

# Spatio-Temporal Latent Graph Structure Learning for Traffic Forecasting

Jiabin Tang

*School of Computing and Artificial Intelligence,  
Southwest Jiaotong University  
Chengdu, China  
tjb2019111290@my.swjtu.edu.cn*

Tang Qian

*SWTJU-Leeds Joint School,  
Southwest Jiaotong University  
Chengdu, China  
qiantang007@my.swjtu.edu.cn*

Shijing Liu

*SWTJU-Leeds Joint School,  
Southwest Jiaotong University  
Chengdu, China  
shijingLiu@my.swjtu.edu.cn*

Shengdong Du\*

*School of Computing and Artificial Intelligence,  
Southwest Jiaotong University  
Chengdu, China  
sddu@swjtu.edu.cn*

Jie Hu

*School of Computing and Artificial Intelligence,  
Southwest Jiaotong University  
Chengdu, China  
jiehu@swjtu.edu.cn*

Tianrui Li

*School of Computing and Artificial Intelligence,  
Southwest Jiaotong University  
Chengdu, China  
trli@swjtu.edu.cn*

**Abstract**—Accurate traffic forecasting, the foundation of intelligent transportation systems (ITS), has never been more significant than nowadays due to the prosperity of smart cities and urban computing. Recently, Graph Neural Network truly outperforms the traditional methods. Nevertheless, the most conventional GNN-based model works well while given a pre-defined graph structure. And the existing methods of defining the graph structures focus purely on spatial dependencies and ignore the temporal correlation. Besides, the semantics of the static pre-defined graph adjacency applied during the whole training progress is always incomplete, thus overlooking the latent topologies that may fine-tune the model. To tackle these challenges, we propose a new traffic forecasting framework—Spatio-Temporal Latent Graph Structure Learning networks (ST-LGSL). More specifically, the model employs a graph generator based on Multilayer perceptron and K-Nearest Neighbor, which learns the latent graph topological information from the entire data considering both spatial and temporal dynamics. Furthermore, with the initialization of MLP-kNN based on ground-truth adjacency matrix and similarity metric in kNN, ST-LGSL aggregates the topologies focusing on geography and node similarity. Additionally, the generated graphs act as the input of the Spatio-temporal prediction module combined with the Diffusion Graph Convolutions and Gated Temporal Convolutions Networks. Experimental results on two benchmarking datasets in real world demonstrate that ST-LGSL outperforms various types of state-of-art baselines.

**Index Terms**—Traffic Forecasting, Graph Structure Learning, Graph Neural Networks, Temporal Convolutional Networks

## I. INTRODUCTION

Owing to the proliferation of urbanization, transportation systems have been under heavy stress and intractable. For-

This work was supported by the National Key R&D Program of China (2019YFB2101801), the Sichuan Science and Technology Program (No. 2021YFG0312) and the National Natural Science Foundation of China (No. 62176221).

\* Shengdong Du is the corresponding author.

tunately, with the pervasive use of the sensors deployed in cities, intelligent transportation systems (ITS) as well as smart city has intrigued the public and highlighted manifold merits. Traffic forecasting, which means predicting the future traffic volume from past observed data, truly plays a pivotal role in smart city efforts and Spatio-Temporal data-mining [1] [2]. Through forecasting future traffic, it provides related departments references in constructing smart cities from managing modern traffic, reducing the risk of public safety emergencies, etc. From another perspective, owning a smooth outing plan based on the information brought by traffic forecast, the living quality of most people could be improved as well. To sum up, accurate traffic forecasting is a meaningful and urgent problem.

More recently, deep learning technique, especially Graph Neural Networks (GNN), is widely used in traffic forecasting. The modeling of spatio-temporal traffic forecasting can be divided into two major portions: extracting temporal features and spatial features. To capture temporal dynamics, the traffic dataset can be regarded as a time series among nodes on a graph, consequently, the recurrent neural network (RNN) represented by GRU [3] [4] and LSTM [5] is commonly utilized to model the complex temporal correlations. However, so as to solve the problem of the high computational complexity of RNN and the accumulation of errors during iteration, gated temporal convolutions, which often consist of 1-D convolution and gated linear units (GLU), are widely used [6], to enable models to be more flexible and lightweight as well as facilitate parallel computation. Furthermore, in order to build a larger receptive field with the layer increasing, the dilated causal convolution networks represented by WaveNet [7] are used to extract the temporal correlation of the traffic dataset [8] [9] [10]. Meanwhile, with the broader applications

and excellent performance of the self-attention mechanism represented by transformer in natural language processing and computer vision, the attempt of self-attention in spatio-temporal data-mining, especially in traffic forecasting, has gradually sprung up [11].

On the other hand, GNN are adopted to represent the spatial topologies while dealing with the non-euclidean data in the real-world traffic forecasting scenario. Nevertheless, the graph structure, the premise of GNN methods' tremendous performance, isn't generally available or it could be fragmented. Hence, there are myriad methods and explorations to deal with this issue. The four most commonly employed solutions are as below: 1) **Pre-defined:** calculating the geographic distance between pair-wise nodes and then using the threshold function [6] [3]. 2) **Adaptive:** multiplying the learnable node embedding to build the graph structure adaptively [12]. 3) **Dynamic:** using hyper-network for fusing the current and historical information of the nodes [4] or tensor decomposition reversely [9] to generate the graph structure at each time step dynamically. 4) **GAT:** calculating the correlation weights based on GAT between the pair-wise nodes [13] on the fully connected graph to avoid defining general graph adjacency matrices [14].

However, the aforementioned manners have their own limitations and drawbacks:

- The pre-defined approaches with the same static graph adjacency during the whole training process are too intuitive and not able to embrace quite complicated topologies about spatial correlations.
- The adaptive and dynamic methods always initialize a random tensor which is automatically updated during training and the interpretability of the models is poor.
- The GAT-based approaches don't take into account not only the correlation between higher-order neighbors but also temporal dynamics.

To address the above challenges, inspired by graph structure learning applied across numerous domains [15], we propose a new framework Spatio-Temporal Latent Graph Structure Learning networks (ST-LGSL), for traffic forecasting. Primarily, in ST-LGSL, we adopt a Latent Graph Structure Learning module (LGSLm) within two different categories of graph generators based on Multilayer perceptron (MLP), called MLP-based generator. The LGSLm utilizes the entire traffic datasets, which could be regarded as each node having a feature vector at all time steps, as the input to mine the latent graph structure and topological information thus extracting both spatial and temporal features. Then the graph structure is fed into a spatio-temporal prediction module within Diffusion Graph Convolutions and Gated Temporal Convolutions Networks (TCN). Besides, to shorten training time and settle in a better local optimum, a *curriculum learning* strategy is employed.

We highlight the major contributions of our work as follows:

- We propose a novel traffic forecasting framework (ST-LGSL) which mines the latent graph structure by MLP,

with the cooperation of the stacked Spatio-Temporal prediction layers, which contain diffusion convolutions layers and temporal convolutions layers, and curriculum learning training strategy.

- ST-LGSL preserves the latent topologies about both node-wise similarity and geographical information, with the help of MLP-kNN graph generator which contains the similarity function and ground-truth initialization method. Compared with the aforementioned constructing methods, ST-LGSL also has stronger interpretability and contains high-order features of neighbors.
- Our extensive experimental results on two real-world datasets (evaluating both traffic speed and traffic flow) show that ST-LGSL outperforms various types of state-of-art baselines. And we conduct the model efficiency research especially Latent Graph Structure Learning Module during the forecasting process by visualizing the graph structures.

## II. RELATED WORK

### A. Traffic Forecasting

Existing models are divided into two major components in traffic forecasting: Spatial features modelling and Temporal features modelling. Initially, the statistical and traditional machine learning methods (e.g., auto-regressive (AR), and auto-regressive moving average (ARMA), auto-regressive integrated moving average (ARIMA) [16] [17], etc.) applied in traffic predictions regarding the task as a pure time series forecasting study, thus focusing on the single variant. Furthermore, with the rising growth of deep learning in various domains like Computer Vision and Natural Language Processing, spatial dependency modelling extract increasing attention. Different deep learning frameworks (i.e., CNN and GNN) and various spatial topology constructions are closely related. In the beginning, Zhang et al. [18] employed grid patterns to represent the traffic network structure so that CNN can be applied to extract the spatial topologies. Moreover, GNN show more vital representational ability for non-European data, naturally suitable for traffic forecasting. The key to the GNN is the construction of a graph and the methods of aggregating the features among graph nodes. The pre-defined methods, which consist of distance-measure [3] [6] and similarity-measure [19] function, are commonly used to constructing the graph structure previously. To explore more complex topological informations, an adaptive approach was employed in [12] and [8], while diverse methods are used to generate dynamic graph at every time step in [9], [4], [20], [21] and [22]. With the advance of the attention mechanism, GAT, which regards the graph as a full-connected graph and uses the weighted attention matrix to denote the correlation, is employed in [13] and [14]. As for aggregating the features from neighbours on graphs, the graph convolutions are commonly used, which is divided into main categories as follows: 1) the Chebyshev which uses Chebyshev polynomial [6] [23]; 2) diffusion convolutions which use bidirectional random walks [3] [24]; 3) mix-hop convolutions which contain information

propagation step and information selection step [10] [4]. As to temporal dependency modeling, the most common approach is to use RNN, e.g., GRU [3] [4] and LSTM [25]. To tackle the limitations of RNN, TCN is widely adopted, e.g., [6] [10] [8] [23], etc. Also, more recently, more applications of self-attention and transformer in traffic forecasting are emerging [11]. Instead of separating spatial and temporal dynamics in isolation, the exploration of fusing the two dimensions is more in-depth. In [26], spatial and temporal features are fused in a single graph which is provided for GNN training.

### B. Latent Graph Structure Learning

Owing to the expressive power of GNN applied in non-Euclidean data, GNN has earned a brilliant achievement in various domains. Nevertheless, GNN performs well only when the graph structures are given properly. To address the challenges, which the graph structures are not available or too noisy in the real world, Graph Structure Learning comes into being, especially latent graph structure learning as mentioned in [27]. Latent Graph Structure Learning primarily is studied in node classification. The sampling from the Bernoulli distribution was used in [28], which is learned during the training process for each possible edge, to construct graph structures. In [29], similarity metric and iterative method adopt to learn the graph structure and graph representation simultaneously. And the graph auto-encoder (GAE) was employed as a graph generator and a novel data augmentation way on graphs was proposed in [30]. In [31], Graph Convolutions Neural networks (GCN) is applied to calibrate the existing graph structures. The Multilayer perceptron was applied to construct the spatial topological structures in [27] and [32]. All methods mentioned above learn from the real data to generate the graph structures and mine the latent topologies. For more details, please refer to [15]. However, Graph Structure Learning rarely applies to Spatio-Temporal data-mining, especially traffic forecasting. Earlier, the LDS in [28] was applied to better solve multivariate time series predictions [33]. In this work, we adopt a similar idea to [33] and propose a novel Latent Graph Structure Learning framework, which is more suitable for traffic forecasting.

## III. METHODOLOGY

In this section, we primarily formalize problems and definitions of traffic forecasting and then present detailed descriptions of different parts of the proposed ST-LGSL. The overall framework of the proposed model is illustrated in Figure 1, which shows that the entire data is fed into the Graph Generator (Latent Graph Structure Learning module) as graph node features, later the generated graph structure and the sequential data are fed into stacked Spatio-Temporal layers with Gated TCN, Diffusion Graph Convolutions using Curriculum Learning Strategy. The more details are as follows:

### A. Problems and Definitions

Traffic forecasting is essentially based on pre-observed traffic data to predict future traffic. Generally, we can regard

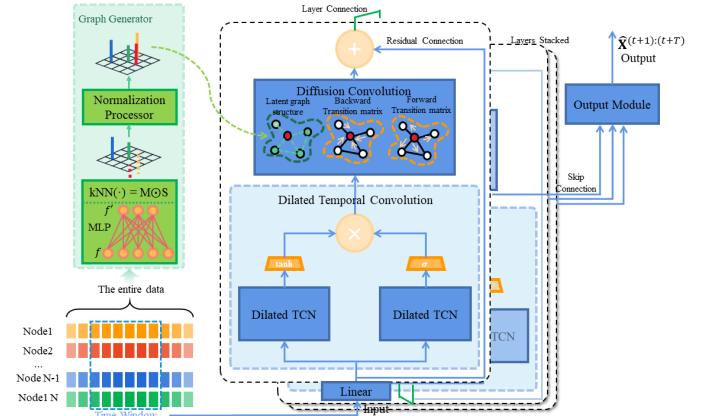


Fig. 1. The framework of the proposed ST-LGSL

the complex traffic road as a sensor network composed of  $M$  sensors, which can be expressed as a weighted directed graph  $\mathcal{G} = (V, \varepsilon, A)$ . Among them,  $V$  represents the position of each node of the transportation network, and  $|V| = M$ ;  $\varepsilon$  is a set of edges and  $A \in \mathbb{R}^{M \times M}$  is a weighted adjacency matrix used to reflect the proximity between nodes. We regard the traffic flow that can be observed in the graph as a graph signal, which can be expressed as  $X \in \mathbb{R}^{M \times N}$ , where  $N$  represents the number of features of each node (e.g., traffic volume, speed, etc.). In each period  $t$ , we denote  $X(t)$  as the observed graph signal. In general, the traffic forecasting problem can be summarized as: In a given transportation network  $\mathcal{G} = (V, \varepsilon, K)$ , it could realize the mapping of the graph signal from the historical period  $T$  to the future period  $T'$  after learning a function  $p(\cdot)$ . The more details are as follows:

$$[X_{1+t-T}, \dots, X_{(t)}; G] \xrightarrow{p(\cdot)} [X_{1+t}, \dots, X_{(T'+t)}]$$

$$\text{where, } (X_{1+t-T}, \dots, X_{(t)}) = X_{t+1-T:t} \in \mathbb{R}^{M \times N \times (T-1)} \\ (X_{1+t}, \dots, X_{(T'+t)}) = X_{t+1:t+T'} \in \mathbb{R}^{M \times N \times (T'-1)} \quad (1)$$

### B. Latent Graph Structure Learning

The Latent Graph Structure Learning module (LGSLm) consists of two components: 1) Graph Generator; 2) Normalization Processor. In the rest of this subsection, we initially formalize the general paradigm of LGSL and our two graph generators in detail respectively. Next, we introduce Normalization Processor more specifically.

1) *Graph Generator*: Inspired by [27], the general paradigm we abide by afterwards of a graph generator is defined as a function Eq. (2):

$$\mathbf{G} : \mathbb{R}^{n \times T} \rightarrow \mathbb{R}^{n \times n} \quad (2)$$

where  $n$  denotes the number of the nodes in a traffic dataset and  $T$  denotes the number of all recorded time steps. Hence, the entire data  $\mathbf{X} \in \mathbb{R}^{n \times T}$  and the generated graph structure  $\mathbf{A}' \in \mathbb{R}^{n \times n}$ . The function  $\mathbf{G}$  contains parameters  $\theta_G$ , which

tend to learn and optimize during training. Generally,  $\mathbf{G}$  is able to map the serialized traffic dataset to the graph structure in a data-driven way thus learning the latent graph topological information. However, the direct mapping is hard to train and computationally heavy. Therefore, we convert Eq. (2) to Eq. (3) and propose the following a Graph Generator to overcome it.

$$\mathbf{G} : \mathbb{R}^{n \times T} \rightarrow \mathbb{R}^{n \times T'} \rightarrow \mathbb{R}^{n \times n}, \quad (3)$$

where  $T'$  denotes the time embedding which is far less than  $T$ .

**MLP-kNN:** For this generator, we adopt the MLP and k-nearest neighbors to complete the first and second phases of the Eq. (3) respectively. MLP-kNN is formulized as:

$$\mathbf{G} : A' = kNN(MLP(\mathbf{X})) \quad (4)$$

where  $MLP : \mathbb{R}^{n \times T} \rightarrow \mathbb{R}^{n \times T'}$  employs MLP to encode the nodes embeddings,  $kNN : \mathbb{R}^{n \times T'} \rightarrow \mathbb{R}^{n \times n}$  uses kNN to produce a graph adjacency matrix and  $\mathbf{X} \in \mathbb{R}^{n \times T}$ . More specifically, as to the details of kNN, we primarily calculate the *top-k Sim-Matrix*  $\mathbf{T} \in \mathbb{R}^{n \times n}$  and *function Sim-Matrix*  $\mathbf{S} \in \mathbb{R}^{n \times n}$  which are defined as Eq. (5), Eq. (6):

$$\mathbf{T}_{ij} = \begin{cases} 1, & v_j \text{ is in the set of top } k \text{ similar nodes to } v_i \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

$$\mathbf{S}_{ij} = \mathbf{Sim}(E_i, E_j), \quad (6)$$

where  $E_i$  and  $E_j$  denote the node embeddings of node  $i$  and  $j$  after passing the MLP, and  $\mathbf{Sim}$  is the similarity function (could be cosine, etc.). Then the  $kNN(\cdot)$  is defined as:

$$A' = kNN(E) = \mathbf{T} \odot \mathbf{S}, \quad (7)$$

where the  $\odot$  denotes the element-wise Hadamard product and  $E$  represents the node embeddings produced by MLP.

To aggregate and enhance the geographical information, we initialize the parameters of MLP-kNN  $\theta_{\mathbf{G}}$  before training to enable it to generate the pre-defined adjacency matrix based on the distance function. Of course, such an initialization method, when applied for other tasks that do not have a pre-defined graph structure, is not required.

All in all, MLP-kNN generator, which learns the entire temporal features of all nodes and initializes with the ground-truth spatial topologies as the starting point, extracts spatial and temporal dynamics simultaneously.

2) *Normalization Processor:*  $A'$  that the graph generator produce may be neither symmetric nor normalized. To address this issue, let  $\tilde{\mathbf{A}} = \mathbf{Sym}(\mathbf{Norm}(A'))$ . In particular, we formulize as Eq. (8):

$$\tilde{\mathbf{A}} = \frac{1}{2} \mathbf{D}^{-\frac{1}{2}} (\text{ReLU}(A') + \text{ReLU}(A')^T) \mathbf{D}^{-\frac{1}{2}} \quad (8)$$

where ReLU is element-wise activation function,  $\mathbf{D}$  represents the diagonal degree matrices of  $\frac{1}{2}(\text{ReLU}(A') + \text{ReLU}(A')^T)$ . Further,  $\frac{1}{2}(\text{ReLU}(\cdot) + \text{ReLU}(\cdot)^T)$  corresponds to  $\mathbf{Sym}(\cdot)$  and  $\frac{1}{2}\mathbf{D}^{-\frac{1}{2}}(\cdot)\mathbf{D}^{-\frac{1}{2}}$  corresponds to  $\mathbf{Norm}(\cdot)$ .

### C. Gated Temporal Convolutions Layer

Due to heavy computational complexity and less parallel computation, we replace GRU or LSTM with TCN for our temporal modeling. In particular, we employ the Gated Temporal Convolutions Layer, which is separated into two portions: 1) **dilated causal convolution**; 2) **Gating mechanisms**.

1) *dilated causal convolution:* The receptive field of dilated causal convolution expands exponentially with the increase of the number of layers, while the standard 1-D convolutions enlarge linearly. Hence, the dilated causal convolution is a special instance of the standard 1-D convolutions. Formally, the dilated causal convolution is written as:

$$\mathbf{X} *_d \mathbf{f}(t) = \sum_{i=0}^{K-1} \mathbf{f}(t)_i \mathbf{X}_{t-d \times s} \quad (9)$$

where  $\mathbf{X} \in \mathbb{R}^T$  and  $\mathbf{f}(t) \in \mathbb{R}^K$  represent the feature vector and convolution filters at time step  $t$ , respectively,  $d$  denotes the dilation factor which actually enables dilated causal convolution to stack fewer layers yet capture increasing features than the standard one.

2) *Gating mechanisms:* Gating mechanisms are an effective technique widely applied in lots of domains (e.g., LSTM, GRU, etc.). We use Gating mechanisms combined with the above dilated causal convolution called Gate TCN like [8]. Mathematically, the Gate TCN is formulized as:

$$\mathbf{h} = \tanh(\Theta_1 *_d \mathcal{X} + \mathbf{b}) \odot \sigma(\Theta_2 *_d \mathcal{X} + \mathbf{c}) \quad (10)$$

where  $\tanh(\cdot)$  and  $\sigma(\cdot)$  denote the activation functions,  $\Theta_1$ ,  $\Theta_2$ ,  $\mathbf{b}$  and  $\mathbf{c}$  denote the parameters of convolutions,  $*_d$  represents the dilated causal convolution and the  $\odot$  denotes the element-wise Hadamard product.

### D. Diffusion Convolution Layer

GCN are broadly adopted to aggregate the spatial features from neighbors, which is a vital operation for GNN. In this work, we choose the diffusion convolution layer which is proved to be more suitable for spatio-temporal prediction. Formally, regarding a directed graph, the diffusion convolution layer we employed is written as:

$$\mathbf{Z} = \sum_{k=0}^K \mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} + \tilde{\mathbf{A}}^k \mathbf{X} \mathbf{W}_{k3}, \quad (11)$$

$$\mathbf{P}_f = \mathbf{A}/\text{rowsum}(\mathbf{A}), \quad (12)$$

$$\mathbf{P}_b = \mathbf{A}^T/\text{rowsum}(\mathbf{A}^T), \quad (13)$$

where  $\mathbf{P}_f$  and  $\mathbf{P}_b$  represent the forward transition matrix and backward transition matrix, respectively,  $\mathbf{A}$  denotes the ground-truth graph adjacency matrix,  $\tilde{\mathbf{A}}$  denotes the latent graph structure which the LGSL module generates, and  $\mathbf{W}_{k1}, \mathbf{W}_{k2}, \mathbf{W}_{k3}$  are learnable parameters. Certainly, the ground-truth graph adjacency matrix is not obligatory, while the ground-truth geographical information is unavailable or incomplete. Thus, the diffusion convolution layer could be defined as:

$$\mathbf{Z} = \sum_{k=0}^K \tilde{\mathbf{A}}^k \mathbf{X} \mathbf{W} \quad (14)$$

### E. Curriculum Learning Strategy

In traffic forecasting tasks, it is obvious that long-term predictions are much more difficult than short-term predictions, so long-term predictions contribute more to training loss than short-term predictions. Therefore, when designing the loss function, the common attempt is to optimize the overall loss of the model by minimizing the long-term predicted loss. Inspired by the idea of curriculum learning and [10], we adopt a similar Curriculum Learning Strategy yet without a subgraph algorithm. In particular, this training strategy begins with predicting purely next-one step afterward and gradually increases the steps of the predictions until the maximum steps of forecasting (12 steps/1 hour in this task).

Based on the above, we outline the train strategy in Algorithm 1.

---

**Algorithm 1** Curriculum learning training strategy of ST-LGSL

---

**Input:** The dataset  $\mathbf{O}$ , node feature  $\mathcal{F}$ , the model  $f(\cdot)$  with the total parameters  $\Theta$ , learning rate  $\gamma$ , batch size  $b$ , step size  $s$

**Set:** iterated number  $it = 1$ , task level  $r = 1$   
 fetch a batchsize of data ( $X \in \mathbb{R}^{b \times T \times N \times D}$ ,  $Y \in \mathbb{R}^{b \times T' \times N}$ ) from  $\mathbf{O}$

**repeat**

- if**  $it \% s == 0$  and  $r \leqslant T'$  **then**
- $r = r + 1$
- end if**
- compute  $\hat{\mathcal{Y}} = f(X[:, :, :, :], \mathcal{F}; \Theta)$
- compute  $\mathcal{L} = loss(\hat{\mathcal{Y}}[:, :r, :], Y[:, :r, :])$
- compute the stochastic gradient of  $\Theta$  according to  $\mathcal{L}$ .
- update model parameters  $\Theta$  with their gradients
- $it = it + 1$

**until** convergence

---

## IV. EXPERIMENT

In this section, we conduct extensive experiments to evaluate our proposed ST-LGSL model on two real-world datasets. Moreover, the experimental results are demonstrated compared with various competitive frameworks, both conventional machine learning models and deep learning models.

### A. Dataset

We validate the performance on two public transportation network datasets, e.g. a Los Angeles dataset and PEMS08, which are demonstrated in Table I.

Both datasets aggregate traffic data in 5 minutes, and each sensor contains 288 data records per day. The datasets contain three aspects of information, namely flow, speed, and occupancy. This work selects traffic speed from PeMSD8 and METR-LA while selecting traffic flow from PeMSD8. The spatial adjacency network that may be used for the ground-truth information for each dataset is constructed from the

actual road network based on distance, which is similar to the pre-processing in [3], which is defined as follows:

$$\mathbf{A}_{v_i, v_j} = \begin{cases} \exp\left(-\frac{d_{v_i, v_j}^2}{\sigma^2}\right), & \text{if } d_{v_i, v_j} \leq \kappa \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

TABLE I  
DESCRIPTION OF METR-LA AND PEMS08

| Datasets | Nodes | Edges  | TimeSteps |
|----------|-------|--------|-----------|
| METR-LA  | 207   | 295374 | 34272     |
| PEMS08   | 170   | 295    | 17856     |

### B. Baseline Methods

We compare ST-LGSL with those following models:

- **DCRNN:** DCRNN uses bidirectional random walks and encoder-decoder for the spatial dependency as well as adopts scheduled [3].
- **Graph WaveNet:** GraphWaveNet adds a self-adaptive adjacency matrix into graph convolution and employs wavenet frameworks [8].
- **ASTGCN:** ASTGCN develops a spatial-temporal attention mechanism to capture the spatial-temporal correlations through calculating the attention matrix [23].
- **GMAN:** Graph Multi-Attention Network (GMAN) uses an Encoder-decoder architecture, which consists of multiple attention mechanisms [34].
- **MTGNN:** MTGNN proposes a novel mix-hop propagation layer and a dilated inception layer for extracting spatial and temporal dynamics [10].

### C. Experiment Settings

In this experiment, we split the dataset METR-LA and PeMSD8 into training, validation and testing sets with a ratio of 7:2:1. Moreover, we set the batch size as 64. During training, we adopt *learning rate decay* strategy and choose  $1.0 \times 10^{-3}$  as the starting learning rate and weight decay is 0.0001. All experiments run on the Linux OS machine with an NVIDIA GeForce GTX 3080 (12GB) GPU and an NVIDIA Titan V (12GB) GPU. Furthermore, the proposed model is built by Pytorch.

On ST-LGSL, the arguments are set as follows: We set the ground-truth adjacency to double transition and set epoch to 1000 while employing the early stop strategy, where the tolerance is 100. In the MLP-kNN graph generator, we set the non-linear activation function to ReLU. As for the initialization of MLP-kNN, we adopt the ground-truth adjacency matrix as the target, the similarity function as cosine, and the  $k$  of kNN as 20. In contrast, the epoch of initialization is the hyperparameter that is further discussed in the later subsection. As to the normalization processor, we choose the symmetric and non-symmetric for comparison.

TABLE II  
PERFORMANCE OF SPEED PREDICTIONS COMPARISON BETWEEN BASELINE MODELS AND ST-LGSL ON METR-LA AND PEMSD08 DATASETS.

| Dataset        | Models        | Horizon 3     |             |               | Horizon 6     |             |               | Horizon 12    |             |               |
|----------------|---------------|---------------|-------------|---------------|---------------|-------------|---------------|---------------|-------------|---------------|
|                |               | MAE           | MAPE(%)     | RMSE          | MAE           | MAPE(%)     | RMSE          | MAE           | MAPE(%)     | RMSE          |
| PEMSD8         | VAR           | 1.1326        | 2.24        | 2.0714        | 1.7166        | 3.56        | 3.3052        | 2.1662        | 4.67        | 4.2502        |
|                | HA            | 2.8118        | 6.31        | 5.6763        | 2.8097        | 6.3         | 5.6731        | 2.8045        | 6.27        | 5.6645        |
|                | DCRNN         | 1.1957        | 2.36        | 2.4677        | 1.5349        | 3.25        | <b>3.3032</b> | 1.9051        | 4.27        | 4.1449        |
|                | ASTGCN        | 1.3792        | 2.96        | 2.9935        | 1.6445        | 3.58        | 3.6193        | 1.9946        | 4.36        | 4.2879        |
|                | GMAN          | 1.1375        | 2.34        | 2.6752        | <b>1.3236</b> | 2.93        | 3.3952        | 1.5121        | 3.55        | 4.0526        |
|                | Graph WaveNet | 1.1184        | 2.33        | 2.5531        | 1.3847        | 3.2         | 3.5325        | 1.6044        | 3.91        | 4.2426        |
|                | MTGNN         | 1.1404        | 2.34        | 2.5736        | 1.3975        | 3.14        | 3.4912        | 1.6324        | 3.94        | 4.2491        |
| <b>ST-LGSL</b> |               | <b>1.1153</b> | <b>2.19</b> | <b>2.4654</b> | 1.3752        | <b>2.92</b> | 3.3372        | <b>1.3333</b> | <b>2.87</b> | <b>3.1887</b> |
| METR-LA        | VAR           | 4.4250        | 10.2        | 7.8948        | 5.4152        | 12.7        | 9.1347        | 6.5278        | 15.8        | 10.1156       |
|                | HA            | 4.1619        | 13          | 7.8017        | 4.1633        | 13          | 7.8063        | 4.1671        | 13          | 7.8022        |
|                | DCRNN         | 2.7720        | 7.3         | 5.3864        | 3.1578        | 8.8         | 6.4517        | 3.6087        | 10.5        | 7.6066        |
|                | ASTGCN        | 4.8613        | 9.21        | 9.2724        | 5.4339        | 10.13       | 10.6118       | 6.5139        | 11.64       | 12.5235       |
|                | GMAN          | 2.7716        | 7.25        | 5.4834        | 3.077         | 8.35        | 6.3421        | 3.4068        | <b>9.72</b> | 7.2201        |
|                | Graph WaveNet | 2.6927        | 6.9         | 5.1537        | 3.0751        | 8.37        | 6.2289        | 3.5364        | 10.01       | 7.3732        |
|                | MTGNN         | 2.6906        | 6.86        | 5.1858        | 3.0588        | <b>8.19</b> | 6.1717        | 3.4989        | 9.87        | 7.2382        |
| <b>ST-LGSL</b> |               | <b>2.6750</b> | <b>6.86</b> | <b>5.0859</b> | <b>3.0537</b> | 8.4         | <b>6.1256</b> | <b>3.5036</b> | 10.1        | <b>7.2195</b> |

TABLE III  
PERFORMANCE OF FLOW PREDICTIONS COMPARISON BETWEEN BASELINE MODELS AND ST-LGSL ON PEMSD08 DATASET.

| Metrics         | VAR     | DCRNN   | STCCN   | ASTGCN  | Graph WaveNet | GMAN    | STGCN   | <b>ST-LGSL</b> |
|-----------------|---------|---------|---------|---------|---------------|---------|---------|----------------|
| Dataset: PeMSD8 | MAE     | 23.4689 | 17.8619 | 18.0223 | 18.6126       | 19.1384 | 14.8743 | 18.0211        |
|                 | MAPE(%) | 15.42   | 11.45   | 11.4    | 13.08         | 12.68   | 9.77    | 11.39          |
|                 | RMSE    | 36.3353 | 27.8352 | 27.8326 | 28.1631       | 31.0535 | 24.0675 | 27.8331        |
| Improvement(%)  | MAE     | 40.05   | 21.23   | 21.94   | 24.41         | 26.49   | 5.41    | 21.93          |
|                 | MAPE    | 41.18   | 20.79   | 20.44   | 30.66         | 28.47   | 7.16    | 20.37          |
|                 | RMSE    | 36.85   | 17.56   | 17.55   | 18.52         | 26.11   | 4.66    | 17.56          |

#### D. Evaluation Metric

In our experiments, we choose mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE), which is widely adopted in traffic forecasting tasks, to evaluate our model comprehensively with others. The definition of MAE that is similar to [8] is as below:

$$Loss(\hat{\mathbf{X}}^{(t+1):(t+T)}; \Theta) = \frac{1}{T \times N} \sum_{i=t+1}^{i=t+T} \sum_{j=1}^{j=N} \left| \hat{\mathbf{X}}_j^{(i)} - \mathbf{X}_j^{(i)} \right| \quad (16)$$

where the  $T$  and  $N$  are the total number of time steps and graph nodes, respectively.

#### E. Experimental Results

The comparison of different models with ST-LGSL predicting the traffic speed on two different datasets is shown in Table II, while predicting the traffic flow on PEMSD-08 is illustrated in Table III.

Experimental results show that our ST-LGSL consistently outperforms the baseline model in speed and flow prediction.

We compare the prediction performance of ST-LGSL and other models in 60 minutes on PeMSD8 and METR-LA datasets from the perspective of speed prediction and focus comprehensively on short-term, mid-term and long-term evaluations.

TABLE IV  
ABLATION STUDY

| Methods | ST-LGSL       | w/o Sym       | w/o Graph-generator | w/o Pre-defined | w/o CL        |
|---------|---------------|---------------|---------------------|-----------------|---------------|
| MAE     | 2.675±0.0036  | 2.6821±0.0004 | 3.0390±0.0226       | 2.7228±0.0192   | 3.0393±0.0083 |
| MAPE(%) | 6.88±0.0008   | 6.92±0.0009   | 8.35±0.0008         | 6.97±0.0009     | 8.45±0.0008   |
| RMSE    | 5.0859±0.0386 | 5.1019±0.0113 | 6.0382±0.0189       | 5.1956±0.0123   | 6.037±0.0145  |

- Our ST-LGSL performs better predictions than the traditional temporal models such as HA and VAR by an overwhelming margin on every dataset because our MLP-kNN generator in the Latent Graph Structure Learning module mines the complex topological information and applies it to the graph convolution so that each node fully aggregates the features of the adjacent nodes according to the learned graph structure.
- Overall, our ST-LGSL is more significant than all the spatial-temporal models. Compared with DCRNN, ST-LGSL is lower down MAE by 6.72%, 10.40%, 30.01% over the horizons 3, 6 and 9 on the dataset PEMSD08; On METR-LA, moreover, compared with GraphWaveNet, ST-LGSL lower down MAE by 0.66%, 0.70%, 0.93%. This is because unlike DCRNN and MTGNN, which use a static adjacency matrix and ASTGCN, GMAN and Graph WaveNet, which employ an adaptive adjacency matrix based on simple tensor multiplication or attention mechanism, we use MLP-kNN. The MLP-kNN module

combines geographic information and node similarity relationships to generate latent graph structures, and continuously optimize and tend to converge during training.

- In terms of long-term time series forecasting, as the forecasting time length increases, the advantages of our ST-LGSL over GraphWaveNet are becoming increasingly apparent, which embodies the benefits of Latent Graph Structure Learning module over adaptive graph generator in long-term forecasting.
- Compared with GMAN in long-term forecasting, our ST-LGSL is slightly inferior in the long-term forecast of the next 1 hour due to the more substantial power of multi-head self-attention in extracting the long-lasting temporal dependency.

On the other hand, our ST-LGSL also excels in traffic flow prediction.

- ST-LGSL make a significant improvement in traffic flow forecasting. Compared with DCRNN, ST-LGSL lower MAE by 21.23%, RMSE by 17.56% and MAPE by 20.79%; For another spatial-temporal model, Graph WaveNet, MAE, RMSE and MAPE has decreased 26.49%, 26.11% and 28.47%. As a latent variable, traffic flow is more closely related to the topology of the road network than traffic speed. Hence, it is easier to mine better latent graph structures to optimize graph convolutional operations.

1) *Ablation Study*: In order to validate the effectiveness of significant components that contribute to the improvement of our ST-LGSL, we conduct an ablation study on METR-LA. We name ST-LGSL without different components as follows:

- **w/o Sym**: ST-LGSL without symmetrizing the graph.
- **w/o Graph-generator**: ST-LGSL without MLP-kNN graph generator.
- **w/o Pre-defined**: ST-LGSL without a pre-difined graph as the ground-truth during the initialization of MLP-kNN.
- **w/o CL**: ST-LGSL without curriculum learning, which means training without gradually increasing the length of prediction.

We repeat above different models ten times with the same experimental settings as before, and the results are shown in Table IV and Figure 2. Regarding pre-defined as the initialized target significantly improves the performance because the MLP-kNN obtains the geographical information at the beginning of the training. The operations of symmetrizing are practical because the real-word adjacency is symmetric. Similarly, curriculum learning is effective, making our model converge quickly and settle in a better local optimum. The effect of the graph generator is highly significant by making full use of the entire temporal features of all nodes.

2) *Hyperparameter*: To improve the performance of ST-LGSL, we also carry out additional work on the hyperparameter set: The number of Epoch while initializing the parameters in the MLP-kNN graph generator. MLP-epoch means when all the data is fed into the MLP-kNN graph generator, the

TABLE V  
HYPERPARAMETER

| Epoch   | 10            | 100           | 1000          | 10000         |
|---------|---------------|---------------|---------------|---------------|
| MAE     | 2.6833±0.0098 | 2.6913±0.0128 | 2.6752±0.0004 | 2.6889±0.0039 |
| MAPE(%) | 6.97±0.0008   | 6.9±0.0009    | 6.88±0.0009   | 6.93±0.0009   |
| RMSE    | 5.1314±0.0413 | 5.1307±0.0329 | 5.0859±0.0386 | 5.134±0.0233  |

number of forwarding calculations and backpropagation times. We repeat the different cases over 15 times and calculate the average of MAE, RMSE, and MAPE in Table V. Extensive experimental results demonstrate that when the mlp-epoch is equal to 1000, the model’s performance is the best, and the performance will be degraded if it is too large or too small.

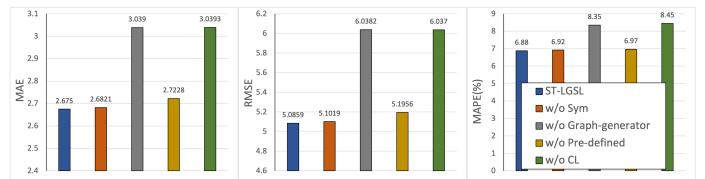


Fig. 2. Ablation Study

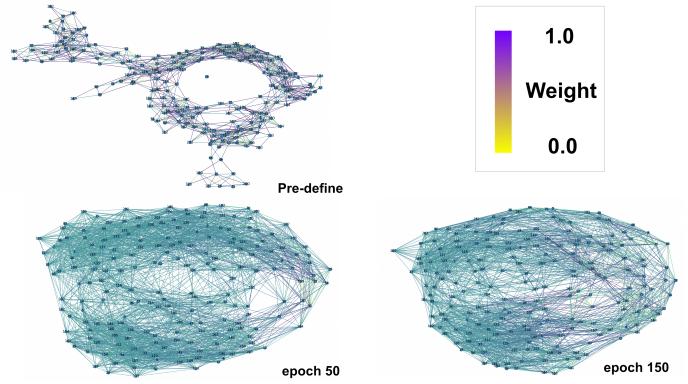


Fig. 3. Adjacency Matrix Visualization

3) *Learned Graph Structure Analysis*: In order to analyse the Learned Graph Structure intuitively, we visualize the adjacency matrix on the METR-LA dataset in Figure 3, which contains the adjacency matrix in a pre-defined way using the distance function as mentioned above, and epochs 50 and 150 during training, respectively. The 207 nodes represent the 207 sensors in the METR-LA dataset, the connection of the two nodes represents the spatial proximity between the two sensors, and the weight of the adjacency matrix is represented by the colour of the edges as the weight increases from 0 to 1. The colour of the edge will gradually deepen. Figure 3 can reflect as the model is trained, adjacency is dynamically learning and updating so that the latent graph topologies are learned. Compared with the pre-defined graph structure, the

latent graph structures obtain more node-wise correlation as the train epoch increases.

## V. CONCLUSION

In this work, we proposed a framework, Spatio-Temporal Latent Graph Structure Learning networks (ST-LGSL), for traffic forecasting. Our ST-LGSL was equipped with a Latent Graph Structure Learning module (LGSLm), which learns from the entire dataset as graph nodes with time-aware features to extract spatial and temporal dependency. The LGSLm consists of an MLP-kNN graph generator that employs the ground-truth initialization and similarity metric to obtain geographical and node-similarity information in the learned graph structure. Moreover, the latent graph structure was fed into the Spatio-temporal predicting framework, which contains the gated temporal convolution and diffusion convolutions layers. The experimental results demonstrated that our ST-LGSL significantly outperforms various baselines. In the future, we tend to add self-attention into temporal features modeling to solve the long-term forecasting better.

## REFERENCES

- [1] X. Yi, Z. Duan, T. Li, T. Li, J. Zhang, and Y. Zheng, “CityTraffic: Modeling Citywide Traffic via Neural Memorization and Generalization Approach,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2665–2671.
- [2] P. Xie, T. Li, J. Liu, S. Du, X. Yang, and J. Zhang, “Urban flow prediction from spatiotemporal data using machine learning: A survey,” *Information Fusion*, vol. 59, pp. 1–12, 2020.
- [3] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *International Conference on Learning Representations*, 2018.
- [4] F. Li, J. Feng, H. Yan, G. Jin, D. Jin, and Y. Li, “Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution,” *ArXiv*, vol. abs/2104.14917, 2021.
- [5] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in neural information processing systems*, 2015, pp. 802–810.
- [6] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2017, pp. 3634–3640.
- [7] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [8] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for deep spatial-temporal graph modeling,” in *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19*, 2019, pp. 1907–1913.
- [9] L. Han, B. Du, L. Sun, Y. Fu, Y. Lv, and H. Xiong, “Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 547–555.
- [10] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 753–763.
- [11] X. Ye, S. Fang, F. Sun, C. Zhang, and S. Xiang, “Meta graph transformer: A novel framework for spatial-temporal traffic prediction,” *Neurocomputing*, 2021.
- [12] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” in *Advances in neural information processing systems*, 2020, pp. 17804–17815.
- [13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations*, 2018.
- [14] X. Zhang, C. Huang, Y. Xu, and L. Xia, “Spatial-Temporal Convolutional Graph Attention Networks for Citywide Traffic Flow Forecasting,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1853–1862.
- [15] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, “Deep graph structure learning for robust representations: A survey,” *ArXiv*, vol. abs/2103.03036, 2021.
- [16] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.
- [17] C. Chen, J. Hu, Q. Meng, and Y. Zhang, “Short-time traffic flow prediction with arima-garch model,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 607–612.
- [18] J. Zhang, Y. Zheng, and D. Qi, “Deep spatio-temporal residual networks for citywide crowd flows prediction,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.
- [19] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, “Passenger demand forecasting with multi-task convolutional recurrent neural networks,” in *Advances in Knowledge Discovery and Data Mining*, 2019, pp. 29–42.
- [20] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, “Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting,” in *Proceedings of the AAAI conference on artificial intelligence*, 2019, pp. 890–897.
- [21] K. Guo, Y. Hu, Z. Qian, Y. Sun, J. Gao, and B. Yin, “Dynamic graph convolution network for traffic forecasting based on latent network of laplace matrix estimation,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.
- [22] T. Mallick, M. Kiran, B. Mohammed, and P. Balaprakash, “Dynamic graph neural network for traffic forecasting in wide area networks,” in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 1–10.
- [23] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 922–929.
- [24] X. Zhang, C. Huang, Y. Xu, L. Xia, P. Dai, L. Bo, J. Zhang, and Y. Zheng, “Traffic Flow Forecasting with Spatial-Temporal Graph Diffusion Network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 15008–15015.
- [25] S. Du, T. Li, Y. Yang, X. Gong, and S.-J. Horng, “An lstm based encoder-decoder model for multistep traffic flow prediction,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.
- [26] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 4189–4196.
- [27] B. Fatemi, L. E. Asri, and S. M. Kazemi, “SLAPS: Self-supervision improves structure learning for graph neural networks,” in *Advances in Neural Information Processing Systems*, 2021.
- [28] L. Franceschi, M. Niepert, M. Pontil, and X. He, “Learning discrete structures for graph neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 1972–1982.
- [29] Y. Chen, L. Wu, and M. J. Zaki, “Deep iterative and adaptive learning for graph neural networks,” *arXiv preprint arXiv:1912.07832*, 2019.
- [30] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, “Data augmentation for graph neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 11015–11023.
- [31] D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang, “Graph-revised convolutional network,” in *Machine learning and knowledge discovery in databases*, 2021, pp. 378–393.
- [32] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, “Learning representations of irregular particle-detector geometry with distance-weighted graph networks,” *The European Physical Journal C*, vol. 79, pp. 1–11, 2019.
- [33] C. Shang, J. Chen, and J. Bi, “Discrete graph structure learning for forecasting multiple time series,” in *International Conference on Learning Representations*, 2021.
- [34] C. Zheng, X. Fan, C. Wang, and J. Qi, “Gman: A graph multi-attention network for traffic prediction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 1234–1241.