Knowledge-Based Systems 242 (2022) 108199

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Adaptive Spatio-temporal Graph Neural Network for traffic forecasting

Xuxiang Ta [a], Zihan Liu [a], Xiao Hu [b], Le Yu [a], Leilei Sun [a,*], Bowen Du [a]

[a] National Laboratory of Software Development Environment, Beihang University, Beijing, 100191, China
[b] Beijing Bytedance Network Technology Co., Ltd., Beijing, 100089, China

ARTICLE INFO

Article history:
Received 8 June 2021
Received in revised form 26 December 2021
Accepted 8 January 2022
Available online 26 January 2022

Keywords:
Traffic forecasting
Spatio-temporal data
Neural networks
Graph convolution

ABSTRACT

Accurate traffic forecasting is of vital importance for the management and decision in intelligent transportation systems. Indeed, it is a nontrivial endeavor to predict future traffic conditions due to the complexity of spatial relationships and temporal dependencies. Recent research developed Spatio-Temporal Graph Neural Networks (ST-GNNs) to capture the spatio-temporal correlations and achieved superior performance. However, the graph adjacency matrices that most ST-GNNs use are either pre-defined by heuristic rules or directly learned with trainable parameters. While node attributes, which record valuable information of traffic conditions, have not been fully exploited to guide the learning of better graph structure. In this paper, we propose an **Ada**ptive **S**patio-**T**emporal graph neural **Net**work, namely Ada-STNet, to first derive optimal graph structure with the guidance of node attributes and then capture the complicated spatio-temporal correlations via a dedicated spatio-temporal convolution architecture for multi-step traffic condition forecasting. Specifically, we first propose a graph structure learning component to obtain an optimal graph adjacency matrix from both macro and micro perspectives. Next, we design a dedicated spatio-temporal convolution architecture to learn spatial relationships and temporal dependencies. Moreover, we present a two-stage training strategy to improve the model performance. Extensive experimental results on real-world datasets demonstrate the effectiveness and interpretability of our approach.

1. Introduction

Traffic forecasting plays an essential role in intelligent transportation systems. Accurate traffic forecasting is beneficial to rational dynamic traffic planning and efficient traffic resource allocation [1,2]. Indeed, it is challenging to achieve precise traffic forecasting due to the following reasons. Firstly, the spatial relationships between different regions are complicated and hard to be explicitly described. Secondly, the temporal dependencies across time dimension may be related with each other during both regular and irregular periods. Mining the underlying spatio-temporal patterns is the key to realize precise traffic forecasting.

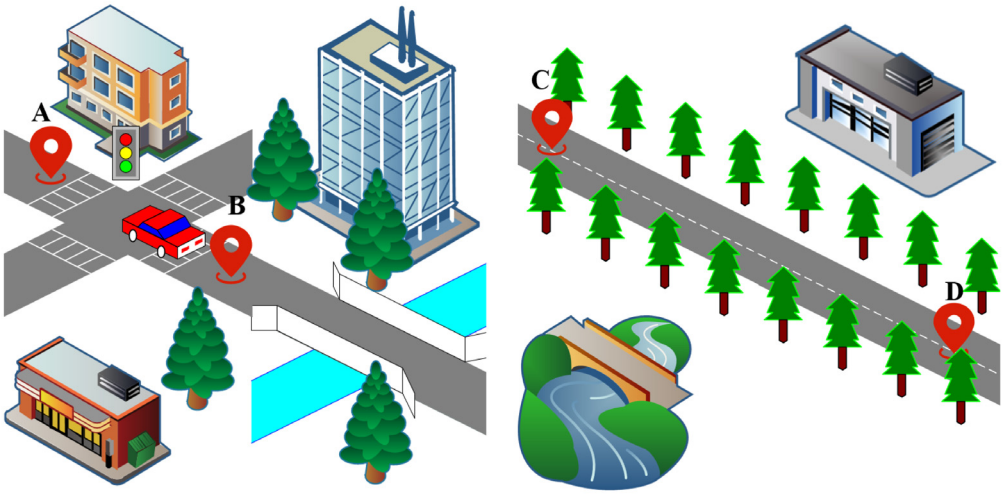
Indeed, existing deep learning approaches on traffic forecasting could be summarized into two categories: grid-based methods and graph-based methods. Grid-based methods usually first divide the studied area into regular grids and then leverage deep neural networks to extract valuable spatio-temporal information among the grids, i.e., a Convolution Neural Network (CNN) to learn spatial relationships and an Recurrent Neural Network (RNN) to capture temporal dependencies [1,3–7]. However, these methods ignore the road network topology and the spatial relationships of different regions could not be fully exploited. Due to the recent success of Graph Neural Networks (GNNs) in handling graph topology [8–10], Spatio-Temporal Graph Neural Networks (ST-GNNs) have been developed to extend GNNs into the field of traffic forecasting and achieved satisfactory performance than grid-based methods [11–21].

ST-GNNs usually design a GNN for spatial relationship learning, and either an RNN [11–14,20,22,23] or a CNN [15–19,21] for temporal dependency extracting. For example, Li et al. [20] proposed to modify the gated recurrent unit (GRU) via replacing the matrix production operation with graph convolution. Yu et al. [21] extracted the spatial relationships of traffic condition by a GCN and captured the temporal dependencies by causal convolution. It could be concluded that most of the existing ST-GNNs first build a graph determined by pre-defined measurements and then study on the constructed graph.

However, a pre-defined graph may be insufficient to contain all the useful information in the complicated traffic forecasting scenarios. In most cases, such a graph is determined by Euclidean distance between each pair of regions. As shown in Fig. 1(a) and Fig. 1(b), some implicit factors (e.g., road characteristics, POI distribution, regional function) could affect the relationships between two regions, so that the spatial relationships between

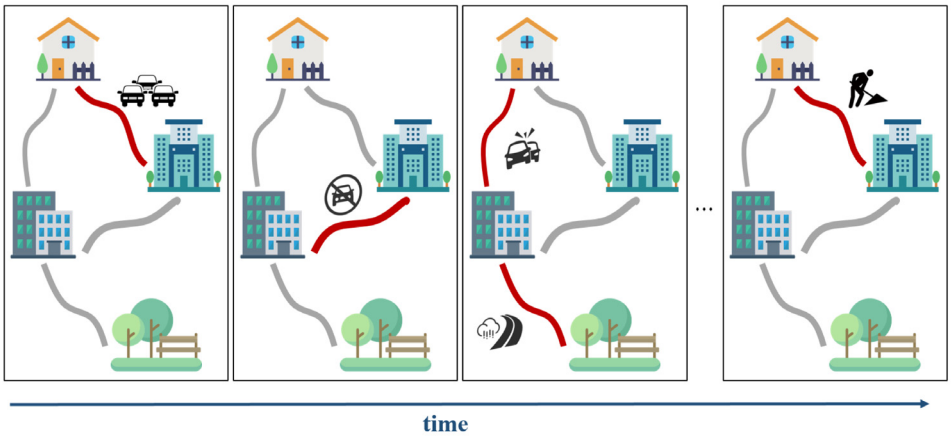
* Corresponding author.
E-mail addresses: taxuxiang@buaa.edu.cn (X. Ta), zihan@buaa.edu.cn (Z. Liu), huxiao.m@bytedance.com (X. Hu), yule@buaa.edu.cn (L. Yu), leileisun@buaa.edu.cn (L. Sun), dubowen@buaa.edu.cn (B. Du).

https://doi.org/10.1016/j.knosys.2022.108199
0950-7051/



(a) Although A and B are close, they are weakly depend on each other due to there is a crossroads between them.

(b) Although C and D are very far apart, they strongly inter-dependent as they locate in the same highway.



(c) Spatial relationships in traffic network vary during different periods due to traffic events such as traffic congestion, restriction, accidents, bad weather or road-works.

Fig. 1. Spatio-temporal correlations are impacted by multiple factors and hard to be explicitly described.

regions separated by the same distance would be totally different. Some scholars utilize different metrics, such as functional similarity [22,23] or transportation connection [22] to construct the graph, while each metric only reflects one aspect of the inherent spatial structure of traffic network. Hence, pre-defined rules are inappropriate or even wrong to describe such relationships in traffic network.

Moreover, as illustrated in Fig. 1(c), the relationships between the same regions may also vary across time both regularly or irregularly due to traffic events. Although Wu et al. [18] and Bai et al. [24] proposed to learn adaptive graph adjacency matrix for traffic forecasting, the derived graph structure remains unchanged during different periods thus ignoring the changes of graph structure over time. In fact, node attributes record valuable information of traffic conditions and it would be beneficial to incorporate node attributes in learning optimal graph structure.

In this paper, we propose an **Ada**ptive **S**patio-**T**emporal graph neural **Net**work, namely Ada-STNet, for traffic forecasting. Specifically, Ada-STNet consists of two components: an adaptive graph structure learning component and a multi-step traffic condition forecasting component. The first module is designed to derive an optimal graph structure with the guidance of node attributes to capture the complicated and time-varying spatial relationships between different regions. The second module forecasts multi-step traffic conditions by stacking multiple spatio-temporal layers based on the learned graph structure, where each spatio-temporal layer performs convolutions in both spatial and temporal dimensions. Extensive experiments are conducted on real-world traffic datasets, and the results demonstrate the superiority of our approach over existing methods. The contributions of our work are summarized as follows:

- An adaptive graph structuring learning component is presented to derive the optimal graph structure from the macro and micro perspectives, respectively. In particular, the macro perspective describes the complicated spatial relationship in the long term and the micro perspective reflects the time-varying spatial relationship in the short term.
- We build a spatio-temporal convolution network to forecast future traffic conditions during multiple periods, which could capture both spatial relationships and temporal dependencies for traffic forecasting.

- A two-stage training strategy is introduced to further enhance the model performance, which could reduce the difficulty in training the proposed model.

The rest of the paper is organized as follows: Section 2 reviews existing research related to our work. Section 3 introduces the problem formalization and background knowledge. Section 4 presents the proposed model. Section 5 evaluates the proposed model through experiments. Section 6 concludes the entire paper.

2. Related work

In this section, we review the existing literature relevant to our work and point out the limitations of previous studies.

2.1. Traffic forecasting

Traffic forecasting has been widely studied in the past decades [2]. Recent traffic prediction approaches first divide the studied area into regular grids and then make traffic forecasting for the grids. Specifically, they employ deep neural networks to extract complicated spatio-temporal correlations, which mainly consist of a CNN for spatial distribution learning and an RNN for temporal series modeling. Lv et al. [1] simultaneously learned spatial and temporal patterns by a stacked autoencoder and a linear regression layer was used to predict traffic flow on road segments. Yu et al. [3] combined deep CNNs and deep LSTM for traffic speed prediction. Jiang et al. [4] proposed a DeepUrbanMomentum model for predicting crowd mobility under big rare events or disasters, when people change their behaviors dramatically. Cui et al. [5] considered both forward and backward dependencies of time series and captured both spatial features and bidirectional temporal dependencies by LSTM. Lv et al. [6] proposed a look-up convolution layer to embed road network to capture spatial features and then utilized LSTM to extract temporal features. Ye et al. [7] extracted spatial relationships by an autoencoder composed of CNNs and then utilized a heterogeneous LSTM to capture the dynamics of multiple transportation demands.

Since the above methods solve the traffic forecasting problem with a grid-like data structure, they are able to take advantage of off-the-shelf solutions and acquire considerable performance improvement. However, these methods ignore the topological structure of traffic network and the correlations of different regions are still not fully explored.

2.2. Spatio-temporal graph neural networks

Due to the success of both graph neural networks and sequential modeling methods, spatio-temporal graph neural networks have been proposed to capture the spatial relationships and temporal dependencies simultaneously [11–23]. For example, Li et al. [20] modified the Gated Recurrent Unit (GRU) by replacing its matrix production operation to graph convolution for traffic forecasting. Pan et al. [12] encoded node and edge attributes, such as Points-Of-Interest (POI), by meta-knowledge learners. Yu et al. [21] combined GCN and CNN for extracting temporal and spatial correlations of traffic condition respectively. Overall, spatio-temporal graph neural networks usually consist of a GNN for spatial distribution modeling, and either a RNN [11–14,20,22,23] or a CNN [15–19,21] for extracting temporal information. Most of existing spatio-temporal graph neural networks first construct an adjacency matrix determined by pre-defined measurements, such as spatial distance, functional similarity or transportation connection [22,23], and then learn on the pre-defined adjacency matrix.

However, the pre-defined adjacency matrix may be insufficient or even incorrect to describe the spatial relationships between nodes, since the manually defined rules could not cover all the valuable information in the complex traffic forecasting scenarios. And solely learning on such a matrix would lead to inferior performance.

2.3. Graph structure learning

Recent literature has focused on learning optimal structure for graphs to facilitate the performance of GNNs [25,26]. Existing graph structure learning methods could be classified into three categories: (1) metric learning approaches, which define different metrics to measure node relationships in the graph [27–29]; (2) probabilistic modeling approaches, which generate graph via sampling from certain distributions and model the sampling probability of edges with learnable parameters [30,31]; (3) direct optimization approaches, which treat the whole graph as learnable parameters and optimize the parameters with GNN parameters [32,33].

In the field of traffic forecasting, although some recent methods aim to learn adaptive graph adjacency matrix [18,24], these methods directly design learnable parameters to generate the graph structure without considering the node attributes, making the models hard to be optimized especially when the training data is small-scale and sparse. Node attributes record the traffic conditions and could contribute to better graph structure generation.

In this paper, we aim to learn optimal graph structure from node attributes adaptively and achieve accurate traffic forecasting by considering complicated spatio-temporal relationships based on the learned graph.

3. Preliminaries

In this section, we first introduce the problem formalization. Notations used in the paper are shown in Table 1. Then, we provide the background knowledge related to our work.

Table 1
Notations and descriptions.

Notation	Description
N	Number of nodes
F	Dimension of node attributes
S, T	Window size of historical and future traffic conditions
$\mathcal{X} \in \mathbb{R}^{S\times N\times F}$	Node attributes that record historical traffic conditions
$\mathcal{Y}, \hat{\mathcal{Y}} \in \mathbb{R}^{T\times N\times F}$	Real and predicted future traffic conditions
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$	A graph defined by nodes, edges and adjacency matrix
$g(\cdot,\cdot)$	Graph structure learner
$h(\cdot,\cdot)$	Multi-step traffic condition predictor

3.1. Problem formalization

Given historical traffic records from N regions on a traffic network, the task of traffic forecasting is to predict the future traffic conditions for each region. Following previous studies, we define N regions and their pair-wise connections as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, where $\mathcal{V}$ is a set of $|\mathcal{V}| = N$ nodes, $\mathcal{E}$ is a set of edges, and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a weighted adjacency matrix representing the nodes' proximities between any pair of nodes. The traffic historical records on $\mathcal{G}$ at time t are denoted as a graph signal $\boldsymbol{X}_{(t)} \in \mathbb{R}^{N\times F}$, where F is the dimension of each node's attributes. The traffic forecasting problem aims to learn a function f that is able to forecast T future graph signals $\hat{\mathcal{Y}} = \left[\hat{\boldsymbol{X}}_{(t+1)}, \ldots, \hat{\boldsymbol{X}}_{(t+T)}\right] \in \mathbb{R}^{T\times N\times F}$ given S historical graph signals $\mathcal{X} = \left[\boldsymbol{X}_{(t-S+1)}, \ldots, \boldsymbol{X}_{(t)}\right] \in \mathbb{R}^{S\times N\times F}$ and the adjacency matrix of traffic network $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, that is

$$\hat{\mathcal{Y}} = f(\mathcal{X}, \boldsymbol{A}), \tag{1}$$

where $\hat{\mathcal{Y}}$ is expected to be close to $\mathcal{Y}$ as much as possible.

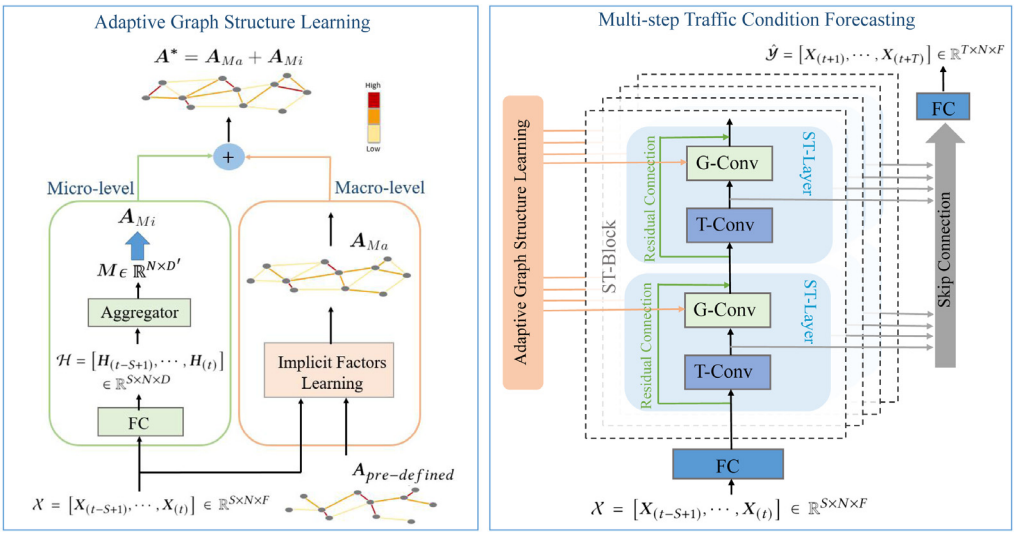


Fig. 2. Framework of the proposed Ada-STNet. Ada-STNet first learns the optimal graph structure to capture spatial relationships of nodes via a hybrid macro/micro level graph structure learning component, and then makes multi-step traffic forecasting by stacking several spatio-temporal convolution layers. FC: Fully Connected Layer; G-Conv: Graph Convolution Layer; T-Conv: Temporal Convolution Layer.

3.2. Graph convolution

Traditional CNN is limited to modeling Euclidean data, and GCN is therefore used to model non-Euclidean spatial structure data, which is more in line with the structure of traffic road network. Existing graph convolutional neural networks can be mainly divided into two categories, spectral-based and spatial-based methods. Spectral-based approaches define graph convolutions by introducing filters from the perspective of graph signal processing where the graph convolution operation is interpreted as removing noise from graph signals. Spatial-based approaches formulate graph convolutions as aggregating feature information from neighbors.

Bruna et al. [34] first developed spectral network, which performed convolution operation for graph data from spectral domain by computing the eigen decomposition of the graph Laplacian matrix L. Several following-up studies [8,35] make the graph convolution more promising by reducing the computational complexity from $O(n^2)$ to linear. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, let $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ denote the signals on graph $\mathcal{G}$. According to 1st-ChebNet [8], the graph convolution operation can be well-approximated by 1st order Chebyshev polynomial expansion and generalized to high-dimensional GCN as:

$$\boldsymbol{Z} = \widetilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{\theta}, \tag{2}$$

where $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ denotes the normalized adjacency matrix with self-loops, $\boldsymbol{Z} \in \mathbb{R}^{N \times D}$ denotes the output, and $\boldsymbol{\theta} \in \mathbb{R}^{F \times D}$ denotes the model parameter matrix. However, the spectral-based methods are only applicable to undirected graphs.

Spatial methods define convolutions directly on the graph through the aggregation process that operates on the central node and its neighbors to obtain a new representation of the central node. In [20], traffic network was firstly modeled as a directed graph. The dynamics of the traffic was captured based on the diffusion process and a diffusion convolution operation is applied to model the spatial correlation, which is a more intuitive interpretation and proves to be effective in spatial–temporal modeling. Specifically, Li et al. [20] modeled the diffusion process of graph signals with a finite K-step truncation as follows,

$$\boldsymbol{Z} = \boldsymbol{X} \star_{\mathcal{G}} \boldsymbol{\theta} = \sum_{k=0}^{K} \boldsymbol{P}^k \boldsymbol{X} \boldsymbol{\theta}_k, \tag{3}$$

where $\star_{\mathcal{G}}$ denotes the diffusion convolution, $\boldsymbol{P}$ denotes the transition matrix and k denotes the diffusion step. The diffusion convolution can be defined on both directed and undirected graphs. In the case of an undirected graph, $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal matrix with its i_{th} element of the diagonal line denoting the degree of the i_{th} node $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. In the case of a directed graph, diffusion convolution models the bidirectional diffusion process, enabling the model to capture the influence of upstream and downstream traffic, where the forward transition matrix $\boldsymbol{P}_f = \boldsymbol{D}_O^{-1}\boldsymbol{A}$ and the backward transition matrix $\boldsymbol{P}_b = \boldsymbol{D}_I^{-1}\boldsymbol{A}^T$. With the forward and the backward transition matrices, the diffusion graph convolution can be defined as:

$$\boldsymbol{Z} = \boldsymbol{X} \star_{\mathcal{G}} \boldsymbol{\theta} = \sum_{k=0}^{K} \boldsymbol{P}_f^k \boldsymbol{X} \boldsymbol{\theta}_{k1} + \boldsymbol{P}_b^k \boldsymbol{X} \boldsymbol{\theta}_{k2}, \tag{4}$$

4. Methodology

This section first presents the framework of the proposed model and then introduces each component step by step.

4.1. Overview of the proposed model

The framework of the proposed Ada-STNet is shown in Fig. 2, which consists of two components: adaptive graph structure learning and multi-step traffic condition forecasting. The first component is designed to learn graph structure automatically, which considers the historical information of all the nodes in the graph and provides an optimal graph adjacency matrix in a data-driven manner. Specifically, this component infers the graph structure in both macro and micro levels. From the macro perspective, the component assumes that the graph structure is relatively stable in the long term and assesses the whole historical information to adjust the pre-defined graph adjacency matrix, achieving the macro-level adjacency matrix. From the micro perspective, the component aims to discover the drastic changes in the short term and generate the micro-level adjacency matrix to capture such fluctuations. The learned macro/micro level matrices are then fused into the optimal graph adjacency matrix to facilitate the traffic forecasting task. The second component aims to predict multi-step traffic conditions, which stacks multiple spatio-temporal layers and makes predictions based on

the learned graph adjacency matrix to improve the performance. In particular, each spatio-temporal layer consists of a graph convolution module and a temporal convolution module to capture the complex spatial relationships and temporal dependencies, respectively.

Mathematically, the learning process of Ada-STNet could be formalized as follows,

$$\boldsymbol{A}^* = g(\mathcal{X}, \boldsymbol{A}), \tag{5}$$

$$\hat{\mathcal{Y}} = h(\mathcal{X}, \boldsymbol{A}^*), \tag{6}$$

where $g(\mathcal{X}, \boldsymbol{A})$ is designed to learn the optimal graph structure adaptively that takes node attributes $\mathcal{X}$ and original adjacency matrix $\boldsymbol{A}$ as inputs and provides $\boldsymbol{A}^*$ as the learned graph structure. $h(\mathcal{X}, \boldsymbol{A}^*)$ achieves multi-step traffic condition forecasting based on $\boldsymbol{A}^*$ and provides $\hat{\mathcal{Y}} \in \mathbb{R}^{T\times N\times F}$ as the final prediction.

4.2. Adaptive graph structure learning

To cope with the issues of the existing graph construction methods, we propose an adaptive graph structure learning component to obtain the optimal graph adjacency matrix in a data-driven manner. The assumption of this component is that the graph structure is relatively stable in the long term and the drastic fluctuations caused by external factors (e.g., traffic accidents, weather condition) tend to have a temporary effect in the short term [15]. Therefore, we design two modules, namely macro-level graph structure learning and micro-level graph structure learning, to adaptively infer the graph structure from both macro (long-term) perspective and micro (short-term) perspective.

4.2.1. Macro-level graph structure learning

From the macro perspective, the spatial relationships of nodes in the graph are relatively stable and reflect their intrinsic correlations. However, a pre-defined adjacency matrix $\boldsymbol{A}$ could only reflect one-sided property, which is determined by the rule, such as the Euclidean relationship based on geographic distance rule [20] and the similarity function rule [22,23]. Therefore, the pre-defined adjacency matrix $\boldsymbol{A}$ is insufficient to contain all the valuable information, since some implicit factors that related to the traffic condition (e.g., POI distribution, regional function) are ignored by the pre-defined rule.

To this end, we design a macro-level graph structure learning module to describe more comprehensive information in the long term. Instead of directly design a learnable without any prior information [18,24], we aim to learn the implicit factors that is struggle to be captured by the pre-defined rule and then inject the learned hidden relationships into the pre-defined adjacency matrix $\boldsymbol{A}$ to achieve the supplementation in information. Specifically, the macro-level adjacency matrix $\boldsymbol{A}_{Ma}$ we aim to obtain is defined as,

$$\boldsymbol{A}_{Ma} = \boldsymbol{A} + \triangle\boldsymbol{A}, \tag{7}$$

where $\triangle\boldsymbol{A}$ represents the trainable hidden relationships of nodes in the graph. $\boldsymbol{A}_{Ma}$ is generated by adding a residual connection [36] to the pre-defined adjacency matrix $\boldsymbol{A}$, where the residual connection makes the optimization easier via learning the residual mapping, rather than the original and unreferenced mapping.

There are several approaches to implement $\triangle\boldsymbol{A}$:

- **Direct Optimization**. This method designs an learnable graph adjacency matrix $\boldsymbol{E} \in \mathbb{R}^{N\times N}$ to directly reflect the implicit factors related to spatial relationships of nodes, that is,

$$\triangle\boldsymbol{A} = \boldsymbol{E}. \tag{8}$$

- **Decomposition-based Optimization**. This method leverages two trainable node embedding dictionaries $\boldsymbol{E}_1, \boldsymbol{E}_2 \in \mathbb{R}^{N\times k}$ to approximate the implicit spatial relationships as follows,

$$\triangle\boldsymbol{A} = SoftMax(ReLU(\boldsymbol{E}_1\boldsymbol{E}_2{}^\top)), \tag{9}$$

where k is the hidden dimension which is used to reduce the number of parameters.

Since the learning of residual mapping between the pre-defined adjacency matrix $\boldsymbol{A}$ and the macro-level adjacency matrix $\boldsymbol{A}_{Ma}$ makes the optimization easier, we adopt the direct optimization approach to learn $\triangle\boldsymbol{A}$ by employing considerable parameters.

4.2.2. Micro-level graph structure learning

From the micro perspective, the spatial relationships of nodes in the graph would change drastically due to traffic events, both regular and unexpected, such as rush hour, weather conditions or traffic accidents. Since nodes are associated with attributes that record the traffic condition, we suppose to describe the sudden fluctuations via mining the related information in node attributes.

In particular, given node attributes $\mathcal{X} \in \mathbb{R}^{S\times N\times F}$, we first apply a fully connected network to transform the original attributes into a latent space via expanding the attribute dimension from F to D by

$$\mathcal{H} = FC(\mathcal{X}) \in \mathbb{R}^{S\times N\times D}, \tag{10}$$

where $\mathcal{H}$ denotes the transformed node attributes. Then, to capture the temporary spatial relationships of nodes during the S periods, we aggregate the transformed node attributes $\mathcal{H}$ along the temporal dimension by a dedicated aggregator as follows,

$$\boldsymbol{M} = AGGREGATE(\mathcal{H}) \in \mathbb{R}^{N\times D'}. \tag{11}$$

$\boldsymbol{M}$ contains the node's information related to the temporary factors, which may influence the node spatial relationships. In practice, we implement the $AGGREGATE(\cdot)$ function as a convolution operation by

$$\boldsymbol{M}_{i,d'} = \boldsymbol{b}_{d'} + \sum_{d=1}^{D} \boldsymbol{W}_{d',d} * \boldsymbol{\mathcal{H}}_{:,i,d}, \ \text{for } 1 \le d' \le D', \tag{12}$$

where $*$ is the valid cross-correlation operator. $\boldsymbol{\mathcal{H}}_{:,i,d}$ stands for the input signal's dth channel, that is, the temporal information of the dth entry of node i's attributes. $\boldsymbol{M}_{i,d'}$ is the output signal's d'-th channel. $\boldsymbol{W}_{d',d}$ is the trainable parameters which describes the correlations between the d'-th output channel and dth input channel. The convolution operation could reduce the temporal dimension to one with kernels whose sizes are all set to be equal to the length of historical sequence, i.e., S. Therefore, the overall parameters size of the convolution kernels is $S \times D \times D'$.

After the aggregation across the temporal dimension, we design a metric learning approach to derive relationships between nodes via learning a metric function $\phi(\cdot,\cdot)$ of a pairwise node representations as follows,

$$\boldsymbol{A}_{Mi}[i,j] = \phi(\boldsymbol{M}_i, \boldsymbol{M}_j) = \boldsymbol{M}_i \cdot \boldsymbol{M}_j{}^\top, \ \text{for } 1 \le i,j \le N, \tag{13}$$

where $\boldsymbol{A}_{Mi}[i,j]$ denotes the learned relationships between node i and node j. Since we adopt the dot product to represent node proximity, the metric learning function could be easily expressed by the matrix multiplication format by

$$\boldsymbol{A}_{Mi} = \boldsymbol{M} \cdot \boldsymbol{M}^T. \tag{14}$$

4.2.3. Multi-level graph structure fusion

Through the above two modules, we could obtain both the macro-level graph adjacency matrix $\boldsymbol{A}_{Ma}$ and micro-level graph adjacency matrix $\boldsymbol{A}_{Mi}$. To facilitate the traffic forecasting task, an optimal graph structure is required and we could derive the graph adjacency matrix by summing up the two adjacency matrices with ReLU activation function and normalization by

$$\boldsymbol{A}^* = Norm(ReLU(\boldsymbol{A}_{Ma} + \boldsymbol{A}_{Mi})). \tag{15}$$

It is worth noticing that more sophisticated design such as the attention mechanism [37] could be leveraged in the above fusion process. In this paper, we choose the simple summation operation, since it could already achieve satisfactory performance in the experiments.

4.3. Multi-step traffic condition forecasting

Considering the complicated spatial relationships and temporal dependencies in traffic conditions, we design a spatio-temporal convolution network to forecast future multi-step traffic conditions. As shown in the right of Fig. 2, the multi-step traffic condition forecasting component consists of three parts: an input layer, stacked spatio-temporal blocks (ST-Blocks) and an output layer.

In particular, we first expand the attribute dimension via a fully connected network for each node at each time window, that is, $\mathcal{X}^{(0)} = FC_{in}(\mathcal{X}) \in \mathbb{R}^{S\times N\times D}$, where D is the dimension of hidden representations. Then, Spatio-Temporal Blocks (ST-Blocks) are designed to capture the complex spatio-temporal correlations in traffic forecasting, where each ST-Block consists of stacked temporal convolution layers and graph convolution layers. Residual connections and skip connections are added to each spatio-temporal layer to avoid model degradation. Finally, the fusion of multi-layer attributes are transformed into the multi-step forecasting via the output fully connected network. Details of each module are illustrated as follows.

4.3.1. Gated TCNs for temporal dependency learning

To extract the temporal dependencies of traffic condition, the model should not only consider the influence of adjacent data in the short-term, but also consider the long-term influence. However, previous RNN-based models are hard to handle long-range sequences due to the exploding or vanishing gradient issues [38]. Thus, we employ the dilated causal convolution [39] in our temporal convolution layer (TCN) to capture temporal dependencies of traffic condition. By adding the dilation factor which controls the skipping distance to the standard causal convolution, the receptive field of dilated causal convolution could expand exponentially with the increase of the layer depth. Such a factor enables TCN to capture longer sequences in fewer layers, saving computational resources. Meanwhile, the non-recursive manner facilitates parallel computation, which decreases the time consumption.

Formally, we perform the dilated causal convolution operation on the time dimension of traffic data to aggregate node historical attributes. Given the input of temporal convolutional layer $\mathcal{X}^{(l-1)} \in \mathbb{R}^{T_{l-1}\times N\times D}$, where l is the ST-Layer number, the convolutional kernel $\boldsymbol{\theta}^l_\tau \in \mathbb{R}^{k\times D\times 2D}$ will map the input to an output element $[\boldsymbol{g}^l, \boldsymbol{f}^l] \in \mathbb{R}^{T_l\times N\times 2D}$, where $\boldsymbol{g}^l$ and $\boldsymbol{f}^l$ are split into half with the same size of channels with

$$T_l = \lfloor \frac{T_{l-1} - d(k-1) - 1}{s} + 1 \rfloor, \tag{16}$$

$$\boldsymbol{g}^l, \boldsymbol{f}^l = \mathcal{X}^{(l-1)} *_\tau \boldsymbol{\theta}^l_\tau, \tag{17}$$

where $*_\tau$ is the dilated casual convolution. We set kernel size $k = 2$, stride $s = 1$ and dilation factor d increase exponentially with a base 2 from 1 inside a ST-Block.

Further, we utilize a gating mechanism, which has been proved powerful in controlling information flow through layers for TCNs [40]. Specifically, we feed the outputs $\boldsymbol{g}^l, \boldsymbol{f}^l \in \mathbb{R}^{T_l\times N\times D}$ through two different activation functions, and then make element-wise multiplication. Mathematically, the gated temporal convolution (Gated TCN) is defined as

$$\mathcal{Z}^{(l)} = \sigma(\boldsymbol{g}^l) \odot \tanh(\boldsymbol{f}^l), \tag{18}$$

where σ denotes the sigmoid function and $\odot$ is the Hadamard product. $\mathcal{Z}^{(l)} \in \mathbb{R}^{T_l\times N\times D}$ is the output of the lth gated TCN layer. $\sigma(\boldsymbol{g}^l)$ serves like a gated mechanism to eliminate irrelevant and select more essential information in $\tanh(\boldsymbol{f}^l)$.

4.3.2. GCNs for spatial relationship learning

Given $\mathcal{Z}^{(l)} \in \mathbb{R}^{T_l\times N\times D}$, we apply the graph convolution network to each time window $\mathcal{Z}^{(l)}_{(t)} \in \mathbb{R}^{N\times D}$ to capture the spatial relationships of nodes. Note that we could obtain an optimal graph structure $\boldsymbol{A}^*$ via the above adaptive graph structure learning component. Hence, we employ the graph convolution layer to learn on the optimal graph structure by replacing $\boldsymbol{P}$ with $\boldsymbol{A}^*$ in Eq. (4). Formally, the graph convolution layer could be defined as

$$\widetilde{\mathcal{Z}}^{(l)}_{(t)} = \mathcal{Z}^{(l)}_{(t)} \star_{\mathcal{G}^*} \boldsymbol{\theta}^{\boldsymbol{l}}_{\mathcal{G}^*}, \tag{19}$$

where $\mathcal{G}^*$ stands for the graph with optimal structure $\boldsymbol{A}^*$ and $\boldsymbol{\theta}^{\boldsymbol{l}}_{\mathcal{G}^*}$ are trainable parameters.

4.3.3. Residual and skip connections

To avoid model degradation and accelerate the convergence speed of the model, we add a residual connection within each ST-Layer. Specifically, the output of the lth ST-Layer can be obtained by

$$\mathcal{X}^{(l)} = \widetilde{\mathcal{Z}}^{(l)} + \mathcal{X}^{(l-1)}, \tag{20}$$

where $\mathcal{X}^{(0)} = FC_{in}(\mathcal{X})$.

Then, representations from different ST-Layers are fused together via skip connections as follows,

$$\mathcal{Z}' = \sum_l FC^{(l)}_{skip}(\mathcal{Z}^{(l)}), \tag{21}$$

where $FC^{(l)}_{skip}$ is a fully connected network at lth ST-Layer.

4.3.4. Multi-step prediction

Traffic forecasting tasks can be classified according to the number of steps ahead predicted by the model. For the simplest case, the model forecasts a single step ahead point of the sequence (single-step) [21], but there are models capable of predicting multiple steps ahead (multi-step) [18]. Another approach, known as multi-stage prediction, generates multiple steps ahead forecasts by using a single-step model, which recursively uses the recently predicted values as input data to inference the next step. [41]

To achieve multi-step traffic prediction, we stack several ST-Blocks and fuse multi-layer attributes to capture the spatial–temporal patterns. Then, we can directly obtain the traffic predictions for the next T steps of all nodes by applying a full connected network as follows,

$$\hat{\mathcal{Y}} = FC_{out}(\mathcal{Z}'), \tag{22}$$

where FC_{out} denotes the output fully connected network. Here, we do not generate the output in a recursive manner as it would

increase the accumulated errors and time consumption significantly.

We choose MAE (Mean Absolute Error) as our training objective and optimize the loss for multi-step prediction together. Thus, the loss function of Ada-STNet for multi-step traffic prediction can be formulated as:

$$\mathcal{L}(\mathcal{Y}, \hat{\mathcal{Y}}) = \frac{1}{T * N * D} \sum_{i=1}^{T} \sum_{j=1}^{N} \sum_{k=1}^{D} |\mathcal{Y}_{i,j,k} - \hat{\mathcal{Y}}_{i,j,k}|. \tag{23}$$

where $\mathcal{Y}_{i,:,:}$ is the ground truth, and $\hat{\mathcal{Y}}_{i,:,:}$ is the prediction of all nodes at time step i.

4.4. Two-stage training process

During the training process, all the parameters in the proposed Ada-STNet could be optimized via gradient descent. However, due to abundant learnable parameters in Ada-STNet and sparse training data, directly optimizing the model in an end-to-end manner would make the model not fully trained and lead to inferior performance (empirically validated in Section 5.6).

To cope with this issue, we perform a two-stage training process, which first optimizes the macro-level graph structure and then trains the whole model by injecting the well-trained macro-level graph adjacency matrix into Ada-STNet. Note that the learnable graph adjacency matrix $\triangle \boldsymbol{A} \in \mathcal{R}^{N \times N}$ in the macro-level graph structure learning module enhances our model capacity, but increases the training difficulty simultaneously.

Therefore, we first optimize the macro learning process to make Ada-STNet easier to train. Specifically, we combine the macro-level graph structure learning module with the multi-step traffic condition forecasting component to pre-train $\boldsymbol{A}_{Ma}$, that is,

$$\hat{\mathcal{Y}} = h(\mathcal{X}, \boldsymbol{A}_{Ma}). \tag{24}$$

The pre-training strategy provides $\boldsymbol{A}_{Ma}$ with a good initialization and makes Ada-STNet better to converge. Then, we combine the pre-trained $\boldsymbol{A}_{Ma}$ with $\boldsymbol{A}_{Mi}$ via Eq. (15) and train Ada-STNet through Eq. (5) and Eq. (6) with the loss function in Eq. (23).

Algorithm 1: Two-stage training process of Ada-STNet.

Input: Training set $\mathcal{D}_{train}$.
Pre-defined adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$.
Output: Trained Ada-STNet model.

1 Initialize all parameters of the model;
2 **repeat**
3 randomly select a batch $\{\mathcal{X}, \mathcal{Y}\}$ from $\mathcal{D}_{train}$;
4 $\boldsymbol{A}_{Ma} \leftarrow \boldsymbol{A} + \triangle \boldsymbol{A}$ using Equation (7) ;
5 $\hat{\mathcal{Y}} \leftarrow h(\mathcal{X}, \boldsymbol{A}_{Ma})$ using Equation (24);
6 compute $\mathcal{L}(\mathcal{Y}, \hat{\mathcal{Y}})$ using Equation (23);
7 $\theta_1 \leftarrow \theta_1 - \alpha_1 \nabla_{\theta_1} \mathcal{L}$;
8 $\boldsymbol{A}_{Ma} \leftarrow \boldsymbol{A}_{Ma} - \alpha_1 \nabla_{\boldsymbol{A}_{Ma}} \mathcal{L}$;
9 **until** *stopping criteria is met*;
10 **repeat**
11 randomly select a batch $\{\mathcal{X}, \mathcal{Y}\}$ from $\mathcal{D}_{train}$;
12 $\boldsymbol{A}^* \leftarrow FUSE(\boldsymbol{A}_{Ma}, \boldsymbol{A}_{Mi})$ using Equation (15);
13 $\hat{\mathcal{Y}} \leftarrow h(\mathcal{X}, \boldsymbol{A}^*)$ using Equation (6);
14 compute $\mathcal{L}(\mathcal{Y}, \hat{\mathcal{Y}})$ using Equation (23);
15 $\theta_1 \leftarrow \theta_1 - \beta_1 \nabla_{\theta_1} \mathcal{L}$;
16 $\theta_2 \leftarrow \theta_2 - \beta_2 \nabla_{\theta_2} \mathcal{L}$;
17 **until** *stopping criteria is met*;
18 Output the trained Ada-STNet model

The two-stage training process is shown in Algorithm 1. Let θ_1 denotes the trainable parameters in the multi-step traffic condition forecasting component, and θ_2 denotes the trainable parameters in adaptive graph structure learning component. Their gradients are $\nabla_{\theta_1} \mathcal{L}$ and $\nabla_{\theta_2} \mathcal{L}$, respectively. Let α_1 denotes the learning rate for the first stage, and β_1, β_2 denote the learning rates for the second stage. As Algorithm 1 outlines, the training set and pre-defined adjacency matrix are taken as inputs. Then we iteratively optimize the macro-level graph structure until convergence (lines 2–9). Finally, we update the whole model by gradient descent (lines 10–17) until the stopping criteria is met.

5. Experiments

This section evaluates the effectiveness of the proposed model by experiments on real-world datasets. Both classical and state-of-the-art methods are compared to provide baseline performance, and multiple metrics are used to provide comprehensive evaluation.

5.1. Datasets

We conduct experiments on two real-world large-scale datasets, METR-LA and PEMS-BAY [20].

- METR-LA: METR-LA records four months of statistics on traffic speed, ranging from Mar 1st 2012 to Jun 30th 2012, including 207 sensors on the highways of Los Angeles County.
- PEMS-BAY: PEMS-BAY contains six months of statistics on traffic speed, ranging from Jan 1st 2017 to June 30th 2017, including 325 sensors in the Bay area.

We adopt the same data pre-processing procedures as Li et al. [20]. The observations of the sensors are aggregated into 5-minute windows. The pre-defined adjacency matrix is constructed by road network distance with thresholded Gaussian kernel [42]. Z-score normalization is applied to the inputs. Both the datasets are split in chronological order with 70% for training, 10% for validation, and 20% for testing. Detailed statistics of the datasets are shown in Table 2.

Table 2
The statistics of METR-LA and PEMS-BAY.

Dataset	#Nodes	#Edges	#Time windows
METR-LA	207	1515	34272
PEMS-BAY	325	2369	52116

5.2. Baselines

We compare Ada-STNet with the following baselines, including both classical and the state-of-the-art methods:

- HA: Historical Average, which models the traffic flow as a seasonal process, and uses the average previous seasons as the prediction.
- ARIMA$_{kal}$: Auto-Regressive Integrated Moving Average Model with Kalman filter, which is a classical time series prediction model.
- FC-LSTM: Recurrent neural network with fully connected LSTM hidden units [43].
- DCRNN: Diffusion Convolutional Recurrent Neural Network [20], which combines recurrent neural networks with diffusion convolution modeling both inflow and outflow relationships.
- ST-GCN: Spatial-Temporal Graph Convolution Network [21], which combines 1D convolution with graph convolution.

Table 3
Performance comparison of multi-step traffic condition forecasting.

Dataset	Model Name	15 min			30 min			60 min		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	HA	4.16	7.80	13.0%	4.16	7.80	13.0%	4.16	7.80	13.0%
	$ARIMA_{kal}$	3.99	8.12	9.6%	5.15	10.45	12.7%	6.90	13.23	17.4%
	FC-LSTM	3.44	6.30	9.6%	3.77	7.23	10.9%	4.37	8.69	13.2%
	ST-GCN	2.88	5.74	7.6%	3.47	7.24	9.6%	4.59	9.40	12.7%
	DCRNN	2.77	5.38	7.3%	3.15	6.45	8.8%	3.60	7.60	10.5%
	Graph WaveNet	2.69	5.15	6.9%	3.07	6.22	8.4%	3.53	7.37	10.0%
	Ada-STNet	**2.65**	**5.06**	**6.8%**	**3.03**	**6.08**	**8.2%**	**3.47**	**7.18**	**9.8%**
PEMS-BAY	HA	2.88	5.59	6.8%	2.88	5.59	6.8%	2.88	5.59	6.8%
	$ARIMA_{kal}$	1.62	3.30	3.5%	2.33	4.76	5.4%	3.38	6.50	8.3%
	FC-LSTM	2.05	4.19	4.8%	2.20	4.55	5.2%	2.37	4.96	5.7%
	ST-GCN	1.36	2.96	2.9%	1.81	4.27	4.2%	2.49	5.69	5.8%
	DCRNN	1.38	2.95	2.9%	1.74	3.97	3.9%	2.07	4.74	4.9%
	Graph WaveNet	**1.30**	2.74	**2.7%**	1.63	3.70	3.7%	1.95	4.52	4.6%
	Ada-STNet	**1.30**	**2.73**	**2.7%**	**1.62**	**3.67**	**3.6%**	**1.89**	**4.36**	**4.5%**

- Graph WaveNet: A convolution network architecture [18], which introduces a self-adaptive graph to capture the hidden spatial dependency, and uses dilated convolution to capture the temporal dependency.

For all neural network based approaches, the best hyperparameters are chosen using grid search based on the performance on the validation set.

5.3. Evaluation metrics

Three common metrics of traffic forecasting are adopted to measure the performance of different models, including (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Squared Error (RMSE).

Suppose $\boldsymbol{X} = x_1, \ldots, x_n$ represents the ground truth, $\hat{\boldsymbol{X}} = \hat{x}_1, \ldots, \hat{x}_n$ represents the predicted values, and Ω denotes the indices of observed samples, the metrics are defined as follows.

(1) Mean Absolute Error (MAE):

$$MAE(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} |x_i - \hat{x}_i| \tag{25}$$

(2) Mean Absolute Percentage Error (MAPE):

$$MAPE(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} |\frac{x_i - \hat{x}_i}{x_i}| \times 100\% \tag{26}$$

(3) Root Mean Squared Error (RMSE):

$$RMSE(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \sqrt{\frac{1}{|\Omega|} \sum_{i \in \Omega} (x_i - \hat{x}_i)^2} \tag{27}$$

5.4. Experimental settings

All experiments use the traffic speed in the last hour to forecast the traffic speed over one hour in the future, i.e., $S = T = 12$. To cover the input sequence length, the number of ST-Blocks is set to 4. We use Eq. (4) as our graph convolution layer with a diffusion step $K = 1$. Each ST-Block contains two ST-Layers with a sequence of dilation factors 1,2. Dropout with $p = 0.3$ is applied to the outputs of graph convolution layer. The number of filters used by all layers in the Gated-TCN and the GCN is set to 40 on METR-LA dataset and 36 on PEMS-BAY dataset. For the adaptive graph structure learning component, the attribute dimension of the node representations $\boldsymbol{M}$ is set to 6. And dropout with $p = 0.5$ is applied to $\boldsymbol{M}$. In the first stage of training, we train the multi-step traffic condition forecasting component and macro-level graph structure $\boldsymbol{A}_{Ma}$ using Adam optimizer with an initial learning rate of 0.001. In the second stage of training, we use Adam optimizer with an initial learning rate of 0.001 to train the adaptive graph structure learning component and fine-tune the multi-step traffic condition forecasting component with a learning rate of 0.00001. The batch size in our experiments is set at 64. The codes and datasets are publicly available at https://github.com/LiuZH-19/Ada-STNet.

5.5. Experimental results and analysis

Table 3 shows the performance comparison of Ada-STNet and baseline models for 15 min, 30 min and 1 h ahead forecasting on both datasets. Ada-STNet achieves superior results regarding all the metrics for all forecasting horizons. We can observe further phenomena from the table.

First of all, the deep learning models starting from FC-LSTM outperform traditional statistical models such as HA, $ARIMA_{kal}$ by a large margin. DCRNN and STGCN use graph convolution neural network to consider the spatial relationships of traffic data, raising the accuracy of prediction. Since STGCN is a single-step forecasting model, the results of multi-step forecasting obtained by recursion accumulate large errors. While DCRNN uses Seq2Seq structure to extract the time trend from historical series for multi-step forecasting, its effect is better than that of STGCN. However, our proposed Ada-STNet model still excels these two GCN-based approaches significantly, because the previous methods perform graph convolution operation on the fixed adjacency matrix determined by spatial distance which does not necessarily reflect the real spatial relationships.

Graph WaveNet and Ada-STNet both combine graph convolution and causal convolution, and can simultaneously output the predicted results of multiple time steps. They not only take advantage of the temporal dependencices of traffic conditions, but also consider the spatial characteristics of road networks hidden in the data. However, Graph WaveNet directly designs learnable parameters to generate the graph structure without using the node attributes and simply takes the learned graph structure as the new fixed Laplacian matrix for forecasting. By contrast, Ada-STNet takes the dynamics and self-adaptability of the graph structure into sufficient consideration. It adaptively infers the graph structure from both macro perspective (stable in the long-term) and micro perspective (fluctuate in the short-term). On METR-LA dataset, the improvement of our Ada-STNet over the state-of-the-art Graph WaveNet reaches almost half of the improvement of Graph WaveNet over the second-best baseline (DCRNN), which is significant and well demonstrates our model superiority. On PEMS-BAY dataset, our model still achieves competitive or improved performance than Graph WaveNet. Moreover, on both datasets, the improvement of Ada-TNet increases with the growth

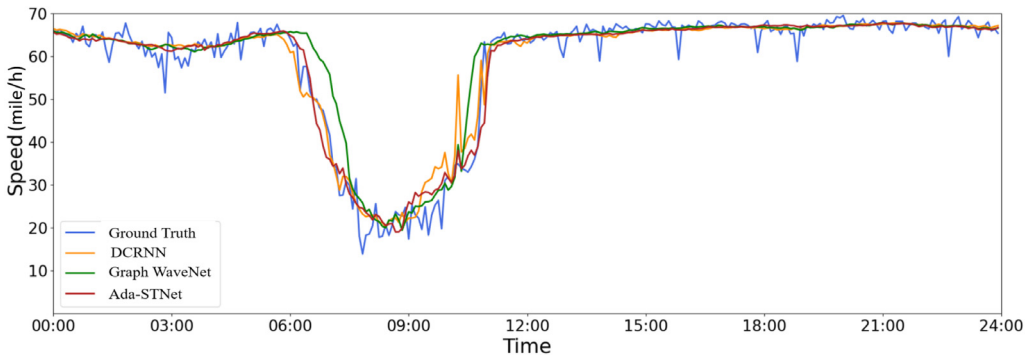


Fig. 3. Comparison of prediction curves for one hour ahead prediction on a snapshot of the test data of METR-LA.

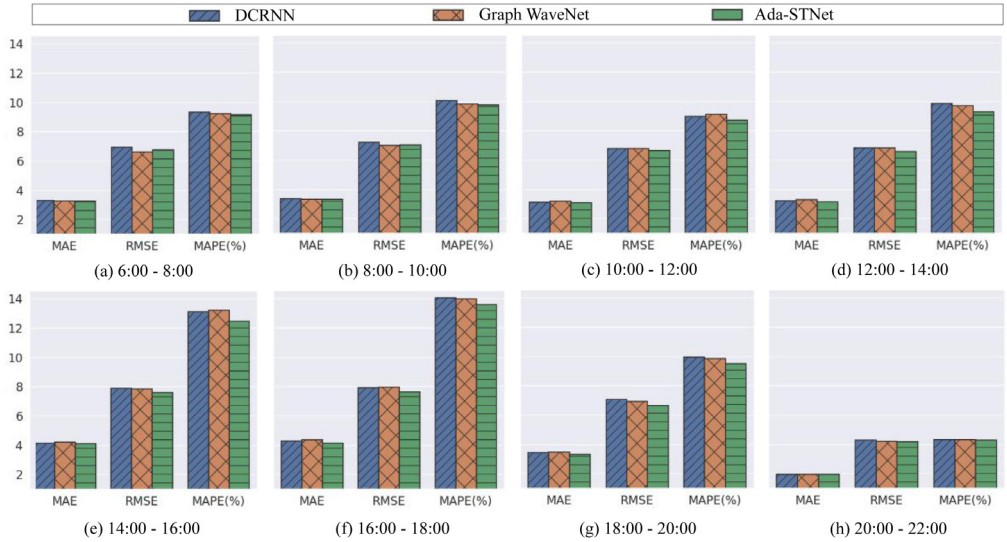


Fig. 4. Performance comparison of different approaches during different time periods on the METR-LA dataset.

of the forecasting horizon, indicating our model ability in long-term forecasting, which is inherently more uncertain and difficult than short-term forecasting. In addition, it is worth noticing that Ada-STNet achieves a large improvement on RMSE, indicating that Ada-STNet predicts fewer deviations since RMSE is susceptible to outliers.

To further compare the performance of Ada-STNet with other classical models (DCRNN and Graph WaveNet), We plot one-hour-ahead predicted values v.s ground truth of three models on a snapshot of the test data in Fig. 3. When the speed fluctuates drastically (eg. from 6:00 am to 8:00 am), Graph WaveNet cannot accurately capture the abrupt change, and there is a large delay between its prediction curve and ground truth curve. Compared with Graph WaveNet, Ada-STNet and DCRNN can predict the start and the end of peak hours more accurately. However, there is an orange sharp spike produced by DCRNN (at about 10:30 am), which deviates far from ground truth. On the contrary, the curve of Ada-STNet goes smoothly and fits the curve of ground truth better. This reflects the robustness of the model.

Since traffic conditions change over time, we evaluate the experiment results during different time periods to further investigate the practicability of our method. As Fig. 4 shows, Ada-STNet achieves the best results in almost any time period. Especially in the time periods when traffic speed changes dramatically, such as 10:00–12:00, 12:00–14:00, 14:00–16:00 and 16:00–18:00, the performance of Ada-STNet is enhanced significantly than other models. It demonstrates again that Ada-STNet has a stronger ability to model complex and changeable traffic conditions than other methods.

5.6. Ablation study

We conduct an ablation study on the METR-LA dataset to validate the effectiveness of proposed key components as shown in Fig. 5.

Pre-defined model just uses the adjacency matrix constructed by Euclidean distance. Mi-only model utilizes the summation of pre-defined adjacency matrix $\boldsymbol{A}$ and micro-level adjacency matrix $\boldsymbol{A}_{Mi}$ to represent the graph structure. Ma-only model assumes the adjacency matrix is a parameter matrix, which can be obtained from macro-level graph structure learning module. Ada-STNet(one-stage) and Ada-STNet(two-stage) both fuse the macro-level and micro-level graph structure, but Ada-STNet(one-stage) trains all the parameters in only one stage, and Ada-STNet(two-stage) trains the model as described in Section 4.4.

According to Fig. 5, Ada-STNet(two-stage) achieves the lowest MAE, RMSE and MAPE regarding all forecasting horizons. Pre-defined model yields large testing errors, as it does not reflect the real spatial relationships. Although the results of Ma-only and Mi-only have been improved, the spatial relationships they learned are still one-sided. Fusing macro-level graph structure and micro-level graph structure improves the performance significantly. It indicates that the macro-level and micro-level graph structure learning modules can introduce new and useful information into the model respectively. Last but not least, the performance of Ada-STNet(one-stage) is even worse than that of pre-defined model as there are too many parameters to train in the model, so the two-stage training method is adopted.

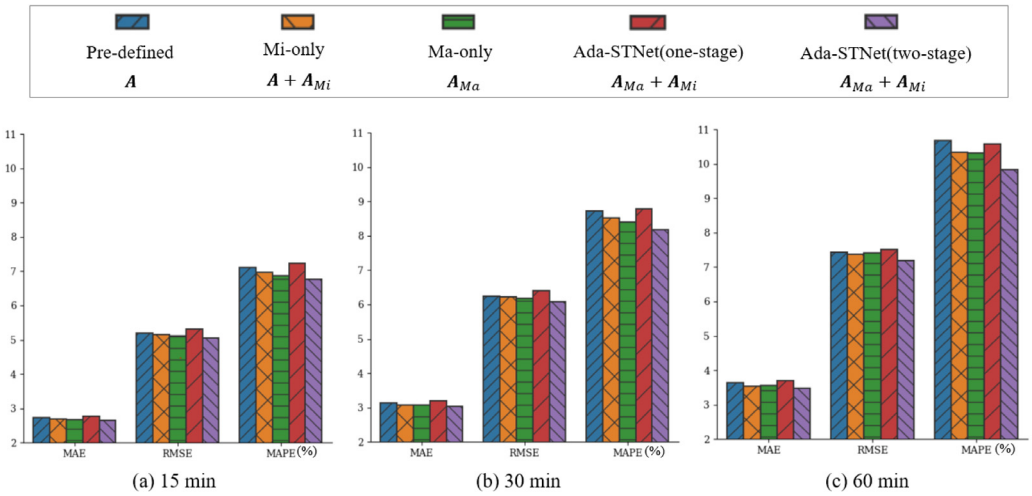


Fig. 5. Experimental results of different adjacency matrix configurations on the METR-LA dataset.

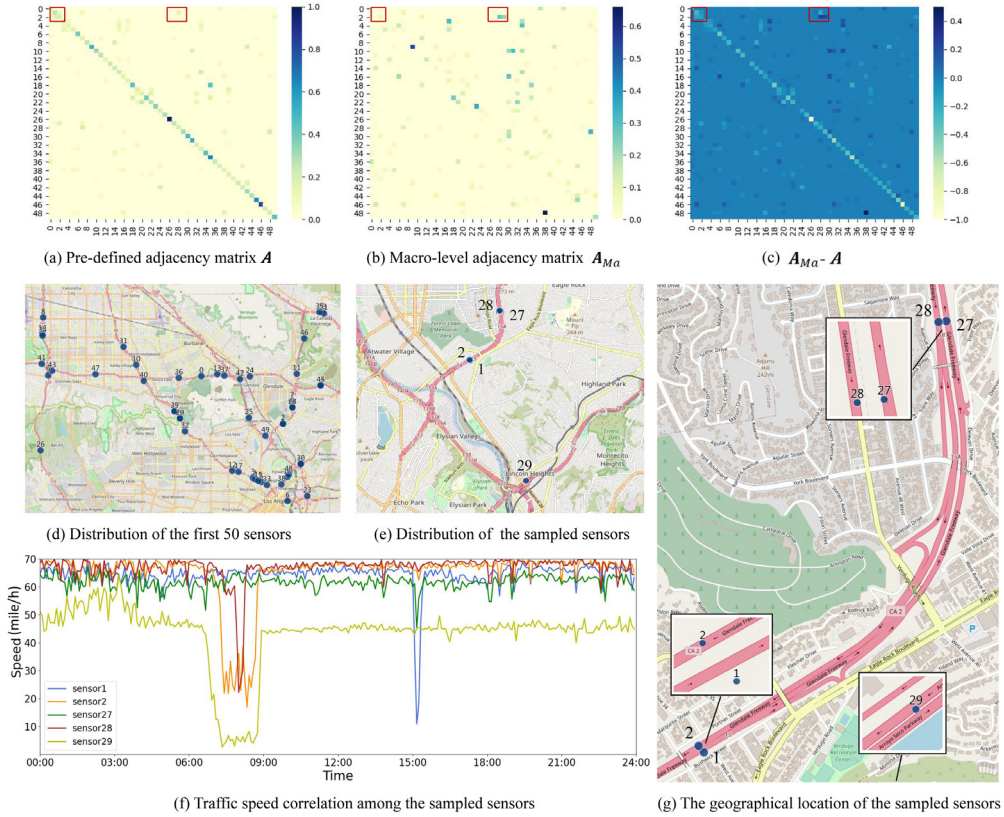


Fig. 6. Case study for analyzing the effect of adaptive graph structure learning.

5.7. Effect of adaptive graph structure learning

To give an intuitive interpretation for the effect of our proposed adaptive graph structure learning component, we conduct a case study as Fig. 6 shows. We select the first 50 sensors in the METR-LA dataset as the research object, and their distribution is visualized in Fig. 6(*d*). Fig. 6(a)(b)(c) show the heatmaps of the pre-defined adjacency matrix $\boldsymbol{A}$, the macro-level adjacency matrix $\boldsymbol{A}_{Ma}$ and the difference between the two ($\boldsymbol{A}_{Ma} - \boldsymbol{A}$). Pre-defined adjacency matrix $\boldsymbol{A}$ is defined by the geographic distance and hence the values on the diagonal are generally high. By contrasting the three heatmaps, we find that the macro-level adjacency matrix $\boldsymbol{A}_{Ma}$ has made many adjustments on the basis of the pre-defined adjacency matrix $\boldsymbol{A}$.

For instance, there is a strong inter-dependence between sensor 1 and sensor 2 on the heatmap of pre-defined adjacency matrix $\boldsymbol{A}$. However, as can be seen from Fig. 6(g), sensor 1 and sensor 2 are located in opposite directions of the highway, although they are close to each other in the Euclidean space. Their curves of traffic speed differ significantly as shown in the blue and orange lines of Fig. 6(f). The macro-level adjacency matrix $\boldsymbol{A}_{Ma}$ weakens the relationship between sensor 1 and sensor 2, and strengthens the effect of sensor 27 on sensor 1, as well as sensor 28, 29 on sensor 2. Fig. 6(e) and (g) show the global and local geographical distribution of these sensors respectively. Although

sensor 27 and sensor 1 are far apart, they locate in the same direction of the same highway and sensor 27 is the downstream of sensor 1. So does the location relationship between sensor 28 and sensor 2. In real life, the downstream traffic conditions on the same road do have a strong impact on the upstream traffic conditions. Fig. 6(f) confirms that there are strong correlations between the speed curves of sensor 27 and sensor 1, sensor28 and sensor 2. Sensor 2 and sensor 29 are located in different segments, but they play the same structural role near the intersection. Although the speed curves of the two are at separate heights, they have the same trend.

No matter from the perspective of geographical position relationships or traffic speed curves, the above cases can prove that the pre-defined adjacency matrix $\boldsymbol{A}$ cannot reveal the true mode of traffic information transmission. The learned macro-level graph structure $\boldsymbol{A}_{Ma}$ which can automatically mine the hidden relationships among nodes from the traffic time series data, has better interpretability and reflects the real spatial dependencies among nodes to a certain extent.

6. Conclusion

In this paper, we proposed an Adaptive Spatio-Temporal graph neural Network (Ada-STNet) to solve the problem of traffic forecasting. To cope with the issues of existing methods in learning on the imprecise graph structure, Ada-STNet leverages node attributes to guide the learning of optimal graph structure. In particular, Ada-STNet consists of two components: (1) an adaptive graph structure learning component to derive the optimal graph adjacency matrix from both macro and micro perspectives; and (2) a multi-step traffic condition forecasting component to predict future traffic conditions considering complicated spatio-temporal correlations. Moreover, we presented a two-step training strategy to further improve the performance. We conducted extensive experiments to validate the effectiveness of the proposed model and experimental results demonstrated the superiority of our approach over existing methods. Accurate traffic forecasting is essential for urban transportation and our approach could be applied to help the decision and management in intelligent traffic systems.

CRediT authorship contribution statement

Xuxiang Ta: Conceptualization, Methodology, Writing – original draft. **Zihan Liu:** Methodology, Writing – original draft, Experimental studies, Visualization. **Xiao Hu:** Investigation, Conceptualization, Data acquisition. **Le Yu:** Supervision, Formal analysis, Manuscript revision. **Leilei Sun:** Supervision, Funding acquisition, Resources. **Bowen Du:** Supervision, Funding acquisition, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 51991391, 51991395, 71901011, U1811463) and the Guangxi Innovation-Driven Development Special Fund Project (Grant No. AA18118053).

References

[1] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: A deep learning approach, IEEE Trans. Intell. Transp. Syst. (2014) 1–9, http://dx.doi.org/10.1109/TITS.2014.2345663.

[2] I. Lana, J. Del Ser, M. Velez, E.I. Vlahogianni, Road traffic forecasting: Recent advances and new challenges, IEEE Intell. Transp. Syst. Mag. 10 (2018) 93–109.

[3] H. Yu, Z. Wu, S. Wang, Y. Wang, X. Ma, Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks, Sensors 17 (2017) 1501, http://dx.doi.org/10.3390/s17071501.

[4] R. Jiang, X. Song, Z. Fan, T. Xia, Q. Chen, S. Miyazawa, R. Shibasaki, Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February (2018) 2-7, AAAI Press, 2018, pp. 784–791.

[5] Z. Cui, R. Ke, Y. Wang, Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction, 2018, arXiv:1801.02143 [cs].

[6] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, X. Zhou, Lc-rnn: A deep learning model for traffic speed prediction, in: IJCAI, 2018, pp. 3470–3476.

[7] J. Ye, L. Sun, B. Du, Y. Fu, X. Tong, H. Xiong, Co-prediction of multiple transportation demands based on deep spatio-temporal neural network, in: Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2019, pp. 305–313, http://dx.doi.org/10.1145/3292500.3330887.

[8] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017, arXiv:1609.02907 [cs, stat].

[9] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 1024–1034.

[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[11] C. Chen, K. Li, S.G. Teo, X. Zou, K. Wang, J. Wang, Z. Zeng, Gated residual recurrent graph neural networks for traffic prediction, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 485–492.

[12] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, J. Zhang, Urban traffic prediction from spatio-temporal data using deep meta learning, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1720–1730.

[13] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, 2019, arXiv:1911.12093.

[14] B. Yu, H. Yin, Z. Zhu, ST-UNet: A Spatio-temporal u-network for graph-structured time series modeling, 2019, arXiv:1903.05631 [cs, stat].

[15] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, S. He, Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting, Proc. AAAI Conf. Artif. Intell. 33 (2019) 890–897, http://dx.doi.org/10.1609/aaai.v33i01.3301890.

[16] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial–temporal graph convolutional networks for traffic flow forecasting, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 922–929.

[17] L. Bai, L. Yao, S.S. Kanhere, X. Wang, Q.Z. Sheng, STG2Seq: Spatial-Temporal graph to sequence model for multi-step passenger demand forecasting, 2019, arXiv:1905.10069 [cs, stat].

[18] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph WaveNet for deep spatial-temporal graph modeling, 2019, arXiv:1906.00121 [cs, stat].

[19] S. Fang, Q. Zhang, G. Meng, S. Xiang, C. Pan, GSTNet: Global Spatial-temporal network for traffic flow prediction, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Macao, China, 2019, pp. 2286–2293, http://dx.doi.org/10.24963/ijcai.2019/317.

[20] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, p. 16, 2018.

[21] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640, arXiv:1709.04875.

[22] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 3656–3663.

[23] Y. Wang, H. Yin, H. Chen, T. Wo, J. Xu, K. Zheng, Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19, ACM Press, Anchorage, AK, USA, 2019, pp. 1227–1235, http://dx.doi.org/10.1145/3292500.3330877.

[24] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[25] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, L. Wang, Deep graph structure learning for robust representations: a survey, 2021, arXiv preprint arXiv:2103.03036.

[26] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, R. Kong, Dynamic network embedding survey, 2021, arXiv preprint arXiv:2103.15447.

[27] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 3546–3553.

[28] Y. Chen, L. Wu, M. Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Adv. Neural Inf. Process. Syst. 33 (2020).

[29] X. Zhang, M. Zitnik, Gnnguard: Defending graph neural networks against adversarial attacks, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[30] L. Franceschi, M. Niepert, M. Pontil, X. He, Learning discrete structures for graph neural networks, in: International conference on machine learning, PMLR, 2019, pp. 1972–1982.

[31] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, W. Wang, Robust graph representation learning via neural sparsification, in: International Conference on Machine Learning, PMLR, 2020, pp. 11458–11468.

[32] L. Yang, Z. Kang, X. Cao, D. Jin, B. Yang, Y. Guo, Topology optimization based graph convolutional network, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 4054–4061.

[33] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, J. Tang, Graph structure learning for robust graph neural networks, in: KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 66–74.

[34] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.

[35] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016) 3844–3852.

[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[37] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[38] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.

[39] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, 2015, arXiv preprint arXiv:1511.07122.

[40] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: International conference on machine learning, PMLR, 2017, pp. 933–941.

[41] E.L. Manibardo, I. Lañ, J.D. Ser, Deep learning for road traffic forecasting: Does it make a difference? 2020, arXiv preprint arXiv:2012.02260.

[42] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, http://dx.doi.org/10.1109/MSP.2012.2235192.

[43] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13, 2014, Montreal, Quebec, Canada, 2014, pp. 3104–3112.