddings proposed in GCN-M. It allows considering the local static features and global historical patterns to avoid the deviation introduced by the missing values and help learn the dynamic graphs. To validate the performance of the learned dynamic graphs, we designed the following variants of our GCN-M model:

- GCN-M-obs: instead of using the enriched traffic embeddings, the raw traffic observations [14] are adopted to construct dynamic graphs.
- GCN-M-adp: instead of learning dynamic graphs and applying dynamic convolution, an adaptive static graph [21] is learned to do the graph convolution.
- GCN-M-pre: instead of learning graphs from the traffic embedding or observations, the predefined graphs [15] calculated with by the directed distances between traffic nodes are adopted for doing graph convolution.

Table 7: Performance comparison on graph module in mix-range missing value scenario with missing rate = 40%

	Models	Horizon=3 (15 mins) MAE	RMSE	MAPE	Horizon=6 (30 mins) MAE	RMSE	MAPE	Horizon=12 (60 mins) MAE	RMSE	MAPE
PEMS-BAY	GCN-M-obs	1.63	3.48	3.53%	1.93	4.16	4.31%	2.25	4.81	5.10%
	GCN-M-adp	1.53	3.11	3.14%	1.82	3.93	4.03%	2.14	4.72	4.92%
	GCN-M-pre	1.61	3.27	3.21%	1.87	4.05	4.11%	2.18	4.74	5.03%
	GCN-M-com	1.54	3.13	3.11%	1.79	3.92	3.97%	2.11	4.62	3.91%
	GCN-M	**1.45**	**3.03**	**3.09%**	**1.70**	**3.81**	**3.89%**	**2.06**	**4.64**	**4.86%**
METR-LA	GCN-M-obs	2.97	5.68	7.71%	3.31	6.57	8.78%	3.71	7.54	10.07%
	GCN-M-adp	2.84	5.51	7.44%	3.19	6.41	8.59%	3.68	7.34	9.96%
	GCN-M-pre	2.92	5.56	7.64%	3.23	6.42	8.63%	3.72	7.42	10.04%
	GCN-M-com	2.84	5.52	7.45%	3.17	6.4	8.63%	3.68	7.41	9.97%
	GCN-M	**2.82**	**5.47**	**7.42%**	**3.16**	**6.38**	**8.55%**	**3.58**	**7.31**	**9.92%**

- GCN-M-com: combine both predefined and learned static graphs [21] to do the graph convolution.

We show in Table 7 the performance comparison on various model variants of the spatial graph modeling. We report the model errors on multiple horizons. We consider the complex scenario of mix-range missing values with a missing rate of 40% on both PEMS-BAY and METR-LA datasets. The results in Table 7 suggest that the dynamic graphs learned from the enriched traffic embeddings perform the best compared to other variants. In contrast, the model obtains the worst performance when learning the dynamic graphs from the raw observations, which is mainly due to the missing values hindering the graph learning process in inferring the dynamic traffic status. GCN-M-obs perform even worse than GCN-M-adp in which the static graph is learned from the entire observations, eliminating the effect from local missing values.

5.6 Discussions

Our approach has several advantages. First, starting from the real-world data, GCN-M considered the complex scenarios of missing values in traffic data. Different from the previous work [3, 7, 18] which consider the missing value from a part of the real-life scenarios: either under the short-range or long-range missing settings, under partial or entire networking missing settings. GCN-M considers the complex mix-missing value context covering various real-life scenarios for missing values.

Second, GCN-M is capable of handling such complex missing value scenarios with a multi-scale memory module which combines local Spatio-temporal features (short-range missing, partial and entire network missing) and global historical patterns (long-range missing) to generate the enriched traffic embeddings; the embeddings allow distinguishing the inherent zero values from the missing values. In this way, GCN-M jointly models the Spatio-temporal patterns and missing values in one-step processing, which generally allows a better model performance than two-step processing [7].

Third, GCN-M allows generating reliable dynamic graphs from the enriched traffic embeddings, which opens a path for learning robust dynamic graphs

under missing value settings. Moreover, the generated dynamic graphs can corporate with various advanced graph convolution modules [22] to improve the model's performance further.

Last but not least, even though GCN-M is designed for traffic forecasting, it is applicable to wider application domains sharing similar Spatio-temporal characteristics and missing-value scenarios, such as crowd flow forecasting [23], weather and air pollution forecasting [9], etc. The Spatio-temporal patterns in those data and the missing values caused by the sensor issues or control center errors form similar research problems to this paper.

Table 8: Model efficiency: training time per epoch (s)

Models	PEMS-BAY	METR-LA	Models	PEMS-BAY	METR-LA
DCRNN	468.22	178.23	GRU	3.65	2.45
STGCN	55.32	27.70	GRU-I	4.22	3.67
GraphWaveNet	118.77	48.16	GRU-D	7.82	5.43
MTGNN	86.20	38.70	LSTM-I	4.32	4.64
AGCRN	67.40	32.9	LSTM-M	8.12	5.76
GTS	191.4	62.3	SGMN	3.45	2.38
GCN-M (ours)	241.69	118.65	-	-	-

However, GCN-M does have a limitation in terms of computational efficiency. Table 8 shows the per epoch training time comparison on the full datasets between GCN-M and the baseline models. The one-step processing baseline models are much more efficient than other models. This is basically because of their simple structure without integrating the costly graph convolution modules. GCN-M still performs better than DCRNN, but worse than other forecasting models. This is mainly caused by two factors: 1) generating the enriched traffic embeddings requires a huge computation cost on the attention score's calculation in the memory module; 2) generating the dynamic graphs for graph convolution requires learning a large number of parameters, thus increasing computation cost. Possible solutions might be to reduce the time complexity for calculating the attention with *ProbSparse Attention* proposed in [26] and to apply more efficient dynamic graph convolution such as graph tensor decomposition [10].

6 Conclusion

In this paper, we propose GCN-M, a graph convolutional network-based model for handling complex missing values in traffic forecasting. We studied the complex scenario where missing traffic values occur on both short & long ranges and on partial & entire transportation network. The enriched traffic embeddings learned by a Spatio-temporal memory module allow handling the complex missing values and constructing dynamic traffic graphs to improve the model's performance. A joint model optimization is applied to consider missing values and traffic forecasting in one-step processing. We compare GCN-M with

the one-step processing models, which are specifically designed for processing incomplete traffic data and the recent advanced traffic forecasting models. The extensive experiments on two benchmark traffic datasets with 12 baselines demonstrate that GCN-M shows a clear advantage under various scenarios of complex missing values, as compared to the advanced traffic forecasting models, while at the same time it maintains comparable performance on complete traffic datasets. These experiments also provide an up-to-date comparison of the traffic forecasting models would it be with or without missing values. In future work, we will explore the aforementioned optimizations to reduce computational costs. From a longer-term perspective, one can consider noisy data or external events that may impact the predictions.

References

[1] (2015) An introduction to the caltrans performance measurement system (pems). https://pems.dot.ca.gov/PeMS_Intro_User_Guide_v5.pdf

[2] BAI L, Yao L, Li C, et al (2020) Adaptive graph convolutional recurrent network for traffic forecasting. Advances in Neural Information Processing Systems 33

[3] Che Z, Purushotham S, Cho K, et al (2018) Recurrent neural networks for multivariate time series with missing values. Scientific reports 8(1):1–12

[4] Chung J, Gulcehre C, Cho K, et al (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning

[5] Cirstea RG, Yang B, Guo C (2019) Graph attention recurrent neural networks for correlated time series forecasting. MileTS19@ KDD

[6] Cui Z, Ke R, Pu Z, et al (2020) Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values. Transportation Research Part C: Emerging Technologies 118:102,674

[7] Cui Z, Lin L, Pu Z, et al (2020) Graph markov network for traffic forecasting with missing data. Transportation Research Part C: Emerging Technologies 117:102,671

[8] Guo S, Lin Y, Wan H, et al (2021) Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. TKDE

[9] Han J, Liu H, Zhu H, et al (2021) Joint air quality and weather prediction based on multi-adversarial spatiotemporal networks. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence

[10] Han L, Du B, Sun L, et al (2021) Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting. In: KDD, pp 547–555

[11] He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: CVPR, pp 770–778

[12] Jiang W, Luo J (2021) Graph neural network for traffic forecasting: A survey. arXiv preprint arXiv:210111174

[13] Lea C, Flynn MD, Vidal R, et al (2017) Temporal convolutional networks for action segmentation and detection. In: CVPR, pp 156–165

[14] Li F, Feng J, Yan H, et al (2021) Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. arXiv preprint arXiv:210414917

[15] Li Y, Yu R, Shahabi C, et al (2018) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: ICLR

[16] Shang C, Chen J, Bi J (2020) Discrete graph structure learning for forecasting multiple time series. In: International Conference on Learning Representations

[17] Tang X, Yao H, Sun Y, et al (2020) Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In: AAAI, pp 5956–5963

[18] Tian Y, Zhang K, Li J, et al (2018) Lstm-based traffic flow prediction with missing data. Neurocomputing 318:297–305

[19] Wang X, Ma Y, Wang Y, et al (2020) Traffic flow prediction via spatial temporal graph neural network. In: WWW, pp 1082–1092

[20] Weston J, Chopra S, Bordes A (2014) Memory networks. arXiv preprint arXiv:14103916

[21] Wu Z, Pan S, Long G, et al (2019) Graph wavenet for deep spatial-temporal graph modeling. In: IJCAI

[22] Wu Z, Pan S, Long G, et al (2020) Connecting the dots: Multivariate time series forecasting with graph neural networks. In: KDD, pp 753–763

[23] Xie P, Li T, Liu J, et al (2020) Urban flow prediction from spatiotemporal data using machine learning: A survey. Information Fusion 59:1–12

[24] Yoon J, Jarrett D, Van Der Schaar M (2019) Time-series Generative Adversarial Networks. In: NeurIPS

[25] Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: IJCAI

[26] Zhou H, Zhang S, Peng J, et al (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of AAAI