Inductive and Adaptive Graph Convolution Networks Equipped with Constraint Task for Spatial-Temporal Traffic Data Kriging

Tonglong Wei[a,b], Youfang Lin[a,b], Shengnan Guo[a,b,*], Yan Lin[a,b], Yiji Zhao[a,b], Xiyuan Jin[a,b], Zhihao Wu[a,b] and Huaiyu Wan[a,b]

[a]School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
[b]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing 100044, China

ARTICLE INFO

Keywords:
Kriging
Spatial-temporal data mining
Graph neural networks

ABSTRACT

In intelligent transportation systems (ITS), deploying fine-grained sensors to continuously collect spatial-temporal traffic data is important but impractical due to the expensive cost. Fortunately, spatial-temporal kriging methods bring advanced solutions for interpolating traffic data for locations without sensors, but they still have the following two drawbacks: (1) The widely adopted predefined and adaptive graphs are either inflexible or limited to transductive learning. (2) The sampling strategies to support inductive learning on new graphs result in losing important partial information. To overcome the above issues, in this paper, we propose an Inductive and Adaptive Graph Convolution Networks (IAGCN) for spatial-temporal traffic data kriging in an inductive manner. Specifically, we propose an adaptive graph constructor to model the hidden spatial relation of nodes and learn the spatial dependency that the predefined graph structure cannot capture. It can work inductively and does not require retraining when introducing new nodes. Additionally, we design a framework that integrates inductive graph convolution and temporal convolution to simultaneously capture the complex spatial-temporal dependencies of traffic data. Finally, to address the information loss issue caused by random sampling, we design a predicted-based constraint task that perceives and utilizes the history information of all observed sensors to predict the current data, as well as approach the result of the interpolation and prediction. Experiments on four real-world datasets show that IAGCN outperforms the state-of-the-art baselines.

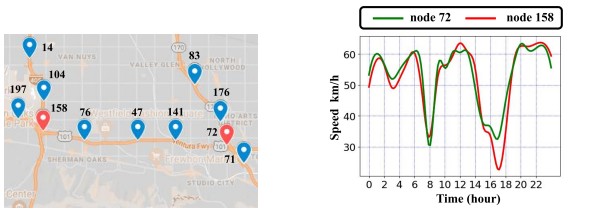
1. Introduction

As a crucial part of smart city construction, intelligent transportation systems (ITS) have been rapidly developing in recent years, bringing great convenience to our daily life [1, 2, 3]. To better support the development of ITS, sensors are deployed in different locations to continuously collect spatial-temporal traffic data to reflect the state of the whole transportation system in real time. For example, the loop sensors deployed in the California Transportation Agencies Performance Measurement System [4] detect the traffic flow and speed of road links, and the cameras deployed at key intersections are used to record traffic flow and vehicle type [5]. However, limited by the costs associated with deploying sensors, it is not feasible to deploy them in all locations of interest in a fine-grained manner. Consequently, the signals of the locations without sensors are never recorded, which not only inhibits our ability to comprehensively monitor the whole state of ITS but also hurts the performance of downstream tasks relying on the complete input. In this work, we focus on the spatial-temporal kriging problem, a kind of interpolation task. Specifically, given data collected from the locations with deployed sensors, the goal is to interpolate the signal of locations without sensors. A spatial-temporal kriging model can provide fine-grained data observations, enabling comprehensive monitoring of the complete state of transportation systems, and supporting a wide range of downstream traffic data mining tasks, e.g., traffic flow prediction [6, 7] and traffic anomaly detection [8, 9].

Actually, great efforts have been devoted to addressing the traffic data kriging task over the past decades. Early works are based on Tobler's first law of geography [10], which assumes that nearby objects are more similar than distant ones. Therefore, these methods utilize K-Nearest Neighbors (KNN) or inverse distance weighting (IDW) to linearly weight the surrounding sensors and interpolate the data for target locations without sensors. However, this kind of method has unsatisfactory performance since the spatial-temporal traffic data is nonlinear and complex. Gaussian process regression is another common solution for this issue [11]. It models the correlation between sensors by carefully designing the covariance function. However, the computational cost becomes extremely high as the number of sensors increases.

Recently, deep learning methods have brought new solutions to the kriging problem. Some researchers regard the traffic data as a multi-dimension tensor and use tensor completion to obtain satisfactory results [12, 13, 14, 15]. However, these methods are essentially transductive, meaning that even minor changes to the network structure (*e.g.,* some sensors are removed or a few new sensors are deployed) require retraining the model. Fortunately, graph neural networks (GNNs) have received much attention since they can well model complex traffic data and some GNNs [16, 17, 18] have inductive ability, *i.e.,* they can generalize to unseen new graph structures. Inspired by the success of GNNs,

*Corresponding author
weitonglong@bjtu.edu.cn (T. Wei); shnguo@bjtu.edu.cn (S. Guo)



(a) Geographical locations of node 158 and 72.

(b) Traffic speed curves of node 158 and 72.

Fig. 1: Geographical and traffic speed relevance between some nodes in the METR-LA dataset [25].

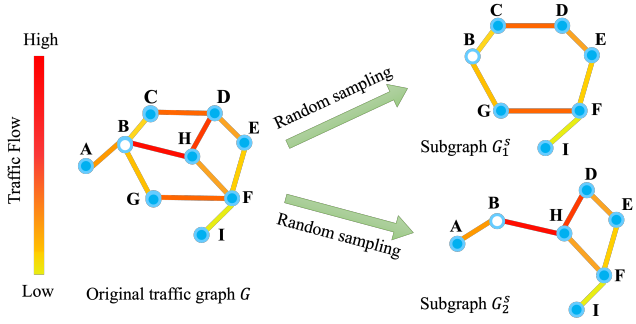


Fig. 2: An example of random sampling. The original traffic graph consists of deployed sensors. The bold line between sensors represents the traffic flow passing, with darker shades of color indicating greater traffic flow.

some GNN-based works [19, 20, 21, 22, 23, 24] have been developed for spatial-temporal kriging. While these studies demonstrate that GNNs can significantly enhance kriging performance, they still exhibit certain limitations.

(1) Although modeling the underlying spatial relationships among nodes is crucial in ITS, it is an open problem working on the inductive setting. To characterize nodes' spatial correlations, existing spatial-temporal kriging methods usually use a topology-based or distance-based graph structure. However, it is insufficient to reflect the complex correlations between nodes in ITS. As shown in Figure 1(a), nodes 158 and 72 are geographically distant but share high similarity in their speed values since they are both located at crossroads (Figure 1(b)). In the field of traffic forecasting, many representative methods have demonstrated the advantages of using learned graph matrices. *e.g.*, GraphWaveNet [26] and AGCRN [27] use an adaptive matrix to model global spatial relations. STFGNN [28] and STGODE [29] apply the dynamic time warping (DTW) [30] algorithm on the node's historical value to obtain node similarity relations. Meanwhile, many works [31, 32, 33] employ an attention mechanism to dynamically calculate the weights among nodes. However, these methods are limited to transductive learning when the nodes in the graph are static. Thus, how to learn an adaptive graph that can reflect the correlation between nodes with and without sensors in an inductive manner for the traffic kriging task is worth careful consideration.

(2) Previous studies [19, 20] use different sampling strategies to make the model work under the inductive setting, but it results in the loss of important information about the entire traffic network. Take the traffic network shown in Figure 2 as an example. Due to the influence of multiple neighbors, *i.e.*, nodes A, C, H, and G, the transportation hub B has a high traffic flow. However, after random sampling subgraphs from the original traffic network, hub B can only gather the flows of the neighbors in the subgraph, *e.g.*, flows from neighbors C and G in subgraph G_1^s, and A and H in G_2^s, which fails to accurately represent its real traffic state. Therefore, how to alleviate the loss of information caused by random sampling in inductive learning is a question worth considering.

To address these challenges, we propose a novel deep learning model: ***I**nductive and **A**daptive **G**raph **C**onvolution **N**etworks* (IAGCN) to perform spatial-temporal traffic kriging. Our model operates in an inductive manner, allowing interpolating signals for new locations without retraining. Different from existing works that focus on capturing the spatial correlation depending on the predefined graph structure, we design an adaptive graph constructor, which can adaptively capture the global spatial correlation and satisfy the inductive setting. To alleviate the loss of information caused by random sampling, a constraint task is well designed to utilize the observed information to supplement the lost information in the sampled graphs. Furthermore, we design a framework that simultaneously captures the spatial-temporal dependence of traffic data. The main contributions of this work are summarized as follows:

- We propose an adaptive graph constructor, capable of working in an inductive manner to learn the spatial correlation between nodes in an end-to-end fashion.
- To simultaneously capture the complex spatial and temporal dependence of traffic data in the kriging task, our proposed IAGCN integrates temporal convolution and inductive graph convolution.
- A constraint task is designed to alleviate the information loss problem caused by random sampling. It takes the historical information of all deployed sensors as input to predict the current values, so as to make full use of the complete observations that may not be perceived by the sampled subgraphs.
- Extensive experiments on four real-world datasets validate that IAGCN outperforms the state-of-the-art methods.

2. Related Work

2.1. Spatial-Temporal Traffic Data Interpolation

Traffic data interpolation is a fundamental problem in intelligent transportation systems, which can interpolate signals of non-deployed sensors based on the signals collected by deployed sensors. Early works [34, 35, 36] performed

data interpolation based on Tobler's first law of geography [10]. For example, KNN uses the nearest neighbor algorithm to find the nearest K neighbors for each sensor and averages their signals to obtain the result of the target sensor. Inverse distance weighting (IDW) calculates the weight of each sensor based on the distance and uses these weights to interpolate the signals from different sensors. However, these linear methods exhibit poor performance due to the complex and nonlinear nature of traffic data. Then Gaussian regression-based methods are used for spatial interpolation [11, 37], which learn the relation between nodes by constructing a flexible structure. These methods are more powerful than linear methods. However, they suffer from high computational costs when dealing with a large number of nodes. Tensor completion [14, 15] is an alternative approach for traffic data interpolation, where it regards traffic data as a fixed-size tensor. Unfortunately, tensor completion methods are transductive and need to retrain the model if the network structure changes.

Recently, deep learning methods have been applied to solve spatial-temporal interpolation problems [38, 39, 40]. KCN [19] established a pioneering connection between kriging and deep learning. It selects the unknown node as the central node and employs the nearest neighbor algorithm to identify the K nearest nodes, forming a subgraph. Subsequently, it utilizes the graph neural network for interpolation. While KCN achieves promising experimental results, it only focuses on local neighbor information in the spatial dimension, disregarding temporal dependencies. IGNNK [20] considers multiple time slices of traffic data, and utilizes graph neural network and the random sampling strategy for the inductive spatial-temporal kriging task. Besides, there are some works [21, 22, 23, 41, 24] focus on capturing the spatial and temporal correlations for interpolating the signals of unknown nodes. These methods have demonstrated that deep learning methods can achieve more accurate interpolation results. However, they face the drawback of information loss in inductive learning, as shown in Figure 2.

2.2. Spatial-Temporal Data Modeling Technology

Spatial-temporal data, which record the state of a system in both temporal and spatial dimensions, has pervaded many fields [42, 43, 44, 45, 46, 47]. To mine the underlying patterns of spatial-temporal data, it is important to effectively model spatial-temporal data. In the temporal dimension, RNNs [48, 49], TCNs [50], and Transformers [51] are widely employed to capture temporal dependencies due to their ability to model data evolution trends. In the spatial dimension, the distribution of the device that records the spatial-temporal data (such as sensors and cameras) is irregular, and several recent efforts regard it as the graph structure and use the graph neural networks (GNNs) to capture the spatial dependence [25, 52, 53]. However, these methods use predefined graph structures (distance-based or topology-based) to model the nodes' spatial correlations, which may be incomplete and insufficient for complex spatial distribution.

In the study of traffic prediction, many methods have demonstrated that learned adaptive matrices are competitive for model spatial correlations between nodes [54, 55, 56], which can be mainly divided into two categories: data-based methods and dictionary-based methods. The former uses node features to calculate the similarity, such as AST-GCN [32], ASTGNN [31], GMAN [33]. ST-GDN [57] uses the attention mechanism to dynamically calculate the weights between nodes. STFGNN [28] and STGODE [29] use dynamic time warping (DTW) [30] to capture the node's similarity. The latter captures global correlations by initializing two vectors for each node and using matrix multiplication to obtain the adaptive matrix, such as GraphWaveNet [26], AGCRN [27] and DAGN [55]. Ada-STNet [58] combines the above two methods to model micro- and macro- adaptive graph structures. The use of adaptive graph structures, coupled with components that capture spatial-temporal correlations, has demonstrated competitive performance across various spatial-temporal data mining tasks. However, in the kriging task, these methods often face difficulties in implementation because some nodes' signals are completely missing, and the number of missing nodes is uncertain. This presents a challenge for effectively applying these adaptive matrices to the kriging task.

3. Preliminaries

A graph is represented as $G = (V, \mathbf{A})$, where V is the set of all deployed sensors, $|V| = N$, and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix of G. The observations on G at time t are denoted as $\mathbf{X}_t = (\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^N)^\top \in \mathbb{R}^{N\times C}$, where $\mathbf{x}_t^v \in \mathbb{R}^C$ denotes the feature vector of node[1] v at time t, and C is the dimension of features.

Figure 3 gives an illustration of the kriging process. We use the training graph G to represent the irregular network between deployed sensors $\{1, 2, \cdots, 8\}$. Following the inductive setting, in each epoch, we randomly sample a subgraph $G^s = (V^s, \mathbf{A}^s)$ and its corresponding signals $\mathbf{X}_t^s \in \mathbb{R}^{N^s\times C}$, where $|V^s| = N^s$ denotes the number of nodes in G^s. Then, we divide V^s into known nodes $V_o^s = \{2, 4, 5, 7\}$ and unknown nodes $V_u^s = \{3, 6\}$. Since unknown nodes (without deployed sensors) cannot generate data in the real world, we set the signals of the unknown nodes in $\mathbf{X}_t^s$ to $\mathbf{0}$. The goal of spatial-temporal kriging is to interpolate the signals of unknown nodes at time t based on the signals of known nodes from the historical T steps and the current time step t. The mapping relation of spatial-temporal kriging is represented as

$$[\mathbf{X}_{t-T}^s, \ldots, \mathbf{X}_t^s, G^s] \overset{f(\cdot)}{\rightarrow} \widetilde{\mathbf{X}}_t^s, \tag{1}$$

where $\widetilde{\mathbf{X}}_t^s \in \mathbb{R}^{N^s\times C}$ denotes the interpolated signals of nodes. $f(\cdot)$ is the learned mapping function. Note that the sensor network during the test is not necessarily the same as the sensor network during training (*e.g.,* in Figure 3, sensor $\{8\}$ is removed, and new sensor $\{9\}$ is deployed during the

[1] In this paper, the terms sensor and node are used interchangeably.

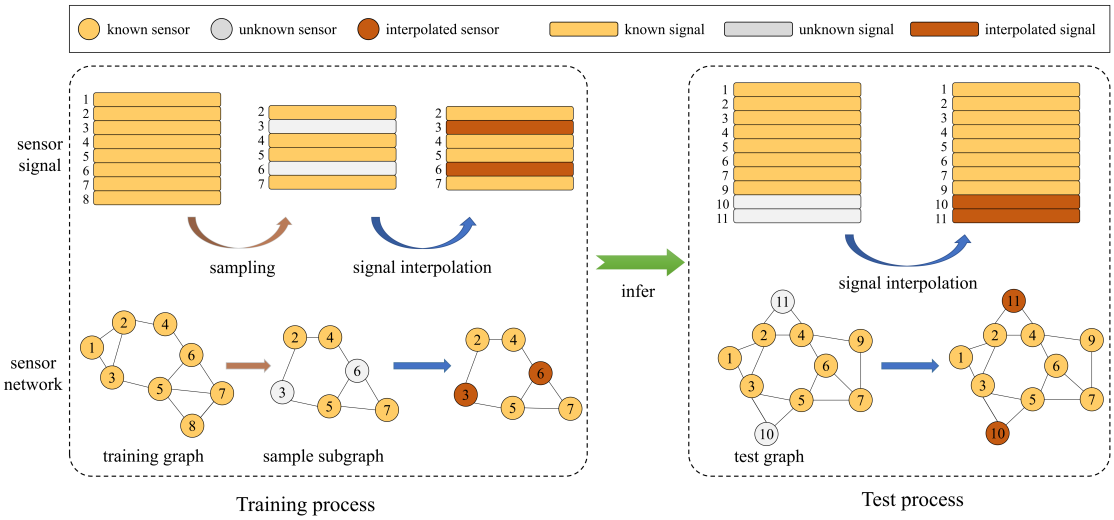


Fig. 3: Illustration of the kriging process. (1) The sensors are randomly masked to construct subgraphs during training. For example, in G^s, the sensors {3,6} are masked. (2) During the test, the goal is to interpolate unseen sensors {10, 11}. Note that the sensor network is not necessarily the same as the training set, *e.g.,* new sensor {9} is deployed, and deployed sensor {8} is removed.

test). Our goal is to use all deployed sensors for training and interpolate the signals of any location without deployed sensors in testing.

4. The IAGCN Model

Figure 4(a) illustrates the framework of our IAGCN model, which consists of three modules: the adaptive graph constructor, the kriging task, and the constraint task. The adaptive graph constructor module aims to capture the complex correlation between nodes in an end-to-end manner. It generates the adaptive adjacency matrix for both the kriging and constraint tasks. The second module consists of subgraph signals initialization and stacked kriging blocks, which captures the complex spatial-temporal dependence of traffic data and interpolates unknown nodes' signals inductively. The third module is designed to perform the prediction task, which makes full use of all deployed sensors' historical observations to alleviate the information loss problem caused by the sampling strategy.

4.1. Adaptive Graph Constructor

The graph is a common structure to model the complex topological relationships between nodes. However, the predefined graph structure in traffic data cannot fully reflect the complex correlations between nodes. Although some adaptive matrix construction methods have been proposed in the study of traffic prediction to solve this problem [26, 27], they are not suitable for direct application to inductive learning since they cannot deal with unseen nodes.

To fully capture the correlation between nodes in an inductive manner, we propose an adaptive graph constructor. As shown in Figure 5, it generates the adaptive adjacency matrix $\mathbf{A}_{\text{krig}}$, $\mathbf{A}_{\text{cons}}$ for the kriging and constraint tasks, respectively. They share two randomly initialized learnable node representation dictionaries $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{N\times D}$ as the source node embedding and target node embedding, respectively, where D denotes the dimension of dictionaries. Next, we describe how to generate $\mathbf{A}_{\text{krig}}$ and $\mathbf{A}_{\text{cons}}$ in detail.

4.1.1. Graph Construction for Kriging Task

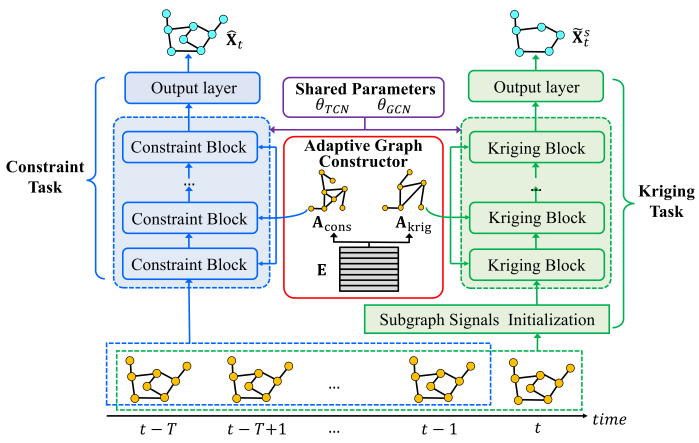
For the kriging task, the adaptive adjacency matrix $\mathbf{A}_{\text{krig}}$ is different from the existing adaptive matrix, which needs to satisfy the inductive setting since only part of the nodes is known.

In the kriging task, we have $N^s = N_o^s + N_u^s$ nodes in a sampled subgraph (see *Section 4.2.1 Subgraph Signals Initialization* module for detail). We aim to capture the spatial correlation between N^s nodes by only using the embedding of N_o^s. To achieve this, we first select the embeddings of the N_o^s nodes from $\mathbf{E}_1, \mathbf{E}_2$ to form $\mathbf{E}_1^{s_o}, \mathbf{E}_2^{s_o} \in \mathbb{R}^{N_o^s\times D}$. For the N_u^s unknown nodes, we set their embeddings to $\mathbf{0}$. Then, we concatenate these two kinds of embeddings along the node dimension to form new matrices $\mathbf{E}_1^s$ and $\mathbf{E}_2^s$, *i.e.*:

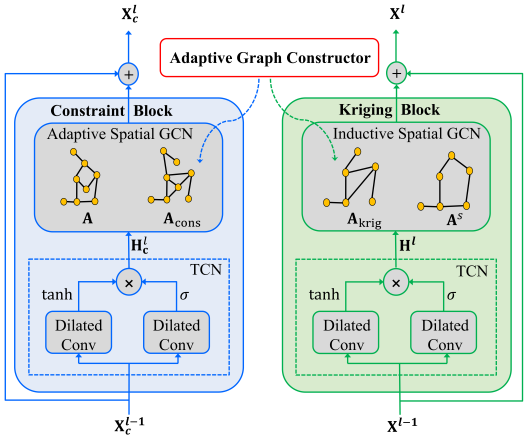
$$\mathbf{E}_1^s = \begin{vmatrix} \mathbf{E}_1^{s_o} \\ \mathbf{0} \end{vmatrix}, \qquad \mathbf{E}_2^s = \begin{vmatrix} \mathbf{E}_2^{s_o} \\ \mathbf{0} \end{vmatrix}, \tag{2}$$

where $\mathbf{E}_1^s, \mathbf{E}_2^s \in \mathbb{R}^{N^s\times D}$.

Inspired by Diffusion Graph Convolution (DGCN) [25], which can effectively model spatial-temporal data by capturing signals from upstream neighbors and downstream neighbors in a bidirectional diffusion process, we use DGCN to transfer node information to obtain the embedding of unknown nodes. Taking $\mathbf{E}_1^s$ as an example, the process of DGCN can be defined as:



(a) Architecture of IAGCN. θ_{TCN} and θ_{GCN} represent the parameters of TCNs and GCNs in both the kriging and the constraint task.



(b) Details of Kriging Block and Constraint Block.

Fig. 4: The framework of IAGCN consists of an adaptive graph constructor, a kriging task, and a constraint task.

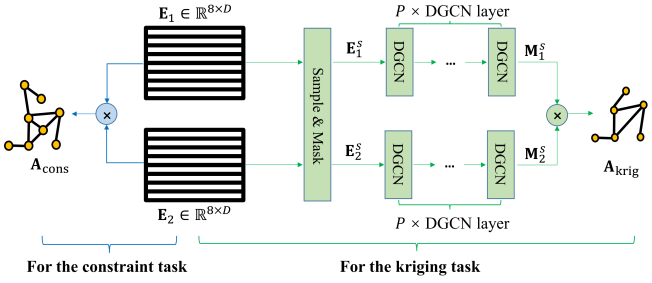


Fig. 5: The pipeline of the adaptive graph constructor. Here we have $N = 8$ and $N^s = 6$, i.e., $\mathbf{A}_{\text{krig}} \in \mathbb{R}^{6\times 6}$, $\mathbf{A}_{\text{cons}} \in \mathbb{R}^{8\times 8}$.

$$\mathbf{M}_1^s = \sum_{k=0}^{K-1} \mathbf{P}_f^{s\,k} \mathbf{E}_1^s \mathbf{W}_{k_1} + \mathbf{P}_b^{s\,k} \mathbf{E}_1^s \mathbf{W}_{k_2}, \tag{3}$$

where $\mathbf{M}_1^s$ is the output of DGCN, K is the number of diffusion steps, $\mathbf{P}_f^s = \mathbf{A}^s/\text{rowsum}(\mathbf{A}^s)$ and $\mathbf{P}_b^s = (\mathbf{A}^s)^\top/\text{rowsum}((\mathbf{A}^s)^\top)$ represent the forward and backward transition matrices, respectively, and $\mathbf{W}_{k_1}$ and $\mathbf{W}_{k_2}$ are learnable parameters. For convenience, we represent DGCN as Θ_{G^s}, *i.e.*,

$$\mathbf{M}_1^s = \Theta_{G^s}(\mathbf{E}_1^s, \mathbf{A}^s). \tag{4}$$

Similarly, we can obtain $\mathbf{M}_2^s$ based on $\mathbf{E}_2^s$:

$$\mathbf{M}_2^s = \Theta_{G^s}(\mathbf{E}_2^s, \mathbf{A}^s). \tag{5}$$

In the first layer of the DGCN, we cannot produce desirable results since unknown nodes only pass message 0 to their neighbors. Therefore, we stack P layers to update the nodes' embeddings. Finally, we obtain the adaptive adjacency matrix $\mathbf{A}_{\text{krig}}$ in an inductive manner by multiplying,

$$\mathbf{A}_{\text{krig}} = \text{softmax}(\text{relu}(\mathbf{M}_1^s(\mathbf{M}_2^s)^\top)) \in \mathbb{R}^{N^s\times N^s}. \tag{6}$$

During the testing phase, in addition to the N known nodes, N_u new nodes are added (need to be interpolated). To learn the spatial relation of $N + N_u$ nodes, we use $\mathbf{E}_1$ and $\mathbf{E}_2$ as the embedding of known nodes and use $\mathbf{0}$ for unknown nodes. Thus, Eq. (2) is rewritten as follows:

$$\mathbf{E}_1^{\text{test}} = \begin{vmatrix} \mathbf{E}_1 \\ \mathbf{0} \end{vmatrix}, \qquad \mathbf{E}_2^{\text{test}} = \begin{vmatrix} \mathbf{E}_2 \\ \mathbf{0} \end{vmatrix}. \tag{7}$$

where $\mathbf{E}_1^{\text{test}}, \mathbf{E}_1^{\text{test}} \in \mathbb{R}^{(N+N_u)\times D}$. We input $\mathbf{E}_1^{\text{test}}$, $\mathbf{E}_2^{\text{test}}$ and the graph structure $\mathbf{A}^{\text{test}}$ of the test set to Eq. (3)-(6) obtain the adaptive matrix $\mathbf{A}_{\text{krig}}^{\text{test}} \in \mathbb{R}^{(N+N_u)\times(N+N_u)}$.

4.1.2. Graph Construction for Constraint Task

In the constraint task, we treat all nodes as known nodes. Therefore, we need to capture the spatial correlation between all nodes. The embedding of all nodes has been initialized by $\mathbf{E}_1$ and $\mathbf{E}_2$ and can be updated end-to-end. Therefore, we directly perform matrix multiplication on $\mathbf{E}_1$ and $\mathbf{E}_2$ to learn $\mathbf{A}_{\text{cons}}$, and the calculation is formulated as follows:

$$\mathbf{A}_{\text{cons}} = \text{softmax}(\text{relu}(\mathbf{E}_1\mathbf{E}_2^\top)) \in \mathbb{R}^{N\times N}. \tag{8}$$

4.2. Spatial-Temporal Kriging

In this section, we describe how to capture the spatial-temporal dependence of traffic data and interpolate the locations where no sensors are deployed in the inductive setting. The kriging task is performed by the following subgraph signals initialization and stacked kriging blocks.

4.2.1. Subgraph Signals Initialization

The aim of subgraph signals initialization is to preliminarily fill in the blank elements without sensors. The first

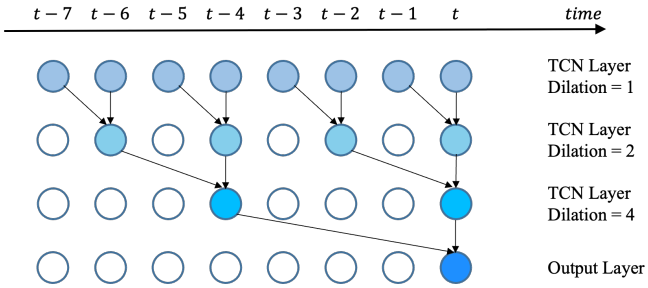


Fig. 6: An example of dilated casual convolution with kernel size 2.

step of the subgraph signals initialization is to randomly select N^s nodes from G with their corresponding signals $\mathbf{X}_p^s \in \mathbb{R}^{N^s \times C}$ at each time step, where $p \in \{t-T, \cdots, t\}$, to generate a sampling subgraph $G^s = (V^s, \mathbf{A}^s)$. $|V^s| = N^s = N_o^s + N_u^s$, where N_o^s and N_u^s are the number of known nodes and unknown nodes, respectively. In reality, unknown nodes V_u^s cannot produce signals since there are no deployed sensors, so we set their signals as $\mathbf{0}$:

$$\mathbf{X}_p^s[i,:] = \begin{cases} \mathbf{X}_p^s[i,:], & \text{if } i \in V_o^s, \\ \mathbf{0}, & \text{else.} \end{cases} \tag{9}$$

Next, we stack three DGCN layers to initialize the signals of unknown nodes in the spatial dimension. This process is essential to ensure that unknown nodes can capture temporal dependencies even when their data are entirely missing. Given G^s and signals $\mathbf{X}_p^s$, the output of DGCN at time step p can be calculated by:

$$\mathbf{X}_p^s = \Theta_{G^s}(\mathbf{X}_p^s, \mathbf{A}^s). \tag{10}$$

The DGCN in subgraph signals initialization only preliminarily interpolates nodes' signals but cannot capture temporal dependency. Therefore, we introduce multiple kriging blocks.

4.2.2. Kriging Block

The aim of the kriging block is to capture the spatial-temporal dependence of traffic data. As shown in Figure 4(b), each kriging block consists of temporal convolutions and inductive spatial graph convolutions.

Temporal Convolutions. To capture the temporal dependency, RNN-based methods and its variants (GRU and LSTM) have been widely used [48, 49]. However, RNN-based methods suffer from the vanishing gradient problem, especially when dealing with long time series. In addition, it also brings expensive computational costs due to the recursive calculation manner. To avoid the problems mentioned above, we use dilated causal convolution [50], a special kind of temporal convolution layer (TCN), to model the temporal dependence of the sequence. As shown in Figure 6, dilated causal convolution can process different time slices in parallel and alleviate the gradient explosion problem. Given a 1-D time sequence $X \in \mathbb{R}^T$ with a length of T and a filter $\varphi \in \mathbb{R}^{K_\varphi}$, the dilated causal convolution operation on the sequence X using filter φ at the p-th time slice can be defined as follows:

$$X \star \varphi(p) = \sum_{s=0}^{K_\varphi - 1} \varphi(s) X(p - d \times s), \tag{11}$$

where d is the dilation factor of the dilated causal convolution. By adjusting the size of d, the receptive field can be easily expanded. Then we use GLU as the activation function [50]. Therefore, given the input $\mathbf{X}^{(l-1)} \in \mathbb{R}^{N^s \times T_{in} \times F}$ (*i.e.*, the output of the $l-1$ kriging block), the dilated causal convolution takes the form:

$$\mathbf{H}^l = \tanh(\mathbf{X}^{(l-1)} \star \Theta_1 + \mathbf{b}) \odot \text{sigmiod}(\mathbf{X}^{(l-1)} \star \Theta_2 + \mathbf{c}), \tag{12}$$

where $\mathbf{H}^l \in \mathbb{R}^{N^s \times T_{out} \times F}$, T_{in} is the time dimension of the input of TCN, T_{out} is the output dimension, $\odot$ is the element-wise product, Θ_1, Θ_2, $\mathbf{b}$ and $\mathbf{c}$ are learnable parameters.

Inductive Spatial Graph Convolutions. In addition to the temporal dependence, traffic data also exhibits strong spatial dependence between nodes. Here we use DGCN [25] to capture the dependence in the spatial dimension. Given node representations $\mathbf{H}^l$ (*i.e.*, the output of the TCN layer in the l-th kriging block), the adjacency matrix $\mathbf{A}^s$, our spatial graph convolution in the l-th kriging block is as follows,

$$\mathbf{X}^l = \Theta_{G^s}(\mathbf{H}^l, \mathbf{A}^s). \tag{13}$$

However, the predefined adjacency matrix $\mathbf{A}^s$ is constructed according to the expert experience (topology-based or distance-based) and cannot fully reflect the complex correlations between nodes. Therefore, we combine the learned adaptive $\mathbf{A}_{\text{krig}}$ and $\mathbf{A}^s$ together to define inductive spatial graph convolution as follows:

$$\mathbf{X}^l = \sum_{k=0}^{K-1} (\mathbf{P}_f^s)^k \mathbf{H}^l \mathbf{W}_{k_1}^l + (\mathbf{P}_b^s)^k \mathbf{H}^l \mathbf{W}_{k_2}^l + (\mathbf{A}_{\text{krig}})^k \mathbf{H}^l \mathbf{W}_{k_3}^l. \tag{14}$$

We use the residual connection and batch normalization for each kriging block to obtain the final interpolation results $\widetilde{\mathbf{X}}_t^s \in \mathbb{R}^{N^s \times C}$.

We choose *Mean Absolute Errors* (MAE) between $\widetilde{\mathbf{X}}_t^s$ and the ground-truth $\mathbf{X}_t^s$ as the loss function of the kriging task:

$$\mathcal{L}_1 = \frac{1}{N_o^s} |\mathbf{X}_t^s - \widetilde{\mathbf{X}}_t^s| \mathbb{1}\{V_o^s\}, \tag{15}$$

where $\mathbb{1}\{\cdot\}$ is an indicator function used to force the loss to only compute errors for known nodes. $\mathbb{1}\{i\} = 1$ if node i is known, otherwise $\mathbb{1}\{i\} = 0$.

4.3. Prediction-based Constraint Task

In the above kriging task, a random sampling strategy is used to make the model adapt to the different graph

structures. However, the sampled subgraph only reflects the spatial-temporal dependency of N^s known nodes in each sample and cannot simultaneously capture the relation among all N deployed sensors. For example, when interpolating node B in Figure 2, the sampled subgraph G_1^s only captures the information from nodes C and G, while it cannot be aware of nodes A and H. Thus, interpolating node B in the subgraph will get a suboptimal result. To alleviate the shortcomings caused by random sampling, we introduce a constraint task, which is a prediction-based task. It takes the historical T time slices of all N deployed sensors as input to predict the signals of the current step, and the map function of the constraint task is as follows:

$$[\mathbf{X}_{t-T}, \dots, \mathbf{X}_{t-1}, G] \overset{g(\cdot)}{\rightarrow} \widehat{\mathbf{X}}_t, \tag{16}$$

where $\widehat{\mathbf{X}}_t \in \mathbb{R}^{N \times C}$ is the predictions for nodes, and $g(\cdot)$ is a mapping function for prediction.

The constraint task can utilize the node information that the subgraph cannot receive to alleviate the information loss problem caused by random sampling. Here we regard N sensors as known nodes, *i.e.,* we have all N sensor information in the constraint task. After feature transformation, we send the node features to the stacked constraint blocks to capture the complex spatial-temporal dependency. Each constraint block consists of a TCN and an adaptive spatial graph convolution module, the same structure as the kriging block, as shown in Figure 4(b). Therefore, the corresponding calculation formula of the constraint task can be represented by Eq. (12) and Eq. (14). However, since the number of nodes for the constraint task and the kriging task is different (N^s in the kriging task and N in the constraint task), Eq. (14) needs to be replaced as:

$$\mathbf{X}_c^l = \sum_{k=0}^{K-1} (\mathbf{P}_f)^k \mathbf{H}_c^l \mathbf{W}_{k_1}^l + (\mathbf{P}_b)^k \mathbf{H}_c^l \mathbf{W}_{k_2}^l + (\mathbf{A}_{\text{cons}})^k \mathbf{H}_c^l \mathbf{W}_{k_3}^l, \tag{17}$$

where $\mathbf{H}_c^l$ denotes the output of the TCN layer in the l-th constraint block, $\mathbf{P}_f = \mathbf{A}/\text{rowsum}(\mathbf{A})$ and $\mathbf{P}_b = \mathbf{A}^\top/\text{rowsum}(\mathbf{A}^\top)$. After stacked constraint blocks, the prediction results $\widehat{\mathbf{X}}_t \in \mathbb{R}^{N \times C}$ of the constraint task are obtained through the output layer. We choose MAE as the loss function of the constraint task:

$$\mathcal{L}_2 = \frac{1}{N} |\mathbf{X}_t - \widehat{\mathbf{X}}_t|. \tag{18}$$

Because of the same structure in the constraint block and the kriging block, we use the parameter sharing strategy to reduce the number of parameters, *i.e.*, the two tasks use the same parameters in GCN and TCN.

Reviewing our kriging task and constraint task again, we find that although their goals are different, their outputs reflect the node's signal at the current time step from two perspectives. Therefore, the prediction results $\widehat{\mathbf{X}}_t^s \in \mathbb{R}^{N^s \times C}$ of the corresponding N^s node and the interpolation results $\widetilde{\mathbf{X}}_t^s$ should be as close as possible. Here, we have a constraint loss:

$$\mathcal{L}_3 = \frac{1}{N^s} |\widetilde{\mathbf{X}}_t^s - \widehat{\mathbf{X}}_t^s|. \tag{19}$$

Finally, we define the overall loss function as follows, which is a weighted sum of the three loss functions:

$$\mathcal{L} = \mathcal{L}_1 + \lambda(\mathcal{L}_2 + \mathcal{L}_3), \tag{20}$$

where λ is a hyperparameter to balance the kriging task and the constraint task.

Table 1
Details of Datasets

Dataset Type	Dataset	Nodes	TimeSteps	Interval
Traffic Speed	METR-LA	207	34,272	5 minutes
Traffic Speed	PEMS-BAY	325	52,116	5 minutes
Traffic Speed	PEMSD7-M	228	12,672	5 minutes
Air Quality Index	BJ-AIR	35	8,707	1 hour

5. Experiments

We validate our proposed IAGCN on two categories of real-world spatial-temporal datasets. One is about traffic speed, consisting of three datasets. The other one is an air quality index dataset in Beijing city, which is chosen to verify the generalization ability of IAGCN in other types of spatial-temporal datasets. Their statistical information is summarized in Table 1.

5.1. Dataset

- **METR-LA**[2]**:** records four months of traffic speed data on 207 loop detectors in Los Angeles County;
- **PEMS-BAY**[2]**:** a traffic speed dataset is collected from 325 sensors in the California Transportation Agencies (CalTrans) Performance Measurement System (PEMS). It contains six months of speed data in the Bay Area.
- **PEMSD7-M:** the traffic speed dataset is also collected by CalTrans. It contains two months of speed data from 228 sensors.

In addition to interpolating the traffic data, it makes sense to interpolate air quality data since air monitoring stations are sparsely deployed in the city due to expensive costs, *e.g.*, there are only 35 air monitoring stations in Beijing.

- **BJ-AIR**[3]**:** air quality index dataset collected by 35 air quality stations in Beijing City, and here we only consider the PM2.5 values.

[2] https://github.com/liyaguang/DCRNN
[3] https://quotsoft.net/air

We compute the pairwise road network distance or geospatial distance between sensors and build the adjacency matrix using a thresholded Gaussian kernel, which is defined by $\mathbf{A}_{i,j} = \exp(-(\frac{\text{dist}(v_i,v_j)}{\sigma})^2)$, where $\text{dist}(v_i, v_j)$ denotes the distance between sensors v_i and v_j, and σ is the standard deviation. Z-score normalization is adopted to standardize the data inputs.

5.2. Baselines

We compare our IAGCN with the following models:

- **KNN**: K-nearest neighbors, which interpolates signals of unknown sensors by averaging the K nearest sensors in the network;
- **IDW**: Inverse distance weighting calculates the weights of sensors based on distance and interpolates unknown sensors by weighting the signals of known sensors.
- **OKriging** [11]: Ordinary kriging, a well-developed spatial interpolation method based on Gaussian regression, constructs the variogram according to the distance;
- **BGCP** [14]: Bayesian Gaussian CANDECOMP / PARAFAC tensor decomposition, which regards spatial-temporal data as a three-dimensional tensor and uses tensor decomposition to deal with missing values.
- **KCN** [19]: Kriging convolutional network, which establishes the connection between kriging and deep learning. It first uses the nearest K neighbors to build a subgraph and then adopts the graph neural network to perform the signal interpolation.
- **IGNNK** [20]: Inductive Graph Neural Network for kriging, which first uses the sampling machine to satisfy the inductive setting, and then utilizes the dynamic graph convolution to capture the spatial correlation.
- **DualSTN** [41]: Dual Joint SpatioTemporal Network considers both long- and short-term temporal for inference, which joints the spatiotemporal graph attention network to learn short-term spatial-temporal correlation, and uses graph recurrent network to model the long-term dependencies.
- **INCREASE** [24]: Inductive Graph Representation Learning for Spatio-Temporal Kriging, which models heterogeneous spatial relations and diverse temporal patterns for interpolating the signals of unseen nodes.

5.3. Detailed Settings of Baselines

For the traditional and tensor-based spatial-temporal interpolation methods (including KNN, IDW, OKriging, and BGCP), we list all of the parameter settings that they need. For deep learning-based methods (including KCN, IGNNK, DualSTN, and INCREASE), we keep the parameters reported in their official code. The detailed parameter settings of each baseline are as follows:

- **KNN**: There are no parameters to learn. We count the average of each node's one-hop neighbors in the predefined graph as a result.
- **IDW**: We first calculate the spatial distance of the neighbor nodes in the predefined graph, and we set the distance weight to -2.
- **OKriging**: We implement OKriging with the autoKriging function in the R package automap. In this manner, we set the variogram kernels as "Gaussian" and automatically learn their parameters.
- **BGCP**: It is a tensor decomposition-based approach for interpolation. We set the maximum number of iterations to 1000 and set the rank $r = 50$. Other hyperparameter settings are the same as BGCP.
- **KCN**: We implement KCN using KCN-Sage based on GraphSage as it achieves the best performance in KCN. The KCN is trained using the Adam optimizer and a learning rate of 0.001 for 100 epochs with a batch size of 32. The other training parameters have the same hyperparameter settings as the KCN.
- **IGNNK**: IGNNK is trained with three stacked GNN layers, and the hidden state dimension of each layer is [100,100,12]. We use the Adam optimizer and a learning rate of 0.001 for 100 epochs with a batch size of 32 to train IGNNK.
- **DualSTN**: We implement DualSTN by 3 JST-GAT layers to capture short-term dependence and set the skip step $t_k = 4$ of GRU to capture long-term dependence. The hidden state of the GRU is set as 16 and the historical time window $T = 25$. We use the Adam optimizer to train the DualSTN with a batch size of 32 and a learning rate of 0.001.
- **INCREASE**: We implement INCREASE with the time window 12, the dimension of the hidden state $D = 64$, and ReLU is selected as the nonlinear activation function. INCREASE is trained using the Adam optimizer for 100 epochs with a learning rate of 0.001 and a batch size of 32.

5.4. Detailed Settings of IAGCN

We implement the IAGCN[4] model based on the PyTorch[5] framework and split each dataset at a ratio of 6 : 2 : 2 into training sets, validation sets, and test sets by the time. For the sensor division, we randomly select 75% of the sensors for training and hold out all sensors for testing. For the input, we set $T = 12$, which means inputting the signals

[4] https://github.com/wtl52656/IAGCN

[5] https://pytorch.org

Table 2
Performance comparison of IAGCN against other baseline models. The best results are marked in bold, the second place results are underlined, and 'NA' means the model is not available. Δ denotes the percentage of the relative improvement in performance between IAGCN and the best baseline.

Method	METR-LA			PEMS-BAY			PEMSD7-M			BJ-AIR		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
KNN	7.23	11.11	20.00%	5.10	8.89	11.81%	11.04	14.30	27.92%	34.15	58.35	50.11%
IDW	8.27	13.74	22.77%	5.85	10.93	13.72%	9.74	13.30	25.36%	33.10	56.95	48.56%
OKriging	7.85	12.79	17.00%	NA	NA	NA	NA	NA	NA	32.07	55.90	47.36%
BGCP	9.16	12.36	24.51%	4.82	8.04	11.69%	9.17	12.89	24.41%	37.00	61.76	55.25%
KCN	6.15	9.71	18.71%	3.82	6.74	9.37%	7.83	12.47	25.04%	31.70	54.66	51.91%
IGNNK	6.21	9.25	16.19%	3.70	6.44	8.60%	8.13	12.23	24.53%	29.90	52.16	47.41%
DualSTN	6.48	9.82	17.62%	3.78	6.41	8.54%	7.88	11.95	24.47%	30.61	52.77	47.68%
INCREASE	5.40	8.65	14.66%	3.78	6.35	7.92%	7.98	12.11	23.52%	29.02	51.34	45.64%
IAGCN	**5.16**	**8.37**	**13.83%**	**3.50**	**6.19**	**7.66%**	**7.35**	**11.73**	**21.05%**	**27.93**	**49.40**	**41.41%**
Δ	**+4.44%**	**+3.24%**	**+4.66%**	**+5.41%**	**+2.52%**	**+3.28%**	**+6.13%**	**+1.84%**	**+10.50%**	**+3.76%**	**+3.78%**	**+9.27%**

of the historical 12 steps and the current step for the kriging task and the historical 12 steps' signals for the constraint task.

In the testing phase, for the kriging task, we replace the sample subgraph with the test graph. However, the constraint task cannot work because the unknown nodes do not have historical information in the test. Since our constraint task aims to alleviate the shortcomings of random sampling, the trained model has achieved our goal. Therefore, we do not use the constraint task when testing.

For parameter settings, we stack 8 kriging blocks and 8 constraint blocks with a sequence of dilation factors [1, 2, 1, 2, 1, 2, 1, 2] to cover the input sequence length. The model dimension F is set to 32, and we set the diffusion step $K = 2$ following [26]. The randomly initialized node representation dimension is 32, and the parameter λ is set to 0.1. We train our model using the Adam [59] optimizer with a learning rate of 0.001 and 100 epochs, and the best model is determined by the performance on the validation sets.

5.5. Evaluation Metrics

We apply three widely used metrics to evaluate the performance, *i.e.*, Mean Absolute Errors (**MAE**), Root Mean Squared Errors (**RMSE**), and Mean Absolute Percentage Errors (**MAPE**), which are defined as:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{\mathbf{x}}_i - \mathbf{x}_i| \tag{21}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{x}}_i - \mathbf{x}_i)^2} \tag{22}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{\mathbf{x}}_i - \mathbf{x}_i}{\mathbf{x}_i}\right| \tag{23}$$

where $\hat{\mathbf{x}}_i$ is the interpolation result of node i and $\mathbf{x}_i$ is the ground truth. N is the total number of samples.

5.6. Experimental Results

5.6.1. Overall Performance

Table 2 shows the comparison of our IAGCN and the baseline methods on four datasets. We observe that our IAGCN obtains superior results among all metrics. For speed datasets PEMS-BAY and PEMSD7-M, the OKriging model does not work since the geographical location between sensors is not provided.

KNN and IDW consider only linear relationships between sensors, which limits their ability to capture complex spatial dynamics, resulting in poor performance. Although Okriging constructs a kernel function based on the spatial geographical location between sensors to model the spatial dependence, it cannot capture the complex dynamics of spatial-temporal data. Therefore, its performance improvements on most metrics are limited.

BGCP is designed for scenarios with missing data, which differs from the kriging scenario, resulting in unsatisfactory performance on most datasets due to the unavailability of historical signals for unknown nodes. For deep learning methods, we find that KCN reports unsatisfactory performance in most metrics since it focuses on modeling the complex spatial dependence of sensors and ignores the temporal dynamics. Compared to KCN, IGNNK simultaneously utilizes traffic data over multiple time slices to perform the kriging task and achieves better performance. However, it cannot model the temporal dependence of the nodes. Although DualSTN considers both long- and short-term temporal correlation, it needs the sampling subgraph to fit the inductive setting, which causes information loss and reduces its performance. INCREASE further considers the temporal and heterogeneous spatial correlation, showing the best performance compared with other baselines. However, it is essentially based on the predefined adjacency matrix, which is inflexible and cannot fully describe the complex spatial correlation. In contrast, our IAGCN achieves start-of-the-art results in all of the metrics. IAGCN further improves the performance by 3.74% $\sim$ 6.59% on different datasets. The underlying reasons for improvement lie in

Table 3
The performance during different intervals in the METR-LA and PEMS-BAY datasets.

Intervals type	Method	METR-LA			PEMS-BAY		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE
Difficult Intervals	KCN	7.29	11.36	26.12%	7.15	11.68	23.83%
	IGNNK	7.17	10.55	21.71%	6.87	10.76	20.48%
	DualSTN	7.23	10.93	23.18%	6.90	10.53	18.94%
	INCREASE	6.26	10.43	17.86%	6.76	10.35	17.92%
	IAGCN	**5.72**	**9.58**	**16.00%**	**6.60**	**10.17**	**17.29%**
Simple Intervals	KCN	6.03	9.41	17.88%	2.34	2.91	3.51%
	IGNNK	6.14	9.06	15.41%	2.14	2.79	3.26%
	DualSTN	5.93	9.25	16.38%	2.25	2.76	3.19%
	INCREASE	5.31	8.47	12.84%	2.20	2.80	3.34%
	IAGCN	**5.05**	**8.29**	**12.34%**	**2.01**	**2.65**	**3.06%**

Table 4
The performance during different intervals in the PEMSD7-M and BJ-AIR datasets.

Intervals type	Method	PEMSD7-M			BJ-AIR		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE
Difficult Intervals	KCN	16.08	21.24	64.04%	64.04	91.28	59.81%
	IGNNK	15.84	20.07	63.57%	62.23	90.20	57.61%
	DualSTN	15.98	19.52	59.87%	63.76	92.19	59.02%
	INCREASE	16.11	19.17	55.50%	60.99	89.12	56.79%
	IAGCN	**14.98**	**18.88**	**50.87%**	**57.56**	**87.42**	**52.49%**
Simple Intervals	KCN	3.26	4.30	4.93%	10.41	15.60	43.75%
	IGNNK	2.84	3.89	4.32%	9.79	15.31	41.47%
	DualSTN	2.75	3.72	4.29%	9.72	15.68	42.45%
	INCREASE	2.80	3.91	4.22%	9.58	15.08	41.33%
	IAGCN	**2.62**	**3.52**	**3.97%**	**9.05**	**14.99**	**39.30%**

two aspects. First, IAGCN can capture the complex spatial correlations between both the deployed and undeployed sensors by learning the adaptive adjacency matrix. Second, it merges a predictive-based constraint task to alleviate the shortcomings of the random sampling strategy.

5.6.2. Performance During Difficult and Simple Intervals

Almost all of the existing studies on spatial-temporal kriging use average accuracy to evaluate performance, which is insufficient to reveal the model's strengths and weaknesses across different intervals. For example, the violent speed fluctuations during the morning and evening peak hours make the kriging task even harder. To explore IAGCN performance under different intervals, we divide the test set into a set of temporal intervals and calculate their variance. The variance in the top 25% is called difficult intervals, and the variance in the bottom 25% is called simple intervals.

Tables 3 and 4 show the experimental results for different kinds of intervals in four datasets. Overall, our proposed IAGCN achieves the best performance whether in difficult intervals or simple intervals. This is mainly because our model can capture the temporal and the complex spatial correlation between sensors adaptively. While KCN and IGNNK ignore the importance of temporal dependence for spatial-temporal kriging, they cannot achieve satisfactory performance. DualSTN and INCREASE consider the spatial-temporal dependence between sensors, but DualSTN has the problem of information loss, and INCREASE is based on the predefined graph, which is inflexible and cannot fully reflect the underlying spatial correlation. For each model, we observe that the performance declines significantly on the difficult intervals against sample intervals, which is reasonable because the temporal dependence of the data becomes more complex at that time. Therefore, we cannot ignore the effect of temporal dependence in the spatial-temporal kriging task.

5.7. Ablation Study

We design three variants to evaluate the effects of different components in IAGCN, including:

- **IAGCN-AC**: deletes the adaptive graph constructor and constraint task.
- **IAGCN-A**: closes the adaptive graph constructor.
- **IAGCN-C**: deletes the constraint task.

Figure 7 shows the comparison of those three variants and the IAGCN model on the METR-LA and BJ-AIR datasets

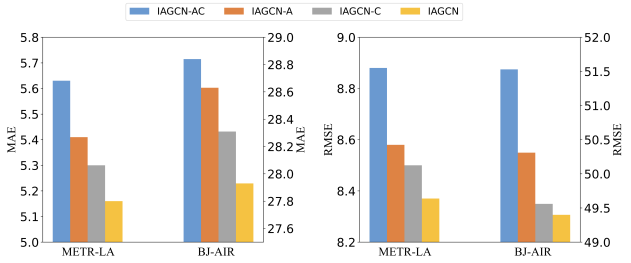


Fig. 7: Component analysis of IAGCN on MAE (left) and RMSE (right).

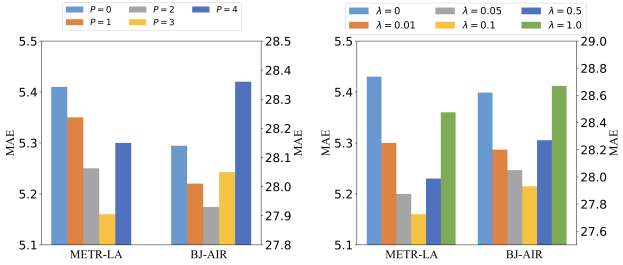


Fig. 8: Hyperparameter study on the METR-LA and BJ-AIR datasets. Left: the DGCN layers P in $\mathbf{A}_{\text{krig}}$. Right: the weight of the constraint task.

in RMSE and MAE metrics. We observed that the performances of IAGCN-A and IAGCN-C are better than IAGCN-AC but worse than IAGCN, indicating that the adaptive graph constructor and the constraint task are beneficial for spatial-temporal kriging. We also notice that introducing these two components has a higher average performance gain on the traffic dataset (8.35% for METR-LA and 3.16% for BJ-AIR) due to the complex spatial dependencies between sensors in the traffic system.

5.8. Hyperparameter Study

To further investigate the effect of different hyperparameters, we conduct experiments on the METR-LA and BJ-AIR datasets.

Effects of the number of DGCN layers P in $\mathbf{A}_{krig}$. We adjust the hyperparameter P for learning $\mathbf{A}_{\text{krig}}$ from 0 to 4. $P = 0$ means that $\mathbf{A}_{\text{krig}}$ is not used. As shown in Figure 8 (left), for the METR-LA dataset, the performance improves as P increases. When $P = 3$, IAGCN reaches the best results. However, the model performance will decrease if the number continues to increase. We think it may be caused by over-smoothing, leading to the embeddings of nodes being similar and cannot well capture the spatial dependence between nodes. The BJ-AIR dataset has the same phenomenon, but it has a small number of sensors. We get optimal performance when the number of DGCN layers is 2.

Effects of the weight of constraint task λ. We set $\lambda = \{0, 0.01, 0.05, 0.1, 0.5, 1\}$. The experimental results are shown in Figure 8 (right). On the METR-LA and BJ-AIR datasets, when λ is 0.1, IAGCN achieves the best results. When λ is smaller or larger, the performance of the IAGCN becomes worse. We think that when λ is too small, the model mainly focuses on interpolation and ignores the problem of random sampling. When λ is large, the model may focus on the constraint task.

5.9. Different Unknown Nodes Study

Different unknown nodes will make different results in the kriging problem since the signal value of each node is different. In this section, we study the performance of IAGCN regarding different unknown node sets. We select 5 different sets of unknown nodes by setting random seed = {1, 2, 3, 4, 5} on the BJ-AIR dataset.

Table 5
Performance comparison of IAGCN and IGNNK regarding different unknown nodes.

Seed	MAE			RMSE		
	IGNNK	INCREASE	**IAGCN**	IGNNK	INCREASE	**IAGCN**
1	31.09	30.41	**29.83**	53.98	53.51	**52.75**
2	26.74	24.39	**23.56**	45.06	44.11	**42.84**
3	21.27	19.46	**18.84**	34.88	33.38	**32.80**
4	25.99	24.03	**23.12**	42.60	40.88	**39.54**
5	18.97	15.57	**15.35**	29.74	27.44	**27.26**

As shown in Table 5, compared with IGNNK and INCREASE, our IAGCN always shows better performance regarding different unknown nodes. We also notice that the performance of three deep learning models is changed when the unknown node set changes. This is because the interpolation difficulty is different for different sets. Some nodes have a single pattern or the advantage of geographic location, making it easy to interpolate, while some nodes are difficult to interpolate.

5.10. Visualization Analysis

To present our model intuitively, we conduct two visualization analysis experiments.

5.10.1. Case Study on Adaptive Graph

In this section, we investigate the neighbor nodes of a given center node in the adaptive graph. Specifically, we first choose a central node 111, and select the top 3 nodes with the largest weights from both the predefined adjacency matrix and adaptive matrix. Then, we visualize their locations and speed curves within 24 hours, as shown in Figure 9. We find that the adaptive adjacency matrix can model long-distance node correlations. For example, nodes 111 and 184 are located at a crossroad, and their speed curves display similar patterns. Therefore, by combining the predefined adjacency matrix and adaptive matrix, IAGCN can simultaneously capture the correlation between neighbors and similar function nodes, improving interpolation performance.

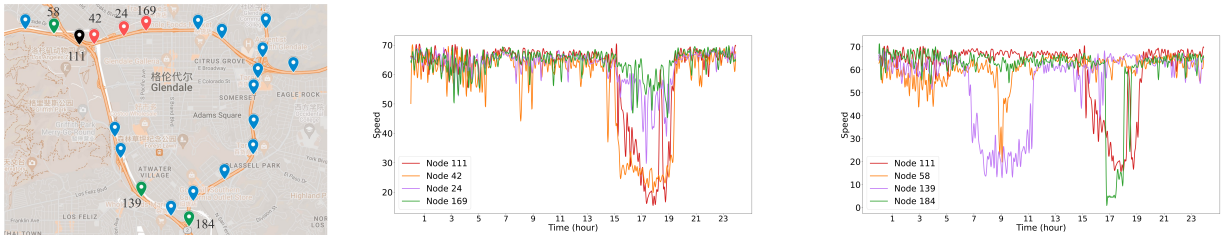


Fig. 9: Left: the location of the center node 111 (black icon), the top 3 neighbors of the center node 111 in the predefined matrix (red icon), and the adaptive matrix (green icon). **Middle:** the speed curve of the top 3 neighbors of node 111 in the predefined matrix. **Right:** the speed curve of the top 3 neighbors of node 111 in the adaptive matrix.

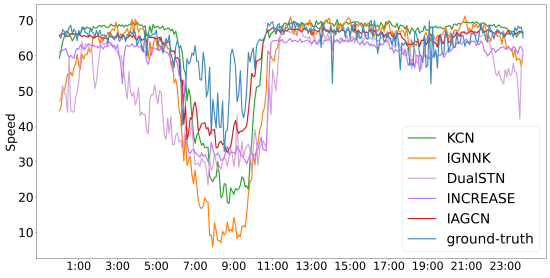


Fig. 10: Visualization result on the unknown node in the METR-LA dataset on June 14th, 2012.

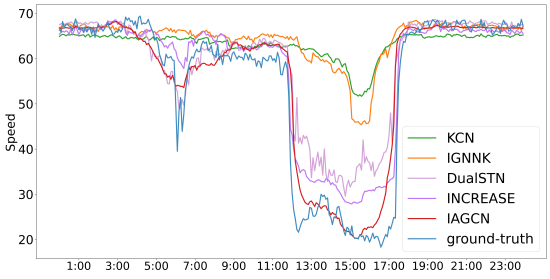


Fig. 11: Visualization result on the unknown node in the PEMS-BAY dataset on June 1st, 2017.

5.10.2. Interpolation Result Visualization

To better illustrate the results between interpolation and ground truth in IAGCN, we randomly select a sensor on the METR-LA and PEMS-BAY datasets. The visualized curve is shown in Figures 10 and 11. The red curve is the interpolated signal of our IAGCN model for the non-deployed sensor. Whether the speed change is smooth or sudden, the interpolation result can accurately track the blue ground truth curve, indicating that our IAGCN can accurately infer the speed of the non-deployed sensor.

5.11. Complexity Analysis

In this section, we analyze the computation and memory complexity of our proposed IAGCN. Specifically, from a theoretical perspective, the computational complexity of

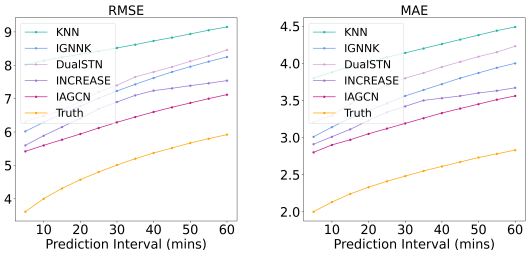


Fig. 12: Performance changes of different filling strategies on the METR-LA dataset as the prediction interval increases.

IAGCN consists of three parts: 1) in each constraint block, the computational complexity of TCN is $O(TK_\varphi F^2)$ and GNN is $O(KN^2F)$; 2) in each kriging block, the computation complexity of TCN is $O(TK_\varphi F^2)$, and GNN is $O(K(N^s)^2F)$; 3) in the Adaptive Graph Constructor, the computational complexity is $O(N^2F + P(N^s)^2F)$. For the memory complexity of IAGCN, each constraint and kriging block is the same. Both are $O(K_\varphi F^2)$ and $O(KF^2)$ in TCN and GNN. In the Adaptive Graph Constructor, the computation complexity is $O(PF^2)$.

From an experimental perspective, the training time of IAGCN grows linearly with the number of data-points. It takes around 0.34s and 3343MB memory per batch, with a batch size of 32 on METR-LA using an NVIDIA RTX A4000 GPU card. In the test stage, the average inference time of IAGCN is 15.19 seconds with a batch size of 32.

5.12. Downstream Task

A good spatial-temporal kriging model not only reflects the state of the whole transportation system in real-time but also can provide data support for downstream tasks. In this section, we evaluate the interpolation results with traffic prediction as the downstream task. Traffic prediction is an important task in intelligent transportation systems and has been extensively studied [26, 27, 28, 29, 31, 32, 33]. We use the historical 1h spatial-temporal data to predict the future 1h traffic signal based on the success of GraphWaveNet [26]. In the downstream task, we use raw signals for the deployed sensors. For non-deployed sensors, we use the following filling strategies:

Table 6
Performance comparison of different filling strategies for flow prediction.

Fill strategy	15 min			30 min			45 min			1 hour		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
KNN	3.96	8.24	11.43%	4.14	8.52	12.38%	4.32	8.83	13.34%	4.49	9.15	14.24%
IGNNK	3.26	6.54	9.11%	3.56	7.23	10.39%	3.80	7.80	11.44%	4.00	8.25	12.27%
DualSTN	3.45	6.65	9.87%	3.80	7.40	10.93%	4.02	7.95	12.03%	4.23	8.46	12.96%
INCREASE	3.11	6.15	8.13%	3.42	6.90	9.89%	3.56	7.31	10.35%	3.67	7.54	10.82%
IAGCN	2.97	5.77	7.93%	3.19	6.29	8.70%	3.39	6.74	9.34%	3.56	7.12	9.88%
Truth	2.24	4.31	5.61%	2.48	5.01	6.51%	2.67	5.52	7.23%	2.83	5.92	7.82%

- **KNN**: The signals of non-deployed sensors are filled by KNN.
- **IGNNK**: The signals of non-deployed sensors are filled by IGNNK.
- **DualSTN**:The signals of non-deployed sensors are filled by DualSTN.
- **INCREASE**: The signals of non-deployed sensors are filled by INCREASE.
- **IAGCN**: The signals of non-deployed sensors are filled by IAGCN.
- **Truth**: use ground-truth for non-deployed sensors.

Note that we only replace the input using each filling strategy, but the label is the ground truth. Each filling strategy has the same settings.

We perform the downstream task with different filling strategies in the METR-LA dataset. Table 6 shows the comparison of different filling strategies for 15 minutes, 30 minutes, 45 minutes, and 1 hour ahead of forecasting. The RMSE and MAE metrics predicted every 5 minutes are shown in Figure 12. Overall, as the prediction interval is longer, the difficulty increases, so all filling strategies' prediction performance also decreases. We observe that none of the filling strategies are better than Truth Filling. This phenomenon is reasonable because there are some errors between the interpolation and real data. Furthermore, IAGCN Filling reports better results than other filling strategies, which suggests that our proposed IAGCN can provide higher-quality traffic data for the downstream task.

6. Conclusions

In this paper, a novel deep learning model IAGCN is proposed and successfully solves the spatial-temporal traffic kriging problem. IAGCN consists of three modules: the adaptive graph constructor, the kriging task, and the constraint task. In the adaptive graph constructor, adaptive matrices are generated for the kriging task and the constraint task to capture the spatial dependence that the predefined graph structure cannot capture well. In the kriging task, we capture the spatial-temporal dependence of the traffic data and use the random sampling strategy to make the model have the inductive ability. In the constraint task, we input all the signals from the deployed sensor to alleviate the shortcomings of random sampling in inductive learning. Experiments on four real-world datasets demonstrate that IAGCN is superior to the state-of-the-art baselines.

Although our IAGCN achieves the best performance, there are two limitations of our work that need to be further explored. The first is that we do not consider the impact of external factors on data interpolation (e.g., accidents and weather). The second is that if only a small number of sensors are deployed, or all of the surrounding sensors are masked, it is difficult to interpolate the value of the center sensors. In future work, we plan to extend our model in the above aspects.

Acknowledgement

This work was supported by the Natural Science Foundation of China (No. 62272033).

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 12 (4) (2011) 1624–1639.
[2] Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: concepts, methodologies, and applications, ACM Transactions on Intelligent Systems and Technology (TIST) 5 (3) (2014) 1–55.
[3] L. Zhu, F. R. Yu, Y. Wang, B. Ning, T. Tang, Big data analytics in intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 20 (1) (2018) 383–398.
[4] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, Z. Jia, Freeway performance measurement system: mining loop detector data, Transportation Research Record 1748 (1) (2001) 96–102.
[5] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, S. Lyu, Ua-detrac: A new benchmark and protocol for multi-object detection and tracking, Computer Vision and Image Understanding 193 (2020) 102907.
[6] T. M. Rajeh, T. Li, C. Li, M. H. Javed, Z. Luo, F. Alhaek, Modeling multi-regional temporal correlation with gated recurrent unit and multiple linear regression for urban traffic flow prediction, Knowledge-Based Systems 262 (2023) 110237.
[7] T. Afrin, N. Yodo, A long short-term memory-based correlated traffic data prediction framework, Knowledge-Based Systems 237 (2022) 107755.
[8] A. Kind, M. P. Stoecklin, X. Dimitropoulos, Histogram-based traffic anomaly detection, IEEE Transactions on Network and Service Management 6 (2) (2009) 110–121.
[9] J. Lan, C. Long, R. C.-W. Wong, Y. Chen, Y. Fu, D. Guo, S. Liu, Y. Ge, Y. Zhou, J. Li, A new framework for traffic anomaly detection, in: Proceedings of the 2014 SIAM International Conference on DATA MINING, SIAM, 2014, pp. 875–883.

[10] H. J. Miller, Tobler's first law and spatial analysis, Annals of the association of American geographers 94 (2) (2004) 284–289.
[11] N. Cressie, C. K. Wikle, Statistics for spatio-temporal data, John Wiley & Sons, 2015.
[12] M. T. Bahadori, Q. R. Yu, Y. Liu, Fast multivariate spatio-temporal analysis via low rank tensor learning, Advances in neural information processing systems 27 (2014).
[13] K. Takeuchi, H. Kashima, N. Ueda, Autoregressive tensor factorization for spatio-temporal predictions, in: 2017 IEEE international conference on data mining (ICDM), IEEE, 2017, pp. 1105–1110.
[14] X. Chen, Z. He, L. Sun, A bayesian tensor decomposition approach for spatiotemporal traffic data imputation, Transportation research part C: emerging technologies 98 (2019) 73–84.
[15] X. Jia, X. Dong, M. Chen, X. Yu, Missing data imputation for traffic congestion data based on joint matrix factorization, Knowledge-Based Systems 225 (2021) 107114.
[16] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).
[17] R. A. Rossi, R. Zhou, N. K. Ahmed, Deep inductive graph representation learning, IEEE Transactions on Knowledge and Data Engineering 32 (3) (2018) 438–452.
[18] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, Graphsaint: Graph sampling based inductive learning method, in: International Conference on Learning Representations, 2019.
[19] G. Appleby, L. Liu, L.-P. Liu, Kriging convolutional networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3187–3194.
[20] Y. Wu, D. Zhuang, A. Labbe, L. Sun, Inductive graph neural networks for spatiotemporal kriging, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 4478–4485.
[21] W. Liang, Y. Li, K. Xie, D. Zhang, K.-C. Li, A. Souri, K. Li, Spatial-temporal aware inductive graph neural network for c-its data recovery, IEEE Transactions on Intelligent Transportation Systems (2022).
[22] X. Zhang, R. R. Chowdhury, J. Shang, R. Gupta, D. Hong, Esc-gan: Extending spatial coverage of physical sensors, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1347–1356.
[23] Y. Wu, D. Zhuang, M. Lei, A. Labbe, L. Sun, Spatial aggregation and temporal convolution networks for real-time kriging, arXiv preprint arXiv:2109.12144 (2021).
[24] C. Zheng, X. Fan, C. Wang, J. Qi, C. Chen, L. Chen, Increase: Inductive graph representation learning for spatio-temporal kriging, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 673–683.
[25] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, 2018.
[26] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 1907–1913.
[27] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, Advances in neural information processing systems 33 (2020) 17804–17815.
[28] M. Li, Z. Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 4189–4196.
[29] Z. Fang, Q. Long, G. Song, K. Xie, Spatial-temporal graph ode networks for traffic flow forecasting, in: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 364–373.
[30] D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series., in: KDD workshop, Vol. 10, Seattle, WA, USA:, 1994, pp. 359–370.
[31] S. Guo, Y. Lin, H. Wan, X. Li, G. Cong, Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting, IEEE Transactions on Knowledge and Data Engineering 34 (11) (2021) 5415–5428.
[32] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 33, 2019, pp. 922–929.
[33] C. Zheng, X. Fan, C. Wang, J. Qi, Gman: A graph multi-attention network for traffic prediction, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 34, 2020, pp. 1234–1241.
[34] B. Tugrul, H. Polat, Privacy-preserving kriging interpolation on partitioned data, Knowledge-Based Systems 62 (2014) 38–46.
[35] J. P. Kleijnen, Kriging metamodeling in simulation: A review, European journal of operational research 192 (3) (2009) 707–716.
[36] M. A. Oliver, R. Webster, Kriging: a method of interpolation for geographical information systems, International Journal of Geographical Information System 4 (3) (1990) 313–332.
[37] M. Seeger, Gaussian processes for machine learning, International journal of neural systems 14 (02) (2004) 69–106.
[38] X. Kong, W. Zhou, G. Shen, W. Zhang, N. Liu, Y. Yang, Dynamic graph convolutional recurrent imputation network for spatiotemporal traffic missing data, Knowledge-Based Systems 261 (2023) 110188.
[39] L. Huang, J. Huang, H. Li, J. Cui, Multi-dimensional spatial–temporal graph convolution for urban sensors imputation and enhancement, Knowledge-Based Systems 278 (2023) 110856.
[40] L. Huang, J. Huang, H. Li, J. Cui, Robust spatial temporal imputation based on spatio-temporal generative adversarial nets, Knowledge-Based Systems 279 (2023) 110919.
[41] J. Hu, Y. Liang, Z. Fan, L. Liu, Y. Yin, R. Zimmermann, Decoupling long-and short-term patterns in spatiotemporal inference, IEEE Transactions on Neural Networks and Learning Systems (2023).
[42] L. Li, B. Du, Y. Wang, L. Qin, H. Tan, Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model, Knowledge-Based Systems 194 (2020) 105592.
[43] R. He, Y. Liu, Y. Xiao, X. Lu, S. Zhang, Deep spatio-temporal 3d densenet with multiscale convlstm-resnet network for citywide traffic flow forecasting, Knowledge-Based Systems 250 (2022) 109054.
[44] R. Yao, G. Lin, S. Xia, J. Zhao, Y. Zhou, Video object segmentation and tracking: A survey, ACM Transactions on Intelligent Systems and Technology (TIST) 11 (4) (2020) 1–47.
[45] D. Yuan, X. Chang, Q. Liu, Y. Yang, D. Wang, M. Shu, Z. He, G. Shi, Active learning for deep visual tracking, IEEE Transactions on Neural Networks and Learning Systems (2023).
[46] D. Yuan, X. Chang, Z. Li, Z. He, Learning adaptive spatial-temporal context-aware correlation filters for uav tracking, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 18 (3) (2022) 1–18.
[47] D. Yuan, X. Shu, Q. Liu, Z. He, Aligned spatial-temporal memory network for thermal infrared target tracking, IEEE Transactions on Circuits and Systems II: Express Briefs 70 (3) (2022) 1224–1228.
[48] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).
[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
[50] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv preprint arXiv:1511.07122 (2015).
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[52] Y. Seo, M. Defferrard, P. Vandergheynst, X. Bresson, Structured sequence modeling with graph convolutional recurrent networks, in: Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25, Springer, 2018, pp. 362–373.
[53] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 34, 2020, pp. 3529–3536.

[54] Y. Qu, J. Rong, Z. Li, K. Chen, St-a-pgcl: Spatiotemporal adaptive periodical graph contrastive learning for traffic prediction under real scenarios, Knowledge-Based Systems 272 (2023) 110591.
[55] X. Ouyang, Y. Yang, Y. Zhang, W. Zhou, J. Wan, S. Du, Domain adversarial graph neural network with cross-city graph structure learning for traffic prediction, Knowledge-Based Systems 278 (2023) 110885.
[56] Z. Gao, Z. Li, H. Zhang, J. Yu, L. Xu, Dynamic spatiotemporal interactive graph neural network for multivariate time series forecasting, Knowledge-Based Systems 280 (2023) 110995.
[57] X. Zhang, C. Huang, Y. Xu, L. Xia, P. Dai, L. Bo, J. Zhang, Y. Zheng, Traffic flow forecasting with spatial-temporal graph diffusion network, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 15008–15015.
[58] X. Ta, Z. Liu, X. Hu, L. Yu, L. Sun, B. Du, Adaptive spatio-temporal graph neural network for traffic forecasting, Knowledge-Based Systems 242 (2022) 108199.
[59] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).