A Dual Branch Graph Neural Network based Spatial Interpolation Method for Traffic Data Inference in Unobserved Locations

Wujiang Zhu[a], Xinyuan Zhou[a], Shiyong Lan [a,*], Wenwu Wang[b], Zhiang Hou[c], Yao Ren[a] and Tianyi Pan[a]

[a]*College of Computer Science, Sichuan University, Chengdu, China*
[b]*Department of Electrical and Electronic Engineering, University of Surrey, Guildford, UK*
[c]*National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu, China*

ARTICLE INFO

Keywords:
Traffic data
Spatial interpolation
Spatial–temporal dependencies
Graph convolutional network

ABSTRACT

Complete traffic data collection is crucial for intelligent transportation system, but due to various factors such as cost, it is not possible to deploy sensors at every location. Using spatial interpolation, the traffic data for unobserved locations can be inferred from the data of observed locations, providing fine-grained measurements for improved traffic monitoring and control. However, existing methods are limited in modeling the dynamic spatio-temporal dependencies between traffic locations, resulting in unsatisfactory performance of spatial interpolation for unobserved locations in traffic scene. To address this issue, we propose a novel dual branch graph neural network (DBGNN) for spatial interpolation by exploiting dynamic spatio-temporal correlation among traffic nodes. The proposed DBGNN is composed of two branches: the main branch and the auxiliary branch. They are designed to capture the wide-range dynamic spatial correlation and the local detailed spatial diffusion between nodes, respectively. Finally, the two branches are fused via a self-attention mechanism. Extensive experiments on six public datasets demonstrate the advantages of our DBGNN over the state-of-the-art baselines. The codes will be available at https://github.com/SYLan2019/DBGNN.

1. Introduction

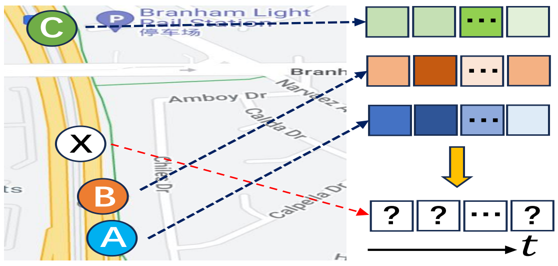
The development of Intelligent Transportation System (ITS) requires high resolution spatio-temporal data to uncover the essential patterns in traffic scenes. As a result, an increasing number of sensors are being deployed to collect traffic data such as traffic flow and speed, which facilitates traffic monitoring and management [1]. However, in practice, the distribution of sensors is often sparse and uneven in space, due to practical challenges in sensor deployment, including associated cost and geographical environment constraints. To obtain fine-grained data with higher spatial resolution, spatial interpolation based on spatio-temporal information is commonly employed to estimate the traffic data for the locations where no traffic sensors are deployed [2–6].

An illustration for spatial interpolation is given in Fig. 1(a), where the data from limited known nodes (with sensors) are used to infer the state of unknown nodes (without sensors) within a certain period of time. The key aspect of spatial interpolation lies in modeling the complex spatio-temporal relations among the nodes, which is also known as spatio-temporal kriging [3, 6].

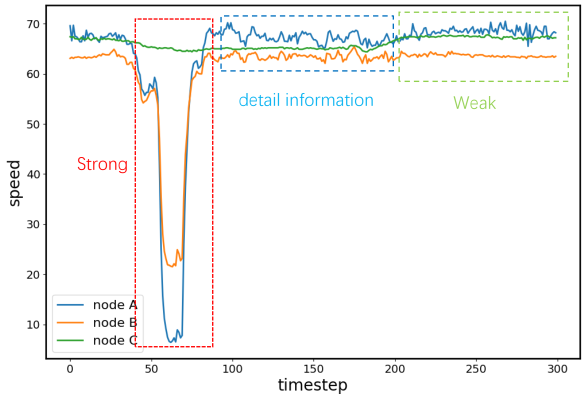
Many methods have been proposed for the spatial interpolation task. Early methods such as K-Nearest Neighbors (KNN) [7] and Inverse Distance Weighting (IDW) [8] only modeled linear spatial relationships. The traditional Kriging method (OKriging) [9] assumes data follows Gaussian distribution, while real-world data may not follow a Gaussian distribution [10]. Some works view spatial interpolation as a tensor/matrix completion task [11, 12]. However, these methods are transductive, which means they cannot handle unknown locations that were not seen during training. Recent works on spatial interpolation have been conducted from the perspective of spatio-temporal data mining [2–4, 6]. They model spatiotemporal correlations jointly by combining Graph Neural Networks (GNNs) and sequence modeling modules such as Recurrent Neural Networks (RNNs) and Temporal Convolutional Networks (TCNs), as well as attention mechanisms. Although these methods have achieved good performance, they still have two major limitations.

First, existing models lack sufficient attention to dynamic spatial correlations. As shown in Fig. 1 (b), the sequence patterns of A and B are highly similar within the red dashed box, indicating a strong spatial correlation between sensor A and B during this time period. However, their correlation is weaker within the blue box. This indicates that the spatial correlations between nodes may dynamically change over time. Unfortunately, most existing works (e.g. KNN [7], IDW [8], GLTL [11], OKriging [10], KCN [2], IGNNK [3], INCREASE [6]) are based solely on static graphs, which cannot accurately capture the dynamic spatial dependencies between nodes. In recent baseline IAGCN, the dynamic graph is created based on node embeddings and remains fixed once trained. For this reason, it cannot truly adapt to the dynamic characteristics of the graph structure during the testing phase. Furthermore, in existing spatial interpolation methods, the generation of dynamic graph edges does not consider randomness [13], as a result, they are prone to the potential risk of overfitting in model training. To address these issues, we propose an attention-based dynamic graph learning module which not only captures dynamic structural attributes of the graph during the testing phase, but also introduces randomness into graph edge generation, alleviating the potential overfitting issue in model training the graph neural

[*]Corresponding author. Email address: lanshiyong@scu.edu.cn



(a) An example of spatial interpolation



(b) The characteristics of traffic data

Figure 1: (a) In this example, the aim of spatial interpolation is to estimate the data of an unobserved location X within a certain time interval using the known data of locations A, B, and C. (b) The spatial correlation between nodes exhibits complex dynamic characteristics. Such correlation is sometimes highly related to spatial adjacency. For example, within the red dashed box, the traffic data collected at node A is more similar to that of node B, rather than that of node C. At other times, this correlation is almost unaffected by the spatial adjacency. For example, within the green dashed box, the trend of node A is more similar to that of node C, rather than that of node B.

networks and thus improving the robustness of the proposed method.

Second, existing models overlook the importance of local spatial correlations, although they contain useful information about the road network structure. Tobler's First Law [14] points out that everything is interrelated, and the closer it is, the more relevant it is. In traffic scenes, the real-time changes of a node can affect its adjacent nodes instantly, but they may have little impact on the distant nodes. For example, as shown in Figure 1 (b), traffic congestion occurs at node A on the road, but it is then quickly resolved. The speed recorded by node A will first decrease and then rebound, and this process is very rapid. The local adjacent point B can perceive this change almost in real time, while the distant node C is very likely unaffected. As mentioned above, changes in details usually affect the local space, and are originated from the transfer of traffic flow. Motivated by this observation, we propose a dual-branch structure for capturing both semantic and detailed information from the spatiotemporal features. Inspired by BiSeNetV2 [15] developed for visual segmentation, our proposed dual-branch structure uses the main and auxiliary branch to capture the wide-range dynamic spatial correlations and local spatial correlations between sensor nodes, respectively.

In summary, to address the aforementioned issues, we propose a Novel Dual Branch Graph Neural Network (DBGNN) for spatial interpolation tasks in transportation scenarios. Our DBGNN network consists of a main branch and an auxiliary branch. The main branch employs a deep network structure with multi-level skip connections to capture the wide-range spatiotemporal information, while the auxiliary branch uses a shallow network to obtain low-level detailed information. Finally, an attention mechanism is utilized to fuse the outputs from these two branches and enhance the interpolation performance. The main contributions of our work are summarized as follows:

- We design a new Dynamic Graph Learning (DGL) module that can capture dynamic spatial correlations among nodes using attention mechanisms. When the input changes, DGL can adaptively generate dynamic graph (A_d) during the training and testing stages. Furthermore, we introduce randomness into graph edge generation, alleviating the potential overfitting and oversmothing issue in training the graph neural networks and thereby improving the robustness of the proposed method.
- We introduce a novel dual branch architecture from different perspectives to model the diffusion mechanism between traffic nodes. The main branch stacks several spatiotemporal layers to represent the global dynamic spatiotemporal correlations among nodes at multiple temporal levels. The auxiliary branch is designed as a shallow network structure to focus on the local spatial correlations between nodes.
- We validate our model on seven well-known real-world datasets. The experimental results show that our model has achieved state-of-the-art performance in all six traffic datasets, and also performs well on another air quality dataset.

2. Related Works

2.1. Spatio-Temporal Graph Neural Networks

As an effective method for spatio-temporal data mining, Spatio-temporal Graph Neural Networks (STGNNs) have gained significant attention. For example, it has been used for traffic flow forecasting [16–19] or demand forecasting [20]. Early works [21–23] mainly regarded spatiotemporal sequences as time series with independent positions, and used Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) [24] and Gate Recurrent Unit (GRU) [25] to model temporal relationships. However, they often ignored the strong spatial correlations in spatiotemporal sequences. Convolutional Neural Networks (CNNs) have also been used to model spatial dependencies [26–31]. However, CNNs are not suitable for directly modeling graph-structured data. Kipf et al. [32] proposed a graph convolutional network (GCN), simpler and faster than Chebnet [33], to extract and fuse spatial features in the spatial domain. In the field of time series modeling, TCN [34] which is improved by CNN, can avoid gradient vanishing or exploding problems, and can also perform parallel computing. Thus, most STGNNs combine GNNs with either RNNs or TCNs [17, 35–37]. Recently, attention mechanisms have been shown to perform well in a wide range of tasks, and numerous researchers have applied the attention mechanism in their STGNNs [16, 38–40].

Modeling dynamic spatial correlations has become a popular paradigm in various tasks (such as prediction) using spatiotemporal data, which can be roughly divided into two categories. In the first category, attention, instead of graph convolution, is used to model long-term correlated features in spatiotemporal data sequence, such as GMAN [38], STAEformer [41], and PDFormer [18]. However, these models ignore the graph structure prior information, which play a crucial role in understanding traffic evolution due to the well-known heterogeneity of the nodes and edges in traffic scenarios. In the second category, dynamic graphs are generated to represent dynamic associations, such as GraphWavenet [35], AGCRN [42], AdaSTNet [43], MSGNet [44], and DAGN [45]. These methods use trainable node embedding to discover potential spatial correlations, but the graphs generated in this way do not change during the testing phase. As a result, they are limited in representing dynamics in traffic data. Some methods are based on data-driven approaches to generate dynamic graphs, such as STFGNN [46] and STGODE [47], which use the DTW algorithm to calculate the similarity between two sequences to measure the spatial correlation of nodes. However, their computational complexity is high. In [48], cosine similarity is used to calculate dynamic graphs, but it is not sensitive to data scale. ASTGNN [49] use attention mechanisms to adjust static graph, but this approach can only adjust the existing spatial correlation between known nodes and is ineffective for unknown nodes. STP-TrellisNets+[50] relies on the passenger transfer data between nodes to construct dynamic graphs, however, most traffic datasets do not have this type of data. In spatiotemporal prediction, research on dynamic spatial correlations is abundant, while in spatial interpolation tasks, there is less attention paid to dynamic spatial correlations.

2.2. Spatial Interpolation

Spatial interpolation can be used to infer data at unobserved locations using information from observed locations. This technique can facilitate the observation of the state of traffic scenes and management of the traffic flows. Early methods primarily focus on modelling linear spatial dependencies to estimate the unobserved nodes, such as KNN [7] and IDW [8]. The KNN finds the nearest K neighbors of each unknown location and averages them, while IDW uses the inverse distance as the summation coefficient. In addition, there are early geostatistical interpolation methods, such as the classic Kriging method [9]. However, the original Kriging method assumes that the data follows Gaussian distribution, which may not necessarily match the actual data distribution.

Previously, treating the Kriging task as matrix/tensor factorization completion is one of the classic approaches [11, 12, 51, 52]. For example, GLTL [11] sets the entries in the input tensor corresponding unobserved locations to zero, and then uses tensor completion methods to recover the values of these locations. GE-GAN [53] is another approach, which matches the observed nodes that are most relevant to each unobserved node in terms of their node embeddings, and then uses generative model [54] to generate estimates for the unobserved nodes. However, these models are transductive, that is to say, they cannot handle unknown nodes that are not seen during training. Recent studies have demonstrated the inductive ability of GNNs [55–57], which can be generalized to new graph structures. KCN [2] and IGNNK [3] using GCN for inductive kriging, achieved promising results. Especially, unlike traditional kriging that only considers spatial correlations, IGNNK has introduced temporal dependencies to improve spatial interpolation accuracy. However, these graph structures are based on local fixed distance, and thus unable to reflect the correlations between nodes that are distant from each other and cannot capture dynamic correlations between nodes over time.

Recently, DualSTN [4] uses both long-term and short-term branches for spatial interpolation. However, it is a one-step interpolation model. INCREASE [6] models and fuses three types of heterogeneous spatial correlations including spatial proximity, functional similarity, and transition probability. Whereas, using predefined spatial associations may not efficiently reflect hidden spatial dependencies, and the latter two spatial associations are only suitable for some specific datasets. To solve the challenges in directly using node embedding to construct dynamic graphs for spatial interpolation, IAGCN [58] chooses to train and update the embeddings of the known nodes, while the embeddings of the unknown nodes in the testing phase are obtained by graph convolution diffusion. However, its use of physical distance based adjacency graphs may not accurately reflect the spatial correlations between the embeddings, resulting in incorrect embeddings of the unknown nodes.

Although these methods have considered modeling the spatio-temporal dependencies for spatial interpolation, they have not yet explored the modelling of dynamic spatial dependencies and local spatial dependencies. In this article, we propose a novel dual branch graph neural network (DBGNN), in which the main branch stacked with deep spatiotemporal layers is used to capture long-range dynamic spatio-temporal dependencies, and the auxiliary branch containing shallow graph convolution is used to model local spatial correlations. By fusing the outputs of the two branches, more accurate interpolation results are obtained.

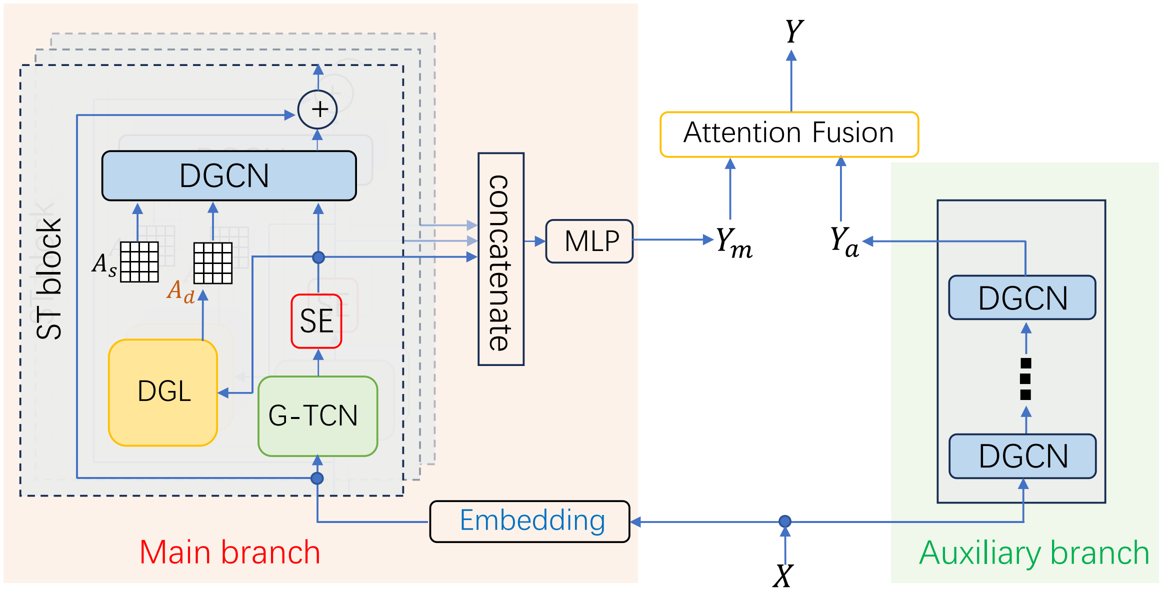


Figure 2: Framework of our DBGNN. The overall structure consists of two branches and a Fusion module. The main branch consists of an Embedding layer, several spatio-temporal blocks, and an MLP layer. Each spatio-temporal block includes a Gated Temporal Convolutional module (G-TCN), a Squeeze and Excitation (SE) module, a Dynamic Graph Learning (DGL) layer, and a Diffusion Graph Convolution (DGCN) module. The auxiliary branch includes shallow layers of DGCN. In each spatio-temporal block of the main branch, the predefined static graph $\mathbf{A}_s$ and the dynamic graph $\mathbf{A}_d$ learned via DGL are jointly used to represent spatial correlations, while the predefined static graph $\mathbf{A}_s$ is only used for each DGCN in the static graphs of the auxiliary branch.

3. Methodology

3.1. Preliminaries

Road network can be defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ is a set of N nodes (equivalent to sensors and their positions) within the road network, and $\mathcal{E}$ is a set of edges which represent the connections between nodes. $\mathbf{A} \in \mathbb{R}^{N \times N}$ corresponds to the adjacency matrix of the graph, where $A_{ij} = 1$ indicates that node v_i and node v_j are connected. Assuming $\mathbf{X}^o_{T-t:T} \in \mathbb{R}^{N_o \times t}$ represents the data of N_o observed points at t time slices, and $\mathbf{X}^u_{T-t:T} \in \mathbb{R}^{N_u \times t}$ represents the data of N_u unobserved points at t time slices, given $\mathbf{A} \in \mathbb{R}^{(N_0+N_u)\times(N_0+N_u)}$ representing the adjacency relationships of all nodes (include observed and unobserved nodes, i.e., $N = N_o + N_u$), along with a trainable model $\mathcal{F}$, the spatial interpolation task is to estimate $\mathbf{X}^u_{T-t:T}$ as follows:

$$\mathbf{X}^u_{T-t:T} = \mathcal{F}[\mathbf{X}^o_{T-t:T}] \quad (1)$$

3.2. Overview

The proposed model DBGNN is shown in Fig. 2, which consists of a main branch, an auxiliary branch, and a fusion module. The main branch is composed of multiple stacked spatiotemporal blocks, which are used to capture dynamic spatiotemporal correlations. The auxiliary branch employs a few layers of graph convolution to capture local spatial correlations, obtaining low-level detailed information. The fusion module merges the interpolation results from these two branches using self-attention. In our method, we use GCN to achieve spatial interpolation, which aggregates information

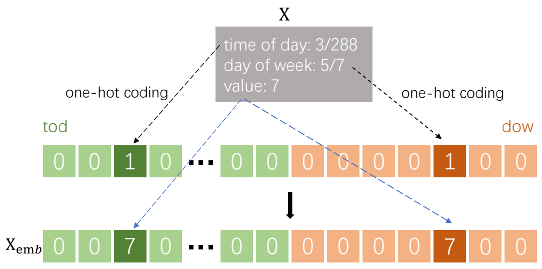


Figure 3: An example of time embedding.

from known nodes to infer values for unknown nodes. We will detail these components next.

3.3. Main Branch

3.3.1. Embbeding

Traffic data, as a representative of spatiotemporal data, exhibits strong periodicity and trends. Encoding time-of-day and day-of-week information for each time step can help the model learn these characteristics. Specifically, the time-of-day and day-of-week information for each time slice can be separately encoded as one-hot vectors $\mathrm{E}_{\mathrm{tod}} \in \mathbb{R}^{F_t}$ and $\mathrm{E}_{\mathrm{dow}} \in \mathbb{R}^{F_d}$, where F_t represents the number of time steps in a day (e.g., if recorded every 5 minutes, then $F_t = 288$), and F_d represents the number of days in a week, i.e., $F_d = 7$. These two vectors can be concatenated to obtain $\mathrm{E}_{\mathrm{time}} \in \mathbb{R}^{(F_t+F_d)}$. The one-hot encoding vector E_{time} consists of 0 and 1. The values at positions L_{Ft} and L_{Fd} in the vector E_{time} are 1, indicating that the time information of this traffic data is on the L_{Ft}-th time slice of one day and the L_{Fd}-th day of one week. Here, $L_{Ft} \in [1, F_t]$, $L_{Fd} \in [F_t + 1, F_t + F_d]$. To further reflect the specific traffic data at this time, we replace the 1 at positions L_{Ft} and L_{Fd} in the vector E_{time} with the actual traffic data value. Thus we obtain the embedded data $\mathcal{X}_{\text{emb}} \in \mathbb{R}^{N\times T\times(F_t+F_d)}$, where N represents the number of nodes, and T represents the number of time steps in the input. Finally, we use a 1×1 convolution to reduce its dimension and obtain the hidden state input $\mathcal{H}_0 \in \mathbb{R}^{N\times T\times F}$ for the subsequent spatiotemporal layers. An example of the time embedding is shown in Fig. 3.

3.3.2. Gated-Temporal Convolutional Network

We use dilated causal convolution [59] as the TCN layer to capture the temporal dependencies of a node. Previous work often used RNN such as LSTM [24] and GRU [25] for sequence modeling tasks. Compared to RNN-based models, the training of TCN can be done in parallel, and it is not prone to gradient vanishing and exploding issues. TCN is adept at capturing local relationships due to its crucial convolutional structure. By combining dilated convolution techniques [59], it can improve its ability to capture long-range dependencies to a certain extent. In addition, the causal convolution ensures that the prediction at the current time step depends only on past information, while dilated convolution allows the receptive field to grow exponentially as the number of layers increases. Given a one-dimensional sequence $\mathbf{s}\in\mathbb{R}^T$ and a convolutional kernel $\boldsymbol{\theta}\in\mathbb{R}^{K_s}$, the causal dilated temporal convolution can be represented as follows:

$$\mathbf{s}\star\boldsymbol{\theta}(t)=\sum_{i=0}^{K_s-1}\boldsymbol{\theta}(i)\mathbf{s}(t-d\times i) \tag{2}$$

where $\star$ represents the convolution operation, t represents the current time step, d is the dilation factor, and K_s is the size of the convolutional kernel.

Furthermore, we adopt the gated TCN (G-TCN) proposed in [35] to model complex temporal dependencies by leveraging the effectiveness of the gating mechanism. Given the input $\mathcal{X}\in\mathbb{R}^{N\times F\times T}$, the mathematical formulation of the gated TCN is written as follows:

$$\mathbf{h}=\tanh\left(\theta_1\star\mathcal{X}+\mathbf{b}\right)\odot \mathrm{sig}moid(\theta_2\star\mathcal{X}+\mathbf{c}) \tag{3}$$

where the θ_1,θ_2, $\mathbf{b}$ and $\mathbf{c}$ are model parameters, $\bigodot$ represents the Hadamard product, $tanh(\cdot)$ serves as the activation function for the input, and $sigmoid\,(\cdot)$ determines the gating ratio for filtering the latent representations.

3.3.3. Dynamic Graph Learning

The adjacency relationship between nodes is usually represented by a static adjacency matrix $\mathbf{A}_s\in\mathbb{R}^{N\times N}$, which is usually predefined based on the spatial Euclidean distance between nodes. However, the predefined adjacency matrix is static and cannot adapt to the dynamic changes of the dependencies between nodes over time. Moreover, the adjacency matrices based on prior knowledge, such as spatial distance, fail to fully capture the hidden spatial relationships, such as functional similarity. Recent works focus on learning dynamic graph structures from data. For instance, in [35], two self-learning node embeddings are used to guide the spatiotemporal predictions. Although this approach can learn spatial connections from training data, this method cannot handle unseen nodes. In addition, the spatial correlations between nodes in the training and testing sets may differ.

In contrast, we choose to directly learn the dynamic adjacency matrix $\mathbf{A}_d\in\mathbb{R}^{N\times N}$ from the input data. Inspired by the work in [60], we utilize self-attention mechanism to capture the spatial relationships between traffic nodes for node-specific data. Assuming the current input is $\mathbf{X}\in\mathbb{R}^{B\times N\times T}$, we can obtain the embedding $\mathbf{X}\in\mathbb{R}^{B\times N\times T\times F}$ by passing $\mathbf{X}$ through an embedding layer, where B represents the batch size, N is the number of nodes, T is the number of time slices in the input, and F denotes the dimension of the embedding layer. Then, for every sample $\mathbf{X}_b\in\mathbb{R}^{N\times T\times F}$, we reshape it into $\mathbf{X}'_b\in\mathbb{R}^{N\times D}$, where $D=T\times F$. The dynamic graph is computed as follows:

$$\begin{aligned}
&\mathbf{Q}_b=\mathbf{X}'_b\mathbf{W}_q,\\
&\mathbf{K}_b=\mathbf{X}'_b\mathbf{W}_k,\\
&\mathbf{A}_d=softmax\left(\sum_{b=0}^{B}\frac{\mathbf{Q}_b\mathbf{K}_b^\top}{\sqrt{d}_k}\right),\\
&A_d^{ij}=\begin{cases}A_d^{ij}, & \text{with probability } 1-p\\ 0, & \text{with probability } p,\ \text{and } A_d^{ij}\notin \text{top } K(\mathbf{A}_d)\end{cases}
\end{aligned} \tag{4}$$

where $\mathbf{W}_q\in\mathbb{R}^{D\times d}$ and $\mathbf{W}_k\in\mathbb{R}^{D\times d}$ are the learned parameters. The superscript $\top$ represents the transpose of a matrix. Since $\mathbf{A}_d$ is computed from data, during training and testing phases, it changes with the input, reflecting the dynamic changes in the spatial dependencies between the nodes as their corresponding time series change. Furthermore, to make the model more robust, we introduce randomness into the generation of dynamic graph edges in the training stage. Specifically, we retain the top K edges with the strongest correlation and randomly discard the remaining edges with a probability p. This approach preserves the edges with strong correlation and avoids the loss of important correlations, while reducing the risk of overfitting and alleviating the problem of over smoothing caused by multiple graph convolutions [13].

3.3.4. Diffusion Graph Convolution Network

Graph convolution serves as a pivotal operation for extracting node features in non-Euclidean space and integrating them according to the graph structure. By simplifying the Chebyshev spectral filter [33], a first-order approximate spatial GCN is proposed in [32] to update the node states by aggregating the information from the adjacent nodes. Given the input $\mathbf{X}\in\mathbb{R}^{N\times F}$ and the normalized adjacency matrix $\tilde{\mathbf{A}}\in\mathbb{R}^{N\times N}$, the GCN is defined as follows:

$$\mathbf{H}=\tilde{\mathbf{A}}\mathbf{X}\mathbf{W} \tag{5}$$

where $\mathbf{H}\in\mathbb{R}^{N\times D}$ represents the output and $\mathbf{W}$ represents the parameter matrix. In [37], Diffusion Graph Convolution Network (DGCN) was proposed based on GCN, which views graph convolution as a K-step diffusion process, and this has been proven to be effective. Therefore, we use DGCN [37] as the fundamental module for extracting and aggregating spatial information, as follows:

$$\mathbf{Z} = \sum_{k=0}^{K_o} \left(\mathbf{A}_{\mathrm{f}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k1}} + \mathbf{A}_{\mathrm{b}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k2}} \right) \tag{6}$$

where the k-th order forward transition matrix $\mathbf{A}_{\mathrm{f}}^{k} = \mathbf{A}/rowsum(\mathbf{A})$, the backward transition matrix $\mathbf{A}_{\mathrm{b}}^{k} = \mathbf{A}/rowsum(\mathbf{A}^{\top})$, the $\mathbf{A}^{\top}$ is the transpose of $\mathbf{A}$, K_o represents the diffusion order, $\mathbf{W}_{\mathrm{k1}}$ and $\mathbf{W}_{\mathrm{k2}}$ represent the learnable parameters for the k-th forward and backward diffusion order, respectively.

However, using only a predefined static adjacency matrix $\mathbf{A}_{\mathrm{s}}$, DGCN is unable to capture the dynamic spatial connections among nodes. Motivated by [35], we incorporate a dynamic adjacency matrix $\mathbf{A}_{\mathrm{d}}$ into the diffusion graph convolution and propose the following graph convolutional layer:

$$\mathbf{Z} = \sum_{k=0}^{K_o} \left(\mathbf{A}_{\mathrm{s,f}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k1}} + \mathbf{A}_{\mathrm{s,b}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k2}} + \mathbf{A}_{\mathrm{d,f}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k3}} + \mathbf{A}_{\mathrm{d,b}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{k4}} \right) \tag{7}$$

where $\mathbf{A}_{\mathrm{s,f}}^{k}$, $\mathbf{A}_{\mathrm{s,b}}^{k}$, $\mathbf{A}_{\mathrm{d,f}}^{k}$, and $\mathbf{A}_{\mathrm{d,b}}^{k}$ correspond to the forward and backward transition matrices of $\mathbf{A}_{\mathrm{s}}$ and $\mathbf{A}_{\mathrm{d}}$, respectively, while $\mathbf{W}_{\mathrm{k1}}$, $\mathbf{W}_{\mathrm{k2}}$, $\mathbf{W}_{\mathrm{k3}}$, and $\mathbf{W}_{\mathrm{k4}}$ represent the corresponding learnable parameters.

3.3.5. Multi-level Feature Fusion

Multi-level feature extraction and fusion have been proven effective in computer vision [61, 62]. By extracting features at different receptive fields, both global and local information can be captured. In the field of spatiotemporal exploration, researchers have also recognized the multi-level nature of time series [4, 35]. As the features are passed through the stacked TCN modules, the receptive field is gradually increased. This results in multi-level features similar to those in GraphWavenet [35], where the shallow features capture the local subtle changes, while the deep features capture the global overall trends. Fusing features of different time resolutions can improve the modeling of temporal relationships within data.

In our work, the main branch consists of stacked spatiotemporal layers, where the embedded data is fed into these layers to extract features at different levels. Specifically, we obtain the downsampled features $\mathcal{H}^{l} \in \mathbb{R}^{N \times F_1 \times T_1}$ of the l-th layer in the temporal dimension using GTCN, then we adjust the weights of different channels to obtain $h^{l} \in \mathbb{R}^{N \times F_1 \times T_1}$ using the classical Squeeze and Excitation (SE) block [63]. At this point, the DGL layer is applied to compute the dynamic graph A_{d}^{l} corresponding to $h^{l} \in \mathbb{R}^{N \times F_1 \times T_1}$ and guide the subsequent diffusion graph convolutional layers. The output of DGCN is sent to the next saptio-temporal layer. It's important to note that with different time resolutions, the computed dynamic spatial correlations also vary. In addition, following the approach of [35], we concatenate all $h^{l} \in \mathbb{R}^{N \times F_1 \times T_1}$ and the output $z^{L} \in \mathbb{R}^{N \times F_L \times T_L}$ of the last DGCN with skip connections to obtain multi-level features and use an multi-linear perceptron (MLP) layer for fusion, resulting in the main branch's output $\mathbf{Y}_{\mathrm{m}} \in \mathbb{R}^{N \times T}$, as follows:

$$\mathbf{Y}_{\mathrm{m}} = MLP(concat(h^{1}, h^{2}, ...h^{L}, z^{L})) \tag{8}$$

3.4. Auxiliary Branch

In a spatial interpolation task, the unobserved points lack historical data for reference. Therefore, it is necessary to exploit information from other nodes to infer the missing data at the unobserved nodes. This can be achieved by modelling the spatial correlations of nodes, including both the local spatial correlations that reflect physical topology and the dynamic global spatial correlations that reflect semantic similarity. To this end, the auxiliary branch is designed to capture the local spatial associations between traffic nodes, using a shallow network to avoid the over-smoothing problem (i.e., the loss of details) caused by multiple graph convolutional layers.

Given the original input $\mathbf{X} \in \mathbb{R}^{N \times T}$ and the static adjacent matrix $\mathbf{A}_{\mathrm{s}} \in \mathbb{R}^{N \times N}$, we have the following formulation:

$$\mathbf{Z} = \sum_{k=0}^{K_{oa}} \left(\mathbf{A}_{\mathrm{s,f}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{ka1}} + \mathbf{A}_{\mathrm{s,b}}^{k} \mathbf{X} \mathbf{W}_{\mathrm{ka2}} \right) \tag{9}$$

where K_{oa} represents the diffusion order, $\mathbf{W}_{\mathrm{ka1}}$ and $\mathbf{W}_{\mathrm{ka1}}$ denote the corresponding learnable parameters, respectively. There are several key points worth mentioning here: (1) The dual-branch structure can represent features from different perspectives, as demonstrated in [4, 60, 64, 65]. We use it to separately model the dynamic global spatial correlations and the local spatial correlations. (2) We do not use downsampling in the temporal convolution of the auxiliary branch to preserve more detailed information from the input data. (3) The dynamic graph module is not used for the auxiliary branch as the dynamic correlations obtained can be prone to noise, which potentially degrades the performance, as shown in the experiment section. (4) The features obtained by the auxiliary branches contain detailed information, which can alleviate the problem of over smoothing using deep GCNs [66].

3.5. Fusion Module

To integrate the information from the aforementioned two branches, we concatenate the outputs of these two branches and then apply the self-attention mechanism, as follows:

$$\begin{aligned} \mathbf{Q} &= \mathbf{W}_{\mathrm{q}}(\mathbf{Y}_{\mathrm{m}}||\mathbf{Y}_{\mathrm{a}}), \\ \mathbf{K} &= \mathbf{W}_{\mathrm{k}}(\mathbf{Y}_{\mathrm{m}}||\mathbf{Y}_{\mathrm{a}}), \\ \mathbf{V} &= \mathbf{W}_{\mathrm{v}}(\mathbf{Y}_{\mathrm{m}}||\mathbf{Y}_{\mathrm{a}}), \\ \hat{\mathbf{Y}} &= \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}})\mathbf{V} \end{aligned} \tag{10}$$

where $\mathbf{Y}_{\mathrm{m}} \in \mathbb{R}^{N \times T}$ is the output of the main branch, and $\mathbf{Y}_{\mathrm{a}} \in \mathbb{R}^{N \times T}$ is the output of the auxiliary branch. This fusion method captures the dynamic features of the traffic data by leveraging the weights obtained from the two branches.

Algorithm 1 Training procedure

```
Input: training graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, training set $X$; initialized model DBGNN().
Output: optimized model parameters
for $i = 1$ to $Num\_Iteration$ do
    Initialize batch list $\mathbf{X}_\mathrm{b} = []$, $\mathbf{Y}_\mathrm{b} = []$.
    for $j = 1$ to $Batch\_Size$ do
        randomly choose a start time $t$.
        Append $X_{t:t+T}$ to $\mathbf{X}_\mathrm{b}$.
        Append $Y_{t:t+T}$ to $\mathbf{Y}_\mathrm{b}$.
    end for
    $\mathcal{G}_s$ = Graph_Sampling($\mathcal{G}$).
    $\mathbf{X}_b^m = mask(X_b, \mathcal{G}_s)$
end for
$\hat{\mathbf{Y}}_\mathbf{b}$ = model($\mathcal{G}, \mathbf{X}_\mathrm{b}^\mathbf{m}$)
Compute $Loss(\hat{Y}_b, Y_b)$ and the gradients
Updata learnable parameters
```

3.6. Training Procedure and Loss Fuction

The data is divided into a training set and a test set, where the unobserved points in the test set do not appear in the training set. In order to achieve better generalization, we follow the training strategy of [3]. In each iteration, we randomly mask some nodes and their corresponding data in the training set, and then use the data from the remaining unmasked nodes to recover the data for the masked nodes. The training procedure is shown in Algorithm 1. We optimize the model by minizing the mean squared error (MSE) loss between the ground truth and the estimated values, as follows:

$$\begin{aligned}\mathcal{L} = &\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}|\hat{\mathrm{Y}}_{i,t} - \mathrm{Y}_{i,t}| \\ &+\frac{1}{2NT}\sum_{i=1}^{N}\sum_{t=1}^{T}||\hat{\mathrm{Y}}_{\mathrm{m},i,t} - \mathrm{Y}_{i,t}||_2^2 \\ &+\frac{1}{2NT}\sum_{i=1}^{N}\sum_{t=1}^{T}||\hat{\mathrm{Y}}_{\mathrm{a},i,t} - \mathrm{Y}_{i,t}||_2^2\end{aligned} \tag{11}$$

where N denotes the number of nodes in the training set, T represents the number of time steps, Y denotes the ground truth, $\hat{\mathrm{Y}}$ is the final interpolated result, and $\hat{\mathrm{Y}}_\mathrm{m}$ and $\hat{\mathrm{Y}}_\mathrm{a}$ denote the outputs of the main branch and auxiliary branch, respectively.

4. Experiments

4.1. Datasets

To evaluate the performance of DBGNN, we conducted experiments on six real traffic datasets, i.e., METR-LA [37], PEMSBAY [37], PEMS04 [17], PEMS08 [17], Chengdu [67], and Sedata [3], and one non traffic dataset, i.e., BJAir [4], which is an air pollution dataset. (1) METR-LA [37] contains traffic speed data recorded by 207 sensors on the highways in Los Angeles from 01/03/2012 to 30/06/2012. (2) PEMSBAY [37] contains traffic speed data recorded by 325 sensors in the Bay Area of California from 01/01/2017 to 13/05/2017. (3) PEMS04 [17] contains traffic data (flow, speed, occupancy) collected by 307 sensors in San Francisco from 01/01/2018 to 28/02/2018. (4) PEMS08 [17] contains traffic data (speed, flow, occupancy) recorded by 170 sensors in the San Bernardino from 01/07/2016 to 31/08/2016. (5) Chengdu [67] is a collection of traffic speeds recorded in Chengdu, consisting of 524 sensors from January 1, 2018 to April 30, 2018, with a recording interval of 10 minutes. (6) Sedata [3] contains traffic speed data from 323 sensors in Seattle. (7) BJAir [4] contains air quality index data from 35 sensors in Beijing, with a recording interval of 1 hour.

4.2. Baseline Methods

We compare our method with the following baselines:

(1) KNN [7]: The record of the unknown nodes is estimated by averaging the data from the K-nearest known nodes.

(2) IDW [8]: The distance from an unknown point to a known point is used as a coefficient to weight the sum over the data of the known points.

(3) OKriging [10]: OKriging is a well-known statistical interpolation algorithm based on geographical coordinates. Since the PEMS04 and PEMS08 datasets do not have sensor coordinate information, we only tested the performance on METR-LA and PEMSBAY.

(4) GLTL [11]: Greedy Low-rank Tensor Learning is a transductive model based on matrix factorization.

(5) KCN [2]: It uses KNN to find reference nodes for unknown points and then combines GNN for interpolation.

(6) IGNNK [3]: This is an inductive spatio-temporal Kriging model that learns the spatio-temporal correlations using the time series of nodes as features.

(7) DualSTN [4]: It uses two branches to model long-term and short-term connections in temporal dimension.

(8) INCREASE [6]: This is an inductive model by fusing three different spatial correlations with an attention mechanism.

(9) IAGCN [58]: It combines graph convolution and temporal convolution to model spatiotemporal dependencies, and uses auxiliary tasks to help the model learn better.

4.3. Experiment Settings

For fair comparison, we use the same data partitioning for all baselines. Following DualSTN, in all datasets, we use the first 70% of the time slices as the training set, the next 20% as the validation set, and the final 10% as the test set. We reserve 50% of the nodes for testing, and the remaining nodes for training. In each iteration, we randomly mask 50% of the training nodes as the unknown nodes. Note that test nodes will not appear during the training phase. All methods are implemented in PyTorch 1.10 and trained on an NVIDIA GeForce RTX 3090 GPU. We set the time window $T = 25$ for all the models. We set the number of training points for the METR-LA, PEMSBAY, PEMS04, and PEMS08 datasets to be $N_o = 104$, $N_o = 165$, $N_o = 157$, and $N_o = 90$, respectively. The number of test points on these four datasets

Table 1
Performance comparison our DBGNN and baselines on four real-world traffic datasets.

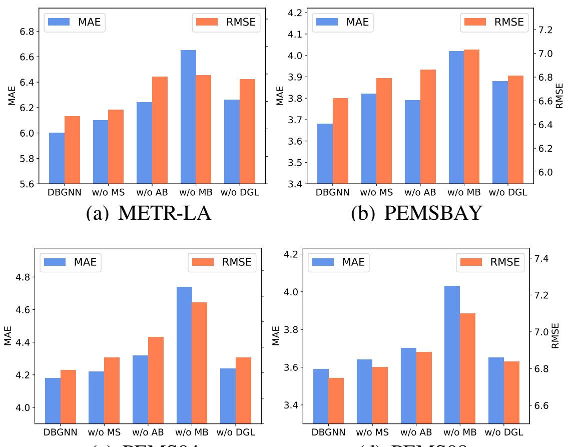
DATASET	METRIC	KNN	IDW	OK	GLTL	KCN	IGNNK	DUALSTN	INCREASE	IAGCN	OURS	Δ
METR-LA	MAE	8.43	8.16	8.35	8.09	7.57	6.78	7.11	6.26	6.69	**6.09**	+2.71%
	RMSE	11.90	11.53	12.34	11.71	11.25	10.12	10.63	9.73	10.94	**9.49**	+2.46%
	MAPE	0.223	0.208	0.256	0.206	0.203	0.196	0.188	0.169	0.195	**0.168**	+0.59%
PEMSBAY	MAE	5.45	5.57	5.53	5.20	4.70	4.04	3.95	3.88	4.14	**3.68**	+5.15%
	RMSE	10.06	9.81	9.05	8.90	8.12	6.96	6.68	6.65	7.29	**6.62**	+0.45%
	MAPE	0.109	0.113	0.105	0.098	0.095	0.092	0.088	0.090	0.095	**0.087**	+1.13%
*PEMS04	MAE	5.67	5.43	-	5.57	5.31	4.76	4.72	4.34	4.70	**4.18**	+11.06%
	RMSE	9.79	9.44	-	9.46	9.13	8.49	8.30	8.06	8.25	**8.02**	+0.49%
	MAPE	0.131	0.129	-	0.126	0.125	0.114	0.107	0.112	0.110	**0.107**	+0%
*PEMS08	MAE	5.32	5.22	-	4.80	4.55	4.27	3.88	4.14	3.79	**3.59**	+5.27%
	RMSE	9.02	8.76	-	8.06	7.84	7.12	6.97	7.28	6.98	**6.75**	+3.15%
	MAPE	0.119	0.104	-	0.096	0.095	0.092	0.084	0.092	0.085	**0.082**	+2.38%
*CHENGDU	MAE	-	-	-	8.10	7.91	7.32	6.89	7.24	7.16	**6.32**	+8.27%
	RMSE	-	-	-	10.57	10.01	9.70	9.28	9.45	9.34	**8.61**	+7.21%
	MAPE	-	-	-	0.388	0.375	0.311	0.289	0.322	0.316	**0.263**	+8.99%
*SEDATA	MAE	-	-	-	5.18	5.23	4.94	4.78	6.19	5.12	**4.54**	+5.02%
	RMSE	-	-	-	8.43	8.66	7.92	7.42	9.38	8.07	**7.39**	+0.40%
	MAPE	-	-	-	0.182	0.189	0.171	0.146	0.193	0.167	**0.142**	+2.73%
*BJAIR	MAE	18.45	17.98	18.03	16.52	16.33	15.42	13.58	14.15	14.45	13.66	-0.58%
	RMSE	29.32	28.25	28.57	26.83	28.41	27.93	25.64	28.71	27.64	**25.34**	+1.17%
	MAPE	1.042	0.947	0.951	0.938	0.767	0.607	0.487	0.564	0.521	0.509	-4.51%

$*$ denotes re-implementation on this dataset. ___ denotes the best baseline. - denotes that no result has been reported for the baseline on this dataset.

are $N_u = 103$, $N_u = 160$, $N_u = 150$, and $N_u = 80$, respectively. For our DBGNN, through experiments, we set the following hyperparameters. The main branch stacks 6 layers of STblock, while the auxiliary branch uses two layers of DGCN. All GTCNs use 32 convolutional kernels with a size of 2 and a dilation factor of 2. In the dynamic graph module, the number of attention heads is 1, and the embedding dimension $d = 32$, the probability $p = 0.1$, the $K = 0.1 * N$, where N is the number of nodes in the current dynamic graph. In the fusion module, the embedding dimension of attention is $d = 64$. The optimizer is Adam with a maximum of 500 epochs, and the learning rate is 0.0005. The batch size is 4. We use the mean absolute error (MAE), the mean absolute percentage error (MAPE), and the root mean squared error (RMSE) as the performance metrics.

4.4. Experiment Results and Analysis

Table 1 shows the performance of DBGNN and eight baselines. DBGNN has achieved the best results on four real traffic datasets. Because we used the same setting as DualSTN, we referenced its experimental data on METR-LA and PEMSBAY, and retrained the baseline on PEMS04 and PEMS08. From the table 1, it can be seen that the performance of traditional statistical models is not as good as that of deep learning methods. This is mainly because they are unable to capture complex spatio-temporal correlations. Both KCN and IGNNK use GNNs to capture spatial correlations, but IGNNK also exploits the temporal dimension, which helps correct the interpolation results. Meanwhile, due to the independent modeling of time dependencies, DualSTN and INCREASE generally perform better than IGNNK. The performance of IAGCN is not very stable, which may be due to the impact of the inaccurate embeddings of the unknown nodes. Our model achieves state-of-the-art performance on the six traffic datasets. On the BJAir dataset, our model is only slightly inferior to DualSTN. This may be due to differences in dataset characteristics. In traffic scenarios, the traffic flow of any node in the road network usually flows directly to its downstream nodes, resulting in strong spatial correlation between locally adjacent nodes. At the same time, due to the diffusion effect of traffic, long-distance nodes may also have potential associations. Our model effectively captures the local and global features from the nodes that are crucial for spatial interpolation. In air scenarios, air quality often exhibits local spatial smoothing characteristics. In such scenarios, the introduction of global information can lead to redundant information that may corrupt extracted features.



(a) METR-LA (b) PEMSBAY
(c) PEMS04 (d) PEMS08
Figure 4: Ablation experiment of module effectiveness.

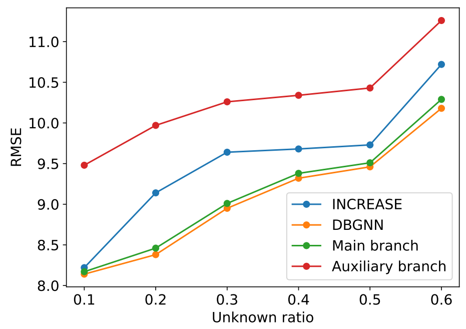


Figure 5: Performance at different missing ratio on the METR-LA dataset. This result shows that our DBGNN has achieved better performance under various missing rates.

Table 2
Performance Comparison of Different Main Branch Layers (L_{main}) with/without auxiliary Branch.

DATASET	L_{main}	MODEL	MAE	RMSE	MAPE
METR-LA	2	W/O AUX	6.50	9.98	0.190
		W AUX	6.28	9.91	0.183
	4	W/O AUX	6.20	9.76	0.179
		W AUX	6.13	9.63	0.175
	6	W/O AUX	6.12	9.56	0.176
		W AUX	**6.09**	**9.49**	**0.168**
PEMSBAY	2	W/O AUX	3.88	6.83	0.093
		W AUX	3.82	6.73	0.092
	4	W/O AUX	3.84	6.77	0.092
		W AUX	3.76	6.72	0.091
	6	W/O AUX	3.80	6.73	0.091
		W AUX	**3.68**	**6.62**	**0.089**
PEMS04	2	W/O AUX	4.31	8.17	0.110
		W AUX	4.17	7.96	0.106
	4	W/O AUX	4.28	8.27	0.110
		W AUX	**4.14**	8.05	**0.106**
	6	W/O AUX	4.31	8.25	0.110
		W AUX	4.18	**8.02**	0.107
PEMS08	2	W/O AUX	3.78	6.90	0.084
		W AUX	**3.55**	**6.71**	**0.080**
	4	W/O AUX	3.85	6.99	0.085
		W AUX	3.58	6.74	0.081
	6	W/O AUX	3.84	6.99	0.086
		W/O AUX	3.60	6.76	0.082

4.5. Ablation Study

To verify the effectiveness of each module in our model, we made the following variants of DBGNN: (1) w/o Mul: This variant removes the multi-level feature integration module and only utilizes the output of the last ST block. (2) w/o De: This variant removes the auxiliary branch. (3) w/o Main: This variant removes the main branch. (4) w/o DGL: This variant removes the dynamic graph learning module. We performed ablation experiments on the above variants on four datasets. Fig. 4 show the evaluation metrics of MAE and RMSE, where lower values indicate better performance. It can be observed that DBGNN outperforms all the variants, providing evidence for the effectiveness of the proposed modules in our model.

In addition, to investigate the performance of the models under different missing ratios, we conducted experiments on the METR-LA dataset for INCREASE, DBGNN, the main branch, and the auxiliary branch. The results are shown in Fig. 5, where the x-axis represents the ratio of unknown points to the total number of nodes, and the y-axis represents the RMSE metric. It can be observed that the main branch performs better than INCREASE, demonstrating the robustness of our proposed model. Additionally, the auxiliary branch has the poorest performance due to its simple structure. However, when combining the main and auxiliary branches in the DBGNN model, the best performance is achieved, indicating the improvement in interpolation accuracy by the auxiliary branch. We attribute the relatively small improvement to the presence of noise in the detailed information, which can degrade the final results. We plan to further improve this aspect in future work.

We further explored the differences between dynamic graph learning and GAT. We replaced the attention score calculation method in our dynamic graph learning (DGL) module with GAT to obtain a variant of the proposed method, DBGNN-GAT. The results are shown in Table 3. It can be seen that the DGL module performs better than the GAT module in the proposed DBGNN method. This may be caused by the following reasons. (1) We use vector scaling dot products, while GAT uses a process involving vector concatenation followed by linear transformation. (2) GAT only calculates attention scores for the adjacent nodes. However, in our dynamic graph, the attention is calculated for all the node pairs, including those without physical connections. As a result, we can explore potential spatial associations and provide more reference information for unknown points to help improve the interpolation accuracy.

In addition, we investigate whether using DGL modules in the auxiliary branch provides any benefits. To this end, we performed additional experiments to evaluate a variant DBGNN-AUXDGL, where we added the DGL module to the auxiliary branch. The results are shown in Table 4. It can be seen that after adding the DGL module, the performance of the proposed method is actually degraded on the METR-LA and PEMSBAY datasets, while the performance deteriorated slightly or remained almost the same on PEMS04 and PEMS08 datasets. This may be because the auxiliary branch focuses more on the details and changes in the local receptive field. The DGL is somewhat prone to the impact of noise, thus resulting in degradation in the local information captured by the auxiliary branch. In contrast, static adjacency graphs offer advantages in capturing the structural properties between nodes within a local region.

Finally, to further investigate the effectiveness of the auxiliary branch, we conducted ablation experiments at different number of main branch layers: without auxiliary branch (W/O AUX), and with auxiliary branch (W AUX). As shown in Table 2, without auxiliary branches, the system performs the best on the METR-LA and PEMSBAY datasets with 6 layers in the main branch, while performing the worst with 2 layers. For the PEMS04 and PEMS08 datasets, the

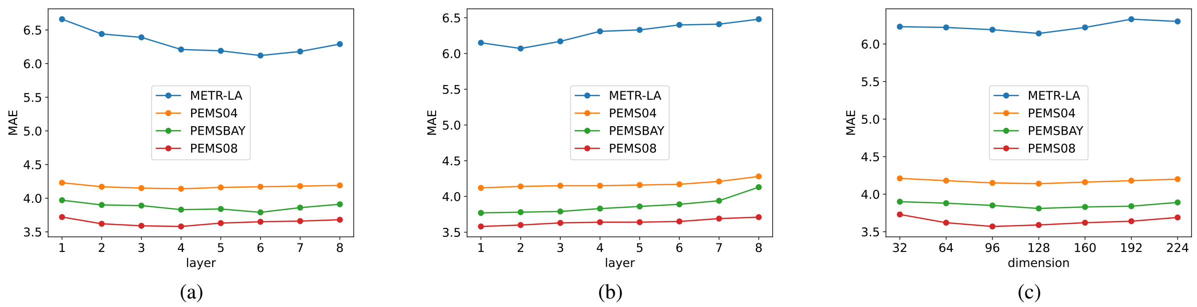
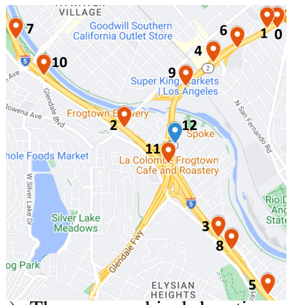
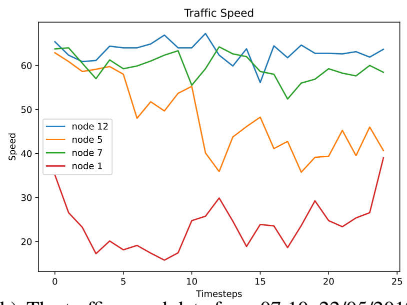


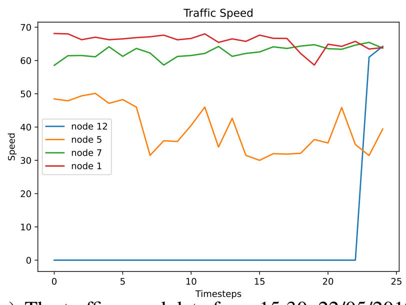
Figure 6: Parameters study on PEMSBAY, PEMS04, PEMS08. (a) Layers for ST blocks in main branch. (b) Layers for DGCNs in auxiliary branch. (c) Dimensions of hidden representation in auxiliary branch



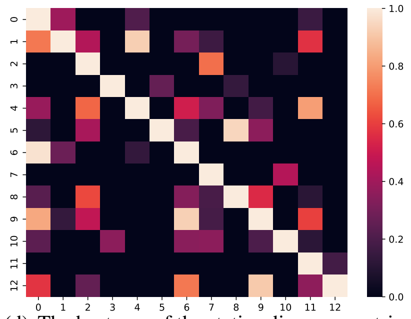
(a) The geographical location of part of the nodes marked on Google



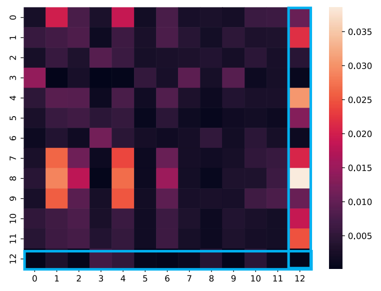
(b) The traffic speed data from 07:10, 22/05/2012.



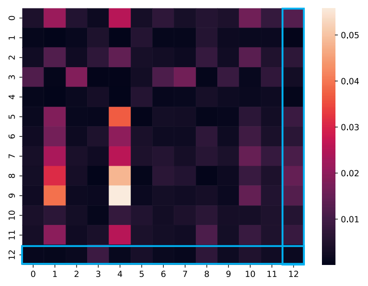
(c) The traffic speed data from 15:30, 22/05/2012.



(d) The heatmap of the static adjacency matrix



(e) The heatmap of the dynamic adjacency matrix from 07:10, 22/05/2012.



(f) The heatmap of the dynamic adjacency matrix from 15:30, 22/05/2012.

Figure 7: Visualization of dynamic graph

performance is relatively good when the number of layers was 2 or 4. When adding the auxiliary branch, the performance on all the datasets has been steadily improved. When the number of layers in the main branch is 2, it is unlikely to be over smoothing and can still improve performance. This indicates that when the number of layers is small, the auxiliary branch can extract features from another perspective, increasing the expressive power of the model. When the number of layers used in the main branch is 6, over smoothing may occur. The auxiliary branch can help increase node discrimination, and alleviate the problem of over smoothing.

4.6. Parameter Study

We investigate the impact of three parameters: the number of spatiotemporal (ST) layers in the main branch (L_m), the number of DGCN layers in the auxiliary branch (L_a), and the dimension of the hidden representation in the auxiliary branch (D_a). We have conducted experiments on the PEMSBAY, PEMS04, and PEMS08 datasets, and the results are presented in Fig. 6. From Fig. 6 (a), it can be observed that the performance is relatively low for a small number of ST-block layers in the main branch, due to its limitation in fitting with the data. Increasing the number of layers helps improve the performance. However, further increasing the number of layers may lead to overfitting and thus degradation in performance. In Fig. 6 (b), as the number of DGCN layers

Table 3
Comparison of performance between original DBGNN and DBGNN-GAT.

DATASET	MODEL	MAE	RMSE	MAPE
METR-LA	DBGNN	**6.09**	**9.49**	**0.168**
	DBGNN-GAT	6.20	9.71	0.174
PEMSBAY	DBGNN	**3.68**	**6.62**	**0.092**
	DBGNN-GAT	3.88	6.90	0.095
PEMS04	DBGNN	**4.18**	**8.02**	**0.107**
	DBGNN-GAT	4.30	8.07	0.109
PEMS08	DBGNN	**3.59**	**6.75**	**0.082**
	DBGNN-GAT	3.66	6.81	0.085

Table 4
Comparison of performance between original DBGNN and DBGNN-AUXDGL.

DATASET	MODEL	MAE	RMSE	MAPE
METR-LA	DBGNN	**6.09**	**9.49**	**0.168**
	DBGNN-AUXDGL	6.15	9.68	0.175
PEMSBAY	DBGNN	**3.68**	**6.62**	**0.092**
	DBGNN-AUXDGL	3.82	6.85	0.091
PEMS04	DBGNN	**4.18**	**8.02**	**0.107**
	DBGNN-AUXDGL	4.20	8.09	0.107
PEMS08	DBGNN	**3.59**	**6.75**	**0.082**
	DBGNN-AUXDGL	3.58	6.77	0.083

increases in the auxiliary branch, the model's performance deteriorates. This could be due to the loss of detailed information as the number of layers increases. Fig. 6 (c) demonstrates that a relatively low hidden dimension D_a can limit the model from accommodating information, while a relatively high dimension may lead to overfitting.

4.7. Visualization of Dynamic Graph

To validate the effectiveness of the dynamic graph (i.e., dynamic adjacency matrix) and also show more details, we performed visualizations on the dynamic graph and its corresponding data sequences. Specifically, we selected a subset of real sensors from the METR-LA dataset, numbered from 0 to 12. Fig. 7 (a) and Fig. 7 (d) respectively depict the distribution of these 13 sensors and the corresponding static adjacency matrix. Fig. 7 (b) and Fig. 7 (c) depict the recorded traffic speed data of node 1, 5, 7, and 12 in two different time periods, while Fig. 7 (e) and Fig. 7 (f) depict their respective adjacency matrices. We can observe the following. (1) The adjacency matrix varies across different time periods, which can be observed from the comparison of Fig. 7 (e) and Fig. 7 (f). (2) The hidden spatial correlations are shown to some extent. In Fig. 7 (b), the similarity between node 12's sequence and the sequences of the other three nodes is noticeably greater than that shown in Fig. 7 (c). Meanwhile, the brightness of coordinates (12, 1), (12, 5), and (12, 7) in Fig. 7 (d) is higher than the corresponding positions in Fig. 7 (e). This indicates that our dynamic graph learning module is capable of generating different graph structures based on different data and can uncover hidden spatial correlations between nodes, even when they are far apart.

4.8. Visualization of Interpolation Result

To enhance the interpretability of the model, we visualize the results of spatial interpolation, including the interpolation results of INCREASE, our model and the two branches in Fig. 8. From the figure, it can be observed that our model is closer to the ground truth compared to INCREASE, and the main branch is smoother than the auxiliary branch, while the auxiliary branch exhibits volatility, indicating that the auxiliary branch pays more attention to detailed information. However, because it only relies on the static adjacency matrix, its attention to overall changes is not as good as the main branch.

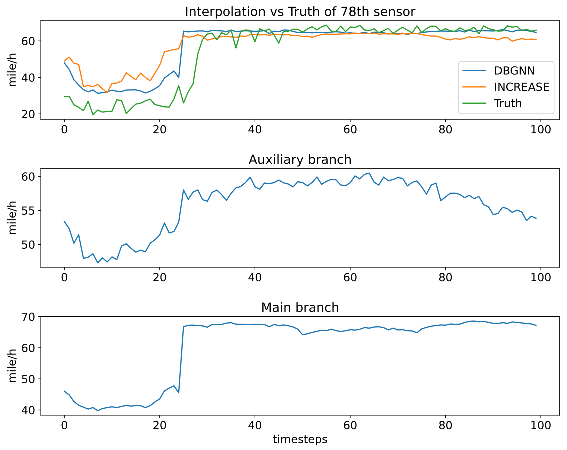


Figure 8: Comparision of interpolation result of INCREASE, DBGNN, Main branch, and Auxiliary branch on a snap of the test data of PEMSBAY. From the results of the auxiliary branch, it can be seen that our designed auxiliary branch can better capture fluctuations in detailed information. Please zoom in the plots for a better view.

5. Conclusion

We have presented a novel dual branch graph neural network (DBGNN) for spatial interpolation in traffic scene. Our DBGNN has utilized a novel dynamic graph learning module and stacked spatio-temporal blocks in the main branch to capture global dynamic spatial dependencies at multiple temporal levels. The auxiliary branch uses a shallow network to represent local spatial correlation between traffic nodes, thereby capturing detailed information in traffic data. The proposed DBGNN achieves new state-of-the-art performance on the six well-known traffic datasets for spatial interpolation, compared to recent baselines. In the future, we will further study multi-subspace representation learning to improve the performance of spatial interpolation for blind zone in traffic scene.

CRediT authorship contribution statement

Wujiang Zhu: Writing – original draft, Validation, Software, Resources, Methodology, Investigation. **Xinyuan Zhou:** Writing – review & editing, Formal analysis, Visualize. **Shiyong Lan:** Writing – review & editing, Supervision, Project administration, Investigation, Funding acquisition, Conceptualization. **Wenwu Wang:** Review & editing, Investigation. **Zhiang Hou:** Writing – review & editing, Data curation, Visualization. **Yao Ren:** Writing – review & editing. **Tianyi Pan:** Review & editing.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62371324. This work was also supported by the 2035 Innovation Pilot Program of Sichuan University, China.

References

[1] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[2] Gabriel Appleby, Linfeng Liu, and Li-Ping Liu. Kriging convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3187–3194, 2020.

[3] Yuankai Wu, Dingyi Zhuang, Aurelie Labbe, and Lijun Sun. Inductive graph neural networks for spatiotemporal kriging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4478–4485, 2021.

[4] Junfeng Hu, Yuxuan Liang, Zhencheng Fan, Li Liu, Yifang Yin, and Roger Zimmermann. Decoupling long-and short-term patterns in spatiotemporal inference. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[5] Yuankai Wu, Dingyi Zhuang, Mengying Lei, Aurelie Labbe, and Lijun Sun. Spatial aggregation and temporal convolution networks for real-time kriging. *arXiv preprint arXiv:2109.12144*, 2021.

[6] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, Jianzhong Qi, Chaochao Chen, and Longbiao Chen. Increase: Inductive graph representation learning for spatio-temporal kriging. *arXiv preprint arXiv:2302.02738*, 2023.

[7] T Cover and P Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[8] George Y Lu and David W Wong. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences*, 34(9):1044–1055, 2008.

[9] Daniel G Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[10] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015.

[11] Mohammad Taha Bahadori, Qi Yu, and Yan Liu. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3491–3499, 2014.

[12] Lei Deng, Xiao-Yang Liu, Haifeng Zheng, Xinxin Feng, and Youjia Chen. Graph spectral regularized tensor completion for traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10996–11010, 2021.

[13] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[14] Harvey J Miller. Tobler's first law and spatial analysis. *Annals of the association of American geographers*, 94(2):284–289, 2004.

[15] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision*, 129:3051–3068, 2021.

[16] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International Conference on Machine Learning*, pages 11906–11917. PMLR, 2022.

[17] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 914–921, 2020.

[18] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4365–4373, 2023.

[19] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 890–897, 2019.

[20] Jie Zhao, Chao Chen, Wanyi Zhang, Ruiyuan Li, Fuqiang Gu, Songtao Guo, Jun Luo, and Yu Zheng. Coupling makes better: an intertwined neural network for taxi and ridesourcing demand co-prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[21] Chenyi Chen, Jianming Hu, Qiang Meng, and Yi Zhang. Short-time traffic flow prediction with arima-garch model. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 607–612. IEEE, 2011.

[22] Yan Tian, Kaili Zhang, Jianyuan Li, Xianxuan Lin, and Bailin Yang. Lstm-based traffic flow prediction with missing data. *Neurocomputing*, 318:297–305, 2018.

[23] Huakang Lu, Zuhao Ge, Youyi Song, Dazhi Jiang, Teng Zhou, and Jing Qin. A temporal-aware lstm enhanced by loss-switch mechanism for traffic flow forecasting. *Neurocomputing*, 427:169–178, 2021.

[24] Long Short-Term Memory. Long short-term memory. *Neural computation*, 9(8):1735–1780, 2010.

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI conference on artificial intelligence*, pages 2588–2595, 2018.

[27] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1655–1661, 2017.

[28] Chuanpan Zheng, Xiaoliang Fan, Chenglu Wen, Longbiao Chen, Cheng Wang, and Jonathan Li. Deepstd: Mining spatio-temporal disturbances of multiple context factors for citywide traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3744–3755, 2019.

[29] Ahmad Ali, Yanmin Zhu, and Muhammad Zakarya. Exploiting dynamic spatio-temporal correlations for citywide traffic flow prediction using attention based neural networks. *Information Sciences*, 577:852–870, 2021.

[30] Nabeela Awan, Ahmad Ali, Fazlullah Khan, Muhammad Zakarya, Ryan Alturki, Mahwish Kundi, Mohammad Dahman Alshehri, and Muhammad Haleem. Modeling dynamic spatio-temporal correlations for urban traffic flows prediction. *IEEE Access*, 9:26502–26511, 2021.

[31] Chi Harold Liu, Yu Wang, Chengzhe Piao, Zipeng Dai, Ye Yuan, Guoren Wang, and Dapeng Wu. Time-aware location prediction by convolutional area-of-interest modeling and memory-augmented attentive lstm. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2472–2484, 2020.

[32] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[33] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[34] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1907–1913, 2019.

[36] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi 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*, pages 753–763. Association for Computing Machinery, 2020.

[37] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[38] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1234–1241, 2020.

[39] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

[40] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929, 2019.

[41] Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 4125–4129, 2023.

[42] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33:17804–17815, 2020.

[43] Xuxiang Ta, Zihan Liu, Xiao Hu, Le Yu, Leilei Sun, and Bowen Du. Adaptive spatio-temporal graph neural network for traffic forecasting. *Knowledge-Based Systems*, 242:108199, 2022.

[44] Wanlin Cai, Yuxuan Liang, Xianggen Liu, Jianshuai Feng, and Yuankai Wu. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. *CoRR*, abs/2401.00423, 2024.

[45] Xiaocao Ouyang, Yan Yang, Yiling Zhang, Wei Zhou, Jihong Wan, and Shengdong Du. Domain adversarial graph neural network with cross-city graph structure learning for traffic prediction. *Knowledge-Based Systems*, 278:110885, 2023.

[46] Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4189–4196, 2021.

[47] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 364–373, 2021.

[48] Wenyu Zhang, Kun Zhu, Shuai Zhang, Qian Chen, and Jiyuan Xu. Dynamic graph convolutional networks based on spatiotemporal data embedding for traffic flow forecasting. *Knowledge-Based Systems*, 250:109028, 2022.

[49] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2021.

[50] Junjie Ou, Jiahui Sun, Yichen Zhu, Haiming Jin, Yijuan Liu, Fan Zhang, Jianqiang Huang, and Xinbing Wang. Stp-trellisnets+: Spatial-temporal parallel trellisnets for multi-step metro station passenger flow prediction. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):7526–7540, 2022.

[51] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM international Conference on Data mining*, pages 403–414. SIAM, 2012.

[52] Ziyue Li, Nurettin Dorukhan Sergin, Hao Yan, Chen Zhang, and Fugee Tsung. Tensor completion for weakly-dependent data on graph for metro passenger flow prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4804–4810, 2020.

[53] Dongwei Xu, Chenchen Wei, Peng Peng, Qi Xuan, and Haifeng Guo. Ge-gan: A novel deep learning framework for road traffic state estimation. *Transportation Research Part C: Emerging Technologies*, 117:102635, 2020.

[54] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[55] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[56] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

[57] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*, 2019.

[58] Tonglong Wei, Youfang Lin, Shengnan Guo, Yan Lin, Yiji Zhao, Xiyuan Jin, Zhihao Wu, and Huaiyu Wan. Inductive and adaptive graph convolution networks equipped with constraint task for spatial–temporal traffic data kriging. *Knowledge-Based Systems*, 284:111325, 2024.

[59] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[61] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[62] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[63] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[64] C Yu, C Gao, J Wang, G Yu, C Shen, and N BiSeNet Sang. V2: Bilateral network with guided aggregation for real-time semantic segmentation. arxiv 2020. *arXiv preprint arXiv:2004.02147*.

[65] Fan Zhang, Meng Li, Guisheng Zhai, and Yizhao Liu. Multi-branch and multi-scale attention learning for fine-grained visual categorization. In *International Conference on Multimedia Modeling*, pages 136–147, 2021.

[66] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence*, pages 841–851. PMLR, 2020.

[67] Yuan Yuan, Chenyang Shao, Jingtao Ding, Depeng Jin, and Yong Li. Spatio-temporal few-shot learning via diffusive neural network generation. In *The Twelfth International Conference on Learning Representations*, 2024.