GSAA: A Novel Graph Spatiotemporal Attention Algorithm for Smart City Traffic Prediction

JIANMIN LIU, XIAODING WANG, HUI LIN*, and FENG YU, College of Computer and Cyber Security, Fujian Normal University, CHINA and Engineering Research Center of Cyber Security and Education Informatization, Fujian Province University, CHINA

With the development of 5G and Internet of Things technologies, the application process of smart transportation in smart cities continues to advance. Sensors are a key source of information for smart transportation, and their data commonly includes complicated traffic scene information. Urban traffic scheduling and efficiency can be significantly increased by deploying data from smart sensors to forecast traffic flows. Despite the fact that some related works have focused on the prediction task of traffic flows, they have not completely mined the traffic spatiotemporal information present in smart sensor data. We offer a novel graph spatio-temporal attention algorithm (GSAA) for traffic prediction in this paper. To completely exploit the geographical and temporal correlations among complicated roadways for traffic forecast, the algorithm combines a spatiotemporal attention mechanism with a graph neural network.To take full advantage of how much effect various hyperparameters provide, deep reinforcement learning is used to obtain the optimal hyperparameters while the predictive model is trained. Experimental results on real-world public datasets show that the algorithm proposed in this paper achieves performance improvements of about 5.47% and 13.10% over the MAE (mean absolute error) than the best baseline strategies for short-term and long-term traffic forecasting, respectively.

CCS Concepts: • **Computing methodologies → Planning under uncertainty**.

Additional Key Words and Phrases: Traffic Prediction, Graph Neural Networks, Attention Mechanism

1 INTRODUCTION

With the acceleration of digital and intelligent development of the whole society, especially the rise of cloud computing [1], big data[2], artificial intelligence [3] and other technologies, intelligent transportation system (ITS) [4] has been greatly developed. The field of smart transportation requires high network capacity, precise positioning and low delay of data transmission. Therefore, a large number of smart sensors are deployed. Sensors are the foundation and core of the development of the Internet of Things, intelligent transportation and smart city construction, and are also key components of the data collected by the intelligent transportation [5]. The sensor network made up of detectors can be an efficient way for the intelligent transportation system to gather information. It detects vehicles approaching each intersection, and the collected data is used to simplify, enhance signal control, increase traffic efficiency, and effectively address other problems like patency, safety, and security that plague urban traffic [6]. Figure 1 depicts the intelligent transportation system's data extraction sensor application.

Authors' address: Jianmin Liu, laujianmin@foxmail.com; Xiaoding Wang, wangdin1982@fjnu.edu.cn; Hui Lin, linhui@fjnu.edu.cn; Feng Yu, fzhiy270@163.com, College of Computer and Cyber Security, Fujian Normal University, No.8 Xuefu South Road, Fuzhou, CHINA, 350117 and Engineering Research Center of Cyber Security and Education Informatization, Fujian Province University, No.8 Xuefu South Road, Fuzhou, CHINA, 350117.

1550-4859/2023/11-ART $15.00
https://doi.org/10.1145/3631608

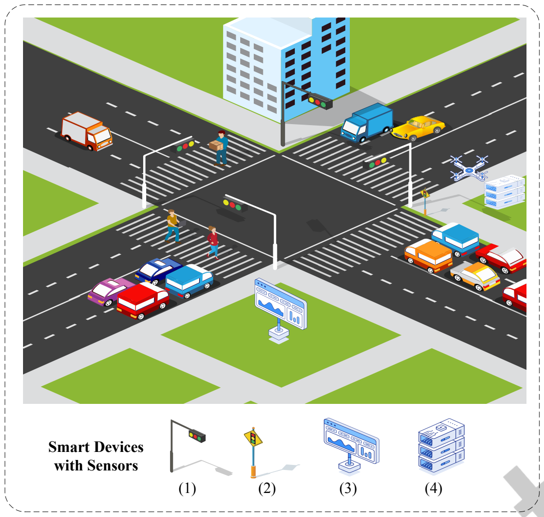


Fig. 1. Sensors in intelligent transportation (Such as the four examples of smart devices with sensors in the picture below: (1)traffic lights at intersections with video camera functions can collect information such as vehicle models, (2) and traffic light sensors used for pedestrian crossings can detect whether there are pedestrians, motorcycle, etc. (3) The sensors on the side billboards monitor the traffic flow on the road, (4) and the speed measuring boxes deployed on specific sections of the road can collect the average speed of passing vehicles.)

Traffic forecasting is an important challenge in the field of intelligent transportation system and mobile robotic systems [7]. By predicting and evaluating dynamic traffic conditions, transportation manager can improve traffic efficiency, ease traffic congestion, increase road capacity, reduce traffic accidents, reduce energy consumption and reduce environmental pollution. However, current methods such as those based on mathematical statistics [8–12] are unable to fully exploit the spatiotemporal value of data. Methods based on deep learning [13–19] have inherent limitations in spatial feature modeling. Methods based on graph neural networks can effectively model spatiotemporal relationships and are currently the mainstream approach [20–30]. However, there are still shortcomings. Most of these methods are unable to quickly adapt to changes in road structures and do not consider the impact of road hierarchy on prediction results. The specific situation is as follows.

- Currently, the main methods for traffic prediction utilize transductive machine learning techniques, focusing on identifying the temporal periodicity and spatial distribution characteristics of traffic patterns, and have achieved satisfactory predictive results. However, they all overlook the fact that road networks may undergo dynamic changes, such as the addition or removal of a road, which can alter the structure of the road network. If transductive graph learning [31], such as GCN, is used for traffic prediction and a new node is added, the prediction network must be retrained to retrieve its embedded representation. This indicates that GCN method is limited to learning the embedded representation of nodes on a specific network and cannot generalise to unknown nodes or learn node representation across a graph. Therefore, inductive task processing method, such as GraphSAGE, SVM and Random Forest, can be adopted, and the embedded representation of newly added nodes in the network can be determined by viewing their adjacency.
- In addition, the traffic conditions are subject to dynamic changes, which are related to the grade and characteristics of the routes. In other words, the hierarchical relationship between roads will also affect the interaction between roads. For example, a car traveling on a main road may not enter an on-ramp for a

long period of time. At this point the interaction on the ramp and the main road will be smaller. In real-life road traffic scenarios, not all nearby roads for a particular road have the same impact on the subsequent traffic conditions, and some are even irrelevant, such as some abandoned roads that cannot provide future traffic flow for that road. Therefore, it is unrealistic to treat all roads equally.

To address the above issues, we propose a novel graph spatiotemporal attention traffic prediction algorithm based on the architecture as shown in Figure 1. The main contributions of this paper are summarized as follows:

- We design a novel inductive graph spatio-temporal attention algorithm (GSAA), which combines the spatio-temporal attention mechanism with an inductive graph neural network to explore the geographical and temporal correlations between complex roads for dynamic traffic flow prediction.
- In order to improve the accuracy of traffic flow prediction, we use deep reinforcement learning to determine the best hyperparameters for prediction model training. Specifically, in order to learn the depth and breadth of the search tree in the aggregation process, the deep reinforcement learning algorithm DQN is used to search the optimal location of the road network nodes and their neighbors.
- We conducted the experiment on real data sets. The experimental results show that the proposed algorithm GSAA achieves better results than the baseline strategy in short-term and long-term traffic forecasting.

The rest of this paper is organized as follows. Section 2 introduces the related works. The system model and problem statement are given in Section 3. The implementation details of the proposed algorithm is elaborated in Section 4. Experimental methods and analysis are given in Section 5. Section 6 concludes this paper.

2 RELATED WORKS

Traffic flow forecasting [32] is an important part of intelligent transportation system and plays an important role in urban traffic control. Traffic prediction has received extensive attention in the field of intelligent transportation, and a lot of realted research work has been presented. The distinctions among all related works are summarized in the Table 1.

2.1 Mathematical and Statistical Methods

The traditional analysis is mainly based on the mathematical statistics of the traffic state at the beginning, such as the historical average model (HA) [8] that used the average value of historical traffic data as the prediction result, and the Bayesian prediction model, which not only used model information and data information, but also fully used prior information had advantages compared to regression models. Later, models that deal with simple time series appeared, such as vector autoregression (VAR) [9], support vector regression (SVR) [10], Kalman filter model [11] , etc. They required the time series to be stationary. Later, the autoregressive integral moving average model (ARIMA) [12] was proposed to solve non-stationary time series, but still cannot handle time series with long-term dependencies.

2.2 Machine Learning Methods

The machine learning-based methods for traffic flow prediction can be divided into the following three categories.

Deep Learning methods. In recent years, the advantages of deep learning in feature capture have attracted the attention of researchers. Many deep learning methods have been applied to traffic flow prediction, such as deep belief networks (DBN)[13, 33] and stacked autoencoder neural networks (SAEs) [15]. The recurrent neural network is a widely used model for processing time series. The long short-term time memory network (LSTM) and the gated recurrent unit (GRU) [16] are typical RNN networks, and they and their variants have good results in time series processing. All of the above methods only consider the temporal dependencies in the sequence, and researchers gradually realized the importance of spatial correlation, and by introducing convolutional neural networks (CNNs) to extract spatial information and combine it with LSTMs [17, 18], improve the prediction

Table 1. **The distinctions among all related works.**

CATEGORY	METHOD	TEMPORAL	SPATIAL	DISTINCTION
Mathematical & Statistical Methods	HA [8]	-	-	Simple to calculate, but with a large margin of error
	VAR [9], SVR [10], Kalman filter [11]	✓	-	Considers the temporal nature of the data, but requires stationary time series
	ARIMA [12]	✓	-	Considered non-stationary data
Deep Learning methods	DBN [13, 33] , SAE [15], GRU [16]	✓	-	Full consideration of timing, big data training for better performance
	Literature [19]	✓	-	Utilizing multiple factors such as weather, date, wind speed, and temperature to make flow predictions
	CNN + LSTM [17, 18]	✓	✓	Spatial structure is considered, but CNNs cannot adapt to non-Euclidean data
Graph Representation methods	TS-STNN[20]	✓	✓	Modeling Spatial Features Using Tree Structures
	T-GCN [21]	✓	✓	Modeling spatio-temporal features using graph representation learning and RNNs
	AST-GCN [22]	✓	✓	Additional consideration was given to external factors such as points of interest, weather, etc.
	DCRNN [23]	✓	✓	Capturing spatial features using bi-directional random wandering
	STGNN [24] and Literature [25–30]	✓	✓	An attention mechanism is introduced to model long-term temporal dependencies
Deep Reinforcement Learning	Literature [34, 35],	-	-	Using Reinforcement Learning to Optimize Traffic Scheduling
	Literature [36]	✓	✓	Using Reinforcement Learning to Generate Dynamical Maps and thus Model Spatio-Temporal Features

accuracy. But CNNs are designed for Euclidean spaces, such as images and grids, which cannot extract the spatial dependencies of traffic flow well in non-Euclidean spaces. Lv *et al.* [20] argue that deep learning spatiotemporal models based on graph convolution theory cannot effectively explore spatial hierarchy and directional information. Therefore, they propose constructing a spatial tree matrix with hierarchical and directional features to extract spatial information.

Pushpendu*et al.* [19] considered multiple factors such as weather, date, average wind speed, and temperature to demonstrate the importance of additional information in traffic prediction.

Graph Representation methods. Graph representation learning has been widely used in many areas [37, 38]. An emerging graph convolutional neural network (GCN) [39] was dedicated to processing network structures [40], which can better model the spatial correlation of road segments in traffic networks [41]. DCRNN [23] captured spatial structure through bidirectional random walks, and modeled time series through encoder-decoder structure. TGCN [21] was a spatiotemporal prediction network model that uses GCN to extract spatial information and GRU to capture temporal information in the sequence. When predicting future traffic flow, AST-GCN [22] not only considers traffic flow information, but also points of interest (POI) and weather conditions around the road. Song *et al.* [25] constructed a local spatiotemporal graph that concatenates individual spatial graphs of adjacent time steps into one graph and captures three spatiotemporal dependencies for spatiotemporal prediction tasks. Guo *et al.* [26] proposed a new attention-based spatiotemporal graph convolutional network model to solve the traffic flow prediction problem. They used three independent components to simulate the characteristics of the three time lengths of the traffic flow respectively to comprehensively predict the traffic flow. Kim *et al.* [27] modeled temporal and spatial dependencies separately, using an attention mechanism to improve prediction accuracy. Wang *et al.* [24] introduced the Transformer framework to improve the modeling ability of long-term temporal dependencies of the model. Xu *et al.* [28] proposed a spatiotemporal transformer structure to capture real-time traffic conditions and directionality of traffic flow by dynamically modeling directed spatial dependencies with a self-attention mechanism. Wang *et al.* [29] simultaneously employ temporal convolutional networks and transformers to model the temporal features of long sequences. They also introduce curriculum learning to optimize the target sequence and avoid getting trapped in local minima. Guo *et al.* [30] designed a self-attention mechanism capable of exploiting local context, enabling the prediction model to capture the temporal dynamics of traffic data, while employing a global receptive field for long-term prediction.

Deep Reinforcement Learning. Walraven *et al.* [34] employed reinforcement learning in the field of smart transportation to improve road traffic flow and prevent traffic jams while also taking into account future traffic flow using various model prediction techniques. For the high-speed moving mobile network [42], Zhou *et al.* [43] employed federated learning to address the knowledge sharing in this distributed scenario, and used reinforcement learning to optimize the aggregation efficiency of the model. By using a hierarchical multi-agent system to split the traffic network region, Abdoos *et al.* [35] applied reinforcement learning to determine the most effective method for implementing joint traffic light scheduling. Peng *et al.* [36] suggested a dynamic traffic flow prediction model that employs reinforcement learning to produce dynamic graphs in order to address potential data faults in traffic prediction.

In summary, the existing methods have not fully considered the impact of road grade and road structure changes on traffic flow, resulting in insufficient exploration of spatiotemporal characteristics. Building upon the aforementioned works, we take into account the dynamic spatiotemporal influence of road nodes on the traffic network.

3 SYSTEM MODEL

Historical traffic information consists of road geographical connectivity and road characteristics. The goal of traffic flow prediction is to use the previous road traffic information to predict the future traffic. In this paper, we use traffic speed as a measure of traffic flow. In addition to traffic speed, other indicators can also be used to evaluate traffic conditions, such as traffic flow, traffic density and traffic performance index (TPI) data. To better formalize the description of the problem, the necessary definitions are provided below.

Definition 1 (Road network G). The topology of the road network is represented by an undirected, unweighted graph called $G(V, E, A)$.

We regard each road or sensor on the road as a node. V represents the set of all road nodes, where V =$\{v_1,v_2,...,v_N\}$. The total number of nodes is denoted by $N = |V|$. The E is the set of edges connecting these nodes. A real, symmetric adjacency matrix $A \in \mathbb{R}^{N\times N}$ can be used to represent these connections, with each element $A[i, j]$ representing the connectivity between nodes i and j. This connectedness often depicts the geographic connectivity between roads or sensors. When them are connected geographically, $A[i, j] = 1$; otherwise, it is 0.

Definition 2 (Feature matrix X). We consider the traffic speed on the road as the feature of each node in the network, denoted as $X \in \mathbb{R}^{N\times T}$, where T signifies the various moments and N denotes the number of nodes. Any other trait, or even a combination of features, might be it.

Each node's traffic status is represented by X^i, and the scalar X_t^i reflects the $i - th$ node's traffic condition at time t. The traffic condition in this instance obviously relates to the pace.

Formal Definition : The formal definition of the problem statement is given below. For the task of road traffic spatiotemporal prediction, which can now be thought of as learning a function map. The network G of roads and the historical traffic data from the feature matrix X serve as the input. The result is the traffic scenario T times in the future, correspondingly. The function map can be expressed as Equation 1:

$$[X_{t+1}, \dots, X_{t+T}] = f(G; (X_{t-N}, \dots, X_{t-1}, X_t))), \tag{1}$$

which N represents the length of the historical traffic data series used, and T represents the length of the time series to be forecasted.

In order to enhance the readability of the article, we have compiled a summary of the abbreviations used in the text and provided explanations for their meanings in the Table 2, and a explanations for symbols in the Table 3.

4 GSAA ALGORITHM DESIGN AND IMPLEMENTATION

Traffic sensors can collect traffic flow data and vehicle locations to help cities optimize traffic flow and reduce congestion and traffic accidents. Smart cities can adjust traffic signals in real time based on traffic flow data and energy consumption data to achieve more efficient energy usage and traffic management. In this paper, we design a spatiotemporal flow prediction model that can fully exploit the spatiotemporal information contained in traffic sensor data. We now give the brief idea of how the proposed GSAA algorithm predicts the traffic flow in urban road networks, as shown in Figure 2.

The GSAA algorithm consists of three components, namely Spatial Dependency Modeling implemented in the GraphSA-Att module, Temporal Dependency Modeling where the GRU and T-Attention modules are used to establish short-term and long-term temporal dependencies, and Deep Reinforcement Learning based Hyperparameter Selection used to improve the prediction accuracy. In general, the algorithm is initially fed with T historical time series and road network topology. The road network's nodes are compared to its neighboring nodes using the similarity function, and the model samples the nodes with the highest similarity in order to aggregate features at GraphSA-Att layer. Instead of using standard aggregation techniques in the aggregation phase, the model converts the similarity matrix between nodes and their neighbors into an attention matrix. The aggregated result will be used as the hidden state of the node at the next moment. In this manner, the spatial feature information of the road network at each instant is gathered. Secondly, in order to capture short-term temporal dependencies through the transfer of hidden states across units, the sequence information comprising spatial characteristics acquired in the prior phase is fed into the gated recurrent unit (GRU). Following that, the output of the gated recurrent unit is routed into the Temporal Attention layer (T-Attention) to record long-term temporal dependencies. We identify global contextual interactions inside the historical time series and capture long-term dependencies by relying on its self-attentive mechanism. Finally, a fully connected layer is used to produce the prediction results.

Table 2. **Meaning of abbreviations**

ABBREVIATION	EXPLANATION
ARIMA	AutoRegressive Integral Moving Average model
AST-GCN	Attribute-augmented SpatioTemporal Graph Convolutional Network
DBN	Deep Belief Network
DCRNN	Diffusion Convolutional Recurrent Neural Network
DQN	Deep Q-Network
GAT	Graph ATtention network
GCN	Graph Convolutional neural Network
GRU	Gated Recurrent Unit
GSAA	Graph Spatiotemporal Attention Algorithm
HA	Historical Average
ITS	Intelligent Transportation System
LSTM	Long Short-Term time Memory network
POI	Points Of Interest
SAEs	Stacked AutoEncoder neural networks
SVR	Support Vector Regression
T-GCN	Temporal Graph Convolutional Network
TPI	Traffic Performance Index
VAR	Vector AutoRegression

Table 3. **Meaning of Symbols**

SYMBOLS	EXPLANATION
$\cdot$	Dot product
$\odot$	Hadamard product
$\phi(x)$	The eigenvectors of x

4.1 Spatial Dependency Modeling

A common inductive graph learning model that has been demonstrated to produce effective results in graph embedding learning is the graph sampling aggregation network (GrapgSAGE). The original GraphSAGE [44], as shown in Algorithm 1.

Note that GraphSAGE makes a random selection of c neighbor nodes for each node $v \in V$ by $RandomSampleNeighbours$ function. If a node has more neighbors than c, then c nodes are picked at random. All of the neighboring nodes will be sampled first if the number of neighbors is less than c, and then a second random sample will be taken to guarantee that c neighbors are sampled. Then, $AGGREGATE$ functions like GCN,

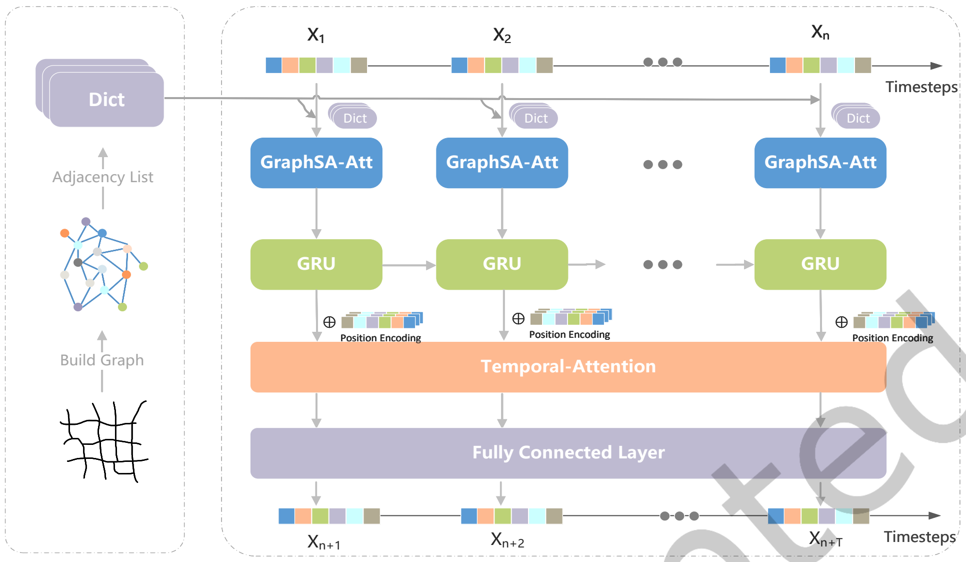


Fig. 2. The Framework of GSAA.

Algorithm 1 GraphSAGE

Require: V : The set of all nodes; c: Number of selected neighbor nodes per layer; $N(v)$:The neighbors of node v; W^k:weight matrices; K:the depth to sample;

Ensure: h_v^k:The feature representation of node v at layer k,with layer 0 being X_v

```
1: $h_v^0 \leftarrow x_v, \forall v \in V$;
2: for $k$=1 ... $K$ do
3:     for $v \in V$ do
4:         $N(v) \leftarrow RandomSampleNeighbours(v, c)$   // For each node $v$, randomly sample $c$ neighboring nodes.
5:         $h_{N(v)}^k \leftarrow AGGREGATE(h_v^{k-1}, \forall v \in N(v))$   // Aggregate the neighboring nodes of the k-1 layer.
6:         $h_v^k \leftarrow \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^k))$   // Obtain the features of the central node $v$ at layer k.
7:     end for
8:     $h_v^k \leftarrow h_v^k / \|h_v^k\|_2$   // Normalization.
9: end for
```

MEAN, MAX and LSTM, etc. combine the nearby node characteristics into $h_{N(v)}^k$. After that, the representation of v at the k-1*th* layer, denoted by h_v^{k-1}, and its neighbors' are concatenated to obtain the k*th* layer representation of the node v, denoted as h_v^k. The parameters k and c are both hyperparameters, and are referring to the depth or number of layers sampled as well as the number of surrounding nodes sampled at each layer. The authors recommend [44] that 2 layers be sampled and $S_1 * S_2 \leq 500$, where S_1 and S_2 are the number of neighbor nodes per layer that were sampled.

No doubt that we can't pick neighbors at random. Instead, we want to pick neighboring roads with comparable traffic flow, thus these roads are in the similar condition. Thereby, we redefine the sampling algorithm in Section 4.1.1, and propose a spatial attention mechanism based on the geographic adjacency of roads in Section 4.1.2. At last, to obtain the spatial dependencies in the traffic flow at the current moment, we use the attention aggregation function elaborated in Section 4.1.3.

4.1.1 Sampling Algorithm Based on Similarity . The link between roads is contained in the traffic network G, and two neighboring nodes signify that the two roads are close by geographically. Generally speaking, their characteristics will be relatively similar. Consider the scenario where the traffic on this road will, in a certain direction, shift to the adjacent road in a few seconds. Thus, two roads adjacent to each other will have comparable traffic circumstances.

However, the traffic on this road may move in any direction at any time, because each road has several forks. In fact, the traffic on this road may come from various roads, including paths and thoroughfares. In most cases, due to the low road traffic flow, the impact on traffic prediction is not particularly significant. In addition, the traffic on the main road tends to maintain this condition for a long period of time, and will not easily turn to the secondary road. In order to obtain more accurate information when sampling, we choose neighboring nodes with high traffic similarity.

GraphSA Att with attention mechanism is divided into two parts, namely sampling and aggregation. We will introduce in depth the implementation of the sampling and aggregation module using the attention mechanism. Algorithm 2 is our proposed sampling algorithm based on similarity.

Every node in the road network is considered the center node in turn. We then extract each node's neighbor list, donate as $Neighbous_v$, and use the *score* function to determine how comparable the center node v and neighboring nodes are. Here, the dot product is used as *score* function in Equation (2):

$$score(v, u) = x_v \cdot x_u^T. \tag{2}$$

Algorithm 2 Sampling Algorithm based Similarity

Require: V : The set of all nodes; c: Number of neighbors were chosen; *score*: The function to calculate similarity of two nodes; *Sort*: Sorting nodes by similarity; K:the depth to sample;
Ensure: N_v : The node chosen as node v's neighbor finally;

```
1:  for $k$=1 ...$K$ do
2:    for $v \in V$ do
3:      $N_v^k \leftarrow Neighbous_v[\ldots]$            // All neighbors of node $v$ at $k$ depth.
4:      if $len(N_v^k) > c$ then
5:        for $u \in N_v$ do
6:          $value \leftarrow score(v, u)$       // Calculate the similarity between node $v$ and u.
7:        end for
8:        $Sort(N_v^k, value)$                   // Sort the nodes based on the similarity.
9:        $N_v^k \leftarrow Sample(N_v^k, c)$       // Sample $c$ nodes from the neighbors.
10:     else
11:       $N_v^k \leftarrow N_v^k$
12:     end if
13:   end for
14:   $N_v \leftarrow Append(N_v^k)$
15: end for
```

The next step is to order the neighbor nodes based on this similarity value, and then sample the top c nodes to gather information. Any type of quick and effective sorting algorithm can be used as the *Sort* function. All of them are sampled to determine the node's neighbors if the number of neighbor nodes is fewer than c. The sample depth here is likewise K layer, same like in graphsage. The neighboring nodes N_v^k engaged in each layer are combined to form N_v. For the purpose of choosing K and c in this experiment, deep reinforcement learning is used. This will be discussed in Deep Reinforcement Learning based Hyperparameter Selection.

4.1.2 *Spatial Attention Mechanism .*

The connection strength between two nodes, or the similarity ratio between two roads or sensors, is represented by each value of the attention coefficient matrix, W_a, which we generate. It guarantees that those neighboring nodes to be aggregated to node v have an attention factor sum of 1. In this method, it is possible to properly model the road's spatial structural features into the traffic forecast model. Our model can recognize dynamic changes in the traffic information flow to better represent the spatial dependencies in the traffic network because we compute the similarity of the traffic information at each instant. To assess the similarity between two routes, we utilized Equation (2), which is the inner product of two vectors. Note that for other distance metrics, Euclidean distance is not applicable to high-dimensional data, Manhattan distance is not very intuitive, and may not provide the shortest path, and Chebyshev distance is not universal, and it is difficult to simplify the calculation. Most of them are not suitable for the similarity measurement of traffic data, or can not better establish the dependence of traffic roads.

Equation (3) is used to determine the attention coefficient between two nodes, where $W_a[u, v]$ stands for the attention coefficient between nodes u and v. W_a is not a true symmetric matrix, as should be noted. Even when two nodes have the same score similarity, their surrounding nodes differ, and as a result, the final attention coefficients also differ

$$W_a[v,u] = \frac{exp(score(x_v, x_u))}{\sum_{k=1}^{N} exp(score(x_v, u_k))}. \tag{3}$$

4.1.3 *Attention Aggregation Function .*

We select the attention based approach to aggregate the characteristics. Equation (4) is used to aggregate the information from neighboring nodes to the central node using the attention-based aggregation approach. And we need to aggregate information from the core node's k-hop neighbors k times.

$$h_{N(v)} = W_a \cdot X, \tag{4}$$

where $h_{N(v)}$ is a synthesis of the neighbor characteristics of the v nodes in this layer, W_a is the attention matrix, X is the input traffic data feature matrix and the symbol $\cdot$ means dot product. h_v^k, the final hidden feature of node v at layer k, is then obtained by combining the characteristics of node v at layer k-1 and $h_{N(v)}$ using Equation (5):

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^k)), \tag{5}$$

where W^k is the parameter matrix gained from training, $CONCAT(\cdot)$ is the splicing operation on the two vectors, and σ is the nonlinear activation function.

Here, we use Figure 3 as an example to explain the similarity of the nodes with k=2 and c=3.

First, the three one-hop neighbors of the center node v are chosen from its neighbor nodes in the first layer. The next step is to determine the second-order neighbors or two-hop neighbors of the central node v for each node in the first-order neighbors. This is how sampling is performed in Figure 3a. Then, we aggregate the information of the second-hop neighbor nodes on the first-order neighbor nodes using the attention coefficient matrix created during sampling, and then aggregate the first-order neighbor node information containing the second-order neighbor information to the central node v. As shown in the Figure 3b. Through the twice aggregation, we are able to incorporate the neighbor node information for up to 2 hops. Similar to the previous example, if the sample depth is k layers, we need to aggregate k times during aggregation, with the exception that sampling is done from the center node outward and aggregation is done from the peripheral nodes inward. Finally, we get the node's subsequent concealed state as shown in Figure 3c and use it later in the computation.

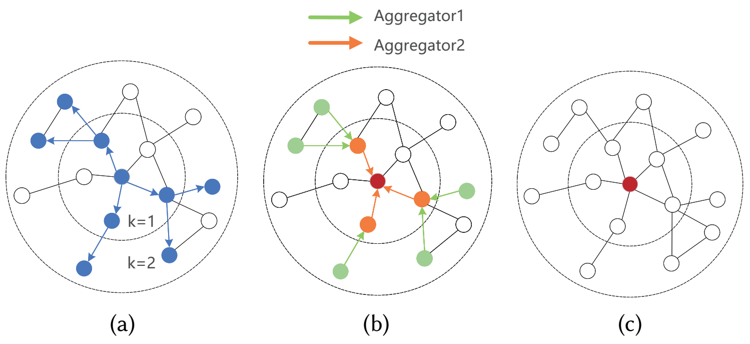


Fig. 3. GraphSA-Att Progatation.

In summary, we use the modified GraphSage algorithm, GraphSA-Att, to learn spatial features in traffic data. As shown in the Figure 3, for the central node v, we sample c neighbours from its neighbor nodes, then aggregate the features of the sampled nodes, and finally concat it and the previous layer features of node v as the node v's next hidden state.

4.2 Temporal Dependency Modeling

In traffic forecasting, it is crucial to capture time-dependent relationships just as much as geographical dependency. To identify short-term and long-term dependencies, we employ a cyclic gating unit (GRU) and an attention layer, respectively. Different from separately inputting the output of the GraphSA-Att module into GRU and Temporal-Attention Layer and then adding their results, this paper first inputs the output of the spatial module into GRU. The output of each time step of GRU is then sequentially fed into the Temporal-Attention Layer, where each input of the Temporal-Attention Layer represents all the spatiotemporal features before the current time step. This approach is more conducive to capturing long-term spatiotemporal dependencies.

4.2.1 Short-term Dependencies. GRU is a classical RNN algorithm. Both GRU and LSTM are variants of RNN. Their principles are almost the same, but GRU has fewer parameters and can maintain the same effect as LSTM. Therefore, we chose GRU to capture short-term dependencies.

At time t, the specific operation of GRU can be expressed as follows:

$$u_t = \sigma(W_u \cdot [X_t, h_{t-1}] + b_u), \tag{6}$$
$$r_t = \sigma(W_r \cdot [X_t, h_{t-1}] + b_r), \tag{7}$$
$$c_t = tanh(W_c \cdot [X_t, (r_t \odot h_{t-1})] + b_c), \tag{8}$$
$$h_t = u_t \odot h_{t-1} + (1 - u_t) \odot c_t. \tag{9}$$

Among them, $\odot$ is the Hadamard product, which means that the matrix is multiplied element-wise; X_t means that the spatially dependent traffic data is already included; W_u, W_r, W_c ,b_u ,b_r and b_c are three different sets of linear variation parameters that the model needs to be trained.

u_t is an update gate, which can be obtained using Equation (6), to control how many states are substituted into the current time at the previous time; r_t is the reset gate, which can be obtained using Equation (7), to control the forgetting degree of the hidden state of the previous time; c_t is a candidate for the hidden layer state, which can be obtained using Equation (8), and the multiplication of r_t and h_{t-1} indicates whether past information is helpful for predicting the future. When r_t approaches zero, the model discards the hidden information in the past, leaving only the current input information. When r_t approaches 1, the past information is considered to be useful, and it is added to the current information.

In Equation (9), the update gate is used to integrate the previous state with the current candidate state to produce the hidden state h_t at the present time.

4.2.2 Long-term Dependencies. Longer-term reliance must also be taken into account when predicting traffic, while GRU captures short-term dependence. So, to capture long-term dependence, we suggest utilizing a multi-head attention method.

The multi-head attention mechanism is used to identify long-term dependencies. It was developed as an inspiration for the temporal attention mechanism in Transform [45]. Transformer encoder is composed of multi-head attention module, residual and normalization module, and feedforward neural network module. The attention module uses scaled dot product attention, which is calculated as shown in Equation (10):

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{QK}^T}{\sqrt{d_k}})\boldsymbol{V}, \tag{10}$$

where d_k refers to the dimension of the $\boldsymbol{K}$ vector, $\boldsymbol{Q}$ represents the query vector, $\boldsymbol{K}$ represents the key vector, and $\boldsymbol{V}$ represents the value vector. $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are all obtained from the input sequence vector by Equation (11). If $\boldsymbol{X}$ is the input sequence vector, then:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{W}_q\boldsymbol{X} + \boldsymbol{b}_q, \\ \boldsymbol{K} &= \boldsymbol{W}_k\boldsymbol{X} + \boldsymbol{b}_k, \\ \boldsymbol{V} &= \boldsymbol{W}_v\boldsymbol{X} + \boldsymbol{b}_v, \end{aligned} \tag{11}$$

where $\boldsymbol{W}_q, \boldsymbol{b}_q, \boldsymbol{W}_k, \boldsymbol{b}_k, \boldsymbol{W}_v, \boldsymbol{b}_v$ are three different sets of linear variation parameters that the model needs to be trained.

The above attention layer can be called single head attention layer, while multi-head attention refers to using different $\boldsymbol{W}_q^i, \boldsymbol{b}_q^i, \boldsymbol{W}_k^i, \boldsymbol{b}_k^i, \boldsymbol{W}_v^i, \boldsymbol{b}_v^i$ to calculate the output of single head attention layer several times and then splicing them all by Equation (12):

$$\begin{aligned} MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= Concat(\boldsymbol{head}_1, \ldots, \boldsymbol{head}_I)\boldsymbol{W}_o; \\ where \quad \boldsymbol{head}_i &= Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}). \end{aligned} \tag{12}$$

Among them, $\boldsymbol{W}_o$ is the linear change matrix to be trained by the model. Its function is to adjust the vector's dimension after multi-head attention operation to the dimension at the time of input.

The so-called residual is to add the initial vector after multi-layer transmission through the neural network. The function is to effectively prevent the gradient from disappearing when the network is deep. Layer normalization (LN) is similar to batch normalization (BN), which is the operation of normalizing data. The so-called feedforward neural network in transformer is the structure of multilayer perceptron(MLP). The only thing worth mentioning is that in this structure, the dimensions of input vector and output vector are the same, and the dimensions of the middle hidden layer can be adjusted at will.

Because the transformer cannot know the relevant position information in the sequence like GRU, the data needs to be position embedded to $\hat{X}$ before being input into the temporal attention layer which is shown as (13):

$$\hat{X} = \boldsymbol{emb}_{in} + \boldsymbol{PE}_{in}, \tag{13}$$

where $\boldsymbol{emb}_{in}$ refers the representation after GRUs and $\boldsymbol{PE}_{in}$ is the position embedding which is defined by Equation (14) as follows:

$$\begin{cases} \boldsymbol{PE}_{in}(pos, 2i) = sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right) \\ \boldsymbol{PE}_{in}(pos, 2i+1) = cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right), \end{cases} \tag{14}$$

where d_{model} represents the dimension of the input vector in this model at this time. pos represents the position of the input sample in the sequence, starting from 0. $2i$ and $2i+1$ must be regarded as two structures. $2i$ represents the even digit in the vector, and $2i+1$ is the odd digit.

After capturing the spatial and temporal characteristics, we use a multilayer feedforward neural network to predict the traffic situation in the future. At the same time, the prediction layer can also help us determine how long to predict the traffic situation in the future. It can be shown as:

$$(Y_{t+1}, \ldots, Y_{t+t'}) = f(H_{t-h}, \ldots, H_t),$$

where H_t is the output result of the temporal attention layer at the previous t time, Y_{t+1} is the prediction result at $t+1$ time.

4.3 Deep Reinforcement Learning based Hyperparameter Selection

Reinforcement learning (RL) is one of the paradigms and methods of machine learning, which is used to describe and solve the problem that agents learn strategies to maximize rewards or achieve specific goals in the process of interacting with the environment. In traffic forecasting, the agent receives the feedback of the model by acting on various model parameters and feeding them back to the environment or model. Finally, the parameters of the model training are selected as the actions that have the best impact on the prediction. In this paper, we use the deep reinforcement learning algorithm Nature DQN [46] to select parameters, that is, to select the number of sampling layers of neighbor nodes which marked as k and the number of neighbors sampled in each layer which marked as c .

We consider the inference capability of the GSAA model as the state, which is an abstract concept. Specifically, we describe the current model's inference capability by combining different evaluation metrics into a feature vector $\phi(s)$. After each state performs a different action, it will receive a reward, denoted as R. We define the reward as the negative value of the evaluation metric MAE.

Nature DQN uses the state action value function to evaluate the strategy. Its input is a state action pair. The state action value function indicates the expected value of the cumulative reward that can be obtained by selecting an action in a state. In order to explore the cumulative reward value of more action spaces in the current state as much as possible, the state action value function approximation technique is used, making the action value function $Q(s, a; w)$ close to the optimal one $Q^*(s, a)$, i.e.,

$$Q(\phi(s), a, w) = Q^*(\phi(s), a), \tag{15}$$

where $\phi(s)$ represents the vector of the state, which means the inferring ability of the model at the current round;; a represents the action performed by the model referring to two hyperparameters: k namely the number of sampling layers and c the number of samples; w represents the parameters of neural network training.

The $Q(\phi(s), a, w)$ is used to calculate the estimated value of each state-action pair. In Nature DQN [46], in order to converge the Q network, a target Q' network with the same structure as the Q network is introduced to compute the true value of each state-action pair. The parameters of the Q network are periodically copied to the target Q' network. Therefore, the calculation of the target Q value can be obtained as formula 16:

$$y_j = \begin{cases} R_j, & end_j \text{ is true,} \\ R_j + \gamma max_{A'} Q'(\phi(S_{j+1}), A'; w'), & end_j \text{ is false.} \end{cases} \tag{16}$$

Among them, γ is the discount factor, Q' is the target Q network with network parameters w', and A' is the action taken by the model in a given state S_{j+1}, end_j indicates whether the termination condition has been met.

To improve the accuracy of the Q network, we need to minimize the difference between the estimated Q value and the target Q value. The calculation for this difference L is as formula 17.

$$L = \frac{1}{m} \sum_{j=1}^{m} (y_j - Q(\phi(S_j), A_j, w))^2 \tag{17}$$

where m represents the number of samples involved in the training. By using the method mentioned above, we can obtain the optimal hyperparameters for the model.

5 EXPERIMENTS

The experiments were conducted on a server running Ubuntu 20.04 LTS. The server was equipped with an NVIDIA GeForce RTX 3090 GPU and 96GB of RAM. The training was performed using the Pytorch 2.0 deep learning framework. The batch size for model training was set to 4, and the neural network had 64 hidden units. The training was carried out for 200 epochs by using the Adam optimizer. During the training process, the learning rate and training set size are set to 0.001 and 0.8, respectively. The number of layers in the temporal attention layer is set to 1.

5.1 Dataset Description

We conduct experiments on two real-world datasets. A brief description of the two datasets (aviable at https://github.com/lehaifeng/T-GCN) is given below:

- Shenzhen: This dataset is the taxi record data from January 1, 2015 to January 31, 2015 in Luohu District, Shenzhen. This dataset has two parts, an adjacency matrix, which records the geographic connectivity of 156 roads, and a feature matrix, which records the traffic speed of each road, recorded every 15 minutes.
- Los-loop: This dataset comes from real-time loop detection points on freeways in Los Angeles, USA. It recorded 207 sensor data from March 1, 2012 to March 7, 2012. Traffic speed data is recorded every 5 minutes. Its adjacency matrix represents the distance between sensors, and the feature matrix contains some missing values, which are filled by linear interpolation.

5.2 Evaluation Metrics

We used four indicators to evaluate the framework. The smaller the value, the better the performance.

(1) Mean absolute error (MAE) is defined as the average of the absolute errors in Equation (18).

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left| Y_i - \hat{Y}_i \right|. \tag{18}$$

(2) Root mean squared error (RMSE) is defined as the rooted average squared difference between the predicted values and the ground truth in Equation (19).

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2}. \tag{19}$$

(3) Mean absolute percentage error (MAPE) is defined as Equation (20):

$$MAPE = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right|. \tag{20}$$

(4) Coefficient of Determination (R2) is defiend as a metric to evaluate the quality of fit in Equation (21).

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \left(\hat{Y}_i - Y_i\right)^2}{\sum_{i=1}^{N} \left(Y_i - \bar{Y}\right)^2}. \tag{21}$$

In the above equations, N denotes the number of samples, and Y_i and $\hat{Y}_i$ denote the ground truth, the prediction of the $i-th$ sample; Y and $\hat{Y}$ represent the set of Y_i and $\hat{Y}_i$ respectively; $\bar{Y}$ is the average of Y.

5.3 Baselines

We compare our GSAA model with (1) historical average (HA), (2) autoregressive integral moving average model (ARIMA) [12], (3) support vector regression (SVR) [10], (4) diffusion convolutional recurrent neural network (DCRNN) [23], (5) temporal graph convolution model (TGCN) [21], (6)Spatial Temporal Graph Neural Network (STGNN) [24]. The following is a brief description of their principles:

- HA: The historical average uses the weighted average of previous periods as a forecast for future periods.
- ARIMA: Autoregressive ensemble moving average model with Kalman filter, which can be used for non-stationary time series, it fits time series data as a structure for future forecasts.
- SVR: Support vector regression regresses time series using support vector machines.
- DCRNN: Diffusion convolutional recurrent neural networks model traffic flow as a diffusion process. It captures spatial dependencies using bidirectional random walks on the graph and temporal dependencies using an encoder-decoder architecture. But it still relies on the road network to disseminate information, which limits the flexibility of the model.
- T-GCN: Temporal GCN combines graph convolutional networks and recurrent gating units for spatiotemporal flow prediction.
- STGNN: Spatial temporal graph neural network, which uses a transformed GCN to learn a latent variable to model spatial relationships and a Transforemer to capture long-term temporal dependencies.

Among them, the hyperparameters are the same as mentioned in the paper or in the code.

5.4 Model Parameters Settings

During the experiment, we use 80% of the dataset as the training set and the remaining 20% as the test set to verify the effectiveness of the model. Also, since graph sampling is done for nodes, we convert the adjacency matrix into a dictionary where the keys are the ordinal number of each node and the values are the ordinal numder of the center node's neighbors. During data preprocessing, we take the data of the previous 12 moments as historical data, and predict the traffic speed of the next 3 steps, 6 steps, 9 steps and 12 steps respectively.

For our GSAA model, the Adam optimizer [47] is used for training, we set the learning rate, batch size and training set scale to 0.001, 4 and 0.8 respectively during training. We set the number of layers of the temproal attention layer to 1 and other parameters determined by experimental testing.

Choice of the number of neighbors we select 5, 10, 15, 20, and 25 to find the best value, because the maximum number of neighbors of a node in the dataset is less than 30. We set other parameters unchanged, when the number of sampled neighbors changes, we train the same number of epochs, and the obtained MAE and RMSE metrics are shown in the figure 4a. The resulting MAPE and R2 metrics are shown in the figure 4b. It can be seen that when the number of sampled neighbors is 20, MAE, RMSE and MAPE all reach the minimum, and R2 reaches the maximum. When the number of neighbors is less than 20 or more than 20, the performance will be lost, so we choose to sample the number of neighbor nodes as 20.

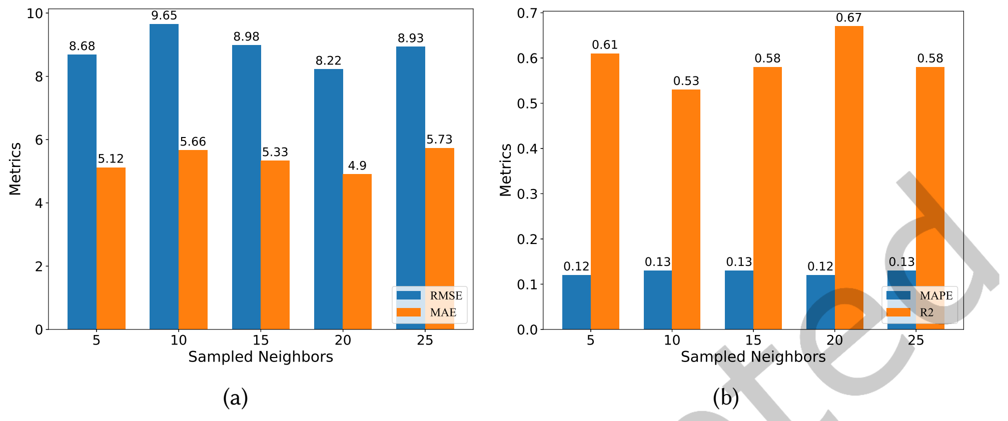


Fig. 4. MAE,RMSE,MAPE and R2 metrics on different sampled neighbors after trained on the same epoches.

Furthermore, we employ deep reinforcement learning to update the parameters for re-validation in order to completely eliminate the contingency of manually picking parameters. We utilized a deep reinforcement learning model based on the DQN learning approach to simulate the performance of the model training results under various parameter states in order to determine the optimal number of sampling layers k and the number of samples per layer c. Figure 5 presents the outcome. In Figure 5, X axis denotes the number of sampling layers, while Y axis represents the number c of neighbor nodes sampled at each layer . Since it has been suggested in the article reference[44] that the number of layers should not exceed two layers, we also take into account 3-hop neighbors for the experiment to be as successful as possible. The darker the color, the smaller the MAE value of the model evaluation index, acquired when the agent applies the X and Y sampling approach.

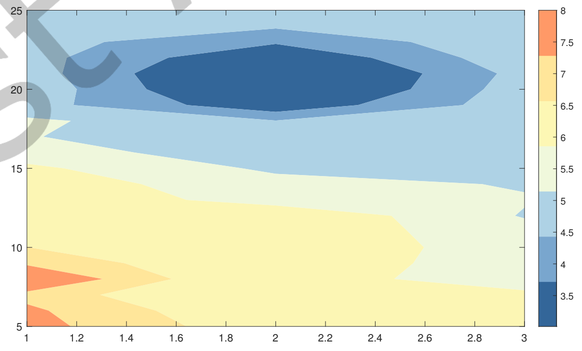


Fig. 5. Experimental results of DQN for different parameters (the darker the color, the better the effect)

As shown in Figure 5, when the number of sampling layers is fixed, the error of the model has been decreasing with an increase in sampling neighbor nodes, reaching a minimum value between 20 and 25. Similarly, when the number of sampling neighbors is relatively fixed, the error of the model has also been decreasing when the more

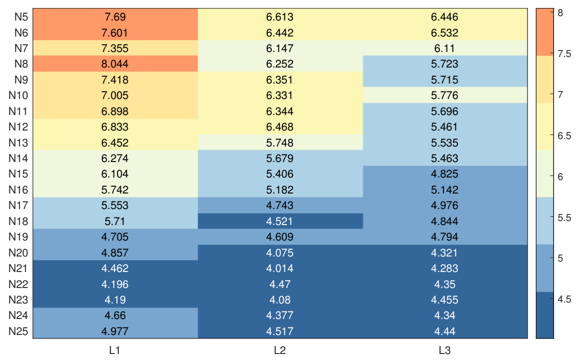


Fig. 6. Heatmap of experimental results for specific parameter combinations.

sample layers are chosen. However, in a similar vein, when the number of sampling neighbors and the number of sampling layers are at their highest, the model does not function at its best. We also give the corresponding heatmap as shown in Figure 6, where $N5, \ldots, N25$ represents the number of neighbors sampled, $L1, L2, L3$, and the number of layers sampled, and the value indicate the MAE values under training the same number of epochs.

Thereby, we determine the parameters of the experiment that the number of hidden units is 64, the number of neighbor nodes is 20, the number of sampling layers is 2, and the number of attention heads is 4.

5.5 Experimental Results

The results of the experiment are analyzed in the following.

We compare the performance of our proposed model with six baseline methods on the shenzhen and losloop datasets. Table 4 shows the prediction results for four time steps on the shenzhen dataset. The prediction results of the four time steps on the losloop dataset are shown in Table 5.

Table 4. Result on dataset Shenzhen

	Shenzhen															
METHOD	15MIN				30MIN				45MIN				60MIN			
	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2
HA	2.7815	4.2951	0.2577	0.8307	2.7815	4.2951	0.2602	0.8307	2.7815	4.2951	0.2613	0.8307	2.7815	4.2951	0.2644	0.8307
ARIMA	4.9824	7.2406	0.2801	*	4.6765	6.7899	0.2583	*	4.6734	6.7852	0.2597	*	4.6655	6.7708	0.2558	*
SVR	2.6233	4.1455	0.2527	0.8423	2.6875	4.1628	0.2527	0.841	**2.7359**	4.1885	0.2526	0.8391	2.7751	4.2156	0.2528	0.837
DCRNN	3.17	4.5033	0.2528	0.8391	3.23	4.5623	0.2573	0.8332	3.27	4.6006	0.2535	0.8275	3.31	4.6412	0.2593	0.8219
GCN	4.2367	5.6596	0.2563	0.6654	4.2647	5.6918	0.2571	0.6616	4.2844	5.7142	0.2507	0.6589	4.3034	5.7361	0.2544	0.6564
GRU	**2.5955**	3.9994	0.2457	0.8329	2.6906	4.0942	0.2483	0.8249	2.7743	4.1534	0.2503	0.8198	2.7712	4.0747	0.2508	0.8266
T-GCN	2.7117	**3.9265**	0.2488	0.8541	2.741	3.9663	0.2533	0.8456	2.7612	3.9859	0.2528	0.8441	2.7889	**4.0048**	0.2519	0.8422
STGNN	3.0645	4.2381	0.2444	0.8358	2.9854	4.2789	0.2466	0.8297	3.1681	4.3734	**0.2499**	0.8252	3.2046	4.4132	0.2522	0.8222
GSAA	2.6406	3.9324	**0.2439**	**0.8672**	**2.6872**	**3.9589**	**0.2464**	**0.8587**	2.7526	**3.9789**	0.2517	**0.8451**	**2.7139**	4.0101	**0.2498**	**0.8443**
GSAA w/o S-ATT	3.6016	4.7029	0.2477	0.8359	2.9025	4.8811	0.2549	0.8521	2.9406	4.8138	0.2552	0.8184	3.7157	4.5842	0.2523	0.7595
GSAA w/o T-ATT	3.0437	4.6361	0.2478	0.8238	3.1745	4.4415	0.2572	0.8331	3.3132	4.5811	0.2557	0.8426	3.1878	4.8478	0.2507	0.7629

The * sign represents a negative value, which is not considered.

It can be seen from the two tables that our proposed GSAA achieves the best performance on both datasets. HA is based on the average value of historical traffic speed information as the predicted value, so we select 12 historical moments to predict the future 15min, 30min, 45min and 60min respectively, and get the same result. So we did not compare its results with other model results. Traditional methods based on mathematical statistics, such as ARIMA and SVR, can also achieve good results when dealing with linear and relatively stable time series. However, it often does not work well for nonlinear and non-stationary time series. Compared with traditional

Table 5. Result on dataset Los-loop

METHOD	Los-loop 15MIN				30MIN				45MIN				60MIN			
	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2	MAE	RMSE	MAPE	R2
HA	4.0145	7.4427	0.0821	0.7121	4.0145	7.4427	0.0821	0.7121	4.0145	7.4427	0.0821	0.7121	4.0145	7.4427	0.0821	0.7121
ARIMA	7.6832	10.0439	0.0783	*	7.6891	9.345	0.0923	*	7.6924	10.0508	0.0944	*	7.6952	10.0538	0.0943	*
SVR	3.7285	6.0084	0.0762	0.8123	3.7248	6.9588	0.0844	0.7492	4.1288	7.7504	0.0897	0.6899	4.5036	8.4388	0.1025	0.6336
DCRNN	4.06	5.7432	0.079	0.8277	4.53	6.3972	0.0824	0.7854	4.67	7.0238	0.0986	0.7352	4.92	7.3384	0.1105	0.7132
GCN	5.3525	7.7922	0.0808	0.6843	5.6118	8.3353	0.093	0.6402	5.9534	8.8036	0.0994	0.5999	6.2892	9.2657	0.1148	0.5583
GRU	3.0602	5.2182	0.0673	0.8576	3.6505	6.2802	0.0789	0.7957	4.0915	7.0343	0.0842	0.7446	4.5186	7.6621	0.1041	0.698
T-GCN	3.1802	5.1264	0.0786	0.8634	3.7466	**6.0598**	0.0941	0.8098	4.1158	6.7065	0.1006	**0.7679**	4.6021	7.2677	0.1084	0.7283
STGNN	3.2513	5.2673	0.0783	0.8587	3.6873	6.2012	0.1017	0.8016	3.5440	6.9109	0.1173	0.7538	4.7782	7.6234	0.1198	**0.7123**
GSAA	**2.9583**	**5.1211**	**0.0649**	**0.8636**	**3.4506**	6.1899	**0.0754**	**0.8116**	**3.5224**	**6.7065**	**0.0841**	0.7538	**4.4646**	**7.5474**	**0.0918**	0.7279
GSAA w/o S-ATT	3.9117	5.2315	0.0742	0.8203	3.6545	7.0359	0.0871	0.8104	3.7925	7.2205	0.0926	0.7057	4.8689	7.8299	0.1071	0.6808
GSAA w/o T-ATT	3.8312	5.5847	0.0777	0.8145	3.7903	6.9417	0.0824	0.8022	4.3556	7.4861	0.0974	0.7211	5.0479	8.4837	0.1079	0.6715

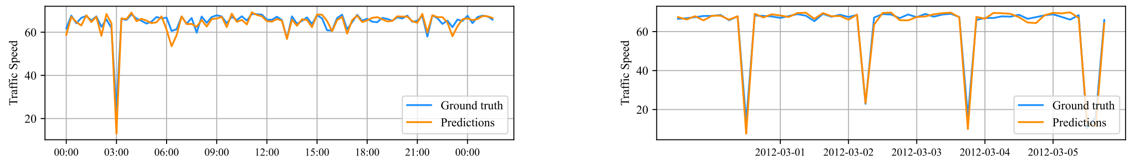
The * sign represents a negative value, which is not considered.

methods, deep learning-based methods have achieved good results in dealing with nonlinear and non-stationary data, such as GCN and GRU. There are also methods that consider both temporal and spatial dependencies, such as TGCN, DCRNN, STGNN including our GSAA, these models achieve better results than simple deep learning methods such as GRU, especially in modeling long-term spatial and temporal dependencies.

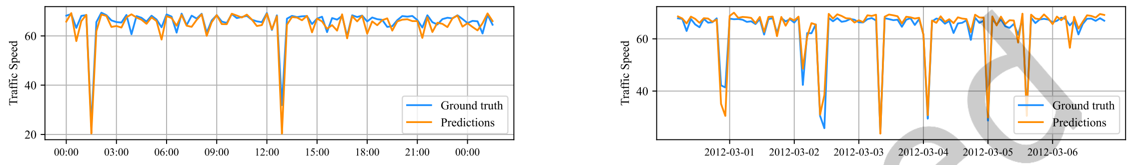
Specifically, compared to the DCRNN and STGNN methods, the GSAA method in this paper achieves better spatial features by deeply mining road hierarchy and features. Together with the fact that the GSAA model utilizes the GRU method to model short-term temporal dependencies, thus demonstrating better performance in short-term traffic flow prediction compared to them. Additionally, thanks to the introduction of the attention layer, the GSAA method can model longer-term spatiotemporal dependencies. Therefore, in longer-term predictions of 30 minutes, 45 minutes, and 60 minutes, the GSAA method outperforms the prediction results of the TGCN model. Moreover, it can be observed that the STGNN method, which also incorporates attention mechanisms, performs better than the DCRNN method, which proves that the attention mechanism has a good effect on the capture of spatiotemporal dependencies. This is also verified on our GSAA model. We add an attention mechanism to the aggregation process of graph node information and the capture of long-term temporal relationships. The accuracy of model predictions is improved, and the error is reduced.

From the ablation experiments, it can be seen that the effect of the model is less degraded in performance in short-term traffic prediction after removing the spatial attention mechanism, because the temporal dependence is greater than the spatial dependence in short-term traffic prediction, so the short-term traffic prediction is much worse after removing the temporal dependence; In the long-term prediction experiments,after removing the component of spatial dependence, the contextual association information established by relying only on temporal attention is not enough to predict the long-term traffic situation, and the performance of the model decreases more; Similarly, because the model loses the previous contextual information after removing the establishment of temporal dependence, the prediction ability of the later period also decreases. From the ablative experiments, it can be seen that both temporal and spatial attention mechanisms are very important in the prediction of traffic flow.

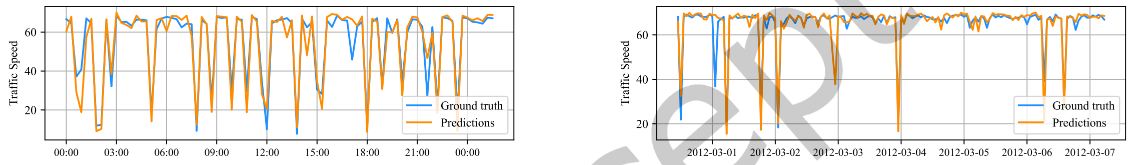
To study the effectiveness of the model in traffic prediction, we selected some nodes on the $Los - loop$ dataset and plotted its real and predicted traffic under different horizons. Figure 7a shows the daily and weekly traffic forecasts at a forecast horizon of 3, representing the short-term and long-term traffic forecasting tasks, respectively. It can be seen that with a prediction horizon of 3, the model fits almost perfectly and achieves a good result, no matter the short-term traffic prediction for each day or the long-term traffic prediction for a week. As the prediction field of view increases, when the prediction field of view is 6, as shown in Figure 7b, there is a slight jitter in the prediction effect of the model, but the overall prediction effect is still very good. When the prediction horizon continues to increase to 9, as shown in Figure 7c, there may be jitter in the short-term



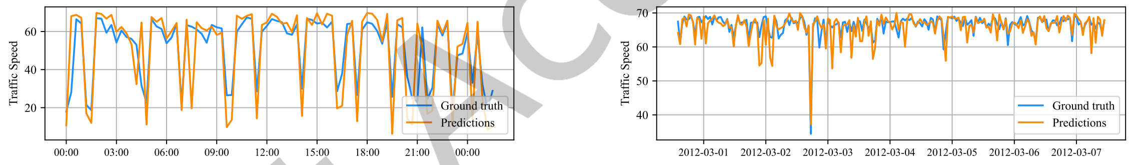
(a) 15-minute prediction by 3 steps.



(b) 30-minute prediction by 6 steps.



(c) 45-minute prediction by 9 steps.



(d) 60-minute prediction by 12 steps.

Fig. 7. The graphs above and below show the comparison for the accumulated forecast results for one day and one week, respectively, compared to the true values on the Los-loop Dataset.

prediction, but the prediction performance is more stable in the long-term prediction effect, thanks to the spatio-temporal attention, the processing of information in the traffic flow. Its performance is more evident when the prediction horizon become to 12, as shown in Figure 7d, where the spatio-temporal relationships in network traffic are still well captured and well modeled in a long-term prediction task where the traffic fluctuates a lot. The temporal dependency of traffic flow is more pronounced in short-term forecasting, and several generic forecasting techniques frequently perform well. The spatial-temporal network based on graph attention mechanism is more suited to handle the complicated spatio-temporal traffic connection as the forecasting range expands and the spatial dependency in traffic flow also becomes significant.

6 CONCLUSION

In this study, we propose a spatio-temporal traffic flow prediction algorithm model based on inductive graph machine learning method in smart cities. The model can be implemented on any sensor device to offer precise and real-time resource scheduling for processing, storage, network, and other resources while making full use of data from intelligent sensors. In this study, we optimized the sampling process of the graph and effectively aggregated traffic features. Additionally, the model effectively captures the spatiotemporal characteristics of traffic flow data through spatial sampling, gated recurrent units, and temporal attention structures. The experimental results on two real world datasets show that our model is more effective and superior than other baseline methods. However, the current model relies too heavily on the selection of hyperparameters. Future considerations will focus on the interaction of traffic information in dynamic spatiotemporal settings, enabling the model to adaptively model the semantic relationships on the road.

7 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.

8 ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China under Grant No. 61702103 and U1905211, and Natural Science Foundation of Fujian Province under Grant No. 2020J01167 and 2020J01169.

REFERENCES

[1] Priyanshu Srivastava and Rizwan Khan. 2018. A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering* 8, 6 (2018), 17–20.

[2] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, and Laurence T Yang. 2020. Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations. *IEEE Transactions on Computational Social Systems* 8, 1 (2020), 171–178.

[3] Caiming Zhang and Yang Lu. 2021. Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration* 23 (2021), 100224.

[4] Agachai Sumalee and Hung Wai Ho. 2018. Smarter and more connected: Future intelligent transportation system. *Iatss Research* 42, 2 (2018), 67–71.

[5] Luca Filipponi, Andrea Vitaletti, Giada Landi, Vincenzo Memeo, Giorgio Laura, and Paolo Pucci. 2010. Smart City: An Event Driven Architecture for Monitoring Public Spaces with Heterogeneous Sensors. In *2010 Fourth International Conference on Sensor Technologies and Applications*. 281–286. DOI: http://dx.doi.org/10.1109/SENSORCOMM.2010.50

[6] W.J. Fleming. 2001. Overview of automotive sensors. *IEEE Sensors Journal* 1, 4 (2001), 296–308. DOI: http://dx.doi.org/10.1109/7361.983469

[7] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, Zheng Yan, Laurence T Yang, Wei Wei, Jianhua Ma, and Qun Jin. 2023. Decentralized P2P Federated Learning for Privacy-Preserving and Resilient Mobile Robotic Systems. *IEEE Wireless Communications* 30, 2 (2023), 82–89.

[8] Brian L Smith and Michael J Demetsky. 1997. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering* 123, 4 (1997), 261–266.

[9] Eric Zivot and Jiahui Wang. 2006. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®* (2006), 385–429.

[10] Haowei Su, Ling Zhang, and Shu Yu. 2007. Short-term traffic flow prediction based on incremental support vector regression. In *Third International Conference on Natural Computation (ICNC 2007)*, Vol. 1. IEEE, 640–645.

[11] Azadeh Emami, Majid Sarvi, and Saeed Asadi Bagloee. 2019. Using Kalman filter algorithm for short-term traffic flow prediction in a connected vehicle environment. *Journal of Modern Transportation* 27 (2019), 222–232.

[12] Billy M Williams and Lester A Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering* 129, 6 (2003), 664–672.

[13] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. 2014. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems* 15, 5 (2014), 2191–2201.

[14] Xiaokang Zhou, Yiyong Hu, Jiayi Wu, Wei Liang, Jianhua Ma, and Qun Jin. 2023. Distribution Bias Aware Collaborative Generative Adversarial Network for Imbalanced Deep Learning in Industrial IoT. *IEEE Transactions on Industrial Informatics* 19, 1 (2023), 570–580. DOI: http://dx.doi.org/10.1109/TII.2022.3170149

[15] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.
[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[17] Yipeng Liu, Haifeng Zheng, Xinxin Feng, and Zhonghui Chen. 2017. Short-term traffic flow prediction with Conv-LSTM. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 1–6.
[18] Yuankai Wu and Huachun Tan. 2016. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022* (2016).
[19] Pushpendu Kar and Shuxin Feng. 2023. Intelligent Traffic Prediction by Combining Weather and Road Traffic Condition Information: A Deep Learning-Based Approach. *International Journal of Intelligent Transportation Systems Research* (2023), 1–17.
[20] Yang Lv, Zhiqiang Lv, Zesheng Cheng, Zhanqi Zhu, and Taha Hossein Rashidi. 2023. TS-STNN: Spatial-temporal neural network based on tree structure for traffic flow prediction. *Transportation Research Part E: Logistics and Transportation Review* 177 (2023), 103251. DOI: http://dx.doi.org/10.1016/j.tre.2023.103251
[21] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2019), 3848–3858.
[22] Jiawei Zhu, Qiongjie Wang, Chao Tao, Hanhan Deng, Ling Zhao, and Haifeng Li. 2021. AST-GCN: Attribute-augmented spatiotemporal graph convolutional network for traffic forecasting. *IEEE Access* 9 (2021), 35973–35983.
[23] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
[24] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*. 1082–1092.
[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 *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 914–921.
[26] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.
[27] Kihwan Kim, Seungmin Jin, Sungahn Ko, and Jaegul Choo. 2020. STGRAT: A Spatio-Temporal Graph Attention Network for Traffic Forecasting. (2020).
[28] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. 2020. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908* (2020).
[29] Chunzhi Wang, Lu Wang, Siwei Wei, Yun Sun, Bowen Liu, and Lingyu Yan. 2023. STN-GCN: Spatial and Temporal Normalization Graph Convolutional Neural Networks for Traffic Flow Forecasting. *Electronics* 12, 14 (2023). DOI: http://dx.doi.org/10.3390/electronics12143158
[30] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2021. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[31] Giorgio Ciano, Alberto Rossi, Monica Bianchini, and Franco Scarselli. 2021. On inductive–transductive learning with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 2 (2021), 758–769.
[32] Roland Chrobok, Oliver Kaumann, Joachim Wahle, and Michael Schreckenberg. 2004. Different methods of traffic forecast based on real data. *European Journal of Operational Research* 155, 3 (2004), 558–568.
[33] Yuhan Jia, Jianping Wu, and Yiman Du. 2016. Traffic speed prediction using deep learning method. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 1217–1222.
[34] Erwin Walraven, Matthijs TJ Spaan, and Bram Bakker. 2016. Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence* 52 (2016), 203–212.
[35] Monireh Abdoos and Ana LC Bazzan. 2021. Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long-short term memory. *Expert systems with applications* 171 (2021), 114580.
[36] Hao Peng, Bowen Du, Mingsheng Liu, Mingzhe Liu, Shumei Ji, Senzhang Wang, Xu Zhang, and Lifang He. 2021. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Information Sciences* 578 (2021), 401–416.
[37] Xiaokang Zhou, Wei Liang, Weimin Li, Ke Yan, Shohei Shimizu, I Kevin, and Kai Wang. 2021. Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system. *IEEE Internet of Things Journal* 9, 12 (2021), 9310–9319.
[38] Wei Liang, Yiyong Hu, Xiaokang Zhou, Yi Pan, I Kevin, and Kai Wang. 2021. Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. *IEEE Transactions on Industrial Informatics* 18, 8 (2021), 5087–5095.
[39] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[40] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[41] Zhishuai Li, Gang Xiong, Yuanyuan Chen, Yisheng Lv, Bin Hu, Fenghua Zhu, and Fei-Yue Wang. 2019. A hybrid deep learning approach with GCN and LSTM for traffic flow prediction. In *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 1929–1933.

[42] Jingjing Guo, Zhiquan Liu, Siyi Tian, Feiran Huang, Jiaxing Li, Xinghua Li, Kostromitin Konstantin Igorevich, and Jianfeng Ma. 2023. Tfl-dt: A trust evaluation scheme for federated learning in digital twin for mobile networks. *IEEE Journal on Selected Areas in Communications* (2023).

[43] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurance T Yang, Shohei Shimizu, I Kevin, and Kai Wang. 2023. Digital Twin Enhanced Federated Reinforcement Learning with Lightweight Knowledge Distillation in Mobile Networks. *IEEE Journal on Selected Areas in Communications* (2023).

[44] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[47] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).