Decoupling Long- and Short-Term Patterns in Spatiotemporal Inference

Junfeng Hu, Yuxuan Liang, Zhencheng Fan, Li Liu, Yifang Yin, and Roger Zimmermann, Senior Member, IEEE

Abstract—Sensors are the key to environmental monitoring, which impart benefits to smart cities in many aspects, such as providing real-time air quality information to assist human decision-making. However, it is impractical to deploy massive sensors due to the expensive costs, resulting in sparse data collection. Therefore, how to get fine-grained data measurement has long been a pressing issue. In this article, we aim to infer values at non-sensor locations based on observations from available sensors (termed spatiotemporal inference), where capturing spatiotemporal relationships among the data plays a critical role. Our investigations reveal two significant insights that have not been explored by previous works. First, data exhibit distinct patterns at both long- and short-term temporal scales, which should be analyzed separately. Second, short-term patterns contain more delicate relations, including those across spatial and temporal dimensions simultaneously, while long-term patterns involve high-level temporal trends. Based on these observations, we propose to decouple the modeling of short- and long-term patterns. Specifically, we introduce a joint spatiotemporal graph attention network to learn the relations across space and time for short-term patterns. Furthermore, we propose a graph recurrent network with a time skip strategy to alleviate the gradient vanishing problem and model the long-term dependencies. Experimental results on four public real-world datasets demonstrate that our method effectively captures both long- and short-term relations, achieving state-of-the-art performance against existing methods.

Index Terms—Attention mechanism, graph neural network, spatiotemporal inference, urban computing.

I. INTRODUCTION

In recent years, numerous sensors have been deployed in different locations to sense the environment. They constantly report spatially correlated and time-varying readings, such as traffic flows on roads and air quality measurements. Real-time monitoring of spatiotemporal data is of great importance to smart city efforts. For example, air quality information, e.g., the concentration of PM2.5 particles, can support air pollution control and alert the public for health concerns. Unfortunately, one of the critical prerequisites for the above benefits is the fine-grained deployment of sensors, which usually leads to considerable expenditure and high energy consumption [1]. Worse still, even existing sensors might lose readings due to factors such as a poor Internet connection. Thus, how to compensate for the pitfall of lacking sensors has become an urgent and challenging problem. In this article, we provide one solution by investigating the problem of spatiotemporal inference: given historical and real-time readings of existing sensors, we infer the real-time information at arbitrary locations under a graph structure. As shown in Fig. 1(b), both historical and current readings of $S_1-S_3$ are leveraged to infer the real-time air quality status of locations $L_1-L_5$ without actual sensors in those places.

Spatiotemporal inference requires delicate spatial and temporal dependency modeling [1]. Early methods infer nodes based on linear dependencies, such as temporal dependency modeling [1]. Early methods infer nodes based on linear dependencies, such as

\begin{align*}
N, N_o, N_t & \quad \text{Number of nodes/observed sensors/target locations.} \\
T & \quad \text{Time length for inference.} \\
t, t_s, t_k & \quad \text{Short-term time window and number of skip steps.} \\
A, \hat{A} & \quad \text{Predefined/learned adaptive adjacency matrix.} \\
x_i, X_i, \lambda & \quad \text{Readings of the } i\text{th sensor/all sensors at time } t\text{/all sensors.} \\
\hat{y}_i^t, \hat{Y}^t & \quad \text{Inferred signals of short-/long-term learning at target time } \tau. \\
\Theta_{AG}(X, A) & \quad \text{Graph convolution only with predefined adjacency matrix.} \\
\Theta_{AG}(X, A, \hat{A}) & \quad \text{Graph convolution with learned adaptive adjacency matrix.}
\end{align*}
Then, nonlinear relations are captured by subsequent approaches such as Gaussian processes (GPs) [3]. However, the Gaussian assumption is rigid and the expensive computation also limits its applicability [4]. Recently, deep learning methods have emerged as a dominant paradigm. Among them, spatiotemporal graph neural networks (STGNNs) are widely adopted due to their superior ability to handle non-Euclidean sensory data in graph structures [5], [6]. These methods capture the spatial relations among nodes by the graph [4].

Then, the temporal correlations among time points can be modeled, e.g., by concatenating a sequence of readings with a time window as the model’s input [7]. In other related areas such as forecasting, approaches usually combine GNNs with recurrent neural networks (RNNs) [8] or temporal convolutional networks (TCNs) [9], [10] to learn spatial and temporal dynamics separately. While these methods effectively capture spatiotemporal dependencies, they encounter two major drawbacks when applied to the inference problem.

First, they neglect the difference between long- and short-term patterns in the time series data. For example, readings can fluctuate within a short period in Fig. 1(c). On the contrary, when focusing on three curves from 00:00 A.M. to the next day at 12:00 P.M., they still follow an increasing trend. This suggests that long- and short-term relations have inconsistent influences. Unfortunately, the GNN-based inference method [7] concatenates temporal data as features, ignoring this phenomenon. To resolve this, RNN might be feasible as the recurrent structure concentrates more on the latest frames. However, the gradient vanishing problem makes it hard to capture either long-term trends or delicate short-term patterns. Although different strategies, such as adopting adversarial training [11], are proposed to mitigate the problem, they are at the cost of computations and training difficulties.

Second, existing inference models lack the ability to effectively learn complex and dynamic spatiotemporal relationships. As shown in Fig. 1(a), readings are affected by both spatial relations in the graph (i.e., blue arrows) and its historical readings in the temporal dimension (i.e., orange arrows). Moreover, there exist more complicated joint spatiotemporal dependencies (i.e., red arrows) that are influenced by sensors at different spatial and temporal positions directly. Unfortunately, the above GNN-RNN structures fail to explicitly consider them, which significantly hobbles the model’s performance. Song et al. [12] attempted to capture the joint dependencies by a temporal-extended static graph structure. However, this static graph definition struggles to grasp the highly dynamic relations. To illustrate this, in Fig. 1(b) and (c), $S_2$ is geographically close to $S_1$ but around 16:00 of 09/11/2017, and PM2.5 of $S_2$ is close to that of $S_1$, possibly due to the fickle wind condition. While several models are introduced to model dynamic relations [13], [14], [15], the majority of them focus on forecasting, and the inference problem is so far an underexplored research area.

To tackle these issues, we propose a Dual Joint SpatioTemporal Network (DualSTN) for real-time spatiotemporal inference based on graph structures. Our DualSTN decouples short- and long-term learning into dual components: a Joint SpatioTemporal Graph Attention network (JST-GAT) and a Skip Graph Gated Recurrent Unit (SG-GRU). The first component adopts attention blocks to measure the impact between a node and its spatial neighboring nodes within temporal short-term frames, as the yellow circle shown in Fig. 1(c). In this way, JST-GAT learns joint spatiotemporal relations explicitly, discarding the separate learning structures. Meanwhile, impacts are measured by real-time sensor signals, which improves the method’s ability to model potential dynamic relations. Inspired by [16], the second component consists of a graph GRU with a time skip strategy, aiming to reach the same time span with fewer recurrent steps. This enables the model to capture the long-term temporal trends while ignoring dedicated short-term patterns, as the purple arrows shown in Fig. 1(c). For long-term dynamic relations, an intuitive way is to learn an adaptive adjacency matrix at each recurrent step as suggested by Wu et al. [9], [17]. However, their transductive design is incompatible with the inductive setting of our task where target locations are not involved during training. Thus, we improve the existing matrix learning method by making node embeddings rely on current input readings. In addition, we further leverage a graph sampling strategy to train the model [18], which further enhances its generalization ability.

We compare our model with state-of-the-art methods on four real-world datasets. Results show that our DualSTN outperforms the competitors clearly. To evaluate the effectiveness and influence of each module, we also visualize the inference results and the attention weights to interpret DualSTN’s ability on modeling long- and short-term patterns as well as dynamic spatiotemporal relations. Our code is available at https://bit.ly/DualSTN and the main contributions are summarized as follows.
1) We propose a new framework for spatiotemporal inference, which decouples the long- and short-term pattern learning into separate modules.

2) We introduce a JST-GAT module that measures the interactions between nodes in different temporal and spatial dimensions concurrently, which captures the joint spatiotemporal relations explicitly.

3) We propose an SG-GRU to facilitate long-term pattern modeling and optimization, where skip operations are introduced to maintain the same time span with fewer recurrent steps.

4) Our DualSTN model achieves the state-of-the-art performance on real-world datasets in diverse applications. These results demonstrate the effectiveness and generalization ability of our method.

II. RELATED WORK

A. Spatiotemporal Inference

Spatiotemporal inference aims to infer signals of target locations with surrounding observed readings in a spatiotemporal domain. To solve the task, early statistical methods leverage linear relations modeling. For instance, KNN search averages neighbor readings as the results, while IDW [2] further utilizes inverse distances as the weights. In addition, several approaches attempt to capture nonlinear dependencies, and kriging [3], [19] is one of the prevalent methods. Based on GPs, it designs specialized kernels for the estimation of covariance between nodes and then infers targets by its posterior. However, the Gaussian assumption may not be followed by datasets, and in this case, a transformation of non-Gaussian data is required [20]. Wallin and Bolin [21] attempted to map data into a geostatistical configuration to weaken the assumption. Besides spatial relations, temporal dependencies are also considered. As an example, Yi et al. [22] modeled them by hybrid variables derived from local and global spatiotemporal views. Alternatively, the problem can be regarded as anomaly detection [23] or matrix/tensor completion [24]. Ozkan et al. [23] proposed an anomaly detection algorithm that adopts a posteriori estimator to fill the missing data, while Yu et al. [25] completed the matrix by a low-rank matrix assumption.

Recently, deep learning methods have merged as a rife paradigm due to their abilities on learning spatiotemporal relations in a data-driven way [26]. Appleby et al. [4] proposed a kriging convolutional network (KCN) for spatial data inference, which adapts graph convolutional networks to extract dependencies from one-hop neighboring sensors. IGNNK [7] concatenates readings of a sequence along the channel dimension as the inputs of the model to further capture temporal relations. This design, however, treats temporal dependencies uniformly, ignoring inconsistency in the temporal relations. Recently, Wu et al. [27] proposed a aggregation method (SATCN) which consisting of a spatial aggregation network to gather diverse spatial information and a TCN to capture temporal dependencies. Some solutions concentrate on specific applications. For instance, Cheng et al. [28] described a neural attention model using external features such as weather and point of interest, named ADAIN. Han et al. [29] introduced a novel multichannel attention model (MCAM) that views external information as feature channels and utilizes long short-term memory (LSTM) for temporal modeling. However, these deep learning models learn spatial and temporal dependencies separately and external information is not always available, which limits the models’ applications.

B. Spatiotemporal Graph Neural Network

STGNNs are popular for spatiotemporal data modeling nowadays, following mainly two categories. They either couple GNNs with RNNs [30], [31], [32], [33] or TCNs [9], [34], [35], [36]. In the first category, GNNs are employed for capturing spatial dependencies, while RNNs are used to model temporal dynamics. For example, Li et al. [37] first proposed a diffusion convolution that learns the spatial relations through bidirectional random walks on a graph and then captured temporal relations by RNNs. More advanced RNN models, such as LSTM and GRU, are also utilized in [32] and [38]. Xu et al. [32] aggregated representations from node neighborhoods as the inputs of a graph GRU, while Lai et al. [16] used LSTM for long- and short-term modeling, which has a similar motivation to us but only focus on temporal relations. In the second category, TCNs are adopted to learn temporal relationships and enjoy faster running speed than RNNs. For example, Liu et al. [34] used the structure to identify more critical data and modeled it by the proposed adversarial algorithm. Wu et al. [9] proposed a dilated inception temporal convolution to discover relations with different temporal scales.

In addition, attention mechanisms can be utilized to enhance the performance of STGNNs [38], [39], [40], [41]. Zheng et al. [42] proposed a multitattentio network to model spatial and temporal relations independently by attention. Wang et al. [43] proposed a multihop graph attention to calculate the weights of context information from multihop neighbors. Cai et al. [44] explored data periodicity by dividing data into segments. The extracted segments are then fed into the attention network to capture temporal dependencies. Huang et al. [45] combined GNNs and attention networks as a spatial gated block and adopt gated linear units (GLUs) for temporal dimension, which achieved compelling performances for both short- and long-term forecasting tasks. These designs, however, fail to capture the joint spatiotemporal relations that we aim to address.

To capture hidden relationships that are not reflected in the adjacency matrix utilized by STGNNs, Li et al. [46] and Bai et al. [47] proposed graph generation methods to learn an adaptive adjacency matrix and STGNNs take these two matrices to learn spatial relations. Unfortunately, the static structure cannot capture dynamic relations and their transductive learning approach is not suitable for our task. To solve the challenge, Shin and Yoon [48] progressively optimized the learned graph for new nodes that are not involved in training, based on their available readings. However, as readings of target locations are completely missing in our problem, this approach is also not applicable.
C. Comparison to Existing Approaches

We compare our model with other inference methods to highlight the differences in this section. KNN, IDW [2], and OKriging [19] are statistical methods, while our DualSTN is a data-driven method. Meanwhile, OKriging is a geolocation method only applied to geographic data. On the contrary, our method is suitable for various datasets. Table I summarizes the model characteristics. For the deep learning methods, KNN and KCN-SAGE [4] are one-hop models. Instead, our approach is an n-hop method and also takes temporal dependencies into consideration. IGNNK [7] adopts GNNs but ignores different long- and short-term patterns. SATCN [27] utilizes TCNs to capture temporal relations, while our method decouples long- and short-term learning and could model spatiotemporal dependencies simultaneously.

III. PRELIMINARIES

A. Problem Formulation

In this work, we focus on the real-time spatiotemporal data inference task under a graph structure [see Fig. 1(a) and (b)]. A graph is represented by $G = (V, E, A)$, where $V$ is the node set, $E$ is the set of edges, and $A$ is the predefined adjacency matrix. Suppose that we have $N_u$ stations with observed spatiotemporal signals, and we denote sensor signals at time $t$ as $X_t = [x_{t1}, ..., x_{tN_u}] \in \mathbb{R}^{N_u \times D}$, where $D$ is the number of attributes in a node. The goal aims to use available station readings $[X_{t-T+1}, X_{t-T+2}, ..., X_t]$ of time window $T$ to infer signals $Y_t$ of $N_u$ locations at time $t$ given their spatial relations in the graph $G$

$$[X_{t-T+1}, X_{t-T+2}, ..., X_t, G] \xrightarrow{f_A(\cdot)} [Y_t] \quad (1)$$

where $f_A(\cdot)$ is the learned mapping function with parameters $A$ and assume that $N = N_u + N_s$. Note that at any time $t$, we only use historical and current station readings $X_{t-T+1:t}$ to infer target locations $Y_t$, which follows the definition of real-time inference as no future readings are available. Furthermore, it is possible that several stations lose readings due to a bad Internet connection or some sensors may be removed or added to the graph. This requires our model to be inductive to various numbers of stations and target locations by design.

B. Graph Convolution Layer

As an essential operation to learn interactions among nodes defined by a graph structure [49], the graph convolution aggregates node features from its neighbors to learn spatial correlations. By stacking convolution layers, the model is capable of learning dependencies from multihop neighbors to improve the modeling ability. From the spatial perspective, a graph convolutional layer is formulated as

$$Z = \phi(PXW) \quad (2)$$

where $P = D^{-1}(A + I) \in \mathbb{R}^{N \times N}$ denotes the normalized adjacency matrix with self-loops, $D$ is the degree matrix, $X \in \mathbb{R}^{N \times D}$ are the input readings, $W \in \mathbb{R}^{D \times F}$ are learnable parameters, and $\phi(\cdot)$ is an activation function. Li et al. [37] further introduced a diffusion convolution that propagates graph features with $K$ steps

$$Z = \phi\left(\sum_{k=1}^{K} P^kXW_k\right) \quad (3)$$

To capture hidden graph structures that the predefined adjacency matrix cannot reflect, Wu et al. [17] proposed an adaptive adjacency matrix learning method for the graph convolution, which results in

$$Z = \phi\left(\sum_{k=1}^{K} P^kXW_k + \hat{A}^kXU_k\right) \quad (4)$$

where $\hat{A}$ is the adaptive adjacency matrix learned by a network. We term these two graph convolutions as $\Theta \circ \phi(X, A)$ and $\Theta \circ \phi(X, \hat{A})$, where $\Theta$ are learnable parameters. The important notations in this article are reported in the Nomenclature.

IV. METHODOLOGY

A. Overview of DualSTN

Fig. 2 shows the overall framework of our DualSTN, which consists of two backbone components for long- and short-term spatiotemporal pattern learning. The short-term JST-GAT first generates pseudo nodes for unknown locations for the following attention blocks. Then, stacked graph convolutions and spatiotemporal attention layers are used to learn joint spatiotemporal dependencies, followed by a fully connected layer to generate short-term inference results. At each recurrent step, the long-term SG-GRU first learns an adaptive adjacency matrix for the graph in an inductive approach. Then, the graph

| Model | Year | Temporal Relation Method | Spatial Relation Method | Learning Approach | # hop |
|-------|------|--------------------------|------------------------|------------------|-------|
| KNN   | None | None                     | Linear                 | 1-hop            | None  |
| IDW [2]| 2008| None                     | Linear                 | 1-hop            | None  |
| OKriging [19] | 2015| None                     | Gaussian               | 1-hop            | None  |
| GLTL [24]     | 2014| Low-Rank Assumption      | Low-Rank Assumption    | 1-hop            | Transductive |
| KCN [4]     | 2020| None                     | GNN                    | 1-hop            | Inductive |
| IGNNK [7]   | 2021| GNN                      | GNN                    | n-hop            | Inductive |
| SATCN [27]  | 2021| TCN                      | Graph Aggregation      | n-hop            | Inductive |
| DualSTN (ours) | New| Joint Attention, GRU     | Joint Attention        | n-hop            | Inductive |
Fig. 2. Framework of the proposed DualSTN model that contains two components: SG-GRU for long-term learning and JST-GAT for short-term learning. Adj means the adjacency matrix.

Fig. 3. Illustrations of graph sampling. For each iteration, we sample a subgraph and divide nodes into observed sensors and unknown locations randomly. (a) Training graph. (b) Graph sampling. (c) Node division. (d) Data inference.

GRU further takes the learned matrix and hidden states to encode current readings. Finally, it integrates short-term results as the input to generate long-term inference results. In the following, we first introduce the graph sampling strategy for inductive learning and then describe the details of DualSTN.

B. Graph Sampling for Inductive Learning

In a real-world environment, new sensors might be added to the graph and even existing sensors could retire after some time. In this situation, models need to be compatible with different graphs and input sensors. In addition, they ought to have a better generalization ability, which makes the task more challenging. Previous model KCN [4] designed an inductive model but trained the model using a static graph, which is suboptimal and prone to overfitting. Nowadays, approaches solve this by either learning node embedding functions [6] or sampling subgraphs during training [7], [18]. In this article, we adopt the sampling method that does not involve more parameters. Given a training graph in Fig. 3(a), for each iteration, we first randomly sample a subgraph in Fig. 3(b). Then, in Fig. 3(c), the subgraph is randomly divided into two groups dubbed observed sensors and target locations. In Fig. 3(d), we leverage observed sensors to infer signals of target locations and optimize the network. In this way, the model is less likely to be optimized according to knowledge from the absolute node locations and can capture universal spatiotemporal relations shared among sensors, strengthening its generalization ability. Algorithm 1 describes the graph sampling process in detail. After obtaining the subgraph, we can fetch a batch of data as the inputs for training. Note that we use $N$ to denote the number of nodes in the subgraph in the following.

C. Joint Spatiotemporal Graph Attention Network

1) K-Nearest IDW: As we do not have signals of target locations in a graph, the attention mechanism is suboptimal to
be applied directly. Thus, we first calculate initial readings for them by k-nearest IDW (k-IDW). To be specific, we fill short-term values of the locations (termed pseudo nodes) at time \(\tau\) and its short-term neighbor frames in the window \([\tau-t_s, \tau-1]\). As shown in the yellow arrows of Fig. 4, k-IDW follows the idea of KNN that first searches the spatially k-nearest observed sensors for each target location. Then, the inverse distances from the location to its neighbors are utilized to weights to calculate the mean

\[
\hat{x}_{t,i} = \frac{\sum_{j=1}^{k} x_{i,j} \odot d_{i,j}^{-\rho}}{\sum_{j=1}^{k} d_{i,j}^{-\rho}}
\]

where \(t \in [\tau-t_s, \tau]\), \(k\) denotes the assigned number of nearest neighbors, \(d_{i,j}\) is the distance between a target location \(i\) and the neighbor \(j\), \(\rho\) means the decay rate, and \(\odot\) represents the Hadamard product. After obtaining pseudo nodes, they can be used to compute attention scores and we regard pseudo nodes and observed sensors uniformly, notated as \(X_i = [x_{i,1}, \ldots, x_{i,N_s}, \tilde{x}_{1,i}, \ldots, \tilde{x}_{N_s,i}] \in \mathbb{R}^{N_s \times D}\) as follows.

2) Joint Spatiotemporal Attention: Spatiotemporal data are influenced by conditions from surrounding locations that are highly interactive and difficult to capture the patterns. To learn these relationships, attention networks are proven to be an effective tool [39]. Among them, researchers mainly utilize individual attention blocks to handle the spatial relations and temporal dynamics, followed by a fusion module to integrate them [41], [42], [50]. However, these models fail to explicitly consider joint spatiotemporal dependencies directly. For instance, assuming a north wind, PM2.5 particles will be blown toward the south over time. In addition, a car accident will clog traffic on an upstream road after a short time. The phenomena involve synchronous spatiotemporal shifts and are hard for separate attention modules to model. We propose a joint spatiotemporal attention mechanism to capture them simultaneously, as shown in Fig. 4.

Given features of a target frame \(Z^l_t\) at layer \(l-1\) and its neighbor frames \(Z^{l-1}_t\) in the short-term window \([\tau-t_s, \tau-1]\), we first employ a graph convolution layer with skip connection to learn the spatial relations in each frame

\[
Z^l_t = y Z^{l-1}_t + \mu \Theta_{xy}(Z^{l-1}_t, A), \quad t \in [\tau-t_s, \tau]
\]

where \(Z^l_t = X_t\) and \(y\) and \(\mu\) are hyperparameters for skip connection. Here, the adjacency matrix \(A\) defines static spatial relations as a prior, so the attention mechanism relieves from learning it again, reducing learning difficulties. Then, the attention operation captures joint spatiotemporal dependencies. We first calculate attention weights between sensor \(z_{t,i}\) and other sensors \(z_{t,j}\) in the window formulated by

\[
e_{t,i,j} = v^T_a \tanh(W_a z^t_{i,j} + U_a z^t_{i,j} + b_a)
\]

\[
\rho_{t,i,j} = \exp(e_{t,i,j})
\]

\[
e_{t,i,j} = \sum_{s=t-t_s}^{t} \sum_{k=1}^{N} \rho_{t,i,j}
\]

where \(v_a, b_a \in \mathbb{R}^F\) and \(W_a, U_a \in \mathbb{R}^{D \times F}\) are learnable parameters. Note that the parameters are shared across all frames to reduce the computational cost. We also compute attention scores within the target frame, and in this situation, the block degrades into spatial attention. Finally, we obtain an attention map \(E^t_{i,j} \in \mathbb{R}^{(t-t_s) \times N_s \times N_s}\), where the second dimension \(N\) refers to sensor features at the target time \(t\) and the third dimension \(N\) denotes sensors in the \(t+1\) frames. Next, we use features of short-term frames and the attention map to learn short-term inference representations

\[
z^t_{t,i} = \sum_{t=t-t_s}^{t} \sum_{j=1}^{N} E^t_{i,j} z^t_{i,j}.
\]

We stack the joint attention blocks for \(L\) layers. On the top layer \(L\), a fully connected layer is used to generate the short-term outputs

\[
\hat{Y}^t = Z^l_z F_s + b_s
\]

where \(F_s \in \mathbb{R}^{F \times D}\) and \(b_s \in \mathbb{R}^D\) are parameters and \(\hat{Y}^t \in \mathbb{R}^{N \times D}\) are short-term inference results. Finally, the joint spatiotemporal dependencies can be learned by a single JST-GAT without separate modules. Moreover, the attention block aids interpretability by visualizing weights to understand how the model learns the spatiotemporal relations.

D. Skip Graph Gated Recurrent Unit

1) Inductive Dynamic Graph Generation: To model the hidden relations among nodes, previous works learn an adaptive adjacency matrix during training and it remains static during testing [9], [17], [47]. However, this disregards the dynamic dependencies of the graph structure over the timeline. Later, Li et al. [46] modeled the dynamic connections by learning an adaptive matrix at each step of a recurrent network. Unfortunately, the method significantly relies on embeddings of training nodes, which is not available in the inductive setting. To solve these challenges, we propose an inductive graph generation module that updates the adjacency matrix in an inductive fashion based on [9]. To be specific, at time step \(t\), we use the historical hidden states \(H_{t-k}, X_t, A\) to learn the graph structure information. Here, \(t_k\) is a skip step described in Section IV-D2. The node embedding is replaced by a fully connected layer FC(\(\cdot\)) taking \(H_{t-k}\) as input. In summary, the
adaptive adjacency matrix $\hat{A}_t$ at time $t$ is calculated by

$$
M_t^1 = \tanh(\Theta_{1 \text{gr}}(X_t, A) \circ FC_1(H_{t-1})) \\
M_t^2 = \tanh(\Theta_{2 \text{gr}}(X_t, A) \circ FC_2(H_{t-1})) \\
\hat{A}_t = \text{ReLU}(\tanh(\sigma(M_t^1M_t^T - M_t^2M_t^T)))
$$

where $M_t^1$ and $M_t^2 \in \mathbb{R}^{N \times F}$ are source node encoder and target node encoder, respectively, $\sigma(\cdot)$ is the sigmoid activation function, and $\alpha$ is the saturation rate hyperparameter.

2) Skip Graph Gated Recurrent Unit: GRU is one of the recurrent structures designed to capture historical temporal information in a recurrent way [51]. However, the gradient vanishing and exploding state estimation of the latest inputs cause difficulties to capture long-term temporal patterns [52]. To alleviate this, motivated by [16], we propose a graph GRU with skips to maintain a temporal span with fewer recurrent steps. To be specific, hidden states are updated using historical hidden states of a certain number of skips $I_k$, which can be formulated as:

$$
\begin{align*}
\mathbf{r}_t &= \sigma(\Theta_{r \text{gr}}(X_t | H_{t-1}, A, \hat{A}) + \mathbf{b}_r) \\
\mathbf{u}_t &= \sigma(\Theta_{u \text{gr}}(X_t | H_{t-1}, A, \hat{A}) + \mathbf{b}_u) \\
\mathbf{c}_t &= \tanh(\Theta_{c \text{gr}}(X_t | (H_{t-1} \odot \mathbf{r}_t), A, \hat{A}) + \mathbf{b}_c) \\
\mathbf{H}_t &= \mathbf{u}_t \odot H_{t-1} + (1 - \mathbf{u}_t) \odot \mathbf{c}_t \\
\hat{Y}_t^l &= \mathbf{H}_t \mathbf{W}_{fl} + \mathbf{b}_{fl}
\end{align*}
$$

where $\odot$ means the concatenation and $\mathbf{r}_t$ and $\mathbf{u}_t \in \mathbb{R}^{N \times F}$ are reset gate and update gate, respectively. Through the recurrent skip, the module is encouraged to focus on the high-level long-term temporal patterns, which ignores delicate relations among the consecutive frames. The decreased recurrent steps also facilitate the optimization process. At the last recurrent step at which the short-term inference results $\hat{Y}_t^l$ are computed, we feed $\hat{Y}_t^l$ to the graph GRU to compute the corresponding hidden states $H_t$. Finally, a fully connected layer is used to obtain the long-term inference outputs

$$
\hat{Y}_t^l = \mathbf{H}_t \mathbf{W}_{fl} + \mathbf{b}_{fl}
$$

where $\mathbf{W}_{fl} \in \mathbb{R}^{F \times D}, \mathbf{b}_{fl} \in \mathbb{R}^D$ are parameters, and $\hat{Y}_t^l \in \mathbb{R}^{N \times D}$ are long-term inference outputs.

E. Loss Function and Training Procedure

To strengthen the generalization ability, we train our model by reconstructing all sensor signals instead of just target locations as in [4] and simultaneously optimize the mean absolute error (MAE) loss of short-term outputs $\hat{Y}_t^l$ as well as long-term results $\hat{Y}_t^s$

$$
\mathcal{L} = \frac{1}{N} |Y_t - \hat{Y}_t^l| + \frac{1}{N} |Y_t - \hat{Y}_t^s|.
$$

The training procedure of DualSTN is summarized in Algorithm 2. For each iteration, we randomly sample a subgraph and its corresponding sensor readings to train the model. Note that we adopt the same subgraph and node division for each batch to simplify the implementation.

Algorithm 2 Training Procedure of DualSTN

Input: graph $G = (V, E)$, sensor readings of training set $X$, time window $T$; initialized model DualSTN().

Output: optimized learnable weights.

1: for $i = 1 \rightarrow \text{Num\_Iteration}$ do
2: Initialize batch list $X_b = [], Y_b = []$.
3: [Sampling Graph]
4: for $j = 1 \rightarrow \text{Batch\_Size}$ do
5: Randomly choose a start time $t$.
6: Append $(X_b)_{t+T}$ to $X_b$.
7: Append $(Y_b)_{t+T}$ to $Y_b$.
8: end for
9: end for
10: $\hat{Y}_b^l, \hat{Y}_b^s = \text{DualSTN}(G_b, X_b)$.
11: Compute MAE($\hat{Y}_b^l, Y_b$) + MAE($\hat{Y}_b^s, Y_b$) and derive the gradients.
12: Update learnable weights using the optimizer.

V. EXPERIMENTS

A. Experimental Settings

1) Datasets: We evaluate the performances of DualSTN on four real-world spatiotemporal datasets in diverse application scenarios.

1) METR-LA\(^1\) [37]: Traffic speed dataset collected from 207 sensors in the highway of Los Angeles from 01/05/2012 to 30/06/2012.

2) PeMS-Bay\(^2\) [37]: A traffic speed dataset collected by California Transportation Agencies containing 325 sensors in the Bay Area from 01/01/2017 to 13/05/2017.

3) NREL\(^3\) [53]: Energy datasets provided by the National Renewable Energy Laboratory and we choose a subset of Alabama Solar Power Data. The dataset contains 137 photovoltaic power plant readings collected in 2006.

4) BJ-Air\(^4\): The air quality index dataset from 35 air quality stations in Beijing and we consider the PM2.5 observations.

We construct the predefined adjacency matrix $A$ based on either road network distance or geospatial distance $\text{dist}(v_i, v_j)$. The road network distance is available in the dataset and we compute the geospatial distance using the Haversine formula, given the longitude and latitude

$$
\text{dist}(v_i, v_j) = 2r \arcsin \left( \sin^2 \left( \frac{\varphi_j - \varphi_i}{2} \right) \cos(\varphi_i) \cos(\varphi_j) \sin^2 \left( \frac{\lambda_j - \lambda_i}{2} \right) \right)^{1/2}
$$

where $r = 6371$ is the radius of the Earth and $(\varphi_i, \lambda_i)$ means the longitude and latitude of the sensor $v_i$. Then, the Gaussian

---

1https://github.com/liyaguang/DCRNN
2https://github.com/liyaguang/DCRNN
3https://www.nrel.gov/grid/solar-power-data.html
4https://www.biendata.xyz/competition/kdd_2018/
kernel method [54] is applied [to get the matrix].

$$A_{i,j} = \exp \left( -\frac{1}{\sigma^2} \operatorname{dist}(v_i, v_j)^2 \right)$$  \hspace{1cm} (16)

where $\sigma$ is the standard deviation. In the case of the directed graph that contains bidirectional adjacency matrices $A_f$ and $A_b$, we use the same network to model them, which is equivalent to $A = (A_f + A_b)/2$. We summarize the statistics of datasets in Table II.

2) Baseline Methods: We compare the performances of our model with seven baselines as follows.

1) **KNN**: It interpolates readings of unknown locations by averaging the $k$-nearest sensors in the spatial dimension.

2) **IDW**: It utilizes distances between nodes to calculate a weighted average of available nodes for each unknown location [2].

3) **OKRiging**: Ordinary kriging is a classical statistical interpolation method based on the geospatial locations of the sensors and GPs [19]. We evaluate the performance of OKRiging using the package PyKrige.\(^5\) Note that OKRiging is not applicable for road network distance, so we just report the performances on NREL and BJ-Air datasets.

4) **GLTL**:\(^6\) It is a low-rank tensor learning framework for spatiotemporal data co-kriging and forecasting, which handles various properties in the data. Moreover, a fast greedy algorithm is proposed to learn the tensor efficiently.

5) **KCN, KCN-SAGE**:\(^7\) KCN first searches KNN for a target location. Then, it constructs a graph structure for the $K + 1$ nodes as inputs of GNNs to interpolate signals [4]. KCN-SAGE is a variant based on graph sampling and aggregating [6].

6) **IGNNK**:\(^8\) Inductive GNN kriging is a state-of-the-art model trained in an inductive approach [7]. It regards sequential reading as the feature of a GNN to learn spatial and temporal relations.

7) **SATCN**:\(^9\) It contains a set of spatial aggregators using signals’ statistic features for spatial modeling. Meanwhile, TCNs are leveraged to capture temporal dependencies [27].

3) **Evaluation Metrics**: We utilize three criteria to evaluate models: the root-mean-square error (RMSE), the MAE, and the mean absolute percentage error (MAPE). All of them are frequently used in regression problems

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i \in N} (y^i - \hat{y}^i)^2}$$  \hspace{1cm} (17)

$$\text{MAE} = \frac{1}{N} \sum_{i \in N} |y^i - \hat{y}^i|$$  \hspace{1cm} (18)

$$\text{MAPE} = \frac{1}{N} \sum_{i \in N} \left| \frac{y^i - \hat{y}^i}{y^i} \right|$$  \hspace{1cm} (19)

where $y^i$ is the ground-truth signal of a sensor and $\hat{y}^i$ denotes inferred signals.

4) **Implementation Details**: Our DualSTN and other deep learning baselines are implemented with PyTorch 1.7 and trained on a Quadro RTX 6000 GPU. We use a historical time window $T = 25$ to infer the real-time readings in which the time window for short-term learning is $t_s = 3$. The skip step of GRU $t_k$ equals 4, i.e., we feed readings of sensors into the network every four steps. For the hyperparameters, we set decay rates $\rho$ and $\lambda$ to 1, saturation rate $\sigma$ to 2, and impact factors $\gamma$ and $\mu$ to 0.1 and 0.9, respectively. The activation function $\phi$ is ReLU. The number of layers for JST-GAT is 3 and the size of hidden states for graph GRU is 16. All learnable parameters are initialized with the Xavier [55]. The model is trained by the Adam [56] optimizer and the learning rate of $10^{-3}$. Note that we keep the same settings for all datasets, verifying the generalization ability of our model.

For the dataset division, we use the first 70% of the time frames to train models and validate or test models using the following 20% and 10%, respectively. For the sensor division, we manually leave 50% of the sensors out for testing, dubbed testing sensors and train each deep learning model five times independently to report the average results and the standard deviations.

### B. Model Comparison

In this section, we compare the performances of our DualSTN with all baselines, and the results are summarized in Table III. We observe that statistical methods have worse results than data-driven approaches. This is chiefly because they are designed to capture linear or Gaussian relations, which is suboptimal to measure complex dependencies. For deep learning methods, as they could learn complicated nonlinear spatiotemporal dependencies from data, we find that even the spatial method KCN outperforms GLTL, which considers both spatial and temporal dependencies. Meanwhile, IGNUK outperforms KCN because IGNUK adopts n-hop GNNs and learns from temporal information. For our model, we observe that it outperforms all baselines on LETR-LA, PeMS-Bay, and NREL and competitive results on the BJ-Air dataset. It results

---

\(^5\)https://geostat-framework.readthedocs.io/projects/pykrige/en/stable

\(^6\)https://roseyu.com/code.html

\(^7\)https://github.com/tufts-ml/KCN

\(^8\)https://github.com/Kaimaoige/IGNNK

\(^9\)https://github.com/Kaimaoige/SATCN

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
from that DualSTN could learn spatiotemporal relationships simultaneously and the decoupled design makes the model easier to distinguish long- and short-term patterns. In addition, DualSTN also has fewer parameters than other models, which also demonstrates its learning ability. The reason might be that our model uses a single module to learn spatial and temporal relations. Thus, we do not need to stack many layers to enlarge the receptive field, reaching nodes with far distances. For the BJ-Air dataset, both our DualSTN and SATCN achieved commensurate performance and surpass other baselines. SATCN designs special aggregators like the standard deviation aggregator, which might be beneficial for this dataset with a large deviation. However, DualSTN has five times fewer parameters than SATCN. Furthermore, we notice that on the BJ-Air dataset, the results of the three deep learning models have larger variations, which means that the training process is not stable. We conjecture the limited number of sensors and the large standard deviation of readings cause difficulties in learning general spatiotemporal dependencies.

C. Ablation Study

Our model is largely built upon two motivations (i.e., decoupling long- and short-term patterns and joint spatiotemporal learning). A natural question is whether they are effective. In this section, we implement three variants of DualSTN to verify the effectiveness of components described as follows.

1) LongSTN: This variant removes the short-term learning module and only contains the SG-GRU. The inference results are outputs of the GRU’s last recurrent step.

2) ShortSTN: This variant removes the skip GRU and remains the joint spatiotemporal attention graph network, which only takes short-term frames as the inputs to infer unknown locations.

3) DualTCN: Our model uses attention blocks to learn temporal dependencies in the short-term module and another intuitive opinion is leveraging TCNs. Thus, this variant replaces the attention module with the TCN that follows the same structure as [17] and the long-term module remains the same.

We evaluate the performance of three variants on four datasets and illustrate the results in Fig. 5. We find that ShortSTN performs better than LongSTN, while both are much worse compared to DualSTN. These observations demonstrate the following conclusions: 1) sensor readings are more relevant to short-term patterns but still follow the trend of long-term ones and 2) decoupling long- and short-term learning improves performance as they provide information from different perspectives. For the TCN variant, we observe that DualSTN has a better performance compared to DualTCN and argue that

![Fig. 5. Ablation studies. The DualSTN consistently achieves the best RMSE and MAE results against other variants.](image-url)
this is because the joint attention module explicitly takes joint spatiotemporal relations into consideration, which reduces the challenge of modeling complex relations.

D. Hyperparameter Study

In this section, we study the performance of our DualSTN under different hyperparameter settings and report MAPE results. In each study, we modify the setting of corresponding hyperparameters and keep others unchanged. All the experiments are conducted on four datasets.

1) Effects of Number of Layers of JST-GAT: We adjust the number of layers $L$ in the joint spatiotemporal attention module and report the results in Fig. 6(a). The model performances first become better and achieve the best at four layers for the BJ-Air dataset and three layers for the rest. Then, MAPE results remain stable or start to increase slightly. According to these observations, we uniformly keep the number of layers as 3 to reduce the computational cost.

2) Effects of Size of Hidden States of SG-GRU: We change the size of the graph GRU’s hidden states from 4 to 128. As shown in Fig. 6(b), we discover that the performances increase until reaching the size of 16 or 32 for four datasets. Then, they start to decrease, indicating that the model tends to overfit. Accordingly, we set the size to 16.

3) Effects of Time Window $T$: We keep the skip step in the graph GRU $t_k = 4$ and adjust the input time window $T$ to evaluate GRU’s capability of learning long-term patterns. As expected in Fig. 6(c), a longer input sequence cannot guarantee better results. Instead, the model crashes over a long time window because of its gradient vanishing problem of GRU on long-term modeling. In this scenario, the encoded long historical features become noises of hidden states, impeding the model performance.

4) Effects of Number of Short-Term Frames $t_s$: Next, we modify the number of short-term frames $t_s$. As shown in Fig. 6(d), as $t_s$ increases, the performance first increases fast and then levels off. As a larger $t_s$ uses frames modeled by SG-GRU, the delicate learning module JST-GAT is redundant to capture them again. To this end, we set $t_s$ to 3 and leverage the graph GRU for learning these patterns.

5) Effects of Time Skip $t_k$: Finally, we keep $t_s = 3$ and $T = 25$ and adjust time skip $t_k$ to evaluate its influence. As shown in Fig. 6(e), the results consistently decrease, and especially, this decrease speeds up as $t_k$ increases. The reason is that as the time span becomes too large, the temporal dependencies between input frames become sparse that the model cannot capture. Thus, we choose $t_k$ to 4 to ensure no frame overlap between the short- and long-term modules, which also saves running time.

In addition, from the study, we observe that DualSTN is able to achieve satisfying performances over all datasets using the same hyperparameter setting. This means that our model is insensitive to different application domains, which relieves the demand for hyperparameter searching and is beneficial to real-world deployment. It could be caused by the fewer learnable parameters and the learning effectiveness of the model.

E. Case Study

1) Long- and Short-Term Patterns Inconsistency: To study how our DualSTN captures the long- and short-term patterns, we visualize the short-term results $\hat{Y}^s$ and the final results $\hat{Y}^f$ that integrates information of both terms. Fig. 7 shows the results and ground truth of META-LA from 16:30, 23/06/2023, and the BJ-Air dataset from 14/01/2018, in which we have three observations.

1) In the red boxes, the truth signals fluctuate while having a flat trend. The short-term inferred signals, aligning
with the ground truth, also oscillate as the JST-GAT only focuses on short-term patterns. Then, by involving trends from long-term patterns, the final outputs become stable.

2) In the blue boxes, signals follow an upward or downward trend. The long-term outputs are accurate compared to the short-term results. This suggests that the tendency is important in this scenario and SG-GRU could capture it precisely.

3) In the gray boxes where a sudden change in readings happens, the short-term outputs outperform the long-term results. This is because the historical tendency in SG-GRU does not tally with this sudden change.

Overall, these discoveries mean that our model handles short- and long-term patterns, and JST-GAT and SG-GRU can contribute to the final results in different aspects.

2) Joint Spatiotemporal Dependencies: The attention block in our model provides interpretability by indicating the dependencies between two nodes. As the motivation here is to capture joint spatiotemporal relations, we conduct a case study using the BJ-Air dataset from 0:00 to 12:00 on 03/09/2017 and visualize the attention weights of an inferred location to investigate this ability. For succinctness, we use a center station $S_8$ as the target sensor to compute attention weights. Note that stations $S_{13}$ and $S_{17}$ are extremely far away from $S_{21}$. (b) Attention scores of station $S_{21}$.

3) However, it cannot guarantee genuine spatiotemporal dependencies. For instance, $S_{13}$ is excessively far away from $S_{21}$ compared to $S_{18}$ but has an even larger impact. In this situation, the dynamic dependencies become dominant.

4) The weights of $S_8$ increase over time, mightily due to the change of wind speed. This means that joint spatiotemporal attention is capable of capturing relations across time and space.

These observations verify that our model captures both static spatial relations, dynamic implicit dependencies, and joint spatiotemporal relations simultaneously even without knowing possible external factors. This compelling learning ability also provides the possibility of applying the module to other tasks that require delicate spatial and temporal modeling, such as video object segmentation [57], [58].

VI. CONCLUSION AND FUTURE WORKS

In this article, we propose a novel DualSTN model for spatiotemporal inference. To better learn the short- and long-term patterns, we decouple the model into two components: JST-GAT and SG-GRU. The first aims to capture delicate short-term joint spatiotemporal correlations, while the second network focuses on long-term patterns by a time skip strategy. Extensive experiments on four real-world applications suggest that our DualSTN offers state-of-the-art performances against previous baselines. Further evaluations also justify the effectiveness of the modules as well as the interpretation ability brought by attention mechanisms.

In the future, we can explore ways to speed up the inference without losing performance, as we find that SG-GRU consumes a long inference time due to its recurrent structure. Then, the idea of joint attention can be transferred to the forecasting task for better capturing spatiotemporal relations. In addition, more complex models can be proposed to integrate forecasting and inference as a unified task.

REFERENCES

[1] Y. Zheng, F. Liu, and H.-P. Hsieh, “U-air: When urban air quality inference meets big data,” in Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2013, pp. 1436–1444.

[2] G. Y. Lu and D. W. Wong, “An adaptive inverse-distance weighting spatial interpolation technique,” Comput. Geosci., vol. 34, no. 9, pp. 1044–1055, Sep. 2008.
[3] C. E. Rasmussen, “Gaussian processes in machine learning,” in Summer School on Machine Learning. Cham, Switzerland: Springer, 2003.

[4] G. Appleby, L. Liu, and L.-P. Liu, “Kriging convolutional networks,” in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 4, 2020, pp. 3187–3194.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in Proc. 2nd Int. Conf. Learn. Represent. (ICLR), Banff, AB, Canada, Apr. 2014, pp. 1–14.

[6] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 1025–1035.

[7] Y. Wu, D. Zhang, A. Labbe, and L. Sun, “Inductive graph neural networks for spatiotemporal kriging,” in Proc. 35th AAAI Conf. Artif. Intell., 2021, pp. 4478–4485.

[8] J. Han, H. Liu, H. Zhu, H. Xiong, and D. Dou, “Joint air quality and weather prediction based on multi-adversarial spatiotemporal networks,” in Proc. 35th AAAI Conf. Artif. Intell., 2021, pp. 4081–4089.

[9] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 753–763.

[10] L. Zhao et al., “T-GCN: A temporal graph convolutional network for traffic prediction,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[11] S. Yang, M. Dong, Y. Wang, and C. Xu, “Adversarial recurrent time series imputation,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 4, pp. 1639–1650, Apr. 2023.

[12] C. Song, Y. Lin, S. Guo, and H. Wan, “Spatial-temporal synchronous graph convolutional networks: A new framework for spatial–temporal network data forecasting,” in Proc. AAAI, vol. 34, no. 1, 2020, pp. 914–921.

[13] C. Shang, J. Chen, and J. Bi, “Discrete graph structure learning for forecasting multiple time series,” in Proc. 9th Int. Conf. Learn. Represent., 2021, pp. 1–14.

[14] Y. Qin, Y. Fang, H. Luo, F. Zhao, and C. Wang, “DMGCNRN: Dynamic multi-graph convolution recurrent network for traffic forecasting,” 2021, arXiv:2112.02264.

[15] M. Li and Z. Zhu, “Spatial–temporal fusion neural network for traffic flow forecasting,” in Proc. AAAI Conf. Artif. Intell., vol. 35, no. 5, 2021, pp. 4189–4196.

[16] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, “Modeling Long- and short-term temporal patterns with deep neural networks,” in Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jun. 2018, pp. 95–104.

[17] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for deep spatial–temporal graph modeling,” in Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI), 2017, pp. 1007–1913.

[18] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna, “GraphSAINT: Graph sampling based inductive learning method,” in Proc. 8th Int. Conf. Learn. Represent. (ICLR), 2019, pp. 1907–1913.

[19] X. Shu, L. Zhang, Y. Sun, and J. Tang, “Host–parasite: Graph LSTM-in-LSTM for group activity recognition,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 2, pp. 663–674, Feb. 2021.

[20] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, “Urban traffic prediction from spatio-temporal data using deep meta learning,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 1720–1730.

[21] Y. Li, J. Tian, L. Miranda-Moreno, and L. Sun, “Adversarial danger identification on temporally dynamic graphs,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 13, 2023, doi: 10.1109/TNNLS.2023.3252175.

[22] Y. Liang et al., “Fine-grained urban flow prediction,” in Proc. Conf. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 5998–6008.

[23] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 5998–6008.

[24] F. Liu, J. Feng, H. Yan, and Y. Bengio, “Graph attention networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 5998–6008.

[25] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, “LSGCN: Long short-term traffic prediction with graph convolutional networks,” arXiv:2011.02264.

[26] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, 2018.

[27] Z. Pan et al., “Spatio-temporal attention networks for traffic flow forecasting,” 2022, arXiv:2202.08982.

[28] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, “A neural attention model for urban air quality inference: Learning the weights of monitoring stations,” in Proc. AAAI Conf. Artif. Intell., vol. 32, no. 1, 2018, pp. 2151–2158.

[29] Q. Han, D. Lu, and R. Chen, “Fine-grained air quality inference via multi-channel attention model,” in Proc. 13th Int. Joint Conf. Artif. Intell., Aug. 2021, pp. 1–7.

[30] Z. Pan et al., “Spatio-temporal meta learning for urban traffic prediction,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 3, pp. 1462–1475, Mar. 2022.

[31] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, “Urban traffic prediction from spatio-temporal data using deep meta learning,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 1720–1730.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. 6th Int. Conf. Learn. Represent. (ICLR), 2018, pp. 1–16.

[33] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial–temporal graph convolutional networks for traffic flow forecasting,” in Proc. AAAI, vol. 33, 2019, pp. 922–929.

[34] C. Zheng, X. Fan, C. Wang, and J. Qi, “GMAN: A graph multi-attention network for traffic prediction,” in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 1, Apr. 2020, pp. 1234–1241.

[35] G. Wang, R. Ying, J. Huang, and J. Leskovec, “Multi-hop attention graph neural networks,” in Proc. 13th Int. Joint Conf. Artif. Intell., 2021, pp. 3089–3096.

[36] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, “Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting,” Trans. GIS, vol. 24, no. 3, pp. 736–755, Jun. 2020.

[37] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, “LSCGN: Long short-term traffic prediction with graph convolutional networks,” in Proc. IJCAI, vol. 7, 2020, pp. 2355–2361.

[38] F. Li, J. Feng, H. Yan, G. Jin, D. Jin, and Y. Li, “Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution,” 2021, arXiv:2104.14917.

[39] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 922–929.

[40] Y. Shin and Y. Yoon, “PGCN: Progressive graph convolutional networks for spatial–temporal traffic forecasting,” 2022, arXiv:2202.08982.

[41] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907.

[42] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, “GeoMAN: Multi-level attention networks for geo-sensory time series prediction,” in Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI), 2018, pp. 3428–3434.

[43] J. Chang, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014, arXiv:1412.3555.
Junfeng Hu received the B.E. degree from the School of Big Data and Software Engineering, Chongqing University, Chongqing, China, in 2020, and the master’s degree from the National University of Singapore, Singapore, in 2022, where he is currently pursuing the Ph.D. degree with the School of Computing.

He has published papers at conferences, such as ACM SIGKDD Conference on Knowledge Discovery And Data Mining (KDD), European Conference on Artificial Intelligence (ECAI), European Conference on Machine Learning (ECML), International Joint Conference on Neural Networks (IJCNN), and IEEE International Conference on Multimedia and Expo (ICME). His research interests include spatiotemporal data mining and graph neural networks.

Yuxuan Liang received the Ph.D. degree from the National University of Singapore (NUS), Singapore, in 2023. He is currently an Assistant Professor with the Intelligent Transportation Trust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. He is working on the research, development, and innovation of spatiotemporal data mining and artificial intelligence (AI), with a broad range of applications in smart cities. He has published over 40 peer-reviewed papers in refereed journals and conferences, such as ACM SIGKDD Conference on Knowledge Discovery And Data Mining (KDD), International World Wide Web Conference (WWW), Conference on Neural Information Processing Systems (NeurIPS), International Conference on Learning Representations (ICLR), European Conference on Computer Vision (ECCV), Association for the Advancement of Artificial Intelligence (AAAI), International Joint Conference on Artificial Intelligence (IJCAI), ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp), and IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE). Those papers have been cited over 2000 times (Google Scholar: H-index of 21).

Dr. Liang was recognized as one out of ten most innovative and impactful Ph.D. students focusing on data science in Singapore by the Singapore Data Science Consortium (SDSC).

Zhencheng Fan received the bachelor’s degree in software engineering from Chongqing University, Chongqing, China, in 2020. He is currently pursuing the Ph.D. degree with the University of Technology Sydney, Sydney, NSW, Australia, specializing in memristor-based neural networks, data mining, and computer vision.

His work has been published in conferences, including ACM SIGKDD Conference on Knowledge Discovery And Data Mining (KDD), European Conference on Artificial Intelligence (ECAI), and International Conference on Knowledge Science, Engineering and Management (KSEM).

Li Liu received the Ph.D. degree in computer science from Université Paris-Sud XI, Paris, France, in 2008. He is currently a Professor with Chongqing University, Chongqing, China. He is also a Senior Research Fellow with the School of Computing, National University of Singapore, Singapore. He aims to contribute to interdisciplinary research of computer science and human-related disciplines. He has published widely in conferences and journals with more than 100 peer-reviewed publications.

Yifang Yin received the B.E. degree from the Department of Computer Science and Technology, Northeastern University, Shenyang, China, in 2011, and the Ph.D. degree from the National University of Singapore, Singapore, in 2016.

She is currently a Scientist at the Institute for Information Research, Agency for Science, Technology and Research (A*STAR), Singapore. She also holds an adjunct faculty position at IIIT Delhi, New Delhi, India. Before joining A*STAR, she worked as a Senior Research Fellow with the Grab-NUS AI Lab, National University of Singapore. She also worked as a Research Intern at the Incubation Center, Research and Technology Group, Fuji Xerox Company Ltd., Yokohama, Japan, from October 2014 to March 2015. Her research interests include machine learning, spatiotemporal data mining, and multimodal analysis in multimedia.

Roger Zimmermann (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the University of Southern California (USC), Los Angeles, CA, USA, in 1994 and 1998, respectively.

He was the Deputy Director of the Smart Systems Institute (SSI) and co-directed the Centre of Social Media Innovations for Communities (CoSMIC), National University of Singapore (NUS), Singapore. He is currently a Professor with the Department of Computer Science and a Key Investigator with the Grab-NUS AI Lab, NUS. He has coauthored a book, seven patents, and more than 300 conference publications, journal articles, and book chapters. His research interests include streaming media architectures, distributed systems, mobile and georeferenced video management, applications of machine/deep learning, and spatial data management.

Dr. Zimmermann is a Distinguished Member of Association for Computing Machinery (ACM). He is also an Associate Editor of IEEE MULTIMEDIA, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Multimedia Tools and Applications (MTAP, Springer), and IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY (OJ-COMS). Further information can be found at http://www.comp.nus.edu.sg/~rogerz/.