Artificial Intelligence 318 (2023) 103899

Contents lists available at ScienceDirect

Artificial Intelligence

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

AutoSTG$^+$: An automatic framework to discover the optimal network for spatio-temporal graph prediction[☆]

Songyu Ke[a,b,c,1,2], Zheyi Pan[b,c,1], Tianfu He[b,c], Yuxuan Liang[b,c,e], Junbo Zhang[b,c,d,*], Yu Zheng[b,c,d]

[a] *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*
[b] *JD iCity, JD Technology, Beijing, China*
[c] *JD Intelligent Cities Research, Beijing, China*
[d] *School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China*
[e] *National University of Singapore, Singapore*

ARTICLE INFO

Article history:
Received 14 March 2022
Received in revised form 15 February 2023
Accepted 28 February 2023
Available online 6 March 2023

Keywords:
Neural architecture search
Spatio-temporal graph
Meta-learning
Urban computing
Spatio-temporal data mining

ABSTRACT

Spatio-temporal graphs (STGs) are important structures to describe urban sensory data, *e.g.*, traffic speed and air quality. Predicting over spatio-temporal graphs enables many essential applications in intelligent cities, such as traffic management and environment analysis. Recently, many deep learning models have been proposed for spatio-temporal graph prediction and achieved significant results. However, *manually* designing neural networks requires rich domain knowledge and heavy expert efforts, making it impractical for real-world deployments. Therefore, we study *automated neural architecture search* for spatio-temporal graphs, which meets three challenges: 1) how to define search space for capturing complex spatio-temporal correlations; 2) how to jointly model the explicit and implicit relationships between nodes of an STG; and 3) how to learn network weight parameters related to meta graphs of STGs.

To tackle these challenges, we propose a novel neural architecture search framework, entitled AutoSTG$^+$, for automated spatio-temporal graph prediction. In our AutoSTG$^+$, spatial graph convolution and temporal convolution operations are adopted in the search space of AutoSTG$^+$ to capture complex spatio-temporal correlations. Besides, we propose to employ the meta-learning technique to learn the adjacency matrices of spatial graph convolution layers and kernels of temporal convolution layers from the meta knowledge of meta graphs. And specifically, such meta-knowledge is learned by graph meta-knowledge learners, which iteratively aggregate knowledge on the attributed graphs and the similarity graphs. Finally, extensive experiments have been conducted on multiple real-world datasets to demonstrate that AutoSTG$^+$ can find effective network architectures and achieve up to about 20% relative improvements compared to human-designed networks.

☆ This paper is an extended version of [1], which has been accepted for presentation at the Web Conference 2021 (WWW'21).

* Corresponding author at: JD iCity, JD Technology, Beijing, China & JD Intelligent Cities Research, Beijing, China.

E-mail addresses: songyu-ke@outlook.com (S. Ke), zheyi.pan@outlook.com (Z. Pan), tfhe@outlook.com (T. He), yuxliang@outlook.com (Y. Liang), msjunbozhang@outlook.com (J. Zhang), msyuzheng@outlook.com (Y. Zheng).

1 These authors contributed equally.

2 This research was done when the first author was an intern at JD Intelligent Cities Research & JD iCity under the supervision of the fifth author.

https://doi.org/10.1016/j.artint.2023.103899
0004-3702/© 2023 Elsevier B.V. All rights reserved.

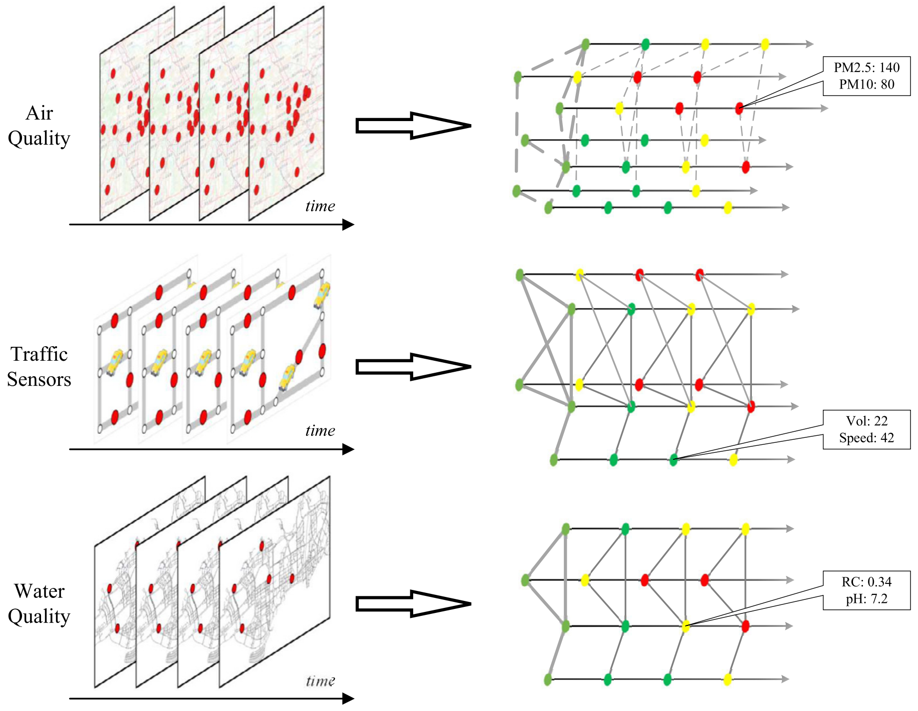

Fig. 1. Various urban ST data and the related ST graphs.

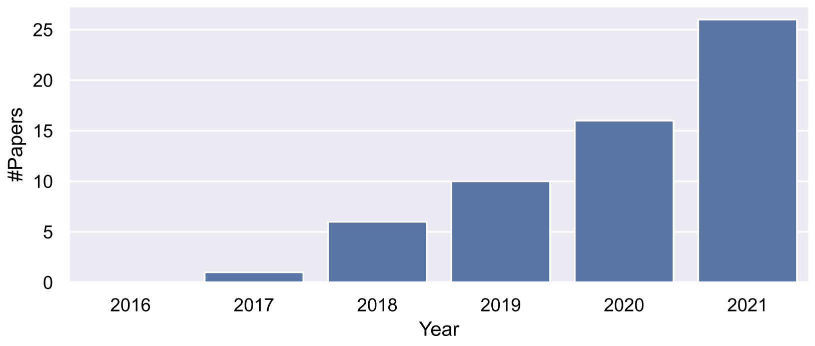

Fig. 2. Numbers of papers to model ST graph in recent years.

1. Introduction

Recent advances in ubiquitous computing and sensing techniques help collect a myriad of spatio-temporal (ST) data in urban areas, such as urban traffic data and air quality data. Such data have complex spatial and temporal correlations, which can be depicted by spatio-temporal graphs, as shown in Fig. 1. Predicting STGs foster many mission-critical applications, e.g., traffic prediction and climate forecasts [2], which are fundamental and essential to the development of intelligent cities.

In the recent five years, we have witnessed a dramatically growing trend of ST neural networks proposed at top-tier data-mining and AI venues for STG prediction (see the statistics in Fig. 2). To capture the spatial and temporal correlations, many representative models, such as [3–6], are carefully assembled from small network structures, which can be grouped into temporal networks (*e.g.*, convolution neural networks [7], recurrent neural networks [8], and temporal attention networks [9]) and spatial networks (*e.g.*, graph convolution networks and spatial attention networks). By their power in modeling ST correlations, these models have shown great promise in STG prediction.

Though effective, the above ST networks are carefully designed for certain tasks and subject to specific data types, formats, and distributions. Simply transferring such networks may cripple the model's performance. For example, the traffic conditions of different cities suffer heterogeneous spatial correlations. Metropolitans like Beijing and Shanghai usually encounter severe traffic jams that can diffuse to a wide range of areas (i.e., long-range spatial correlations). In contrast, the long-range spatial impacts in small cities are insignificant due to less traffic congestion. Because of such disparity, it is necessary to design neural networks for different tasks, which requires substantial domain knowledge and large amounts of

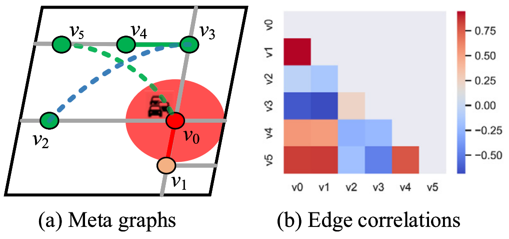

Fig. 3. Impacts of meta graphs on ST data.

experts' efforts, making it time-consuming and costly to popularize deep learning technology for various important applications. Considering these facts, one may ask: can we design a unified framework that can be easily generalized to various STG prediction tasks? The framework should produce suitable computational graphs for different spatio-temporal correlations.

Neural architecture search (NAS), originally proposed for image recognition and sequence modeling, enables us to search for optimal architectures automatically, thereby significantly reducing human efforts in manual network designs [10–12]. Afterward, adaptations of these techniques achieved much success in other domains, such as learning graphs [13,14] and forecasting grid-based ST data [15]. However, literature on applying NAS for STG modeling is still scarce due to the following three challenges:

Challenge 1: *Complex ST correlations*. In STG predictions, the future of a node is conditioned on its history and the previous readings of its neighbors [6]. In other words, it is extremely important to capture the complex correlations in both spatial and temporal domains in STGs. However, traditional search space schemes are designed for conventional data types, such as sequence, grid-based, and graph data, which cannot work satisfactorily in the NAS framework for STG prediction tasks.

Challenge 2: *Various meta graphs that impacts the ST correlations*. A meta graph describes spatial correlations between sensors in a specific view. In general, an STG is associated with a set of meta graphs (e.g., the road network, the geospatial distances, and the temporal pattern similarity), where the nodes and edges have meta features related to their characteristics and further impact the ST correlations. In detail, three major meta features pose such an impact:

1) *The explicit attributes of nodes and edges.* Taking Fig. 3 as an example. In Fig. 3(a), a real-world traffic example in METR-LA dataset [3] is presented, where nodes and edges denote sensors and relations between sensors, respectively. In this example, v_0 is located at a hub while v_2 is an intermediate node on a road network. Consequently, v_0 easily becomes congested, while v_2 is often unobstructed, demonstrating diverse node characteristics. Similarly, as edge $\langle v_0, v_1\rangle$ is much shorter than edge $\langle v_0, v_2\rangle$, node v_0 has stronger impacts on v_1 than on v_2 as depicted in Fig. 3(b), indicating diverse edge characteristics.

2) *The node similarities.* The similarities also reveal the node and edge characteristics, pointed out in other research works. For example, [16] proposed to construct similarity graphs for prediction enhancement. As presented in Fig. 3, though there is no direct edge between the node pairs $\langle v_0, v_5\rangle$ and $\langle v_2, v_3\rangle$, the correlations between the nodes remain strong.

3) *Dynamic spatial correlations.* There are dynamic spatial correlations in various spatio-temporal forecasting tasks. For example, the traffic patterns at the morning peak and the evening peak are entirely different, and the air pollutant transmissions vary under different weather conditions. Therefore, capturing such dynamic spatial correlations would help the prediction of spatio-temporal graphs.

All the features above can be regarded as meta graphs of an STG for effective ST correlation learning. Therefore, it is vital to comprehensively extract the meta graphs and model them simultaneously.

Challenge 3: *Knowledge extraction from meta graphs.* For a node/edge of an STG, a straightforward method to build its representation of the characteristics is to extract knowledge from its meta features [17]. However, such a solution may lead to degenerated performances in STG prediction tasks where strong ST correlations exist in nodes/edges, as it fails to incorporate the graph structure of STGs. The following two examples elaborate on the challenge: 1) the characteristics of a node are related to its neighbors and edges. In Fig. 3(a), v_1 and v_3 have very similar node attributes (*i.e.*, the density and structure of roads). However, as edge $\langle v_0, v_1\rangle$ is much shorter than $\langle v_0, v_3\rangle$, v_1 is more likely to witness traffic congestion than v_3, showing different node characteristics; and 2) Likewise, the characteristics of an edge are related to its two connecting nodes. In Fig. 3(a), edge $\langle v_0, v_1\rangle$ and $\langle v_3, v_4\rangle$ have similar attributes (i.e., length and width). However, v_0 and v_1 are easily blocked by traffic, while v_3 and v_4 are of easier traffic dispersion. Accordingly, in Fig. 3(b), edge $\langle v_0, v_1\rangle$ shows much stronger correlations than $\langle v3, v4\rangle$, indicating different edge characteristics. Hence, it is essential to learn characteristics considering the graph structure.

To tackle the above problems, we propose a NAS framework, entitled AutoSTG$^+$, for spatio-temporal graph prediction. To capture ST correlations, our search space is constructed from a candidate operation set, including *spatial graph convolution (SC)*, *temporal convolution (TC)*, and *fully connected layer (FC)*, as shown in Fig. 4(a) and (b). As ST correlations are related to meta graphs, we employ meta learning [17] to generate weight parameters of SC and TC from node and edge meta knowledge (*i.e.*, characteristics) of meta graphs, as shown in Fig. 4(a) and (c). Notably, a graph representation learning network is used to extract the meta knowledge by aggregating neighbors' information from meta graphs, as shown in Fig. 4(c). Our contributions are four-fold:

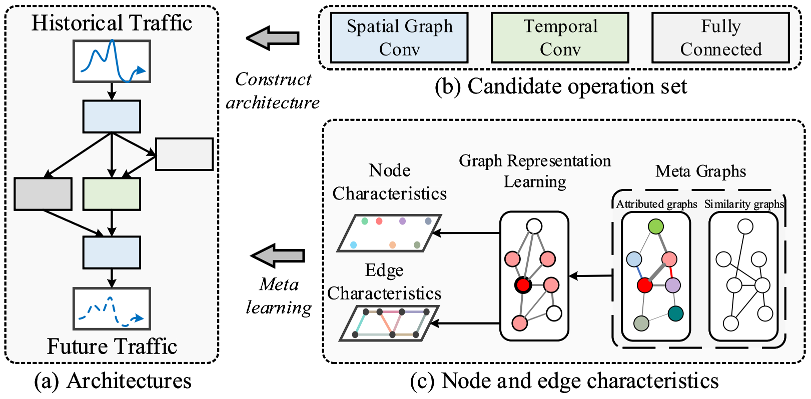

Fig. 4. Insights of our NAS framework.

- We proposed a novel framework, entitled AutoSTG$^+$, for automated spatio-temporal graph prediction. Specifically, spatial graph convolution and temporal convolution operations are adopted in our AutoSTG$^+$ to capture spatial and temporal correlations, respectively. To the best of our knowledge, we are the first to study NAS framework for modeling STGs.
- To better model the relationships between sensors, we propose to construct a set of meta graphs, including graphs of geographical connections, similarities between nodes, and corresponding dynamic graphs, which enhances the AutoSTG$^+$ with better generalization for the STGs without physical connections (e.g., air quality).
- To capture the ST correlations related to meta information, the meta learning technique is adopted to learn the adjacency matrices of spatial graph convolution layers and kernels of temporal convolution layers from the meta knowledge of the meta graphs. In particular, a graph meta knowledge learner, composed of node learners and edge learners, iteratively aggregates the neighbors' information of meta graphs to learn node and edge meta knowledge.
- We conduct extensive experiments on seven real-world datasets to verify our framework, including five traffic benchmark datasets, an air quality dataset, and a water quality dataset. These datasets come from different sources and possess various scales, verifying the unity of our framework for STG prediction. The experimental results demonstrate that our AutoSTG$^+$ can find effective neural architectures and achieve state-of-the-art prediction accuracy.

Compared with the original conference version [1], we extend this work from four major aspects.

- Since different spatio-temporal graphs have their unique characteristics, to better capture complex spatial correlations, we introduce the construction process for meta graphs, including the extraction of spatial similarity, temporal similarity, and feature similarity, and improve the structure of graph meta-knowledge learner to learn meta-knowledge from multiple meta graphs.
- To efficiently capture the dynamic spatial correlations between nodes, we propose a partial dynamic graph module termed *Vector Quantised Dynamic Graphs (VQDG)* to learn the representation from the dynamic temporal information and then generate the corresponding dynamic meta graph. The ablation studies show the effectiveness of the new proposed module.
- After an elaborate analysis of the distribution of different operators in the searched architecture, we have removed two operators from the searching space. We surprisingly find that it yields promising gains on real-world datasets. On spatio-temporal graph prediction tasks, adding constant skip-connections and removing Identity and Zero from the candidate operator set can gain stable improvement over all seven datasets.
- We conducted experiments on more STG datasets, including three extra traffic datasets, an air quality dataset, and a water quality dataset. The results show that the proposed model can achieve good results on new datasets with different scales and different types of meta graphs. It shows that AutoSTG$^+$ can handle many STG prediction tasks, and AutoSTG$^+$ would help us develop better intelligent applications for modern cities.

2. Preliminaries

In this section, we first delineate the related notations and the problem we study. The background of graph convolutions and the temporal similarity measurement is introduced as well. For brevity, frequently used notations are listed in Table 1.

2.1. Definitions and problem statement

Definition 1. Meta graph. A meta graph is defined as $\mathbb{G} = \left\{\mathbb{V}, \mathbb{E}, \mathbf{V}^{(0)}, \mathcal{E}^{(0)}\right\}$, where $\mathbb{V} = \left\{v_1, v_2, \cdots, v_{N_l}\right\}$ denotes the node set of the graph and $\left|\mathbb{V}\right| = N_l$; $\mathbb{E} = \left\{\langle v_i, v_j\rangle \mid 1 \le i, j \le N_l\right\}$ indicates the edge set that represent the proximity between

Table 1
Frequently used notation.

Notation	Description
N_l, N_t	Number of locations/timestamps.
S	Number of meta graphs.
$N_{\text{in}}, N_{\text{out}}$	Number of timestamps in history/future.
$\mathbf{X}_t$	The sensor readings at timestamp t.
$\mathbf{V}^{(m)}, \mathcal{E}^{(m)}$	The node/edge representation at m-th iteration.
$\mathcal{A}$	The adjacency matrices of graph convolution.
$\mathbb{K}$	The kernels of temporal convolution.

different locations; $\mathbf{V}^{(0)} \in \mathbb{R}^{N_l \times D}$ denotes D dimensional attribute values for nodes, and $\mathcal{E}^{(0)} \in \mathbb{R}^{N_l \times N_l \times K}$ denotes K dimensional attribute values for edges, respectively. $\mathbf{V}^{(0)}$, $\mathcal{E}^{(0)}$ represents the original attributes of nodes and edges, $\mathbf{V}^{(1)}$, $\mathcal{E}^{(1)}$ denotes the features of nodes and edges after the first iteration of a graph meta-knowledge learner (Section 3.3), and so on. Note that if there is no edge between the i-th and j-th node (i.e., $\langle v_i, v_j \rangle \notin \mathbb{E}$), its attribute values in $\mathcal{E}^{(0)}$ are set as zero. In addition, we use $\mathbb{N}_i = \left\{ j \mid \langle v_i, v_j \rangle \in \mathbb{E} \right\}$ to denote the neighbors of node i. An STG is usually associated with several meta graphs, and we denote the set of meta graphs as $\mathcal{G} = \left\{ \mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_S \right\}$ with $|\mathcal{G}| = S$.

Definition 2. STG prediction. Given the readings of previous N_{in} time steps for N_l nodes $\left[\mathbf{X}_1, ..., \mathbf{X}_{N_{\text{in}}}\right] \in \mathbb{R}^{N_{\text{in}} \times N_l \times D_{in}}$ and the attributed graph $\mathbb{G}$, we aim to predict the future readings of the next N_{out} readings $\left[\hat{\mathbf{Y}}_1, ..., \hat{\mathbf{Y}}_{N_{\text{out}}}\right] \in \mathbb{R}^{N_{\text{out}} \times N_l \times D_{out}}$. The notations D_{in} and D_{out} refer to the number of feature dimensions of the input and the output, respectively.

Problem 1. NAS for STG prediction. We aim to find a neural architecture for STG prediction, which learns from training dataset $\mathbb{D}_{\text{train}}$ and achieves the minimum loss on validation dataset $\mathbb{D}_{\text{val}}$:

$$\begin{aligned} \min_{\Lambda} \quad & \mathrm{L}\left(\Theta^*(\Lambda), \Lambda, \mathbb{D}_{\text{val}}\right), \\ \text{s.t.} \quad & \Theta^*(\Lambda) = \arg\min_{\Theta} \mathrm{L}\left(\Theta, \Lambda, \mathbb{D}_{\text{train}}\right), \end{aligned} \tag{1}$$

where L, Θ, Λ denote loss function, network weight parameters, and architecture parameters, respectively.

2.2. Graph convolution

Graph convolution [18] is the dominant operator for learning node representations in a graph, achieving state-of-the-art performance in many graph-related applications, such as node classification and link prediction [19]. [3] further proposed diffusion convolution for more accurate modeling of spatial correlations in urban traffic modeling by considering both upstream and downstream impacts. In light of this work, we employ diffusion convolution to implement graph convolution.

Formally, given node feature $\mathbf{H} \in \mathbb{R}^{N_l \times D}$ and K adjacency matrices $\mathcal{A} = [\mathbf{A}_1, \cdots, \mathbf{A}_K] = \left(a_{kij}\right)_{K \times N_l \times N_l}$ as inputs, diffusion convolution outputs node state $\mathrm{DC}(\mathbf{H}, \mathcal{A}, \mathbb{W}) \in \mathbb{R}^{N_l \times D'}$, computed by:

$$\mathrm{DC}\left(\mathbf{H}, \mathcal{A}, \mathbb{W}\right) = \sum_{k=1}^{K} \sum_{p=1}^{P} \left(\mathbf{A}_k\right)^p \mathbf{H} \mathbf{W}_{kp}, \tag{2}$$

where P is a constant denoting the number of diffusion steps, $(\mathbf{A}_k)^p$ is the power series of diffusion matrix $\mathbf{A}_k$, and $\mathbb{W} = \left\{ \mathbf{W}_{kp} \in \mathbb{R}^{D \times D'} \right\}$ is the set of weight matrices for feature learning.

In STG prediction, the adjacency matrices $\mathcal{A}$ can be constructed from edge representation $\mathcal{E} = \left(e_{ijk}\right)_{N_l \times N_l \times K}$ by function $\mathrm{DM}\left(\mathcal{E}\right) = \left(a_{kij}\right)_{K \times N_l \times N_l}$. As the adjacency matrices represent the diffusion probability on the graph, we adopt the softmax function to compute each a_{kij} by normalizing $\mathcal{E}$:

$$a_{kij} = \begin{cases} \dfrac{\exp(e_{ijk})}{\sum_{j' \in \mathbb{N}_i} \exp\left(e_{ij'k}\right)}, & \langle v_i, v_j \rangle \in \mathbb{E}, \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

2.3. Temporal similarity measurement

In most cases, it is non-trivial to explicitly acquire the physical connections between nodes, such as in air quality prediction. Another commonly used method is to build meta graphs based on the similarity between the readings of sensors. Dynamic time warping (DTW) is mainly suitable for computing the distance between two time series since it has a tolerance for common issues in time-series analysis, e.g., shift, insertion, and deletion [20].

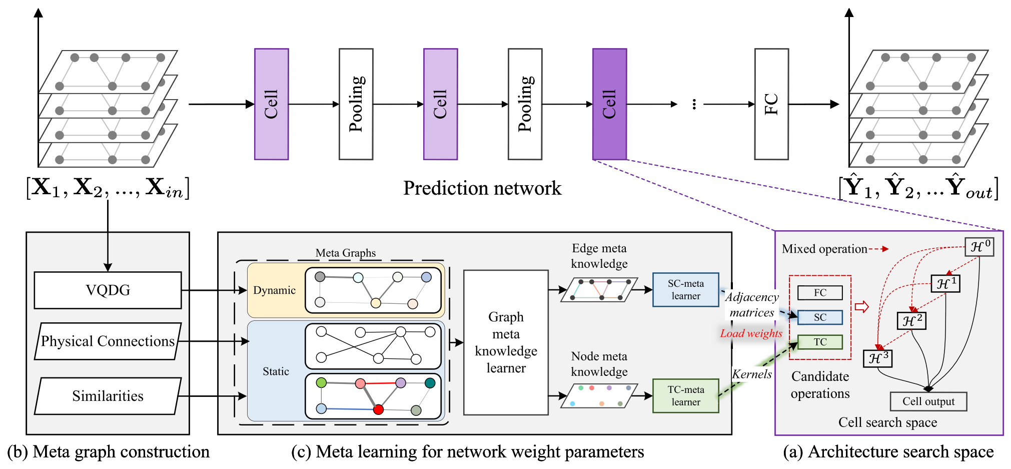

Fig. 5. Framework Overview of AutoSTG⁺. Physical connections and similarities (including spatial features and temporal patterns) are used to construct static meta graphs. Dynamic meta graphs are generated by the encoding of temporal inputs via the VQDG module. Then, graph meta knowledge learners extract the node and edge meta knowledge from meta graphs. Meta learning techniques are used to generate the kernels and adjacent matrices of TC and SC operators. And the gradient-based searching algorithm is used to optimize the structure of each cell in the prediction network.

DTW distance between two time-series $\mathbf{a}[1..i]$ and $\mathbf{b}[1..j]$ is computed as

$$\mathrm{DTW}(\mathbf{a}[1..i], \mathbf{b}[1..j]) = \mathrm{dist}(\mathbf{a}[i], \mathbf{b}[j]) + \min \begin{cases} \mathrm{DTW}(\mathbf{a}[1..i-1], \mathbf{b}[1..j-1]) \\ \mathrm{DTW}(\mathbf{a}[1..i-1], \mathbf{b}[1..j]) \\ \mathrm{DTW}(\mathbf{a}[1..i], \mathbf{b}[1..j-1]) \end{cases},$$

where $\mathbf{a}[i]$ (or $\mathbf{b}[i]$) denotes the ith element of the time series $\mathbf{a}$ (or $\mathbf{b}$), and the operator $\mathrm{dist}(\cdot,\cdot)$ is the distance function. In particular, the DTW distance between an empty series (i.e., $i, j = 0$) and any other time series is zero.

By computing the DTW distance between the readings of nodes, we can obtain the implicit relationships between nodes, allowing us to perform graph convolutions to capture complex ST correlation among nodes.

3. Methodologies

Our framework (AutoSTG+) consists of three major processes: 1) constructing architectures from the search space as shown in Fig. 5(a); 2) constructing multiple meta graphs as shown in Fig. 5(b); and 3) employing meta learning techniques to learn the weights of SC and TC layers in the built architectures, as shown in Fig. 5(c). Specifically in Fig. 5(c), the meta-learning part employs a graph meta knowledge learner to extract node and edge meta knowledge from the meta graphs and then uses SC-meta learner and TC-meta learner to generate the adjacency matrices of SC layers and kernels of TC layers from the extracted edge meta knowledge and node meta knowledge in the constructed architectures, respectively. In the following subsections, we first introduce the search space for architecture construction in Section 3.1. Then we present how to construct the meta graphs of an STG in Section 3.2. Next, we describe the graph meta knowledge learner and meta learners in Section 3.3. And finally, we show the optimization algorithm in Section 3.5.

3.1. Search space for architecture construction

Inspired by the existing convolution-based neural architectures [4,6] that have been proven effective in STG prediction, we design a convolution-based search space for our framework. As shown in Fig. 5(a), our prediction network is composed of a series of cells and temporal pooling layers, where the cells are employed for modeling ST correlations and the temporal pooling layers (*e.g.*, average pooling in the temporal domain) are adopted to increase the temporal receptive fields of hidden states. All the outputs of cells and pooling layers are aggregated by shortcut connections for modeling multi-scale ST correlations. Then a fully-connected (FC) layer is adopted to predict the next N_{out}-timestamp values. Initially, these cells have undetermined architectures, and our goal is to search the optimal architectures for them.

Following DARTS framework [10], the search space of each cell is a direct acyclic graph consisting of N_v vertices. Each vertex denotes a latent representation of the STG, i.e., $\mathcal{H}^i \in \mathbb{R}^{T\times N_l \times D}$, where T is the number of timestamps and D is the number of features (T and D are fixed in each cell). The cell input is $\mathcal{H}^0$, while the output is the sum of all intermediate representations $\sum_{i=0}^{N_v} \mathcal{H}^i$. Each vertex pair (i, j) is associated with a candidate operation set $\mathbb{O} = \{o_1, o_2, \cdots\}$ (*i.e.*, a function set) that transforms $\mathcal{H}^i$, weighted by architecture parameters $\boldsymbol{\alpha}^{(i,j)} = \{\alpha_o^{(i,j)} \mid o \in \mathbb{O}\}$. The representation of a vertex can be computed based on all its predecessors:

$$\mathcal{H}^j = \sum_{i<j} \sum_{o \in \mathbb{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathbb{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o\left(\mathcal{H}^i\right). \tag{4}$$

For each pair (i, j), the operation with the highest score $\alpha_o^{(i,j)}$ in $\boldsymbol{\alpha}^{(i,j)}$ is selected as the operation in the final architecture.

To capture ST correlations, spatial graph convolution and temporal convolution are included in our candidate operation set $\mathbb{O}$:

- *Spatial Graph Convolution.* We use diffusion convolution to capture spatial correlations of STGs. Formally, suppose that the input hidden state is $\mathcal{H} = [\mathbf{H}_1, \cdots, \mathbf{H}_T] \in \mathbb{R}^{T \times N_l \times D}$, the input adjacency matrices are denoted as $\mathcal{A}$, and the learnable weights are denoted as $\mathbb{W}$. The spatial graph convolution function is defined as $\mathrm{SC}\left(\mathcal{H}, \mathcal{A}, \mathbb{W}\right) = \mathcal{H}' = \left[\mathbf{H}'_1, \cdots, \mathbf{H}'_T\right] \in \mathbb{R}^{T \times N_l \times D}$, which employs diffusion convolution on each $\mathbf{H}_i$, expressed as:

$$\mathbf{H}'_i = \mathrm{DC}\left(\mathbf{H}_i, \mathcal{A}, \mathbb{W}\right), \tag{5}$$

  where the computation of function $\mathrm{DC}(\cdot)$ and the constraints of the related parameters (*e.g.*, $\mathbb{W}$) refer to Equation (2).
- *Temporal Convolution.* We employ a temporal convolution on each node to capture temporal correlations of STGs. Suppose the input hidden state $\mathcal{H} \in \mathbb{R}^{T \times N_l \times D}$ is composed of $\left\{\mathbf{H}_1, \cdots, \mathbf{H}_{N_l}\right\}$, where each $\mathbf{H}_i \in \mathbb{R}^{T \times D}$ denotes the hidden state of node i. Then, given the input convolution kernels $\mathbb{K} = \left\{\mathcal{K}_1, \cdots, \mathcal{K}_{N_l}\right\}$, the temporal convolution function $\mathrm{TC}\left(\mathcal{H}, \mathbb{K}\right) = \mathcal{H}'$, where $\mathcal{H}' \in \mathbb{R}^{T \times N_l \times D}$ is composed of $\left\{\mathbf{H}'_1, \cdots, \mathbf{H}'_{N_l} \mid \mathbf{H}'_i \in \mathbb{R}^{T \times D}\right\}$. Then, the output hidden state of node i is computed by:

$$\mathbf{H}'_i = \mathbf{H}_i \star \mathcal{K}_i, \tag{6}$$

  where $\star$ denotes 1D convolution [18] for sequence modeling.

Furthermore, the recent works [21,22] point out that the skip-connections (*i.e.*, the *Identity* operation) have unfair advantages in competing with other operations. It makes the optimization algorithm increase the weights of skip connections, leading to an easier back-propagation of the gradients. However, this trend may cause many operators to collapse into skip connections, reducing the capacity of the final network. To solve this issue, we exclude the two common operations (i.e., *Zero* and *Identity*) from the candidate operation set used in the original conference version, i.e., AutoSTG.

3.2. Construction of meta graphs

Since an STG is affected by multiple complex factors, it is necessary to model the relationship between nodes from different views, i.e., the meta graphs. In this section, we enhance AutoSTG with an additional method of meta graphs construction to consider such impacts. There are two main types of meta-graphs in the set: 1) physical connections and 2) similarity graphs.

3.2.1. Physical connections

The first type of meta-graph is about physical connections. Physical connections, e.g., roads and water pipes, usually have several attributes, such as the length, width, number of lanes, and level of the road, the diameter, material, and service life of the pipe. These attributes show us the basic relationship between nodes and describe the transmission capacity of physical connections [23]. Moreover, some multivariate time-series prediction tasks without explicit physical connections can also be formulated as STG prediction tasks [24,16]. In this case, we can describe the physical connection by the geospatial distance between nodes.

3.2.2. Similarity graphs

Physical connections only describe how the sensors are connected, which are not enough for capturing complex ST correlations in an STG. Implicit relationships between nodes also contain essential information. For example, because of their complementary functions, the traffic in residential and office areas is highly correlated, especially at the peak time. Moreover, this situation also occurs in the readings of the air quality monitoring stations located in the most frequent wind direction. Therefore, we construct extra similarity meta graphs to better model the complex ST correlations of an STG from two aspects:

Spatial similarity The spatial attributes of nodes describe the characteristics of the node itself and the surrounding environment, and similar node attributes mean that the nodes locate in similar environments, perform similar roles, and have similar readings. Therefore, we can build a spatial similarity meta-graph based on the node attributes and apply cosine similarity to measure the difference between the attributes of the two nodes.

Temporal pattern similarity Due to the similar or complementary functions among the locations, there would be a strong correlation between some distant nodes. Therefore, it is essential to construct a meta-graph based on the temporal

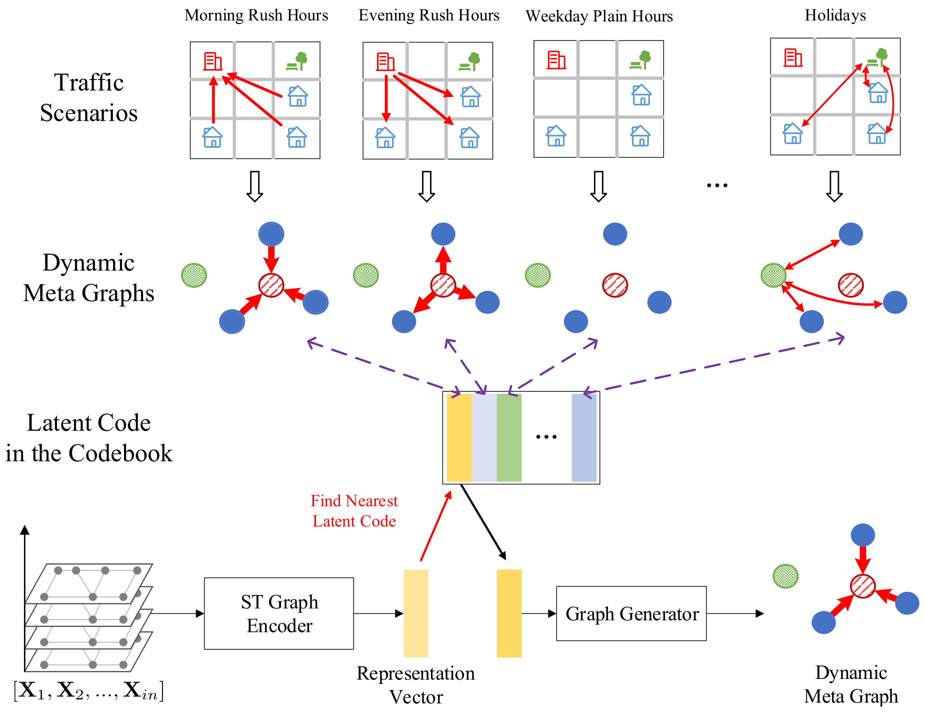

Fig. 6. The Vector Quantised Dynamic Graphs (VQDG) module. The top part shows different traffic scenarios in the real world. For example, during the morning rush hours, people depart for the business area from their homes and usually come back during the evening rush hours. And people usually go to the park for a rest in the holidays. Such frequent patterns can be described by corresponding dynamic meta graphs, as shown in the figure, where blue solid nodes represent residential areas, red slash nodes represent business areas, and green light ones represent parks. In VQDG, there are several latent codes in the codebook, and the graph generator is learned to generate the adaptive dynamic meta graph from a given latent code. Moreover, the ST graph encoder employs a CNN-based temporal encoder and a graph attention network to learn the representation of temporal inputs. Then, we find the nearest code from the codebook and use it to generate the dynamic meta graph. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

pattern similarity. DTW distance is a useful metric to measure the temporal pattern distance because it computes the distance based on the optimal matching of the time series and has a large tolerance for the acceleration and deceleration issues in the time series.

3.2.3. Vector quantised dynamic graphs

The spatial correlations between sensors are usually highly dynamic. For example, peoples depart from home and rush to office buildings and factories in the morning and go back in the evening. The traffic patterns in the morning peak and the evening peak are pretty different. The diffusion process of air pollutants varies under different weather conditions. However, the meta graph constructions mentioned above, including physical connections and similarities, are all static, which are hard to learn the dynamic spatial correlations of an STG. Therefore, we try to introduce some dynamic graphs for graph meta knowledge learning.

Some existing works try to model dynamic correlations. ST-MetaNet+ [23] introduces a dynamic context learner and fusion gates to generate the weights of networks with both static and dynamic information. However, such a solution faces a critical issue. Since the high dynamic traffic information and the non-linearity of the context learner, even a small fluctuation in the input traffic tensor may lead to entirely different adjacent matrics, which makes it hard to train the neural network. Self-attention mechanisms introduced in [25] have shown great power in modeling dynamic correlations between tokens, but the quadratic time complexity of self-attention mechanisms limits their applications on large spatio-temporal graphs.

Moreover, in most common spatio-temporal graph prediction tasks, e.g., traffic prediction and crowd prediction, there are only several similar patterns related to different seasons, weekdays or weekends, morning or evening, and weather conditions. Therefore, there is a trivial idea, i.e., we can use a module to classify the input spatio-temporal graph and use the corresponding dynamic meta graph for further graph meta knowledge learning. However, this trivial idea is hard to be implemented since the gradients cannot backpropagate from the graph generator to the spatio-temporal graph classifier directly.

Inspired by VQ-VAE [26], we developed our partial dynamic graph module, i.e., Vector Quantised Dynamic Graphs (VQDG). As shown in Fig. 6, VQDG first uses a spatio-temporal graph encoder to compute a representation vector of the spatio-temporal graph. Then, it chooses the nearest neighbor of the vector from a learnable representation dictionary. Next, we use the nearest neighbor to generate the edge and node features for the dynamic meta graph.

More specifically, in the spatio-temporal graph encoder, we employ a CNN-based temporal encoder to learn the temporal representations of each node and then use a graph attention network to aggregate the information from nodes and generate the representation vector of the spatio-temporal graph, denoted as $\boldsymbol{r}_g \in \mathbb{R}^{D_l}$. After that, we find the nearest latent code $\boldsymbol{c}^\star \in \mathbb{R}^{D_l}$ from the codebook, where the distance between two vectors is measured by the **Euclidean** distance. Then, the latent code is fed into a graph generator to get the node features $\mathbf{V}_d^{(0)}$ and edge features $\mathcal{E}_d^{(0)}$.

The graph generator generates $\mathbf{V}_d^{(0)}$ and $\mathcal{E}_d^{(0)}$ in a two-step way. First, we use a fully-connected layer to generate the node features $\mathbf{V}_d^{(0)}$. Then, for each pair of node $\langle i, j\rangle$, we concatenate the node features of node i and j to generate the edge features of $\langle i, j\rangle$ by a fully-connected network, which shares weights for each edge in the graph and can be implemented by a 1x1 convolutional network.

The codebook size N_c and the latent code dimension D_l are hyperparameters. Note that we use a trick in the implementation, i.e., directly copy the gradients of $\boldsymbol{c}^\star$ to $\boldsymbol{r}_g$,[3] so that we can train the spatio-temporal graph encoder and the graph generator simultaneously. Additionally, to make the representation vector close to the latent code in the codebook, there will be an extra loss

$$\mathcal{L}_{VQDG} = \|\boldsymbol{r}_g - [\boldsymbol{c}^\star]\|_2 + \frac{1}{4}\|[\boldsymbol{r}_g] - \boldsymbol{c}^\star\|_2,$$

where $[\cdot]$ is a constant mark, which would be regarded as a constant during the backpropagation, it makes $\boldsymbol{r}_g$ and $\boldsymbol{c}^\star$ approach each other at different speeds, with the $\boldsymbol{r}_g$ moving faster. This ratio (4:1) is the same as that in [26].

3.3. Graph meta knowledge learner

In STG prediction, the ST correlations of data are related to the characteristics of the set of meta graphs $\mathcal{G}$. Therefore, it is essential to learn the representations, namely the meta knowledge, of nodes and edges from $\mathcal{G}$. Previously, [17] employed FC layers to learn node and edge meta knowledge, respectively, from node and edge attributes. However, such a method ignores the impacts of the graph structure. To tackle this problem, we propose a graph meta-knowledge learner, which iteratively learns nodes' and edges' representations by aggregating their neighbors' information on $\mathcal{G}$.

Take a single meta graph $\mathbb{G} = \left\{\mathbb{V}, \mathbb{E}, \mathbf{V}^{(0)}, \mathcal{E}^{(0)}\right\} \in \mathcal{G}$ as an example. As shown in Fig. 7, we apply M iterations to learn node and edge representations by node learners and edge learners. Let $\mathbf{V}^{(m)}$ and $\mathcal{E}^{(m)}$ denote the node and edge representations at m-th iteration, respectively. The inputs are node and edge attributes, denoted as $\mathbf{V}^{(0)}$ and $\mathcal{E}^{(0)}$, and the outputs are node and edge meta knowledge, denoted as $\mathbf{V}^{(M)}$ and $\mathcal{E}^{(M)}$. The details of node learners and edge learners are as follows:

Node Learner. As the characteristics of a node are related to its neighbors and connecting edges, we employ graph convolution to learn each node's representation by aggregating its neighbors' information through its edges. At m-th iteration, the node learner takes the representations of the previous iteration, *i.e.* $\mathbf{V}^{(m-1)} \in \mathbb{R}^{N_l \times D}$ and $\mathcal{E}^{(m-1)} \in \mathbb{R}^{N_l \times N_l \times K}$ as inputs, and then returns the higher-level node representations $\mathbf{V}^{(m)} \in \mathbb{R}^{N_l \times D'}$. Without loss of generality, we employ diffusion convolution as a concrete example to show the implementation of node learner.

The first step is to compute adjacency matrices of graph convolution. In our work, we use edge representations to generate the adjacency matrices $\mathcal{A}$ by the formula: $\mathcal{A} = \mathrm{DM}\left(\mathcal{E}^{(m-1)}\right)$. The computation of this function refers to Equation (3). Then the diffusion convolution can be employed for $\mathbf{V}^{(m-1)}$ to get the higher-level node representations by Equation (2), which can be expressed as:

$$\mathbf{V}^{(m)} = \mathrm{ReLU}\left(\mathrm{DC}\left(\mathbf{V}^{(m-1)}, \mathcal{A}, \mathbb{W}^{(m)}\right)\right), \tag{7}$$

where $\mathbb{W}^{(m)}$ denotes the learnable parameters of this node learner, and the constraint of $\mathbb{W}^{(m)}$ is illustrated in Section 2.2. In this way, we can learn the representations of each node by aggregating neighbors' information through the connected edges. Therefore, the attributes and structure of the attributed graph are considered in learning node representations.

Edge Learner. For each edge, its characteristics are related to its connected nodes. Thus, for each edge, we push the representations of its two connected nodes to this edge and then use an FC layer to learn the higher-level edge representation. At m-th iteration, given node representations $\mathbf{V}^{(m-1)} = \left[\boldsymbol{v}_1^{(m-1)}, \ldots, \boldsymbol{v}_{N_l}^{(m-1)}\right] \in \mathbb{R}^{N_l \times D}$ and edge representations $\mathcal{E}^{(m-1)} \in \mathbb{R}^{N_l \times N_l \times K}$ as inputs, edge learner outputs $\mathcal{E}^{(m)} \in \mathbb{R}^{N_l \times N_l \times K'}$ to represent the higher-level edge representations. Specifically, $\mathcal{E}^{(m)}$ is composed of the representation of each edge, *i.e.* $\{\boldsymbol{e}_{ij}^{(m)} \in \mathbb{R}^{K'} \mid \langle v_i, v_j\rangle \in \mathbb{E}\}$ (note that for $\langle v_i, v_j\rangle \notin \mathbb{E}$, $\boldsymbol{e}_{ij}^{(m)}$ is set as zero). Then the higher-level edge representation for edge $\langle v_i, v_j\rangle$ is computed by:

$$\boldsymbol{e}_{ij}^{(m)} = \mathrm{ReLU}\left(\mathbf{W}^{(m)}\left(\boldsymbol{v}_i^{(m-1)} \parallel \boldsymbol{v}_j^{(m-1)} \parallel \boldsymbol{e}_{ij}^{(m-1)}\right) + b^{(m)}\right), \tag{8}$$

[3] PyTorch version: `r + (c - r).detach()`.

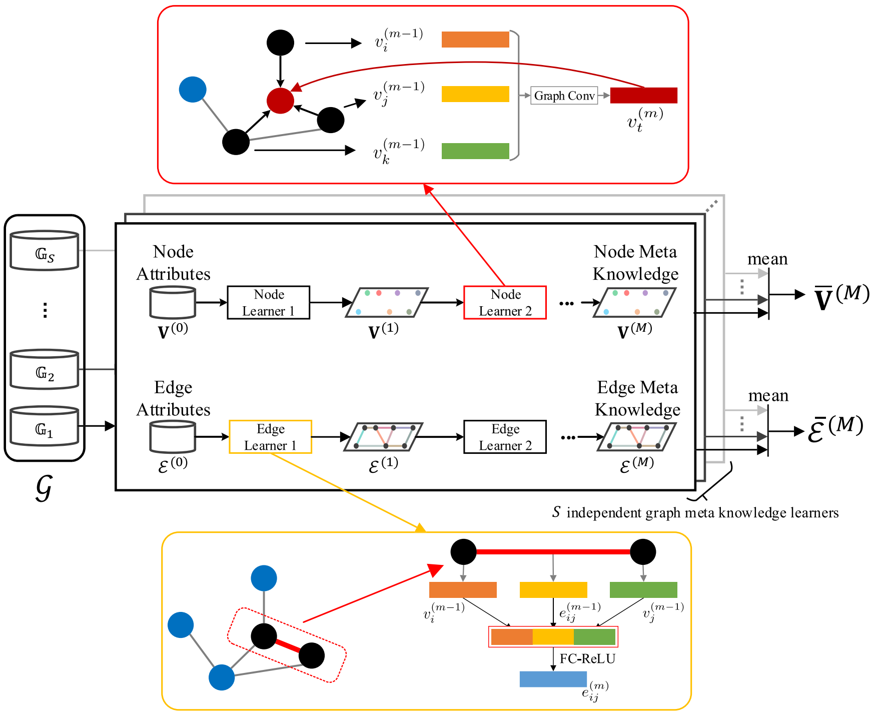

Fig. 7. Structure of graph meta knowledge learner. Differing from the conference version that only learns representations from one graph, AutoSTG+ establishes multiple meta graphs to learn more useful meta knowledge from various aspects.

where $\|$ denotes vector concatenation, $\mathbf{W}^{(m)} \in \mathbb{R}^{K' \times (2D+K)}$ is the weight matrix and $b^{(m)} \in \mathbb{R}$ is the bias, respectively. In this way, both the attributes and structure of the attributed graph can be modeled for learning edge representations.

Our graph meta knowledge learner has M iterations, and the final output representations will be used as the meta knowledge for modeling ST correlations. In particular, the receptive field of each node or edge increases exponentially with respect to M.

Thus, the selection of hyper-parameter M refers to how far the characteristics of nodes and edges can impact each other according to the real-world datasets. In summary, our graph meta knowledge learner can extract node and edge meta knowledge by aggregating neighbors' information, such that it can tackle the complex impacts of attributes and graph structure.

Similarly, we can get the edge meta knowledge $\mathcal{E}_1^{(m)}, \ldots, \mathcal{E}_S^{(m)}$ and the node meta knowledge $\mathbf{V}_1^{(m)}, \ldots, \mathbf{V}_S^{(m)}$ of the meta graphs by employ independent graph meta learners for each graph, where S is the number of meta graphs. Then, we get the edge meta knowledge $\bar{\mathcal{E}}^{(m)}$ and node meta knowledge $\bar{\mathbf{V}}^{(m)}$ by

$$\bar{\mathcal{E}}^{(m)} = \text{mean}\left(\mathcal{E}_1^{(M)}, \ldots, \mathcal{E}_S^{(m)}\right), \tag{9}$$

$$\bar{\mathbf{V}}^{(m)} = \text{mean}\left(\mathbf{V}_1^{(m)}, \ldots, \mathbf{V}_S^{(m)}\right). \tag{10}$$

3.4. Meta learners

As ST correlations of STGs are impacted by the characteristics of the attributed graph, we propose to learn the adjacency matrices of SC layers and kernels of TC layers from the meta knowledge of the attributed graph by meta learners. In this subsection, we introduce the meta learners for SC layers and TC layers in detail.

SC-Meta Learner. As spatial correlations are impacted by edge meta knowledge, we employ SC-meta learners to learn adjacency matrices from the edge meta knowledge $\bar{\mathcal{E}}^{(M)} \in \mathbb{R}^{N_l \times N_l \times K}$, which consists of $\left\{\bar{\boldsymbol{e}}_{ij}^{(M)} \in \mathbb{R}^K \mid \langle v_i, v_j \rangle \in \mathbb{E}\right\}$. First, it employs an FC layer to learn edge representation $\mathcal{E} \in \mathbb{R}^{N_l \times N_l \times K'}$, which is composed of $\left\{\boldsymbol{e}_{ij} \in \mathbb{R}^{K'} \mid \langle v_i, v_j \rangle \in \mathbb{E}\right\}$. Each vector $\boldsymbol{e}_{ij}$ is computed by:

$$\boldsymbol{e}_{ij} = \mathbf{W}_{\text{mat}} \bar{\boldsymbol{e}}_{ij}^{(M)} + b_{\text{mat}}, \tag{11}$$

where $\mathbf{W}_{\text{mat}} \in \mathbb{R}^{K' \times K}$, $b_{\text{mat}} \in \mathbb{R}$ are learnable weights and bias, respectively. Then Equation (3) is adopted to generate the adjacency matrices, *i.e.* $\mathcal{A} = \text{DM}(\mathcal{E})$. Finally, we can use these adjacency matrices to compute $\text{SC}\left(\mathcal{H}, \mathcal{A}, \mathbb{W}\right)$ according to Equation (5).

TC-Meta Learner. As temporal correlations depend on node meta knowledge, we employ TC-meta learner to generate the convolution kernels $\mathbb{K} = \left\{\mathcal{K}_1, ..., \mathcal{K}_{N_l}\right\}$ from $\bar{\mathbf{V}}^{(M)} = \left[\bar{\boldsymbol{v}}_1^{(M)}, ..., \bar{\boldsymbol{v}}_{N_l}^{(M)}\right] \in \mathbb{R}^{N_l \times D}$. Suppose each $\mathcal{K}_i$ has N_{ker} weights. We learn all these weights by an FC layer from vector $\boldsymbol{v}_i^{(M)}$, and then reshape the output vector to the tensor shape of $\mathcal{K}_i$, which can be expressed as:

$$\mathcal{K}_i = \text{reshape}\left(\mathbf{W}_{\text{ker}} \bar{\boldsymbol{v}}_i^{(M)} + b_{\text{ker}}\right), \tag{12}$$

where $\mathbf{W}_{\text{ker}} \in \mathbb{R}^{N_{\text{ker}} \times D}$, $b \in \mathbb{R}$ are learnable weight and bias, respectively. After generating kernels $\mathbb{K}$, we can adopt Equation (6) to compute $\text{TC}\left(\mathcal{H}, \mathbb{K}\right)$.

3.5. Searching algorithm

In our AutoSTG^{+}, all computations are differentiable. Thus we can employ a bi-level gradient-based optimization algorithm like DARTS [10], that updates network weight parameter set Θ (including the parameters in operations, graph meta knowledge learners, and meta learners) and architecture parameter set Λ (including the scores of candidate operations) alternately. As shown in Algorithm 1, we first construct datasets and then initialize all parameters (Line 1-2). After that, we alternately adopt a training dataset to update weight parameters (Line 4-6) and a validation dataset to update architecture parameters (Line 7-8) until the stopping criteria are met. At last, we can acquire the optimal architecture by selecting the candidate operations with the highest operation scores and adopting the training dataset to further train the network weight parameters of this neural architecture like normal neural networks (Line 10).

Algorithm 1: Optimization algorithm of AutoSTG^{+}.

```
   input : STG data $\left[\mathbf{X}_1, ..., \mathbf{X}_{N_t}\right]$, attributes $\mathbf{V}^{(0)}$ and $\mathcal{E}^{(0)}$
 1 Build $\mathbb{D}_{\text{train}}$, $\mathbb{D}_{\text{valid}}$ from $\left[\mathbf{X}_1, ..., \mathbf{X}_{N_t}\right]$, $\mathbf{V}^{(0)}$, and $\mathcal{E}^{(0)}$
 2 Initialize $\Theta$ and $\Lambda$
 3 do
 4     Sample $\mathbb{D}_{\text{batch}}$ from $\mathbb{D}_{\text{train}}$
 5     // For efficiency, we adopt first-order approximation
 6     $\theta \leftarrow \theta - \lambda_\Theta \nabla_\theta \text{L}\left(\Theta, \Lambda, \mathbb{D}_{\text{batch}}\right)$, $\forall \theta \in \Theta$ // $\lambda_\Theta$ is learning rate
 7     Sample $\mathbb{D}_{\text{batch}}$ from $\mathbb{D}_{\text{valid}}$
 8     $\alpha \leftarrow \alpha - \lambda_\Lambda \nabla_\alpha \text{L}\left(\Theta, \Lambda, \mathbb{D}_{\text{batch}}\right)$, $\forall \alpha \in \Lambda$ // $\lambda_\Lambda$ is learning rate
 9 until stopping criteria is met
10 Get the learned architecture, and further train it using $\mathbb{D}_{\text{train}}$
```

4. Evaluation

4.1. Experimental settings

4.1.1. Task descriptions

We conduct experiments on seven real-world STG prediction tasks, and Table 2 shows the data statistics:

- **PEMS-BAY [3]** contains the traffic speed readings of the 325 sensors collected by the California Transportation Agencies Performance Measurement System.
- **METR-LA [3]** consists of the traffic speed readings of the 207 sensors on the highway of Los Angeles County.
- **PEMS03 [27]** records the traffic speed readings of the 358 sensors from PeMS [28].
- **PEMS04 [27]** collects the traffic speed readings of the 307 sensors from PeMS [28].
- **PEMS08 [27]** is composed of the traffic speed readings of the 170 sensors from PeMS [28].
- **Air** includes 229 air quality monitoring stations in China's Beijing-Tianjin-Hebei region, 6 types of air pollutants, and meteorological conditions in the corresponding regions. Since the concentration of PM2.5 varies greatly and determines the air quality most times, we take it as the predicted target.
- **Water [29]** contains four main monitoring values of 14 water quality monitoring stations in a water supply network. We take RC (residual chlorine) as the target to predict.

For METR-LA and PEMS-BAY datasets, we adopt the GPS coordinates of sensors as node attributes and the road distance between sensors as edge attributes to build the physical connection graph. The edge attributes are determined by upstream

Table 2
Statistics of datasets.

Dataset	Time range	Period	#Timestamp	#Sensors
PEMS-BAY	2017-01-01∼2017-06-30	5 min	52116	325
METR-LA	2012-03-01∼2012-06-30	5 min	34272	207
PEMS03	2018-09-01∼2018-11-30	5 min	26208	358
PEMS04	2018-01-01∼2018-02-28	5 min	16992	307
PEMS08	2016-07-01∼2016-08-31	5 min	17856	170
Air	2015-01-01∼2019-12-31	1 h	43825	229
Water	2012-01-01∼2014-12-31	1 h	26304	14

and downstream road attributes like [3]. And for PEMS03, PEMS04, and PEMS08 datasets, we use the indegree and outdegree of each node as the node attributes and the road distance between sensors as edge attributes. For the above traffic prediction task, we aim to predict the traffic at the next 12 steps (i.e., 1 hour ahead) based on the historical observations from the past hour as well as the generated similarity graph, $\mathbb{G}$, and the constructed similarity graph. We utilize the same ratio for dataset partition [3], and [27], *i.e.*, the training, validation, and test sets are split by the ratio of 7:1:2 for METR-LA and PEMS-BAY and 6:2:2 for PEMS03, PEMS04, and PEMS08.

The node attributes of Air and Water datasets are both GPS coordinates. The edge attributes of Air dataset are the geographical distance between sensors, and we remove the connections between the sensors that were more than 150 km away since it is difficult for air pollutants to travel more than this distance within 12 hours under normal wind speed. And the edge attributes of Water dataset are the shortest paths of the pipe network between sensors. We predict the following 6-step target values (i.e., 6 hours) given the historical readings of the last 12 hours and the meta graphs. The training, validation, and test sets are split by the ratio of 6:2:2.

4.1.2. Metrics

Mean absolute error (MAE) and rooted mean square error (RMSE) are employed to evaluate our framework:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{Y}_i - Y_i\right|, \qquad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2},$$

where N is the number of instances, $\hat{Y}_i$ is the predicted value, and Y_i is the ground truth.

4.1.3. Baselines

We first compare AutoSTG$^+$ with the following STG prediction models for urban traffic:

- **GBRT**. Gradient Boosting Regression Tree is a non-parametric statistical learning method for regression problems. We train GBRTs to predict each future timestamp, where the previous traffic speed and node attributes are given as the inputs.
- **DeepAir**[30]. DeepAir considers various factors that affect air quality and designs a deep fusion network to aggregate the information of these factors and predict future air quality.
- **ST-MVMTL**[29]. It is a multi-view and multi-task learning framework to predict the water quality in the pipe network.
- **DCRNN**[3]. It combines diffusion convolution operations and sequence-to-sequence architecture to predict STGs.
- **Graph WaveNet**[6]. It uses graph convolution networks and WaveNet to model spatial and temporal correlations. A self-adaptive adjacency matrix is also learned to discover unseen graph structures from data without prior knowledge.
- **AutoSTG**[1]. It is the previous version of AutoSTG$^+$, only the geographic graph used in the graph meta knowledge learner.

In addition, we compare AutoSTG$^+$ with two basic NAS methods:

- **RANDOM**. We randomly sample neural architectures from our search space and train them from scratch for STG prediction.
- **DARTS** [10]. It employs continuous relaxation on candidate operations, enabling gradient-based optimization on network weight parameters and architecture parameters.
  Our experiments employ the basic DARTS framework (i.e., without graph meta knowledge learners and meta learners) on our search space for STG prediction.

4.1.4. Framework settings and training details

We add ReLU activation and Batch Normalization (BN) into SC and TC for non-linear projection and easy training in our implementation. The order is ReLU-X-BN, where X denotes SC or TC layer. In addition, we employ the operation sampling technique introduced in [31] to save the GPU memory consumption in the searching process. The hyper-parameter settings of our framework are the same for all seven datasets:

Table 3
Predictive performance on PEMS-BAY and METR-LA datasets.

Dataset	Metric	GBRT	DCRNN	Graph Wavenet	AutoSTG	AutoSTG+
METR-LA	MAE	3.86	3.04±0.01	3.05±0.01	3.02±0.00	**3.00±0.01**
	RMSE	7.49	6.27±0.03	6.16±0.03	6.10±0.01	**6.09±0.01**
PEMS-BAY	MAE	1.96	1.59±0.00	1.59±0.00	1.56±0.01	**1.54±0.00**
	RMSE	4.48	3.70±0.02	3.66±0.04	3.57±0.02	**3.56±0.00**
PEMS03	MAE	19.14	19.99±0.02	19.83±0.02	16.75±0.81	**14.69±0.07**
	RMSE	32.33	32.59±0.23	32.29±0.06	27.21±1.30	**24.88±0.84**
PEMS04	MAE	24.34	26.55±0.59	24.97±0.25	21.34±0.60	**18.82±0.06**
	RMSE	38.79	40.93±0.78	38.23±0.24	33.37±0.84	**30.46±0.14**
PEMS08	MAE	18.20	21.48±0.91	18.96±0.10	16.55±0.77	**14.95±0.07**
	RMSE	30.63	33.39±1.07	29.64±0.14	25.96±1.07	**23.99±0.03**

Table 4
Predictive performance on Air and Water datasets.

Air			Water		
Model	MAE	RMSE	Model	MAE	RMSE
GBRT	26.19	37.82	GBRT	5.76E-02	8.85E-02
DeepAir	15.27±0.42	22.79±0.23	ST-MTMVL	4.21E-02±9.96E-4	6.12E-02±1.04E-03
DCRNN	14.9±0.21	25.86±0.21	DCRNN	1.66E-1±2.65E-4	2.20E-1±2.38E-4
Graph Wavenet	14.35±0.12	25.86±0.21	Graph Wavenet	**3.11E-02±7.66E-4**	**4.83E-02±4.35E-4**
AutoSTG	13.57±0.10	24.22±0.12	AutoSTG	3.46E-02±1.12E-03	5.16E-02±1.81E-03
AutoSTG+	**11.85±0.09**	**21.27±0.21**	AutoSTG+	3.43E-02±1.01E-3	5.11E-02±9.99E-04

- The number of cells is 6.
- The number of vertices in each cell is 3.
- The number of cell hidden units is 64.
- The number of iterations in graph meta knowledge learner is 2.

Our framework is tested on CentOS 7 with a single Tesla V100 16GB GPU. The batch size is set as 32 for all the datasets. We adopt the Adam optimizer for 150 epochs (about 12 hours in total). After that, we reinitialize the optimizer and train the architecture for another 80 epochs (about 6 hours). In both the searching and training processes, the initial learning rate is 0.003, and it will multiply by 0.1 every 50 and 30 epochs in the searching and training processes, respectively.

4.2. Performance comparison

Table 3 and 4 show the performance of the baselines and our framework. We train and test deep learning models five times with different random seeds and present the results in the format: "mean±standard deviation". In particular, our AutoSTG$^+$ provides five architectures in each task by using different random seeds.

We first discuss the performance on the METR-LA and PEMS-BAY datasets (Table 3). On these two datasets, the non-deep-learning model, i.e., GBRT, achieves the worst performance on the two tasks since it only considers the human-crafted statistical features of input data. However, STGs are so complex that human prior knowledge cannot fully describe the ST correlations related to meta attributes. Thus, the non-deep-learning models have limitations on model expressiveness. DCRNN and Graph WaveNet use diffusion convolution for capturing spatial correlations. Particularly, edge attributes (road distance) are adopted to generate diffusion matrices by a pre-defined function. Such knowledge improves prediction accuracy. However, designing such pre-defined functions depends on the prior knowledge of datasets. As prior knowledge differs in different tasks and sometimes could be very complex, it is hard to find a perfect way to use these attributes. Unlike these baselines, AutoSTG can automatically learn neural architectures to capture ST correlations and achieve stable and competitive prediction accuracy compared with these expert-designed networks. AutoSTG$^+$, the improved version of AutoSTG, further improves and stabilizes the model performance by optimizing the search space and adding new graphs. It has achieved state-of-the-art results on the two datasets.

The results on PEMS03, PEMS04, and PEMS08 datasets are slightly different from METR-LA and PEMS-BAY datasets. The performance gap between the non-deep-learning model and the expert-designed models, i.e., DCRNN and Graph WaveNet, has been significantly reduced. A possible reason is that the spatial and temporal correlations of these three datasets are very different from the above two datasets. As a result, DCRNN and Graph WaveNet with default hyperparameters cannot capture spatio-temporal correlations well on these three datasets. Moreover, AutoSTG and AutoSTG$^+$ still perform well on these three datasets with the same hyperparameter setting, which shows that they can find the optimal network architectures on different datasets and achieve good results. That also proves that AutoSTG$^+$ can save expert efforts in real-world applications.

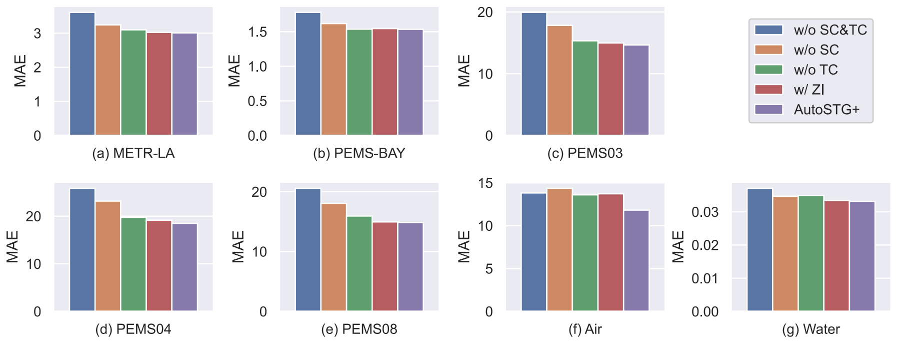

Fig. 8. Ablation studies on candidate operations.

Furthermore, the results on Air and Water datasets also show the advantages of our proposed model. Compared with GBRT, two domain-specific deep learning models, i.e., DeepAir and ST-MTMVL, significantly improve accuracy, indicating that well-designed models based on expert experience can achieve good results on specific tasks. DCRNN and Graph WaveNet can achieve better results without human-crafted spatial features by introducing graph structure and diffusion convolutional networks. Moreover, our proposed model, AutoSTG^{+}, can achieve the state-of-the-art result on the Air dataset and a competitive result on the Water dataset.

Note that AutoSTG+ uses the same configuration (including the number of cells, the number of candidate operations, and the learning rate) for all seven datasets. Water dataset contains only 14 nodes, while others have more than 100 sensors. Therefore, this search space may be too large for Water dataset, and the searched models are overfitted with such a configuration. Compared with other human-designed models, such a competitive result is acceptable.

4.3. Ablation studies

In this subsection, we conduct ablation studies for AutoSTG^{+} to validate the effectiveness of each component.

First, we evaluate the effectiveness of our candidate operation set. We compared our framework with three variants: 1) **w/o SC&TC**, which replaces SC and TC with FC operation in our candidate operation set; 2) **w/o SC**, which removes SC from the candidate operation set; 3) **w/o TC**, which removes TC from the candidate operation set; and 4) **w/ ZI**, which adds Zero and Identity connections to the candidate operation set. As shown in Fig. 8, both SC and TC can improve the prediction accuracy in modeling STGs. On the water quality prediction task, removing SC or TC alone similarly affects the prediction accuracy. While on the traffic datasets, SC contributes more to the final results, whereas TC shows minor improvement. The reason is that the temporal pooling layers and the fully connected layers can somehow capture temporal correlations by aggregating information in the temporal domain. In contrast, spatial correlations can only be captured by our SC operation. As a result, removing SC operations degrades performance significantly on the traffic datasets. Moreover, the importance of operations is different in the air quality prediction task. Fig. 8(f) illustrates that temporal convolution plays a more critical role in forecasting air quality. A possible reason is that unlike the number of vehicles on the road network is constant for a short time, air pollutants will degrade when transmitting from one sensor to another. Therefore, spatial convolution is less important than temporal convolution in the air quality prediction task. Furthermore, we also compare the new operation set with that presented in the conference version [1]. The reduced operation set can achieve better performance in all seven datasets.

Second, we evaluate the effectiveness of meta graphs. We compared the following variants: 1) **PC**, which only learns the graph meta-knowledge from physical connections (i.e., the road network or the geospatial distances between sensors); 2) **TPS**, which only learns the graph meta-knowledge from the temporal pattern similarity; 3) **PC+TPS**, which learns the graph meta-knowledge from both the road network and the temporal pattern similarity between sensors; 4) **PC+TPS+DG**, which learns the graph meta-knowledge from both the static and dynamic information. Fig. 9 demonstrates the prediction errors of three variants on PEMS03, PEMS04, and PEMS08 datasets. As shown in Fig. 9, the performance when using PC or TPS alone is unsatisfiable since they only represent partial spatial correlations. It also shows that the importance of these two meta graphs differs on different tasks. On PEMS03 dataset, PC plays a more important role, while TPS is more powerful on the other two datasets. Therefore, it is necessary to incorporate different meta graphs to capture the complex spatial correlations, and PC+TPS achieves better performance than the two variants above. Moreover, by capturing dynamic spatial correlations with the new proposed VQDG, PC+TPS+DG improves the predicting accuracy of all three datasets.

Third, we conduct experiments to verify the effectiveness of each framework component by using the following NAS methods: 1) **RANDOM**, which samples an architecture from our search space and then trains this architecture for prediction; 2) **DARTS**, which removes meta-learning method in our framework; and 3) **w/o graph**, which replaces the graph representation learning network (*i.e.* the graph meta knowledge learner) with the basic fully connected networks for representation learning of nodes and edges like [17]. As shown in Fig. 10, our AutoSTG^{+} performs significantly better than these vari-

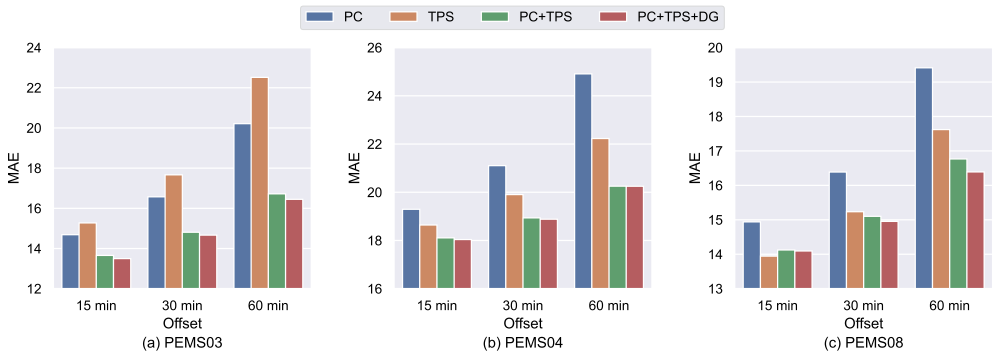

Fig. 9. Ablation studies on meta graphs.

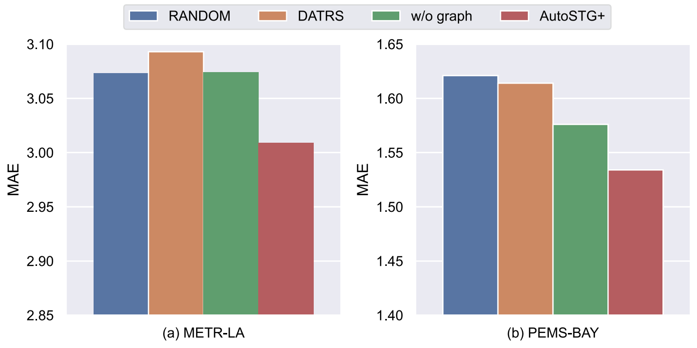

Fig. 10. Ablation studies on framework components.

ants that remove architecture search process (RANDOM), meta-learning technique (DARTS), or graph representation learning network (w/o graph), illustrating that all these components are effective to model STGs.

4.4. *Case study of graph meta knowledge*

To illustrate the effects of graph meta knowledge learners, we plot the heatmaps of adjacent matrics and node feature similarities on the PEMS08 dataset in Fig. 11.

Fig. 11(a) shows the adjacent matrix of physical connections, i.e., the road network in the PEMS08 dataset. This adjacent matrix is quite sparse, so it is hard for the network to learn spatial correlations from such an adjacent matrix. Therefore, AutoSTG$^+$ with only physical connections performs worse than others with temporal pattern similarities, as shown in Fig. 9(c).

Moreover, Fig. 11(b) shows the weights of the temporal pattern similarities. The higher similarity means that the two locations may perform similar roles in the road network. And Fig. 11(c) shows the pairwise Euclidean distances between learned node meta knowledge. A smaller distance means that the pair of nodes are more similar to each other. Fig. 11(b) and Fig. 11(c) show consistent results, which is reasonable since we use the node meta knowledge to generate the kernels of TC operators for capturing temporal correlations, and similar temporal patterns allow us to generate similar parameters for TC operators.

Furthermore, Fig. 11(d) shows the scaled weights of an adjacent matrix generated from the edge meta knowledge. It shows that the SC operators can aggregate the information from the nodes similar to itself and capture the spatial correlations.

4.5. *Empirical studies of learned architectures*

In general, the properties of datasets are different, and the neural architectures are diverse. To verify it, we study the properties of the traffic datasets.

We compute the correlation between any two sensors in these datasets, i.e., Pearson's correlation coefficient. The upper part of Fig. 12 shows the cumulative density functions of Pearson's correlation coefficients on these datasets. The ratios of sensor pairs with low correlation on METR-LA and PEMS-BAY datasets are larger than those of PEMS03, PEMS04, and PEMS08 datasets. That means that spatial dependencies on these two datasets are more complex than those on the other three. For example, when traffic congestion occurs and spreads over the road network, there would be delayed changes in sensor readings of the related road segments, which results in low Pearson coefficients for the readings of the two sensors.

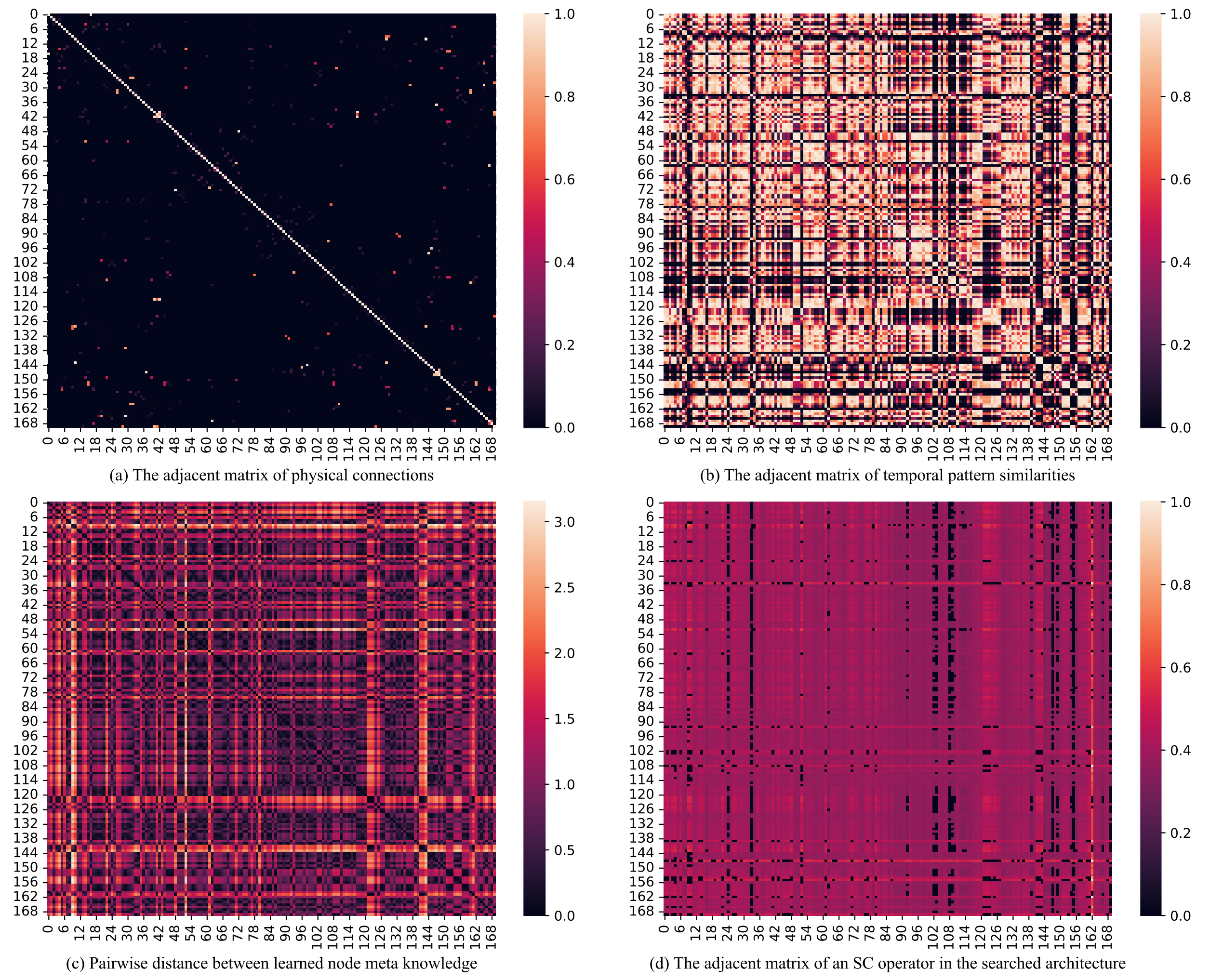

Fig. 11. The heatmaps of raw adjacent matrics, node meta knowledge similarity, and the generated adjacent matrix on PEMS08 dataset. (a)(b) are the normalized matrices fed into graph meta knowledge learners. (c) shows the pairwise Euclidean distance between two node meta knowledge vectors. (d) is an adjacent matrix used in the searched architecture, which is generated from the learned edge meta knowledge.

Such complex spatial correlation requires more spatial convolution operations to capture. As presented at the bottom of Fig. 9, the ratios of spatial convolution in the learned architectures decrease as the ratios of sensor pairs with low correlation decrease.

Therefore, our AutoSTG$^+$ can find customized architectures according to dataset properties.

4.6. *Studies of execution speed and memory cost*

Fig. 13 presents the execution speed and the maximum GPU memory occupation during the searching process on PEMS08 dataset. Since we remove two operators and narrow down the search space, AutoSTG$^+$ consumes less GPU memory than AutoSTG even if there are several new modules. Moreover, compared with AutoSTG, our new AutoSTG$^+$ costs about 14% extra time and reduces 10% predicting errors.

5. Related work

Deep Learning for Spatio-Temporal Graph. Deep learning was widely adopted in STG prediction. DCRNN leverages graph convolutional layers and gated recurrent units (GRU) to model both spatial and temporal dependencies [3]. However, since the recurrent network is hard to parallelize, it is easy to be the bottleneck of the model's efficiency. Recent models try to use parallel operators to capture temporal correlations. For example, STGCN employs gated 1D convolutional layers [4] while Graph WaveNet adopts distillated temporal convolutional layers [6].

For example, [4,6,32–34] combined convolutional neural networks and graph convolutional neural networks for modeling STGs, while some other studies [3,35,36] employed recurrent neural networks and graph convolutional neural networks for

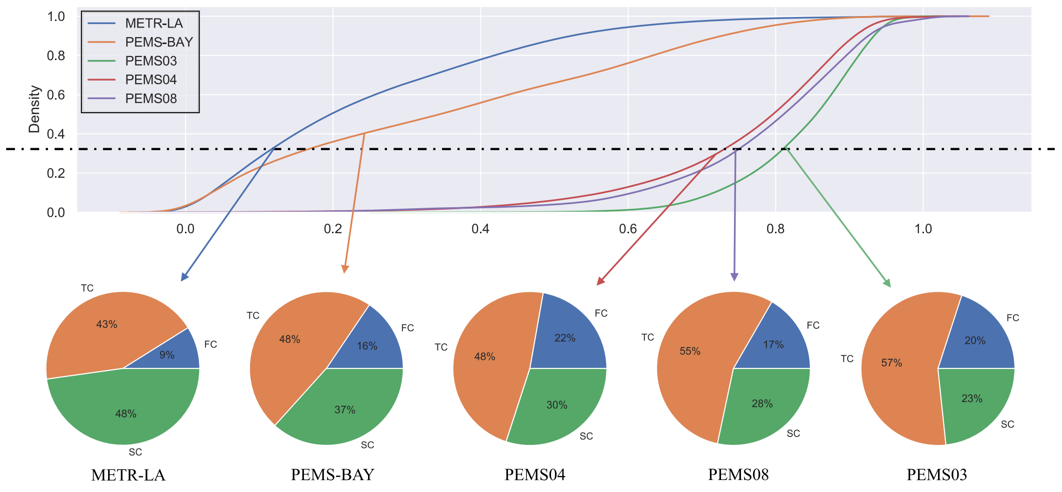

Fig. 12. The Pearson's correlation coefficient distribution of sensor pairs and the ratios of each operation in learned architectures on traffic datasets.

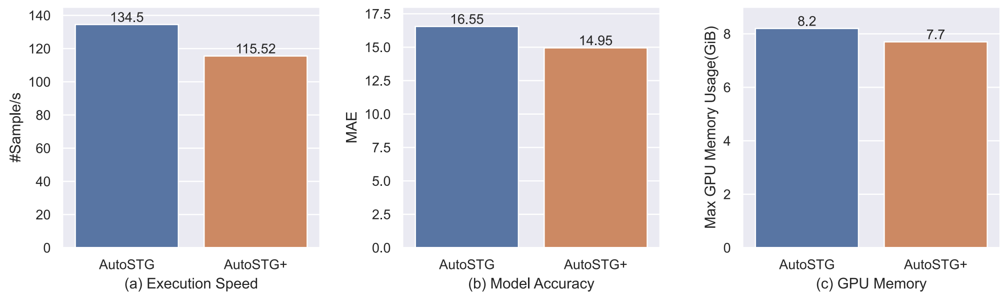

Fig. 13. The execution speed and GPU memory usage on PEMS08 dataset.

STG prediction. Moreover, [37,5,38] studied the usage of attention mechanisms to capture ST correlations. Recently, [17,23] leveraged meta learning method to model the inherent relationships between graph attributes and diverse ST correlations, and [39] proposed to learn graph structures from dynamic states. These works provide insights (*e.g.*, possible network structures) for STG prediction. However, designing models for specific tasks requires substantial domain knowledge and large amounts of expert effort. By contrast, our work can automatically search promising neural architectures for STGs, which is more efficient and cost-saving.

Neural Architecture Search. Many works have explored neural architectures for grid-based data (*e.g.*, image) or sequential data (*e.g.*, natural language). At first, reinforcement learning was adopted to search neural architectures [40,41,12]. Later, some one-shot algorithms tried to generate the weights of the sampled architectures by hypernetworks [42,43]. By contrast, [10] proposed continuous relaxation on candidate operators, enabling gradient-based optimization on architecture parameters. Besides these conventional data types, [15] proposed to search architectures for grid-based ST data and [13,14] employed NAS to model graph structure data. Moreover, [1] first presents a NAS framework for STG predictions, and [44] conducts extensive experiments to validate the effectiveness of different spatial operations and temporal operations. This work, based on [1], studies how to integrate and extract knowledge from meta graphs and uses it on STG prediction tasks, which benefits the tasks without explicit physical connections.

6. Conclusion

In this paper, we propose a novel NAS framework, entitled AutoSTG$^+$, for automatically selecting optimal network architecture for STG prediction. It employs SC and TC operations in search space to model spatial and temporal correlations, respectively. To model the implicit relationship between sensors, we enhance AutoSTG$^+$ by constructing meta graphs from different views, including physical connections, spatial similarity, and temporal similarity. To capture the spatio-temporal relationships in the meta graphs, we use graph meta knowledge learners to extract the graph characteristics, and then apply meta learning to generate the diffusion matrices of SC layers and the kernels of TC layers from the learned characteristics. We conduct extensive experiments on seven real-world benchmark datasets to demonstrate that our AutoSTG$^+$ can find promising architectures and achieve significant prediction accuracy. We find that our AutoSTG$^+$ still cost 12 hours for training a well-performed model, which is a major bottleneck for us to explore its further use. In the future, we plan to achieve a preferable speed-accuracy trade-off by designing novel algorithms.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Acknowledgement

This work was supported by the National Key R&D Program of China (2019YFB2101805), the National Natural Science Foundation of China (62172034), the Beijing Nova Program (Z201100006820053), and the Industrial Technology Basic Public Service Platform Project of the Ministry of Industry and Information Technology (2021-0166-1-1).

References

[1] Z. Pan, S. Ke, X. Yang, Y. Liang, Y. Yu, J. Zhang, Y. Zheng, Autostg: neural architecture search for predictions of spatio-temporal graph, in: J. Leskovec, M. Grobelnik, M. Najork, J. Tang, L. Zia (Eds.), WWW '21: The Web Conference 2021, Virtual Event, Ljubljana, Slovenia, April 19–23, 2021, ACM / IW3C2, 2021, pp. 1846–1855.

[2] Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: concepts, methodologies, and applications, ACM Trans. Intell. Syst. Technol. 5 (3) (2014) 1–55.

[3] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, in: International Conference on Learning Representations, 2018, https://openreview.net/forum?id=SJiHXGWAZ.

[4] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3634–3640.

[5] Y. Liang, S. Ke, J. Zhang, X. Yi, Y. Zheng Geoman, Multi-level attention networks for geo-sensory time series prediction, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3428–3434.

[6] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 1907–1913.

[7] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012).

[8] L.R. Medsker, L. Jain, Recurrent neural networks, Des. Appl. 5 (2001) 64–67.

[9] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, http://arxiv.org/abs/1409.0473.

[10] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019, https://openreview.net/forum?id=S1eYHoC5FX.

[11] S. Xie, H. Zheng, C. Liu, L. Lin, SNAS: stochastic neural architecture search, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019, https://openreview.net/forum?id=rylqooRqK7.

[12] H. Pham, M.Y. Guan, B. Zoph, Q.V. Le, J. Dean, Efficient neural architecture search via parameter sharing, in: J.G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, in: Proceedings of Machine Learning Research, PMLR, vol. 80, 2018, pp. 4092–4101, http://proceedings.mlr.press/v80/pham18a.html.

[13] K. Zhou, Q. Song, X. Huang, X. Hu, Auto-gnn: neural architecture search of graph neural networks, arXiv preprint, arXiv:1909.03184, 2019.

[14] Y. Gao, H. Yang, P. Zhang, C. Zhou, Y. Hu, Graphnas: graph neural architecture search with reinforcement learning, arXiv preprint, arXiv:1904.09981, 2019.

[15] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, Y. Zheng, Autost: efficient neural architecture search for spatio-temporal prediction, in: R. Gupta, Y. Liu, J. Tang, B.A. Prakash (Eds.), KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 794–802.

[16] C. Wang, Y. Zhu, T. Zang, H. Liu, J. Yu, Modeling inter-station relationships with attentive temporal graph convolutional network for air quality prediction, in: L. Lewin-Eytan, D. Carmel, E. Yom-Tov, E. Agichtein, E. Gabrilovich (Eds.), WSDM '21, the Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021, ACM, 2021, pp. 616–634.

[17] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, J. Zhang, Urban traffic prediction from spatio-temporal data using deep meta learning, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1720–1730.

[18] Y. Kim, Convolutional neural networks for sentence classification, in: A. Moschitti, B. Pang, W. Daelemans (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, a Meeting of SIGDAT, a Special Interest Group of the ACL, ACL, 2014, pp. 1746–1751.

[19] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Comput. Soc. Netw. 6 (1) (2019) 1–23.

[20] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, Seattle, WA, USA, vol. 10, 1994, pp. 359–370.

[21] X. Chu, T. Zhou, B. Zhang, J. Li, Fair DARTS: eliminating unfair advantages in differentiable architecture search, in: A. Vedaldi, H. Bischof, T. Brox, J. Frahm (Eds.), Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV, in: Lecture Notes in Computer Science, vol. 12360, Springer, 2020, pp. 465–480.

[22] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, J. Yan, {DARTS}-: robustly stepping out of performance collapse without indicators, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=KLH36ELmwIB.

[23] Z. Pan, W. Zhang, Y. Liang, W. Zhang, Y. Yu, J. Zhang, Y. Zheng, Spatio-temporal meta learning for urban traffic prediction, IEEE Trans. Knowl. Data Eng. (2020) 1, https://doi.org/10.1109/TKDE.2020.2995855.

[24] X. Ouyang, Y. Yang, Y. Zhang, W. Zhou, Spatial-temporal dynamic graph convolution neural network for air quality prediction, in: International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021, IEEE, 2021, pp. 1–8.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Łukasz Kaiser, I. Polosukhin, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Łukasz Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 2017-Decem (2017) 5999–6009.

[26] A. van den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning, in: I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 6306–6315, https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

[27] C. Song, Y. Lin, S. Guo, H. Wan, Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 914–921, https://aaai.org/ojs/index.php/AAAI/article/view/5438.

[28] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, Z. Jia, Freeway performance measurement system: mining loop detector data, Transp. Res. Rec. 1748 (1) (2001) 96–102, https://doi.org/10.3141/1748-12.

[29] Y. Liu, Y. Zheng, Y. Liang, S. Liu, D.S. Rosenblum, Urban water quality prediction based on multi-task multi-view learning, in: S. Kambhampati (Ed.), Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, IJCAI/AAAI Press, 2016, pp. 2576–2581, http://www.ijcai.org/Abstract/16/366, 2016.

[30] X. Yi, J. Zhang, Z. Wang, T. Li, Y. Zheng, Deep distributed fusion network for air quality prediction, in: Y. Guo, F. Farooq (Eds.), Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, ACM, 2018, pp. 965–973.

[31] H. Cai, L. Zhu, S. Han, Proxylessnas: direct neural architecture search on target task and hardware, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019, https://openreview.net/forum?id=HylVB3AqYm.

[32] R. Huang, C. Huang, Y. Liu, G. Dai, W. Kong, Lsgcn: long short-term traffic prediction with graph convolutional networks, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 2355–2361, main track.

[33] R. Dai, S. Xu, Q. Gu, C. Ji, K. Liu, Hybrid spatio-temporal graph convolutional network: improving traffic prediction with navigation data, in: R. Gupta, Y. Liu, J. Tang, B.A. Prakash (Eds.), KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 3074–3082.

[34] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: multivariate time series forecasting with graph neural networks, in: R. Gupta, Y. Liu, J. Tang, B.A. Prakash (Eds.), KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 753–763.

[35] W. Zhang, H. Liu, Y. Liu, J. Zhou, H. Xiong, Semi-supervised hierarchical recurrent graph neural network for city-wide parking availability prediction, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1186–1193, https://aaai.org/ojs/index.php/AAAI/article/view/5471.

[36] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 3529–3536, https://aaai.org/ojs/index.php/AAAI/article/view/5758.

[37] C. Zheng, X. Fan, C. Wang, J. Qi, GMAN: a graph multi-attention network for traffic prediction, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1234–1241, https://aaai.org/ojs/index.php/AAAI/article/view/5477, 2020.

[38] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 922–929.

[39] Q. Zhang, J. Chang, G. Meng, S. Xiang, C. Pan, Spatio-temporal graph structure learning for traffic forecasting, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1177–1185, https://aaai.org/ojs/index.php/AAAI/article/view/5470.

[40] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017, https://openreview.net/forum?id=r1Ue8Hcxg.

[41] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 19–34.

[42] A. Brock, T. Lim, J.M. Ritchie, N. Weston, SMASH: one-shot model architecture search through hypernetworks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018, https://openreview.net/forum?id=rydeCEhs-.

[43] C. Zhang, M. Ren, R. Urtasun, Graph hypernetworks for neural architecture search, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019, https://openreview.net/forum?id=rkgW0oA9FX.

[44] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, C.S. Jensen, Autocts: automated correlated time series forecasting - extended version, CoRR, arXiv:2112.11174 [abs], 2021, https://arxiv.org/abs/2112.11174.