A method for traffic flow forecasting based on the Node-Walk algorithm

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Traffic flow forecasting is a complicated task because of the complicated spatial dependence of different roads and the dynamically changing time series data. However, a large amount of existing work cannot effectively model the temporal and spatial features of traffic flow and merge them effectively. In this paper, we propose a novel node-walk spatial-temporal attention network(NWSTAN) model to solve such difficulties. NWSTAN have three components: spatial feature extraction module based on node-walk algorithm; temporal attention mechanism module; spatial attention mechanism module. Node-walk module is used to capture spatial features preliminary. The last two modules are utilized to extract temporal and spatial information further. The output of above two components are sent to our well-designed gated fusion unit for processing, and finally output the predicted value. Experimental results on several public traffic datasets demonstrate that our method achieves state-of-the- art performance consistently than other baselines.**

***Index Terms*—trafficflow forecasting, deep learning, node2vec, Node-Walk**

I. Introduction

With the fast acceleration of urbanization and the expansion in vehicle stock in recent years, traffic demand has risen dramatically, causing urban traffic congestion to worsen and increasing traffic system's pressure. One significant way of addressing such problem is the intelligent transportation system, which is being developed in several nations. Intelligent transportation systems can offer real-time traffic information to cars, assisting drivers in finding the best route from their starting point to their destination. The prediction of short-term traffic flow of an urban traffic road network is essential for traffic guidance system which is the heart of ITS (Intelligent Transportation System) [1].

However, precisely forecasting traffic flow is challenging. The reasons relies on the complexity of the urban road network topology and operation characteristics of traffic flow. The urban transportation network may be seen as a topological graph, with nodes representing corresponding detectors and edges representing the geographic connection of the two detectors. On the one hand, adjacent graph vertices have spatial connections and impact one another, this means that the road network's complicated structure will have an impact on traffic flow modeling and forecasting. Each vertex, on the other hand, corresponds to a time series have a temporal correlation, these time series reflect the traffic conditions at these locations, and their fluctuations are unstable and non-linear [2].

With the rise in the quantity and quality of traffic data in recent years, an increasing number of researchers have attempted to tackle the aforementioned difficulties. Their research approaches may be broadly split into two categories: model-driven and data-driven. The former depicts complicated changes in traffic state by creating a mathematical model of each traffic flow parameter and predicting traffic state based on this model, Non-parametric regression, Kalman filtering, and historical trends based on statistical characteristics are only a few examples [3]. However these approaches have drawbacks: they can't handle non-stationary or nonlinear time series data, and can't deal with complicated road network models. The latter, also known as a machine learning method, can represent complex temporal and geographical interactions. Traditional machine learning requires corresponding domain knowledge for cumbersome feature design, which limits its practical value. In recent years, deep learning has become widely popular among researchers. For example, convolutional neural networks(CNN) is used for grid-like data processing [4]; graph convolutional neural networks(GNN) is used to process non-Euclidean spatial data that is not grid-like: graphs [5]; the attention mechanism can solve the feature extraction problem of time series data in a parallel method fastly [6]. It's a new challenge to figure out how to use the aforementioned approaches in a integrated way to address the problem of processing spatio-temporal correlation data.

We suggest a new model in this article: Node-Walk based Spatial-Temporal Attention Network(NWSTAN) used to forecast traffic conditions at various nodes among the road network. This model can process graph-based traffic data and extract the Spatio-temporal correlations to achieve more accurate predictions. Its characteristics can be summarized as follows:

- To extract the spatio-temporal correlation of traffic flow data, this paper introduces a spatio-temporal correlation attention method. The method first involves the well-known Node-Walk algorithm and then modifying it to entirety model such that it can extract spatial similarity between each road network nodes.
- We also developed the time attention mechanism accord-

ingly, computed each time point's correlation coefficient to extracted time correlations of traffic data for more accurate forecasting of the traffic flow volume.
- Our deep neural network model was evaluated on various real-world datasets and compared with current benchmarks. It is clear from the results of the experiments that our model has certain advantages.

II. Related Work

A. Statistical Methods

Traffic flow forecasting is a classic time series forecasting problem, and numerous researchers have done extensive studies on it [6]. In mathematics, time series are values taken at a series of equally spaced time points. Time series forecasting is the process of predicting future values using historical data and statistical models. The most well-known methods include ARIMA, VAR [7]. Autoregressive integrated moving average(ARIMA) model is a generalization of an autoregressive moving average(ARMA) model. ARMA is used to model stationary or weak stationary time series. ARIMA is an improvement over ARMA because its algorithm includes data difference operations, which helps to eliminate the non-stationarity of the data and makes data analysis simple. ARIMA is a time series model with a single variable. While many time series data in natural sciences and engineering are multivariate, with complicated connections between variables. The Vector Autoregression(VAR) was created to address this issue. VAR assumes there is a linear relationship between the components of the multivariate vector. The above approaches are extensively utilized, but their disadvantage is that they all contain some basic assumptions. And the actual data is too intricate to conform to such assumptions. As a result, the predictions are inaccurate.

B. Machine Learning Methods

Machine learning offers several benefits over traditional statistical approaches, including the ability to handle vast amounts of data automatically, requiring relatively few assumptions. SVM, KNN, and other machine learning methods can model complex data, but they need careful feature engineering [8]. Deep neural network, also known as deep learning, is a relatively new machine learning method distinguished by automated feature extraction, which improves the shortcoming of traditional machine learning's laborious features design. Deep learning evolves from a fully connected network to a convolutional neural network for image classification and then to a recurrent neural network for time series data processing. Recurrent neural networks can extract the time series characteristics of data. Many scholars have tried to use this network to process traffic flow data and achieved good results. Gated recurrent neural network (GRU) and long short-term memory network (LSTM) are two more typical variations [9]. The time parameter is explicitly added in the modeling of recurrent neural network, but this complicates the design of its parallel algorithm [].

C. Attention Mechanism

There is also a class of techniques in neural networks that imitate cognitive attention: attention mechanisms. This approach mimics human attention mechanisms by focusing on less, more relevant data to increase prediction accuracy and reduce the amount of computation. Soft attention, hard attention, key-value attention, and multi-head attention are examples of attention mechanisms. It can be used in a variety of fields such as Natural Language Processing(NLP), computer vision(CV). Google's transformer model for machine translation is maybe the most well-known example [5]. Many researchers have recently attempted to introduce attention mechanisms to the object detection and other tasks in the field of computer vision, and the experimental results have improved.

D. Graph Representation Learning

Many scientific problems can be abstracted into graphs. Therefore, a variety of algorithms are required to process graph data intelligently. Traditional machine learning algorithms are mainly designed for image, text, and voice data, and their ability to process graph data is insufficient. Graph data is different from the image. It is not Euclidean data and cannot be directly input to neural networks such as CNN. Therefore, graph representation learning is needed to represent the graph as data in Euclidean space and then use machine learning methods to process it [10]. There are classic kernel methods for graph representation learning, as well as machine learning methods. Graph kernel is a kernel function that computes an inner product on graphs. Graph kernels can be intuitively understood as functions measuring the similarity of pairs of graphs. Examples of kernel methods are random walk kernel and Weisfeiler-Lehman graph kernel, etc. Machine learning methods include Deepwalk [11], Node2vec [12], Graphgan [13], etc. Node2vec can project each node of the graph into a vector in Euclidean space [14]. This algorithm is widely used in recommendation systems, natural language processing, and social networks.

E. Spatial-Temporal Forecasting

Spatio-temporal data prediction is a significant scientific problem in many fields such as traffic flow forecasting. The key to this question is how to fuse the two types of data. DCRNN [15] using the diffusion random process to design the convolution kernel, extracts the spatial features of the road network, and combines it with time series forecasting, and has achieved good results. STGCN [16] employed graph convolution based on the Fourier transform of graphs to acquire spatial features and temporal convolution on the time axis to obtain temporal characteristics. Graph WaveNet [17] is a framework com- bines adaptive adjacency matrix into graph convolution with 1D dilated convolution. STSGCN [18]completely merges space and time blocks by using local spatio-temporal synchronization graph convolution. STFGNN [19] used dynamic time warping algorithm to measure differences of each time series produced by different detector. It model

spatial and temporal correlation in a different method and get good result.

III. Preliminaries

A. Traffic Data

In this paper, we define the road network as an undirected graph $G = (V, E, A)$ as shown in Fig.**??**, where V is the set of vertexs; E is the set of the edges, indicating the undirected edges between two vertexs; $A \in \mathbb{R}^{N \times N}$ denotes adjacent matrix of graph G. Each detectors on the network samples data in a same time interval. The sampled data is a series of vectors of number F corresponding to each time point. Its each component represents the metrics we care about, such as speed, traffic, and occupancy.

(a) pic1. (b) pic2.

Fig. 1. pics

B. Traffic Flow Forecasting

$x_t^i \in \mathbb{R}$ denotes feature vector of node i at time point t. $\mathcal{X} = (X_1, X_2, ..., X_\tau)^T \in \mathbb{R}^{N \times F \times \tau}$ denotes all the features of all the nodes over τ time slices.

Problem: We define T_p as the num of next several time slices which we care about. Given $\mathcal{X}$, the model predict future traffic flow sequences $Y = (y^1, y^2, ..., y^N)^T \in \mathbb{R}^{T_p}$.

IV. Node-walk search strategy

To search and find relationship between each node, the machine learning model should traverse the entire topology garph. The mainstream search strategies are depth first search(DFS) and breadth first search(BFS).

A. Breadth First Search

As shown in Figure 4-2, we define the node A as the starting vertex, The depth-first search strategy is to start from node A to find nodes that are adjacent to node A and have not been searched. At this time, the nodes that meet the above conditions are: S_1, S_2, S_3, S_4, S_5. If this algorithm can't find the proper adjacent node, then it return to the previous node to perform depth priority search. As shown in Figure 4-2, the depth priority search process started from the node A, and then find node S_5. S_5 have the neighborhood S_6, so the next node S_6 can be added in neighborhood set. The above process is shown in Figure 4_3. In brief, the depth priority search can be expressed as a strategy gradually expending neighborhood set in a near-to-far way.

(a) pic1. (b) pic2.

Fig. 2. pics

B. Depth First Search

The breadth priority search algorithm is a blind search method to systematically expand and check all node information in the graph. As shown in Figure 4-2, node A is the starting node. The first step of the strategy can find all node near the node A(S_1, S_2, S_3, S_4, S_5), then search second-order neighbor of nodes in neighborhood set until the full topology graph is traveled, this search process is shown in Figure 4-4.

(a) pic1. (b) pic2.

Fig. 3. pics

C. Homogeneity and structural equivalence

Both breadth-first search and depth-first search are extreme search techniques, but when it comes to predicting the link between nodes in a graph, they frequently alternate between the two qualities of homogeneity and structural equivalence, that is, taking into account the depth Priority search and breadth-first search. According to the homogeneity hypothesis, there is a substantial link between nodes that are well interconnected and belong to similar network clusters or communities (node S_1 and node A in Figure 4-2). On the other hand, the structural equivalence hypothesisshows that nodes in the topological graph with identical structural roles have a strong relationship. In Figure 4-2, for example, node A and node S_7 are both the hubs of their respective network clusters. Importantly, unlike the homogeneity theory, structural equivalence does not focus on node connectedness. For example, some nodes in the network may be far distant, yet they all have the same structural features.

D. Node-Walk search algorithm

We designed a topological graph node search algorithm-Node-Walk based on the above two search strategies and knowledge of homogeneity and structural equivalence. This algorithm combines depth-first and breadth-first search strategies, as well as real-world traffic data. The search bias α ensures that the neighboring nodes of the target node are

searched, as well as encouraging the search for deeper nodes in the graph.

Consider each vehicle detector deployed on the road as a node on the graph G(V,E)(V represent the vertices on the topological graph, E represent the connecting edges between the nodes on the graph). The roads are viewed as edges connecting nodes in a topological graph. To represent the spatial distance between the vehicle detector and the road network graph, we choose the following exponential function. So the edge weight can be expressed as:

$$w_{ij} = \begin{cases} exp\left(-\frac{{d_{ij}}^2}{\sigma^2}\right) & i \neq j \\ 0 & others \end{cases} \tag{1}$$

σ^2 is a controlling factor that is used to control the distribution of edge weights.

N_i is the neighborhood set of node i. We also define the traffic characteristics data score of the target node in the algorithm to better explore the actual relationship between the related nodes:

$$t_{score_i} = \omega_f \frac{f_i - f_{min}}{f_{max} - f_{min}} + \omega_s \frac{s_i - s_{min}}{s_{max} - s_{min}} + \omega_o \frac{o_i - o_{min}}{o_{max} - o_{min}} \tag{2}$$

ω_f, ω_s, and ω_o are the traffic flow weighting coefficient, the vehicle speed weighting coefficient, and the lane occupancy weighting coefficient respectively. From them we can calculate the edge weight matrix $W \in R^{N \times N}$(N is the number of nodes on the graph, that is, the number of vehicle detectors).

In the urban traffic network it is necessary to find some nodes that relate to the target node which will affect the traffic conditions of a certain area. This is why we have developed the Node-Walk algorithm to search the adjacent nodes which are closely linked to the target node. The main issue about the Node-Walk algorithm is the probability of jumping to a certain node from the next jump. So we have developed a second-order step procedure and defined p_{vx} as the probability of node jumps and α as the next node search bias: s and q are the parameters for jump guidance.

Fig. 4. Example of a figure caption.

As shown in Figure 5, we assume that the node t is the target node and the first-order search of the edge (t, v) is performed. Now the search node turn to the node v. Because the Node-Walk algorithm searches for a collection of nearby nodes linked to the target node, when the search node jumps to this node, it becomes the starting node for the search for the second-order node. At this point, we set the jump probability of the starting node to $p_{vx} = \alpha_{sq}(t, x_i)w_{vx}$:

$$\alpha_{sq}(t, x_i) = \begin{cases} 0 & d_{t_{x_i}} = 0 \\ \frac{d_{score_{x_i}} t_{score_{x_i}} w_t x_i}{q} & d_{t_{x_i}} = 1 \\ \frac{w_{2nd} d_{score_{x_i}} t_{score_{x_i}} w_v x_i}{s} & d_{t_{x_i}} = 2 \end{cases} \tag{3}$$

d_{tx_i} is the topological distance between the node and x_i (for clarity, the length of each edge in Figure 5 is seted to one because this work is studying the node's second-order walk, the value of d_{tx_i} can only be 0, 1, 2), w_{tx_i} and w_{vx_i} are the edge weights between node t, node v, and x_i, respectively. $d_{score_{x_i}}$ is the normalized score of node x_i, $t_{score_{x_i}}$ is the normalized traffic feature data score of node x_i, and $w_{2^{nd}}$ is the search node's second-order jumping coefficient. According to equation (3), the parameter q controls whether the search node searches the neighbourhood of node v, and the parameter s controls whether the node searches deeper areas in the network. We set the search length to L, then walk through all nodes in the road network topology graph to obtain the neighborhood node set of all nodes. The neighborhood node set is represented by S_{node}, $S_{node} \in R^{N_{node} \times L \times n}$.

V. NWSTAN FRAMEWORK

We will introduce our NWSTAN short-term traffic flow prediction model in this section. NWSTAN searches for nodes that have a strong correlation with the target node and builds a set of related nodes before mining the traffic flow with the spatial feature extraction module. The spatial features of the data are mined by Node-Walk algorithm preliminarily; the temporal features of traffic flow data are mined using self-attention and time embedding; the gated fusion module is used to integrate the mined spatial and temporal features.

Fig. 5. Example of a figure caption.

The NWSTAN framework, as shown in Figure 6, first includes a neighborhood node selection module that employs the Node-Walk algorithm to determine the relevant node set of each node in the traffic graph. The traffic feature data is then preprocessed (vehicle flow, vehicle speed, lane occupancy), divided, and sliced to produce the label data set and training data set. The historical training data is fused and reconstructed based on the relevant data set of each node obtained by the Node-Walk algorithm to obtain the traffic characteristic data set related to each node.The benefits of traditional two-dimensional convolution in spatial feature extraction are then used to perform preliminary spatial extraction on the reconstructed traffic feature data. To obtain the spatial and temporal features of the traffic feature data mined, the preliminary

features are sent to the spatial feature extraction module and the temporal feature extraction module, respectively. Finally, the feature gated fusion module is used to obtain the final output result, which is based on the relationship between the time feature and the space feature of the traffic feature data. Finally, the feature gated fusion module designed according to the relationship between the time feature and the space feature of the traffic feature data is used to obtain the final prediction, and this result is compared with the label data to adjust the relevant parameters of the neural network to obtain the best parameters.

A. Correlation mining between nodes

$D_t \in \mathbb{R}^{E\times H\times N\times F}$ denotes the feature data of all nodes in the training set for all periods (E is the number of time blocks after fragmentension, H is the size of the historical time window, N is the total number of nodes on the road network graph, and F is the feature of the traffic feature data Number of parameters). As shown in Figures 7(1) and (2), take the e-th time slice as an example, merge $D_e \in \mathbb{R}^{H\times N\times F}$ with the neighborhood node-set $S \in \mathbb{R}^{N\times L\times n}$ of the target node and reconstruct to obtain a new training set $\mathcal{D}_e$. $\mathcal{D}_e \in \mathbb{R}^{H\times N\times L\times n}$, (L is the length of the node walk, n is the number of walks), and the reconstructed traffic data $D_m \in \mathbb{R}^{H\times N\times L\times n}$ is used as the input data of the network structure shown in Figure 7(3). This sec is shown in Figure 7(3). This part contains three layers of two-dimensional convolutional layers and one fully connected layer, and uses the output of the fully connected layer as the feature value of each node at each moment. Finally, we get the abstract feature data $T_m \in \mathbb{R}^{N\times H\times C_{out_1}}$ (as shown in Figure 7(4)), C_{out_1} is the abstract feature dimension, $T = \{T_1, T_2, ..., T_t\}$ and $T \in \mathbb{R}^{E\times H\times N\times C_{out_1}}$ (E is the number of time blocks after fragmentation). Take the e-th time slice as an example, and merge with the neighborhood node-set $S \in \mathbb{R}^{N\times L\times n}$ of the target node, as shown in Figures 7 (1) and (2).

Fig. 6. Example of a figure caption.

B. Spatial feature extraction module

This work uses an independent self-attention mechanism instead of spatial convolution to establish a complete attention model to extract the spatial characteristics of traffic flow. Similar to the two-dimensional convolution, we assume there is a node $X_{ho} = R^{c_{in}}$, (h represents the h-th time point in the historical time window, o represents the o-th node in the node sequence, and C_{in} is the dimension of input feature). To get the spatial feature, first, in the neighborhood set $ab = N_k(h, o)$ of X_{ho} (take the node X_{ho} as the center), this module extract the local area feature of the node in the space of size k. And define the center node X_{ho} attention score output as $y_h o = R^{d_{out}}$, at this time:

$$y_{ho} = \sum_{a,b\in N_k(h,o)} softmax_{ab}(q_{ho}^T K_{ab}) v_{ab} \tag{4}$$

Among them, $q_{ho} = W_o X_{ho}$; $k_{ab} = W_k X_{ab}$; $v_{ab} = W_v X_{ab}$. Above process performs a linear transformation on the central node X_{ho} and its neighbor nodes, and w_q, W_k, and W_v are trainable linear transformation parameters. The above calculation process is shown in Figure 4.

The spatial position information of the neighbohood node of the target node X_{ho} relative to the target node is not included in the above calculation process. So the positional relationship of the node's neighborhood cannot be reflected if there is no embedded position information. This work uses relative position embedding to represent the relative position relationship between nodes and incorporates position information into the self-attention calculation to solve this difficult. An example of relative position calculation is shown in Figure 8.

Fig. 7. Example of a figure caption.

Fig. 8. Example of a figure caption.

And by adding relative position information to the formula (4), the attention formula with relative position information can be obtained:

$$y_{ho} = \sum_{a,b\in N_k(h,o)} softmax_{ab}(q_{hn}^T K_{ab} + q_{hn}^T r_{a-h,b-n}) v_{ab} \tag{5}$$

The row offset is $a-h$, the column offset is $b-h$, the row and column offset embeddings are r_{a-h} and r_{b-n}, respectively, and the row and column offset embeddings are connected to get the position embedding vector $r_{a-b,b-n}$.

Although embedding location information in independent self-attention has been successfuly collected in process of capturing spatial features, using content-based self-attention mechanism to capture edge location information becomes difficult. To compensate the disadvantages without significantly increasing the number of calculations, We added distance-based information into w_v through a linear transformation of

spatial variation to bridge the gap between the self-attention mechanism and spatial convolution. At this time:

$$v_{ab} = (\sum_{m} p(a,b,m) W_V^m) \tag{6}$$

$$p(a,b,m) = softmax_m((emb_{row}(a) + emb_{col}(b))^T V^m) \tag{7}$$

Where the W_v^m is multivariate function, $emb_{row}(a)$ and $emb_{col}(b)$ is the row and column embedding aligned with the pooling window, V^m is mixture embedding. So the final independent self-attention formula can be expressed as:

$$y_{ij} = \sum_{a,b \in N_k(h,n)} softmax_{ab}(q_{hn}^T k_{ab} + q_{hn}^T r_{a-h,b-n}) v_{ab} \tag{8}$$

Perform the above independent self-attention calculation process on all data nodes to obtain the spatial feature of the data $T_s \in \mathbb{R}^{E \times H \times N \times C_{out_s}}$, C_{out_s} is the dimension of spatial features.

C. Temporal feature extraction module

Fig. 9. Example of a figure caption.

The traffic flow is a typical time series data with a high degree of nonlinearity and complexity. To improve prediction performance, accurate and efficient extraction of time features is important. To extract time features better, this work employs the time feature extraction module depicted in Figure 10. We begin by embedding the features of each time block of each node with time $T_{t_{emb}} = T_t W_{t_{emb}}$(where $T_t \in \mathbb{R}^{E \times N \times H}$, $W_{t_{emb}}$ is a learnable linear transformation, $T_{t_{emb}}$ is the feature data after time embedding, and $T_{t_{emb}} \in \mathbb{R}^{E \times N \times H \times D_{emb}}$, D_{emb} is the embedding Dimension), as illustrated in Figure 11.

Fig. 10. Example of a figure caption.

We encode the time position of each time point in the time block to better express the sequence of the time series. The coding dimension is the same as the time embedding dimension, making it easier to incorporate time sequence information into the time embedding information. To encode time position information, this work used sine and cosine functions of varying frequencies:

$$PE(pos, 2i) = sin(pos/10000^{2i/D_{emb}}) \tag{9}$$

$$PE(pos, 2i+1) = cos(pos/10000^{(2i+1)/D_{emb}}) \tag{10}$$

Pos is the time point's position in the time block, D_{emb} is the time embedding dimension, and i is a specific position component in D_{emb}, implying that each dimension of the time position encoding corresponds to a sinusoidal signal. Finally, to obtain the time position code, connect the sine and cosine coding components:

$$PE = PE(pos, 2i) \oplus PE(pos, 2i+1) \tag{11}$$

Incorporating location information into time code:

$$T_t = T_{t_{emb}} + PE \tag{12}$$

First, we perform self-attention on the fused feature data T_t and determine the correlation between each time point in the time window. We use scaled dot product attention mechanism to obtain different time points in temporal feature extraction, just like we do in spatial self-attention. $T_{t_{hm}}^i \in \mathbb{R}^{H \times D_{emb}}$ represent the node's feature data in the m-th time window and Figure 4-12 depicts the self-attention calculation process performed on $T_{t_{hm}}^i$.

In order to obtain the association relationship between various time points, in the time feature extraction, we introduce a multi-head attention mechanism:

$$MultiHead(Q,K,V) = Concat(h_1, h_2, ..., h_n) W^o \tag{13}$$

Here each head can be calculated as equation $head_i = Attention(QW_i^O, KW_i^K, VW_i^V))$.

Then performing batch normalization to the output result Z and $T_{t_h^m}^i$ to avoid problems such as the disappearance of gradients, and connect Z and $T_{t_h^m}^i$ as the output of this layer:

$$Z_{a\&n} = Z + T_{t_{hm}}^i \tag{14}$$

Finally, temporal feature extraction module send the output result of the normalization at each time point to a feedforward network to get $Z_{a\&n\&f}^i$, connect the $Z_{a\&n}^i$ and $Z_{a\&n\&f}^i$, and then normalize the connected result again. After network do such operation twice, module output time features $T_t \in \mathbb{R}^{E \times H \times N \times D_{emb}}$.

D. Spatio-temporal feature gated fusion

Fig. 11. Example of a figure caption.

To get the prediction results, we combine the temporal feature and the spatial feature using the gated fusion method illustrated in Figure 12. Among them the temporal feature is $T_t \in \mathbb{R}^{E\times H\times N\times D_{emb}}$ and the spatial feature is $T_s \in \mathbb{R}^{E\times H\times N\times C_{out_s}}$, $D_{emb} = C_{out_s}$. $T_t \in \mathbb{R}^{E\times H\times N\times D_{emb}}$ and $T_s \in \mathbb{R}^{E\times H\times N\times C_{out_s}}$ are currently combined as follows:

$$T_s' = T_s + (1 - \sigma(T_s W_s)) \tag{15}$$

$$T_t' = T_t + (1 - \sigma(T_t W_t)) \tag{16}$$

$$Z = \sigma(T_s' W_s' + T_t' W_t' + b_z) \tag{17}$$

$$T_{S_t}{}' = Z T_S' + (1 - z) T_t' \tag{18}$$

where, the σ is a nonlinear activation function; W_s, W_t ,W_s', W_t' and b_z are all learnable parameters. After spatio-temporal feature gated fusion, the output of the final model, which is the prediction result $T_{S_t} \in \mathbb{R}^{E\times H\times N}$, obtained. The prediction result T_{S_t} of the feature fusion module is compared to the original labeled data set, and the loss value is computed. Finally, the loss value is optimized to modify the relevant parameters of the model.

VI. Experiments

A. Datasets

The data set PeMS obtained by the California Transportation Performance Measurement System is used in this paper. The traffic flow, the average lane occupancy rate, and the average vehicle speed are among the major features of traffic flow data contained in this data collection. It also includes information on the car detector itself, such as the detector number, location information, and the timestamp. The following are the pertinent details of the data set:

PeMSD7-50 It has 39,000 traffic detectors in the Los Angeles region. During the working day, data gathered by car detectors on 50 key traffic routes in a specific spatial region was picked from May 1, 2012 to June 30, 2012. Each detector has a 5-minute sample period. Table 1 displays information about the device itself.

TABLE I
PeMSD7 detector information

detector number	latitude	longitude
715944	34.01368	-118.166
715947	34.01533	-118.171
716072	34.05411	-118.203
716076	34.05557	-118.193
716078	34.05529	-118.186
716206	34.02907	-118.211
716875	34.02279	-118.173
716891	34.01496	-118.173
716929	34.00720	-118.156
716934	34.01988	-118.182
...	...	...

This work constructs a real road network topology graph based on the latitude and longitude information of each node in the road network and the actual location of each node in the road network, and selects three traffic characteristics: traffic volume, average lane occupancy rate and vehicle speed, and data time interval for 5 minutes. Therefore, each node contains 288 time data points per day. The missing values in the original data are filled in by linear interpolation. In addition, Z-score normalization is performed on three different traffic attribute data respectively.

TABLE II
Experiment result

model	PeMSD5-50		
	MAE	MAPE(%)	RESM
STGCN	2.37	5.45	4.3
GAGCN	2.30	5.35	4.32
NWSTAN	2.32	5.29	4.21

B. Experimental parameters

The neighborhood node-set of each node is obtained using the Node-Walk method, with the search node's walk step length set to 9 and the number of searches set to 3. The total number of selected nodes is 50, and the historical time slice duration of all trials is 60 minutes, resulting in 12 observation points that are used to forecast vehicle speed for the following hour (the time interval is 5 minutes). The RMSprop mode is designed to optimize the mean square error during the training phase. The training batch size is 30 and NWSTAN and all comparison verification baselines are trained 50 times. We set the model's initial learning rate to 10^{-4} and the decay rate to 0.9.

C. Evaluation parameters

We use Mean Absolute Error (MAE), Mean Absolute Error Percent (MAPE) and Root Mean Square Error (RMSE) to evaluate the performance of our models. They are defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i| \tag{19}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n} \left|\frac{y_i - x_i}{y_i}\right| \tag{20}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{21}$$

where y is the actual value. x is the forecast value.

D. experimental result

We choose 50 detectors in data-set PeMSD7-100 to build our train date-set PeMSD7-50 and test our model on it. Table 2 shows the through comparison between different models. Results show our NWSTAN outperforms baseline models consistently and overwhelmingly on every dataset.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks . . .”. Instead, try “R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

[1] Qi, Luo. ”Research on intelligent transportation system technologies and applications.” 2008 Workshop on Power Electronics and Intelligent Transportation System. IEEE, 2008.

[2] Elefteriadou, Lily. An introduction to traffic flow theory. Vol. 84. New York: Springer, 2014.

[3] Lv, Yisheng, et al. ”Traffic flow prediction with big data: a deep learning approach.” IEEE Transactions on Intelligent Transportation Systems 16.2 (2014): 865-873.

[4] Zhang, Junbo, Yu Zheng, and Dekang Qi. ”Deep spatio-temporal residual networks for citywide crowd flows prediction.” Thirty-first AAAI conference on artificial intelligence. 2017.

[5] Kipf, Thomas N., and Max Welling. ”Semi-supervised classification with graph convolutional networks.” arXiv preprint arXiv:1609.02907 (2016).

[6] De Gooijer, Jan G., and Rob J. Hyndman. ”25 years of time series forecasting.” International journal of forecasting 22.3 (2006): 443-473.

[7] Cryer, Jonathan D., and Kung-Sik Chan. Time series analysis: with applications in R. Vol. 2. New York: Springer, 2008.

[8] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.

[9] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[10] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems. 2017.

[11] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. ”Deepwalk: Online learning of social representations.” Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014.

[12] Grover, Aditya, and Jure Leskovec. ”node2vec: Scalable feature learning for networks.” Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016.

[13] Wang, Hongwei, et al. ”Graphgan: Graph representation learning with generative adversarial nets.” Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[14] Goyal, Palash, and Emilio Ferrara. ”Graph embedding techniques, applications, and performance: A survey.” Knowledge-Based Systems 151 (2018): 78-94.

[15] Li, Yaguang, et al. ”Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.” arXiv preprint arXiv:1707.01926 (2017).

[16] Yu, Bing, Haoteng Yin, and Zhanxing Zhu. ”Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.” arXiv preprint arXiv:1709.04875 (2017).

[17] Wu, Zonghan, et al. ”Graph wavenet for deep spatial-temporal graph modeling.” arXiv preprint arXiv:1906.00121 (2019).

[18] Li, M., and Z. Zhu. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, May 2021, pp. 4189-96, https://ojs.aaai.org/index.php/AAAI/article/view/16542.

[19] Song, C., Y. Lin, S. Guo, and H. Wan. Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, Apr. 2020, pp. 914-21, doi:10.1609/aaai.v34i01.5438.