Automated Spatio-Temporal Synchronous Modeling with Multiple Graphs for Traffic Prediction

Fuxian Li
Department of Electronic Engineering, Tsinghua University
Beijing, China

Huan Yan*
Department of Electronic Engineering, Tsinghua University
Beijing, China

Guangyin Jin
Department of Electronic Engineering, Tsinghua University
Beijing, China

Yue Liu
AutoNavi, Alibaba Group
Beijing, China

Yong Li
Department of Electronic Engineering, Tsinghua University
Beijing, China

Depeng Jin
Department of Electronic Engineering, Tsinghua University
Beijing, China

ABSTRACT

Traffic prediction plays an important role in many intelligent transportation systems. Many existing works design static neural network architecture to capture complex spatio-temporal correlations, which is hard to adapt to different datasets. Although recent neural architecture search approaches have addressed this problem, it still adopts a coarse-grained search with pre-defined and fixed components in the search space for spatio-temporal modeling. In this paper, we propose a novel neural architecture search framework, entitled AutoSTS, for automated spatio-temporal synchronous modeling in traffic prediction. To be specific, we design a graph neural network (GNN) based architecture search module to capture localized spatio-temporal correlations, where multiple graphs built from different perspectives are jointly utilized to find a better message passing way for mining such correlations. Further, we propose a convolutional neural network (CNN) based architecture search module to capture temporal dependencies with various ranges, where gated temporal convolutions with different kernel sizes and convolution types are designed in search space. Extensive experiments on six public datasets demonstrate that our model can achieve 4%∼10% improvements compared with other methods.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

KEYWORDS

Traffic prediction, Graph convolution, Spatio-temporal modeling

*Corresponding author. Email: yanhuan@tsinghua.edu.cn.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557243

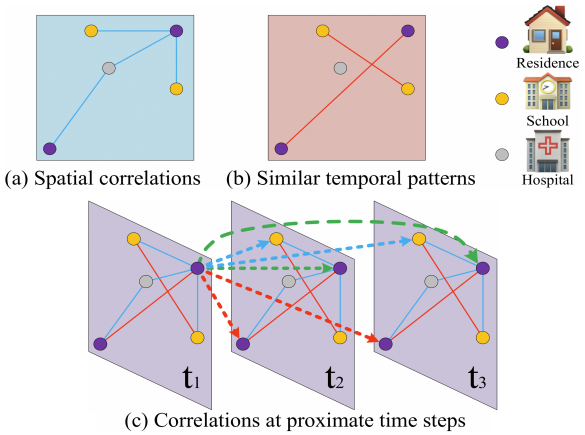


Figure 1: An example of complex spatio-temporal correlations.

ACM Reference Format:
Fuxian Li, Huan Yan*, Guangyin Jin, Yue Liu, Yong Li, and Depeng Jin. 2022. Automated Spatio-Temporal Synchronous Modeling with Multiple Graphs for Traffic Prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557243

1 INTRODUCTION

Traffic prediction is essential for a wide range of applications in many intelligent transportation systems, such as traffic management, navigation planning and congestion controlling. Precise prediction can provide valuable reference for these applications, thus greatly enhancing user service experience. The core of traffic prediction is spatio-temporal modeling, which aims to capture the complex and dynamic spatio-temporal correlations.

Figure 1 shows a typical example of the spatio-temporal correlations.First, there exist geo-spatial dependencies, where adjacent locations are connected in traffic network, as seen in Figure 1(a). Second, although some nodes are distant with each other, they may have similar temporal patterns, e.g., Figure 1(b). In most cases, the geo-spatial dependencies and temporal-pattern similarities may exist at the same time. Moreover, each node not only influences itself at

the proximate time steps, but also has potential impacts on its neighboring nodes. We visualize such complex correlations in Figure 1(c). Recent works [18, 31, 34] pay more attention on graph neural network (GNN) based spatio-temporal modeling. These works treat different locations of a road network as different nodes in a graph, then employ GNNs for spatial modeling, and recurrent neural networks (RNNs) or convolutional neural networks (CNNs) for capturing temporal dependencies. However, they fail to adaptively adjust the network architectures according to various kinds of data, which restricts the representation ability. Fortunately, Neural architecture search (NAS) algorithm has been promoted to design the neural networks automatically [4, 17, 19, 20, 23, 24, 26, 29, 37]. In particular, AutoSTG [21] proposes an automated neural architecture search framework, where the spatial convolution and temporal convolution are directly used as candidate operations in one search space. Nevertheless, its pre-defined and fixed candidate operations limit the latent combinations of searchable architectures. It is expected to explore more fine-grained search for the inner architectures of both spatial and temporal modules. Thus, the following two problems need to be addressed.

(1) How to define the search space of graph convolution? Many prior works make meaningful attempts on various graph construction methods, such as the weighted directed graph [18], binary graph [10, 25], adaptive graph [31], POI graph [10], DTW [27] graph [16, 33], free-flow reachability graph [5], edge-wise graph [3], and localized spatio-temporal graph [25], etc. These graphs can reflect complex and multi-faceted correlations among nodes. It motivates us to *design a general GNN-based architecture search mechanism with multiple graphs, which can adaptively find a more competitive message passing process for spatio-temporal modeling*. Following this motivation, neural architecture search (NAS) methods like [9, 14, 36] are proposed to address it. However, they neglect one of the most essential parts of graph convolution, i.e., the adjacency matrix, which determines the routes and weights of the message passing process. Thus, it is challenging to define the search space of graph convolution.

(2) How to learn multi-scale temporal dependencies? Although RNNs [5, 18] and CNNs [34] can be used for temporal modeling, it is difficult for their fixed neural architectures to learn dynamic and complex temporal correlations from multiple scales. For example, under unobstructed traffic conditions, the recent temporal dependencies are sufficient to reflect the situation of traffic flow. In contrast, under heavy traffic conditions, the occurrences of congestion could cause the long-range impacts on traffic flow. Thus, it is beneficial to employ fine-grained neural architecture search into the temporal modules to capture the complex dependencies adaptively. However, it is not trivial to design the search space for mining multi-scale temporal dependencies.

To solve the above problems, we propose a novel neural architecture search approach, called AutoSTS, for automated spatio-temporal synchronous modeling in traffic prediction. To be specific, a GNN-based architecture search mechanism is designed to capture short-range spatio-temporal correlations, and a CNN-based architecture search mechanism is employed to model long-range temporal dependencies. Our main contributions are summarized as follows.

- We design a GNN-based neural architecture search mechanism with multiple pre-defined candidate graphs to find a better message passing process for short-range spatio-temporal modeling. Specifically, a novel final activation and fusion mechanism is designed to tremendously improve the search efficiency, while still improve the prediction accuracy.
- We propose a CNN-based neural architecture search mechanism with gated temporal convolutions of various kernel sizes and convolution types. It can automatically search a better stacking way of different temporal convolutions to capture long-range temporal dependencies.
- Extensive experiments are conducted on six public datasets, which demonstrates that our model can achieve at least 4%∼10% improvements of prediction performance compared with the state-of-the-art baselines.

2 RELATED WORKS

2.1 Neural Architecture Search

Neural architecture search (NAS) algorithm has been promoted to design the neural networks automatically in recent years, which has three main types: reinforcement learning-based, evolutionary-based and differentiable methods. The reinforcement learning-based methods treat the optimal architecture search as a sequential decision task [19, 23, 37]. The evolutionary-based methods utilize some evolutionary algorithms to search the optimal super-net [24, 26, 32]. However, their main limitations are huge computing overhead and low training efficiency. For the differentiable methods, it enables probabilistic neural architecture selection [4, 20, 28, 29]. Compared with other two kinds of methods, the training efficiency of differentiable methods has been greatly improved. Since efficiency is important for the deployment of traffic prediction models, we adopt differentiable NAS framework DARTS in this paper.

2.2 Traffic Prediction

To capture complex spatial and temporal correlations for traffic prediction, some graph-based deep learning models are introduced recently. STGCN [34] first combines graph convolution with temporal convolution to capture spatio-temporal dynamics efficiently. DCRNN [18] encodes the temporal information by gated recurrent unit (GRU) instead of temporal convolution. In addition, STSGCN [25] and STFGNN [16] introduce the spatio-temporal synchronous graph based framework to capture both spatial and temporal correlations in parallel. Graph-WaveNet [31] adopts the adaptive graph convolution to improve the limited spatial relations from the pre-defined traffic networks. In the most recent work, AutoSTG [21] first makes attempts to integrate the neural architecture search approach with spatio-temporal graph neural network based framework to improve the prediction accuracy. However, this work cannot select appropriate spatio-temporal learning modules from a more fine-grained perspective.

3 PRELIMINARIES

The traffic network can be modeled as a graph $\mathcal{G} = (V, E, A)$, where V denotes the set of nodes $|V| = N$, which represents the observation of N traffic sensors; E denotes the set of edges and A denotes the adjacency matrix to characterize the relations between different

nodes. The graph signal $X_{\mathcal{G}}^{(t)} \in \mathbb{R}^{N\times d}$ denotes the feature matrix of the graph $\mathcal{G}$ at time step t, which consists of observed d-dimensional traffic features (e.g., the speed, volume) of each node. The task of traffic prediction aims to learn a non-linear function $f(\cdot)$ from previous T speed series for forecasting next $T^{'}$-step traffic speed from N given sensors in the traffic network. The mathematical form is defined as follows:

$$[\mathbf{X}_{\mathcal{G}}^{(t-T+1)}, \cdots, \mathbf{X}_{\mathcal{G}}^{t}] \xrightarrow{f(\cdot)} [\mathbf{X}_{\mathcal{G}}^{t+1}, \cdots, \mathbf{X}_{\mathcal{G}}^{t+T^{'}}] \tag{1}$$

4 OUR MODEL

The main architecture of AutoSTS is demonstrated in Figure 2(a). We first employ a linear mapping to transform the input graph signal into a latent space. At each layer, Short-range Architecture Search Module (SASM) and Long-range Architecture Search Module (LASM) are designed for short-range spatio-temporal modeling and long-range temporal capturing, respectively. We then employ residual mechanism and parameterized skip connections to avoid gradient vanishing and enhance the robustness of our model. A multilayer perceptron (MLP) is employed to get the final output. In the following subsections, we introduce two core components of AutoSTS, i.e., SASM and LASM, respectively. Finally, we show a brief overview of the optimization algorithm.

4.1 Short-Range Architecture Search Module

4.1.1 Graph Convolution with Multiple Graphs. Traditional graph convolution operation [15] can be formulated as

$$g(X) = \phi(\tilde{A}X\Theta), \tag{2}$$

where $\tilde{A} \in \mathbf{R}^{N\times N}$ denotes the normalized adjacency matrix that provides the paths and weights of message passing, $X \in \mathbf{R}^{N\times D}$ denotes the input graph signals, $\Theta \in \mathbf{R}^{D\times D'}$ denotes the learnable parameters for linear mapping, and $\phi(\cdot)$ denotes the nonlinear activation function, such as *Tanh* and *ReLU*.

However, one simple graph cannot comprehensively describe the complex and multi-faceted inter-node correlations. To make good joint use of multiple graphs, we generalize Equation 2 to the following formulation:

$$g(X) = \phi(\mathcal{M}(X; \mathbb{G}, \boldsymbol{\zeta})\Theta), \tag{3}$$

where $\mathcal{M}$ is a general message passing function based on the set of multiple graphs $\mathbb{G}$, and $\boldsymbol{\zeta}$ represents learnable parameters for message passing process $\mathcal{M}$.

When stacking multiple graph convolution layers, we have:

$$\begin{aligned} H^{(j)} &= \sum_{i<j} g^{(i,j)}(H^{(i)}), \\ H_{out} &= AGG([H^1, \cdots, H^L]), \end{aligned} \tag{4}$$

where $H^{(i)}$ is the output of the i^{th} graph convolution layer, $g^{(i,j)}$ denotes the graph convolution which projects $H^{(i)}$ into an intermediate hidden state. Then, $H^{(j)}$ is computed based on all of its predecessors. *AGG* is an aggregation function such as sum, average, mean and max pooling. L is the number of graph convolution layers.

A naive NAS-style implementation of Equation 3 and Equation 4 is to directly use the complete graph convolution procedures in Equation 2 with various graphs in set $\mathbb{G}$ as candidate operations. However, high complexity of candidate graph convolutions will bring about huge consumption on computational resources, and increase the risks of over-fitting. *To solve this challenge, we first separate the $\mathcal{M}$ and the Θ in Equation 3. Then we conduct the architecture search on $\mathcal{M}$, and design a novel final activation & fusion mechanism for Θ.* Especially, we simplify $\mathcal{M}$ with much fewer parameters, formulated as

$$\mathcal{M}(X; \mathbb{G}, \boldsymbol{\zeta}) = \sum_{A\in\mathbb{A}} \gamma_A AX, \tag{5}$$

where A is a candidate adjacency matrix in matrix set $\mathbb{A}$, and γ_A is learnable to weight the messages from various graphs. Here we simplify the $\boldsymbol{\zeta}$ in function $\mathcal{M}$ with γ_A. In other words, we choose $\tilde{A}X$ in Equation 2 as the basic style of candidate operations in search space.

4.1.2 Details of SASM. To perform the effective search, we design the search architecture of SASM inspired by [20], as shown in Figure 2(b). This is a direct acyclic graph (DAG), consisting of L vertices which represent L graph convolution layers with L intermediate latent representations denoted as $X^{(i)}, i = 1, \cdots, L$. Following Equation 5, the mixed message passing operation $\bar{\mathcal{M}}^{(i,j)}$ from layer i to layer j is formulated as follow:

$$\bar{\mathcal{M}}^{(i,j)}(X) = \sum_{A\in\mathbb{A}} \frac{\exp(\alpha_A^{(i,j)})}{\sum_{A'\in\mathbb{A}} \exp(\alpha_{A'}^{(i,j)})} AX, \tag{6}$$

where $X \in \mathbf{R}^{N\times D}$ is the input graph representations, $A \in \mathbf{R}^{N\times N}$ is a candidate adjacency matrix, $\mathbb{A} = \{A_1, A_2, \cdots\}$ is a set of pre-defined candidate adjacency matrices, and $\alpha_A^{(i,j)}$ is a learnable architecture parameter to weight the operation AX associated with pair (i, j).

Denote the original input graph signal as $X^{(0)}$, each intermediate representation is computed based on all of its predecessors:

$$X^{(j)} = \sum_{i<j} \bar{\mathcal{M}}^{(i,j)}(X^{(i)}). \tag{7}$$

4.1.3 Final Activation & Fusion Module. During the message passing processes of graph convolutions, the spatio-temporal information of nodes is propagated along the given graph structures, so it is meaningful to aggregate intermediate representations with different propagation ranges to get the comprehensive information of message passing. Moreover, the nonlinear activation, linear mapping and layer aggregation are necessary components of graph convolutions. Thus, we design the Final Activation & Fusion Module (FAFM), which essentially acts as a combination of ϕ, Θ in Equation 3 and *AGG* in Equation 4. In details, cropping operations [25] are first used for all intermediate representations $X^{(i)}, i = 1, \cdots, L$ in search space to simplify the results of message passing. After that, we use gated linear units (GLUs) [6] for linear mappings and nonlinear activation. The aforementioned operations can be formulated as:

$$\mathcal{H}^i = GLU_i(Crop(X^{(i)})), i = 1, \cdots, L, \tag{8}$$

where $X^{(i)}$ is intermediate representation of the i^{th} graph convolution layer, $Crop$ is the cropping operation, GLU_i is the GLU module

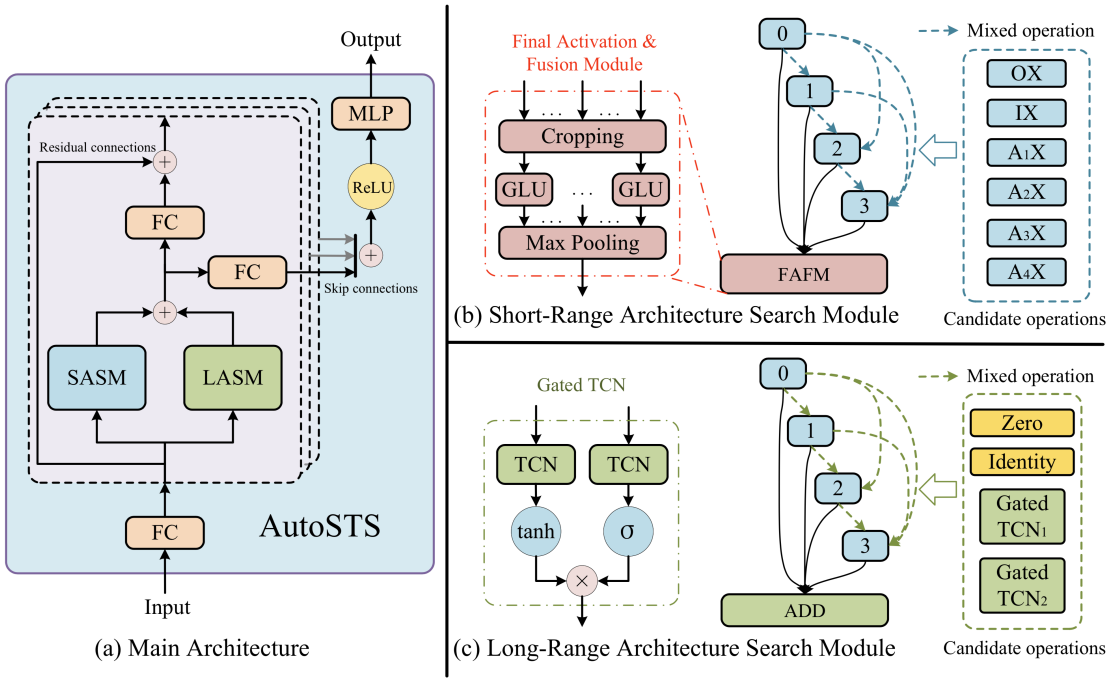


Figure 2: Detailed framework of AutoSTS. (a) is the main architecture of AutoSTS. The input is first projected by a Fully Connected layer (FC). Then the Short-range Architecture Search Module (SASM) and the Long-range Architecture Search Module (LASM) filter the input of each layer of AutoSTS in parallel to achieve comprehensive spatio-temporal modeling. After fusing the output of SASM and LASM together, we employ residual mechanism and parameterized skip connections to avoid gradient vanishing. Finally, the output is obtained by aggregating the hidden features of skip connections. (b) and (c) are detailed structures of SASM and LASM, respectively.

applied to $X^{(i)}$ individually, formulated as follow:

$$GLU(X) = (XW_1 + b_1) \odot \sigma(XW_2 + b_2), \tag{9}$$

where $W_1, W_2 \in \mathbb{R}^{D \times D'}$, $b_1, b_2 \in \mathbb{R}^{D'}$ are learnable parameters, $\odot$ means Hadamard product and σ is the sigmoid activation function.

Finally, we employ a max pooling operation to aggregate all intermediate representations and get the final output of SASM, formulated as follow:

$$X_{out} = MaxPool([\mathcal{H}^1, \cdots, \mathcal{H}^L]). \tag{10}$$

At the end of search, we replace each mixed operation $\bar{\mathcal{M}}^{(i,j)}$ with the most likely message passing operation to get the final architecture, which can be formulated as follow:

$$\mathcal{M}^{(i,j)}(X) = (\text{argmax}_{A \in \mathbb{A}} \ \alpha_A^{(i,j)})X. \tag{11}$$

4.1.4 Construction of Candidate Graphs. Inspired by [16], we adopt the spatio-temporal synchronous graph convolution [25] to capture localized spatio-temporal correlations. In spatio-temporal synchronous graph convolution, the number of nodes is kN instead of N, N is the number of original nodes in traffic network and k denotes the number of contiguous time steps in localized spatio-temporal graph $A \in \mathbf{R}^{kN \times kN}$. In this paper, k is set to 4.

Specifically, we design 6 candidate adjacency matrices from multiple angles, and finally get $\mathbb{A} = \{O, I, A_1, A_2, A_3, A_4\}$, as illustrated in Figure 3. In details, O and I mean zero mapping and identity mapping respectively. A_{TG} is the adjacency matrix of the temporal graph TG, which is calculated by DTW algorithm [27] to describe the temporal similarities among traffic time series of node pairs. We formulate A_{TG} as follow:

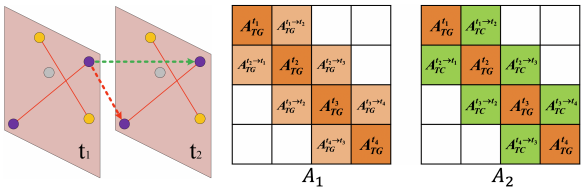
$$A_{TG}^{ij} = \begin{cases} 1, DTW(S_i, S_j) < \epsilon \\ 0, \text{otherwise} \end{cases}, \tag{12}$$

where S_i and S_j represent traffic time series of node i and node j respectively. We use ϵ as a threshold value to control the sparsity of A_{TG}. Besides, A_{SG} is the adjacency matrix of the spatial graph SG, which is calculated according to the spatial connectivity among nodes in traffic network. We formulate A_{SG} as follow:

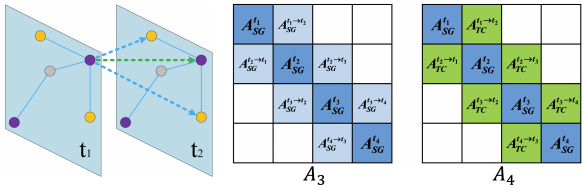
$$A_{SG}^{ij} = \begin{cases} 1, if \ v_i \ connects \ to \ v_j \\ 0, \text{otherwise} \end{cases}, \tag{13}$$

where v_i and v_j represent node i and node j respectively. Furthermore, A_1 and A_3 are designed for inter-node connections at proximate time steps, while A_2 and A_4 are built for intra-node similarities at proximate time steps.

To the best of our knowledge, we take the first step to employ neural architecture search on various candidate adjacency matrices during graph convolutions. First, we focus on searching graph convolutions with different graphs, rather than ways of node aggregation or layer aggregation. In consideration of the existing kinds



(a) TG based spatio-temporal graph and its adjacency matrices



(b) SG based spatio-temporal graph and its adjacency matrices

Figure 3: Pre-defined candidate graphs and their corresponding matrices. A_{TC} is an identity matrix to describe self-connectivity, TG is a temporal graph and SG is a spatial graph.

of graphs for traffic prediction, our method has great extensibility to make good use of graphs built from various perspectives. Second, we improve the GNN-based NAS architecture by splitting the complete graph convolution procedure, simplifying the candidate operations and designing the aforementioned FAFM. Thus, the model efficiency would be greatly improved.

4.2 Long-Range Architecture Search Module

Temporal convolutional network (TCN) is essentially 1D CNN, which benefits from its high efficiency. Comparatively, gated TCN [7] is more powerful than TCN without losing too much efficiency, and has been widely used for sequential modeling. Specifically, a commonly used gated TCN is composed of two parallel 1D CNNs followed by tangent hyperbolic activation function and sigmoid function respectively, which can be formulated as follow:

$$\mathbf{Y} = \phi(\mathbf{\Theta_1} \star X + \mathbf{b}) \odot \sigma(\mathbf{\Theta_2} \star X + \mathbf{c}), \tag{14}$$

where X denotes the input of gated TCN, $\mathbf{\Theta_1}$, $\mathbf{\Theta_2}$, $\mathbf{b}$ and $\mathbf{c}$ are learnable parameters, $\star$ denotes 1D convolution operation, $\odot$ represents the element-wise product, $\phi(\cdot)$ denotes the tangent hyperbolic activation function, and $\sigma(\cdot)$ represents the sigmoid function. Besides, 1D dilated convolutions [16, 31] are also introduced to enhance the receptive field of TCN.

However, due to the oversimplified convolution kernel and convolution type, it is tough for traditional TCNs to deal with long sequences and capture long-range temporal patterns in spatio-temporal modeling. To tackle this challenge, on one hand, we should consider various convolution types such as standard convolution, separable convolution and dilated convolution, etc. On the other hand, different kernel sizes should be jointly used to capture temporal dependencies with various ranges, especially long range. Following the above analysis, we design the LASM, which will be introduced as follows.

As shown in Figure 2(c), the search space of LASM is a a DARTS-style [20] DAG consisting of N_v vertices. Denoting the input and the intermediate latent representation of the i^{th} vertex in search space as X^0 and $X^{(i)}, i = 1, \cdots, N_v$ respectively, the output of LASM is formulated as follow:

$$X_{out} = \sum_{j}^{N_v} \sum_{i<j} \sum_{o \in \mathbb{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathbb{O}} \exp(\alpha_{o'}^{(i,j)})} o(X^i), \tag{15}$$

where o is a candidate operation in function set $\mathbb{O} = \{o_1, o_2, \cdots\}$, and $\alpha_o^{(i,j)}$ is a learnable architecture parameter to weight the operation o associated with vertex pair (i, j).

Specifically, we choose zero mapping, identity mapping and gated TCNs with various kernel sizes and convolution types as candidate operations in $\mathbb{O}$. For instance, the candidate operations can be gated standard convolution with 1×3 kernel, gated separable convolution with 1×5 kernel and gated dilated convolution with 1×7 kernel, etc. After search process, we select the most likely operation o with the highest $\alpha_o^{(i,j)}$ as the final operation for each pair (i, j), formulated as $o^{(i,j)} = \text{argmax}_{o \in \mathbb{O}} \ \alpha_o^{(i,j)}$.

4.3 Searching Algorithm

In AutoSTS, the architecture parameters include the scores of different candidate adjacency matrices, i.e., the α_A in SASM (Equation 6), and the scores of candidate operations, i.e., the α_o in LASM (Equation 15). The network weight parameters represent all learnable parameters of the model except architecture parameters, such as the parameters in FCs, MLP, and candidate gated TCNs. Similar to [20], we first alternately update the weight parameters and the architecture parameters with the training dataset and the validation dataset separately, until the stopping criterion is met. After that, we train the optimal architecture with the training dataset until another stopping criterion is met.

5 EXPERIMENTS

In this section, we first describe the six public traffic datasets. Then we validate the model performance of our AutoSTS on the datasets for traffic prediction task, where several representative baseline methods are chosen for comparison. Next, we conduct ablation experiments to verify the effectiveness of key components and mechanisms of AutoSTS. The parameter analysis and case study are also conducted to further investigate the effectiveness of our model. We conduct experiments to answer the research questions summarized as follows:

- **RQ1:** How is the prediction performance of our proposed AutoSTS compared with the state-of-the-art baselines?
- **RQ2:** How does our model perform without key components and mechanisms?
- **RQ3:** Does the final activation and fusion mechanism (FAFM) help improve the efficiency of our model?
- **RQ4:** How do the hyperparameters of AutoSTS influence the model performance?

5.1 Datasets

We use six public traffic datasets for performance evaluation. METR-LA and PEMS-BAY are commonly used datasets released by [18].

METR-LA is collected from loop detectors in the highway of Los Angeles, and it contains 207 nodes and ranges from Mar 1st 2012 to Jun 30th 2012. PEMS-BAY is collected by California Transportation Agencies (CalTrans), and it contains 325 nodes and ranges from Jan 1st 2017 to May 31th 2017. We also evaluate the performance of AutoSTS on PEMS03, PEMS04, PEMS07 and PEMS08, which are four highway traffic datasets released by [25]. These four datasets are collected from Caltrans Performance Measurement System (PeMS). The time granularity of all six datasets is set to 5 minutes following the setting of previous works [18, 25]. The pre-defined adjacency matrices for each dataset are calculated based on road network topology. Z-score normalization is applied to the traffic flow data. Detailed statistics of datasets are shown in Table 1.

Table 1: Dataset description and statistics.

Datasets	Number of nodes	Time range
METR-LA	207	3/1/2012 - 6/30/2012
PEMS-BAY	325	1/1/2017 - 5/31/2017
PEMS03	358	9/1/2018 - 11/30/2018
PEMS04	307	1/1/2018 - 2/28/2018
PEMS07	883	5/1/2017 - 8/31/2017
PEMS08	170	7/1/2016 - 8/31/2016

5.2 Baselines

We compare our model with several state-of-art baselines, which are introduced as follows:

- **ARIMA [2]:** This is a traditional method and has been widely used for time series prediction. It integrates auto-regression with moving average model.
- **VAR [12]:** Vector Auto-Regression can be used for time series forecasting.
- **SVR:** This is a classical time series analysis model which uses linear support vector machine for the regression task.
- **FNN:** This is a feed forward neural network with two hidden layers and L2 regularization.
- **FC-LSTM [13]:** This is a well-known recurrent network architecture with fully connected hidden layers.
- **DCRNN [18]:** This model integrates the Gated Recurrent Unit (GRU) with dual directional diffusion graph convolution for spatio-temporal graph modeling.
- **STGCN [34]:** This model incorporates graph convolutions with 1D convolutional neural networks.
- **ASTGCN [11]:** This model designs spatial attention and temporal attention mechanisms to achieve a better ability of spatio-temporal modeling. Here we only take the recent components of the model for fair comparison.
- **Graph WaveNet [31]:** This model combines the graph convolution with gated 1D dilated convolutional neural network, where the adaptive adjacency matrix is proposed to model the latent spatial correlations.
- **STSGCN [25]:** This model designs a novel spatio-temporal synchronous graph convolution mechanism to model the localized spatio-temporal correlations.
- **STFGNN [16]:** This model improves STSGCN with the spatio-temporal fusion graph and gated 1D dilated convolutional neural networks.
- **STGODE [8]:** This is a continuous GNN-based model with neural ODE modeling.
- **ST-MetaNet [22]:** This model employs the meta learning mechanism to generate the parameters of GAT and GRU.
- **AGCRN [1]:** This model integrates GRU with graph convolutions, where the adaptive graph and the node adaptive parameter learning mechanism are proposed.
- **GMAN [35]:** This model designs spatial, temporal and transform attentions for better spatio-temporal modeling.
- **MTGNN [30]:** This model employs dilated inception layers, mix-hop propagation layers, and adaptive graph to capture spatio-temporal correlations.
- **AutoSTG [21]:** This is a GNN-based and CNN-based model with neural architecture search, where the pre-defined spatial graph convolution and temporal convolution are included in one candidate operation set to capture spatio-temporal dependencies.

To evaluate the performances of all methods, we adopt three commonly used metrics in traffic prediction field, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE), which can be formulated as follows:

$$\begin{aligned} \mathrm{MAE}(y,\hat{y}) &= \frac{1}{|\Omega|}\sum_{i\in\Omega}|y_i-\hat{y}_i|, \\ \mathrm{RMSE}(y,\hat{y}) &= \sqrt{\frac{1}{|\Omega|}\sum_{i\in\Omega}(y_i-\hat{y}_i)^2}, \\ \mathrm{MAPE}(y,\hat{y}) &= \frac{1}{|\Omega|}\sum_{i\in\Omega}\left|\frac{y_i-\hat{y}_i}{y_i}\right|, \end{aligned} \tag{16}$$

where $y = y_1, \cdots, y_n$ represents the ground truth, $\hat{y} = \hat{y}_1, \cdots, \hat{y}_n$ denotes the predicted values, and Ω represents the indices of observed samples.

5.3 Experiment Settings

Following the settings of previous methods [18, 25], we chronologically split the METR-LA and PEMS-BAY datasets with 70% for training, 10% for validation and 20% for testing. Besides, the PEMS03, PEMS04, PEMS07, PEMS08 datasets are chronologically split with 60% for training, 20% for validation and 20% for testing. We use traffic flow in the past one hour to predict the flow in the next one hour. AutoSTS is implemented by Pytorch 1.7 on virtual workstations with Nvidia GeForce RTX 2080Ti and NVIDIA TESLA V100 GPU. By default, AutoSTS contains 3 layers, where the depths of graph convolution in SASM (the L in equation 10) is set to 3 and the number of vertices in LASM (the N_v in equation 15) is 2. For simplicity, we use gated standard convolutions with 1×3 kernel and 1×5 kernel as the candidate gated TCNs in LASM. The dimension of hidden representations is set to 40. The model is trained by Adam optimizer. The batch size is 32 and the learning rate is 0.001. MAE is chosen as the loss function. AutoSTS is evaluated five times in each dataset. During search process, we employ early stopping for architecture search with tolerance 15 for 60 epochs. After that, we reinitialize the optimizer and employ early stopping for training with tolerance 30 for 200 epochs.

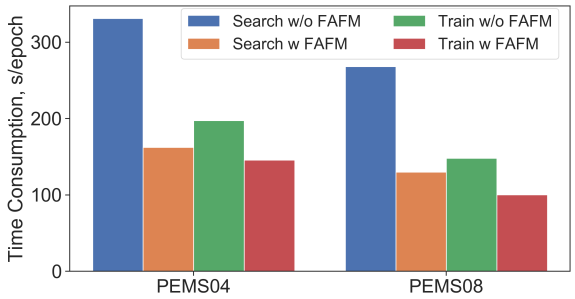
5.4 Experiment Results and Analysis (RQ1)

Table 2 shows the performance of AutoSTS and baselines for 15 minutes, 30 minutes, and 60 minutes ahead prediction on METR-LA and PEMS-BAY datasets. Table 3 demonstrates the performance comparison of baseline models and AutoSTS, where the metrics are MAE, RMSE and MAPE averaged for 60 minutes ahead prediction. From the experimental results in Table 2 and Table 3, we can find that AutoSTS consistently outperforms other baselines with 4%∼10% improvements on all datasets, which demonstrates the superiority of our proposed method. The performance improvements of AutoSTS on PEMS-BAY dataset are relatively less significant. This is because the models on PEMS-BAY can achieve much better prediction accuracy compared with other datasets, making it hard to further improve the performance. For example, the MAPE of some simple methods like ARIMA, VAR, and SVR on PEMS-BAY dataset can be 3.50%∼3.80% for 15 minutes ahead prediction, which is even better than some deep learning methods like FNN (5.19%) and FC-LSTM (4.80%). Besides, the worst MAPE performance (5.19%) for 15 minutes ahead prediction on PEMS-BAY dataset is much better than the best MAPE performance (6.55%) on METR-LA dataset, as shown in Table 2. STSGCN and STFGNN fail to perform well on METR-LA and PEMS-BAY datasets, possibly due to the limited representation ability of models and the missing values in data. All the baselines except AutoSTG are based on manually-designed architectures, which fail to adjust themselves corresponding to the data. AutoSTG employs neural architecture search on spatio-temporal modeling and has a data-dependent architecture, hence its performance is superior to most other baselines. However, the inner architectures of candidate spatial convolution and temporal convolution of AutoSTG are both pre-defined and fixed without any search inside them, which restricts the representation ability of the model. Comparatively, we design a multi-scale and fine-grained search framework for AutoSTS, i.e., SASM and LASM, through which the model not only jointly uses multi-view graphs to capture short-range spatio-temporal correlations, but also extracts long-range temporal dependencies with optimized temporal convolutions.

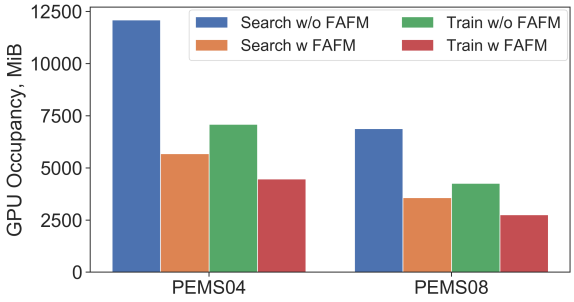
5.5 Ablation Study (RQ2 & RQ3)

To evaluate the effectiveness of key components in AutoSTS, we conduct ablation study on PEMS04 and PEMS08 datasets. The variants of AutoSTS are introduced as follows:

- *w/o FAFM*: this variant replaces FAFM in SASM with simple *add* function, and here we use complete graph convolutions (Equation 2) with the same set of candidate graphs as candidate operations.
- *w/o SASM*: this variant replaces SASM with fixed STFGN Module [16].
- *w/o LASM*: this variant replaces LASM with fixed gated TCN.
- *SASM Random*: this variant samples an architecture from search space in SASM during search.
- *LASM Random*: this variant samples an architecture from search space in LASM during the search process.
- *Random*: this variant samples an architecture from search space in both SASM and LASM, then directly trains the randomly-sampled architecture for prediction.



(a) Time Consumption



(b) GPU Occupancy

Figure 4: Time Consumption and GPU Occupancy.

- *w/o* A_iX: this variant removes A_iX from the candidate operation set of SASM.

From the experimental results shown in Table 4, we can find that AutoSTS outperforms all the ablation variants. Moreover, the efficiency of our model is tremendously improved thanks to FAFM. As demonstrated in Figure 4, when compared with *w/o FAFM*, the percentage improvements of time consumption and GPU occupancy are around 100% during search process, and 20%∼30% during training process. Compared with the results of *w/o SASM*, AutoSTS improves 6.34%, 7.15% and 5.07% in terms of MAE, MAPE and RMSE on PEMS04. Meanwhile, it also improves 7.22%, 6.41% and 6.44% in terms of MAE, MAPE and RMSE on PEMS08, which illustrates the effectiveness of our GNN-based neural architecture search method and the power of joint use of multiple graphs. With regard to *w/o LASM*, in our ablation experiments, we use two intuitive designs of candidate TCNs (standard convolutions with 1×3 kernel and 1×5 kernel respectively) to verify the effectiveness of LASM. The results show that they are simple yet effective. We believe that other designs of candidate TCNs with larger capacity (more diverse kernel sizes and convolution types) will further improve the performance. Moreover, there is a sharp fall in the learning ability of model without architecture search process (*SASM Random*, *LASM Random* and *Random*), demonstrating the effectiveness of our neural architecture search mechanism. Finally, the results of *w/o* A_iX verify the effectiveness of pre-designed candidate graphs. Here, the ablation of one candidate graph will not bring about significant performance degradation. This is because we still have three candidate adjacency matrices if we just remove one of them, where the spatio-temporal correlations can still be described from three different angles.

Table 2: Performance comparison of baseline models and AutoSTS on METR-LA and PEMS-BAY datasets.

Dataset	Models	15min			30min			60min		
		MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
METR-LA	ARIMA	3.99	9.60	8.21	5.15	12.70	10.45	6.90	17.40	13.23
	VAR	4.42	10.20	7.89	5.41	12.70	9.13	6.52	15.80	10.11
	SVR	3.99	9.30	8.45	5.05	12.10	10.87	6.72	16.70	13.76
	FNN	3.99	9.90	7.94	4.23	12.90	8.17	4.49	14.00	8.69
	FC-LSTM	3.44	9.60	6.30	3.77	10.90	7.23	4.37	13.20	8.69
	DCRNN	2.77	7.30	5.38	3.15	8.80	6.45	3.60	10.50	7.60
	STGCN	2.88	7.62	5.74	3.47	9.57	7.24	4.59	12.70	9.40
	STSGCN	2.79	7.42	5.45	3.14	8.81	6.42	3.65	10.67	7.81
	STFGNN	2.72	7.24	5.26	3.10	8.63	6.30	3.55	10.56	7.47
	ST-MetaNet	2.69	6.91	5.17	3.10	8.57	6.28	3.59	10.63	7.52
	Graph WaveNet	2.69	6.90	5.15	3.07	8.37	6.22	3.53	10.01	7.37
	AGCRN	2.87	7.70	5.58	3.23	9.00	6.58	3.62	10.38	7.51
	GMAN	2.80	7.41	5.55	3.12	8.73	6.49	3.44	10.07	7.35
	MTGNN	2.69	6.86	5.18	3.05	8.19	6.17	3.49	9.87	7.23
	AutoSTG	2.70	6.94	5.17	3.08	8.40	6.19	3.46	9.85	7.31
	AutoSTS	**2.57**	**6.55**	**4.93**	**2.89**	**7.85**	**5.87**	**3.28**	**9.43**	**6.86**
PEMS-BAY	ARIMA	1.62	3.50	3.30	2.33	5.40	4.76	3.38	8.30	6.50
	VAR	1.74	3.60	3.16	2.32	5.00	4.25	2.93	6.50	5.44
	SVR	1.85	3.80	3.59	2.48	5.50	5.18	3.28	8.00	7.08
	FNN	2.20	5.19	4.42	2.30	5.43	4.63	2.46	5.89	4.98
	FC-LSTM	2.05	4.80	4.19	2.20	5.20	4.55	2.37	5.70	4.96
	DCRNN	1.38	2.90	2.95	1.74	3.90	3.97	2.07	4.90	4.74
	STGCN	1.36	2.90	2.96	1.81	4.17	4.27	2.49	5.79	5.69
	STSGCN	1.38	2.92	3.02	1.76	3.95	4.07	2.11	4.96	4.85
	STFGNN	1.35	2.83	2.91	1.72	3.82	3.89	2.02	4.79	4.63
	ST-MetaNet	1.36	2.82	2.90	1.76	4.00	4.02	2.20	5.45	5.06
	Graph WaveNet	1.34	2.76	2.81	1.65	3.68	3.71	1.95	4.61	4.48
	AGCRN	1.37	2.94	2.87	1.69	3.87	3.85	1.96	4.64	4.54
	GMAN	1.35	2.84	2.93	1.66	3.68	3.79	1.91	4.43	4.39
	MTGNN	1.32	2.77	2.79	1.65	3.69	3.74	1.94	4.53	4.49
	AutoSTG	1.33	2.79	2.78	1.63	3.66	3.62	1.91	4.57	4.42
	AutoSTS	**1.28**	**2.66**	**2.63**	**1.57**	**3.54**	**3.50**	**1.83**	**4.41**	**4.27**

Table 3: Performance comparison of baseline models and AutoSTS on PEMS03, PEMS04, PEMS07 and PEMS08 datasets.

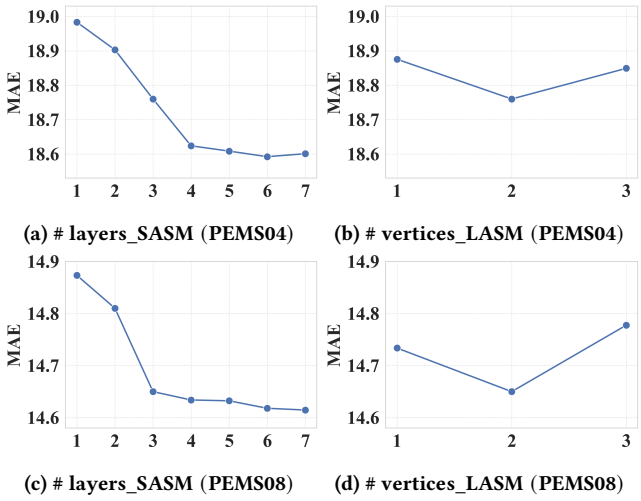
Datasets	Metric	FC-LSTM	DCRNN	STGCN	ASTGCN	Graph WaveNet	STSGCN	STFGNN	STGODE	AutoSTG	AutoSTS
PEMS03	MAE	21.33 ± 0.24	18.18 ± 0.15	17.49 ± 0.46	17.69 ± 1.43	16.74 ± 0.05	17.48 ± 0.15	16.77 ± 0.09	16.53 ± 0.10	16.27 ± 0.27	**14.61 ± 0.03**
	MAPE(%)	23.33 ± 4.23	18.91 ± 0.82	17.15 ± 0.45	19.40 ± 2.24	17.56 ± 1.66	16.78 ± 0.20	16.30± 0.09	16.68± 0.05	16.10 ± 0.03	**14.18 ± 0.07**
	RMSE	35.11 ± 0.50	30.31 ± 0.25	30.12 ± 0.70	29.66 ± 1.68	27.75 ± 0.13	29.21 ± 0.56	28.34± 0.46	28.34± 0.46	27.63 ± 0.78	**24.71 ± 0.40**
PEMS04	MAE	27.14 ± 0.20	24.70 ± 0.22	22.70 ± 0.64	22.93 ± 1.29	20.95 ± 0.09	21.19 ± 0.10	19.83± 0.06	20.84± 0.07	20.38 ± 0.09	**18.76 ± 0.08**
	MAPE(%)	18.20 ± 0.40	17.12 ± 0.37	14.59 ± 0.21	16.56 ± 1.36	14.55 ± 0.17	13.90 ± 0.05	13.02± 0.05	13.76 ± 0.04	14.12 ± 0.02	**12.84 ± 0.01**
	RMSE	41.59 ± 0.21	38.12 ± 0.26	35.55 ± 0.75	35.22 ± 1.90	32.64 ± 0.11	33.65 ± 0.20	31.88± 0.14	32.84 ± 0.19	32.51 ± 0.12	**30.31 ± 0.17**
PEMS07	MAE	29.98 ± 0.42	25.30 ± 0.52	25.38 ± 0.49	28.05 ± 2.34	23.49 ± 0.08	24.26 ± 0.14	22.07± 0.11	23.02± 0.15	23.22 ± 0.33	**20.26 ± 0.02**
	MAPE(%)	13.20 ± 0.53	11.66 ± 0.33	11.08 ± 0.18	13.92 ± 1.65	10.17 ± 0.23	10.21 ± 1.65	9.21± 0.07	10,09± 0.09	9.95 ± 0.01	**8.54 ± 0.04**
	RMSE	45.94 ± 0.57	38.58 ± 0.70	38.78 ± 0.58	42.57 ± 3.31	36.32 ± 0.05	39.03 ± 0.27	35.80± 0.18	37.48 ± 0.39	36.47 ± 0.47	**33.09 ± 0.01**
PEMS08	MAE	22.20 ± 0.18	17.86 ± 0.03	18.02 ± 0.14	18.61 ± 0.40	16.51 ± 0.06	17.13 ± 0.09	16.64± 0.09	16.79± 0.08	16.37 ± 0.12	**14.65 ± 0.04**
	MAPE(%)	14.20 ± 0.59	11.45 ± 0.03	11.40 ± 0.10	13.08 ± 1.00	10.64 ± 0.09	10.96 ± 0.07	10.60± 0.06	10.58± 0.04	10.36 ± 0.03	**9.49 ± 0.07**
	RMSE	34.06 ± 0.32	27.83 ± 0.05	27.83 ± 0.20	28.16 ± 0.48	25.88 ± 0.16	26.80 ± 0.18	26.22± 0.15	26.01 ± 0.14	25.46 ± 0.18	**23.52 ± 0.08**

5.6 Parameter Analysis (RQ4)

To further investigate the effectiveness of our model, we conduct parameter study on PEMS04 and PEMS08, including the number of graph convolution layers of SASM L and the number of vertices in LASM N_v. The experimental results are shown in Figure 5. As shown in Figure 5(a) and (c), MAE on two datasets are close to be optimal When L is equal to 5. With the increase of the depth of message passing L, MAE on PEMS04 and PEMS08 does not become worse. This is because the SASM can adaptively search for a better message passing process according to the data, thus prevent the

Table 4: Ablation experiments.

Dataset	Model&Variants	MAE	MAPE%	RMSE
PEMS04	AutoSTS	**18.76**	**12.84**	**30.31**
	w/o FAFM	19.10	13.10	30.65
	w/o SASM	20.03	13.83	31.93
	w/o LASM	18.89	13.09	30.38
	SASM Random	23.00	16.03	36.14
	LASM Random	19.32	13.51	30.98
	Random	23.67	16.53	36.88
	w/o A_1X	18.91	12.90	30.54
	w/o A_2X	18.88	13.01	30.59
	w/o A_3X	18.94	12.92	30.66
	w/o A_4X	19.01	12.94	30.71
PEMS08	AutoSTS	**14.65**	**9.49**	**23.52**
	w/o FAFM	14.97	9.64	23.86
	w/o SASM	15.79	10.14	25.14
	w/o LASM	14.85	9.79	23.65
	SASM Random	17.69	11.40	28.25
	LASM Random	15.21	9.80	24.12
	Random	18.86	11.87	29.47
	w/o A_1X	14.72	9.61	23.55
	w/o A_2X	14.84	9.64	23.87
	w/o A_3X	14.78	9.63	23.66
	w/o A_4X	14.85	9.55	23.80



(a) # layers_SASM (PEMS04) (b) # vertices_LASM (PEMS04)

(c) # layers_SASM (PEMS08) (d) # vertices_LASM (PEMS08)

Figure 5: Studies on hyper-parameters.

graph convolution from the over-smoothing problem incurred by excessive aggregation of node information. From Figure 5(b) and (d), we can find that our model achieves the best results on these two datasets when the value of N_v is equal to 2. It indicates that the overly complex architecture could arise the over-fitting problem, thus hindering the performance improvement.

5.7 Case Study

We select PEMS04 and PEMS08 to further investigate the relations between their properties and the optimal neural architectures on them. Two important properties of datasets are chosen in this case, including the mean degree of TG and the mean partial auto-correlation coefficient (PACF) of node-wise traffic flow. To be specific, we use mean degree of TG to measure the overall pattern similarity between different nodes on the datasets. Since the temporal graphs are constructed by DTW distances, higher degree means higher pattern similarity between different nodes. We also use the mean partial auto-correlation coefficient (PACF) of node-wise traffic flow to measure the auto-correlation of time series. Higher PACF means higher temporal correlation in the time series. To reveal the properties of the learned neural architectures, we count temporal synchronous graphs convolutions (TSGC: A_1X and A_2X) and temporal convolutions in the output neural architectures on the two datasets. Note that we count and average these properties across the searched neural architectures from five independent experiments.

Table 5: The attributes of two datasets and the corresponding neural architectures.

Properties	PEMS04	PEMS08
Mean degree of TG	4.51	1.27
Mean node-wise PACF	0.1378	0.1144
Average num of TSGCs	75.6	48
Average num of TCNs	4.5	3.4

As shown in Table 5, we observe that the neural architectures on PEMS04 have more TSGCs and TCNs. Since the mean degree of TG on PEMS04 is significantly higher than that on PEMS08, the overall pattern similarity on PEMS04 is higher than PEMS08. Therefore, more TSGCs are expected to extend the message passing hops for learning long-range spatial dependencies. Similarly, the mean node-wise PACF is higher on PEMS04, indicating that it is expected to need more TCNs to generate a larger range of receptive field for capturing long-range temporal dependencies.

Therefore, these results demonstrate that our model can obtain more appropriate neural architectures to capture the spatio-temporal correlations according to different datasets.

6 CONCLUSION

We propose a novel neural architecture search method via multiple graphs to capture complex spatio-temporal correlations for traffic prediction. Extensive experiments on six real-world datasets demonstrate that our model can significantly enhance the prediction accuracy. Besides, a novel final activation and fusion mechanism is designed to improve both the search efficiency and prediction accuracy. In this paper, we give a first attempt to use multiple candidate graphs in the architecture search framework. In future work, we will plan to design the candidate graphs in a more fine-grained way.

ACKNOWLEDGEMENT

This work was supported in part by The National Key Research and Development Program of China under grant 2020YFA0711403, the National Nature Science Foundation of China under 61971267, 61972223, U1936217.

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting.

[2] George Edward Pelham Box and Gwilym M. Jenkins. 1970. *Time series analysis, forecasting and control.*

[3] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. 2020. Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting. *AAAI* 34 (04 2020), 3529–3536. https://doi.org/10.1609/aaai.v34i04.5758

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*. 1294–1303.

[5] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Y. Wang. 2019. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *IEEE TITS* (2019). https://doi.org/10.1109/TITS.2019.2950416

[6] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *ICML* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 933–941.

[7] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*. 933–941.

[8] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-Temporal Graph ODE Networks for Traffic Flow Forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 364–373.

[9] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[10] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *AAAI*.

[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *AAAI* 33 (07 2019), 922–929. https://doi.org/10.1609/aaai.v33i01.3301922

[12] James D. Hamilton. 1994. *Time Series Analysis* (1 ed.). Princeton University Press. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0691042896

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[14] ZHAO Huan, YAO Quanming, and TU Weiwei. 2021. Search to aggregate neighborhood for graph neural network. In *ICDE*. IEEE, 552–563.

[15] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[16] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. *AAAI* 35, 5 (May 2021), 4189–4196. https://ojs.aaai.org/index.php/AAAI/article/view/16542

[17] Ting Li, Junbo Zhang, Kainan Bao, Yuxuan Liang, Yexin Li, and Yu Zheng. 2020. Autost: Efficient neural architecture search for spatio-temporal prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 794–802.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*.

[19] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *ECCV*. 19–34.

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. In *ICLR*.

[21] Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. 2021. AutoSTG: Neural Architecture Search for Predictions of Spatio-Temporal Graph. In *WWW*. 1846–1855.

[22] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. 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)*. 1720–1730.

[23] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *ICML*.

[24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *AAAI*, Vol. 33. 4780–4789.

[25] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *AAAI*, Vol. 34. 914–921.

[26] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. 2020. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics* 50, 9 (2020), 3840–3854.

[27] P. Tormene, Toni Giorgino, S. Quaglini, and M. Stefanelli. 2009. Matching incomplete time series with dynamic time warping: An algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine* 45. http://www.ncbi.nlm.nih.gov/pubmed/19111449

[28] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. 2021. Rethinking architecture selection in differentiable NAS. In *ICLR*.

[29] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*. 10734–10742.

[30] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. *Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks.* Association for Computing Machinery, New York, NY, USA, 753–763. https://doi.org/10.1145/3394486.3403118

[31] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proc. of IJCAI*.

[32] Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *ICCV*. 1379–1388.

[33] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *arXiv: Learning* (2018).

[34] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proc. of IJCAI*. 3634–3640.

[35] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. *AAAI* 34 (04 2020), 1234–1241. https://doi.org/10.1609/aaai.v34i01.5477

[36] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184* (2019).

[37] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).