Future Generation Computer Systems 126 (2022) 70–81

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

STGNN-TTE: Travel time estimation via spatial–temporal graph neural network

Guangyin Jin [a], Min Wang [a,1], Jinlei Zhang [b], Hengyu Sha [a], Jincai Huang [a,*]

[a] College of System Engineering, National University of Defense Technology, Changsha, China
[b] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China

ARTICLE INFO

Article history:
Received 22 April 2021
Received in revised form 21 June 2021
Accepted 12 July 2021
Available online 18 July 2021

Keywords:
Travel time estimation
Spatial–temporal learning
Graph convolutional network

ABSTRACT

Estimating the travel time of urban trajectories is a basic but challenging task in many intelligent transportation systems, which is the foundation of route planning and traffic control. The difficulty of travel time estimation is the impact of entangled spatial and temporal dynamics on real-time traffic conditions. However, most existing works does not fully exploit structured spatial information and temporal dynamics, resulting in low accuracy travel time estimation. To address the problem,we propose a novel spatial–temporal graph neural network framework, namely STGNN-TTE, for travel time estimation. Specifically, we adopt a spatial–temporal module to capture the real-time traffic conditions and a transformer layer to estimate the links' travel time and the total routes' travel time synchronously. In the spatial–temporal module, we present a multi-scale deep spatial–temporal graph convolutional network to capture the structured spatial–temporal dynamics. Also, in order to enhance the individual representation of each link, we adopt another transformer layer to extract the individualized long-term temporal dynamics. Finally, these two parts are integrated by a gating fusion module as the real-time traffic condition representation. We evaluate our model by sufficient experiments on three real-world trajectory datasets, and the experimental results demonstrate that our model is significantly superior to several existing methods.

1. Introduction

According to the annual report of Didi Chuxing, Didi has been developed into the world's largest travel service platform whose business has expanded to the Asia-Pacific region, Latin America and Russia, with more than 550 million users and more than 10 billion passengers in 2019 [1]. With the development of urbanization, the urban transportation system and ride-hailing service platform are becoming more and more complete. Therefore, travel time estimation (TTE) has become a basic but important task in intelligent transportation systems (ITS) [2]. For individuals, TTE module in map service can help them to plan travel route reasonably. For government and service platform, TTE module can allow them to perceive the urban traffic situation in advance, and dispatch or control vehicles in real-time based on the feedback information to prevent large-scale congestion.

In early stage, some classic time series modeling models and some statistical learning models were presented in TTE [3–5]. Also, some data-driven statistical learning models have been introduced into this field, especially some ensemble regression tree models [6–9]. These ensemble statistical learning models can extract diversity features from historical data. Although these traditional methods are simple, intuitive and computationally light, they cannot integrate more complex spatial and temporal dynamics, which limits the improvement of accuracy. In recent years, deep learning models have become the fruitful approach for spatial–temporal learning and inference. For instance, Convolutional Neural Networks (CNNs) [10] and their variants are adopted to extract grid spatial features while Recurrent Neural Networks (RNNs) [11] and their variants are introduced into capturing temporal dynamics. Some previous works attempted to combine these two different scale models for spatial–temporal representation learning, which can be successfully extended to TTE [12–17]. Also, with the development of graph deep learning, Graph Neural Networks (GNNs) have become the most effective methods for representation learning in spatial topology structure. The transportation network is a natural graph structure, which creates the basic conditions for the promotion of the spatial–temporal graph neural network architectures in TTE [18,19]. Although some deep learning models have achieved some breakthroughs, there are still some limitations to be improved. First, most of previous deep learning models in TTE are traditional

* Correspondence to: National University of Defense Technology, Deya Road 109, 410073, Changsha, China.
E-mail address: huangjincai@nudt.edu.cn (J. Huang).

[1] The authors contributed equally to this work.

https://doi.org/10.1016/j.future.2021.07.012
0167-739X/

single deep models or hybrid CNN-RNN based models. These models ignore the topology structure of traffic network, which leads to the inability to learn fine-grained spatial–temporal representation. Second, most of previous works are difficult to capture multi-scale spatial–temporal dynamics in traffic networks. Many deep learning frameworks capture the high-level spatial–temporal dynamics by stacking multiple layers, but ignore the impact of shallow-level information at different scales on high-level information [20–23].

To address these problems, we propose a novel end-to-end GCN-based learning framework named STGNN-TTE for travel time estimation. In order to implement fine-grained and multi-scale spatial–temporal dynamics representation learning, we adopt the multi-scale spatial–temporal graph convolutional network (ST-GCN) cell in spatial–temporal learning module. In each STGNN cell, we involve the adaptive dynamics graph to jointly learn spatial representations with predefined graph. Also, for deeper ST-GCN cells, they can receive not only the original inputs but also the information from shallower STGNN cells, to reserve different level spatial–temporal coupling information for multi-scale learning. Meanwhile, to enhance the individual information for each road link, we involve the Transformer layer to capture the long-range temporal dependencies individually and then we fuse the individual information with the structural aggregation information from multi-scale ST-GCN. Finally, we also apply the Transformer model for multi-task learning. The time of the total routes and the single links can be estimated synchronously. Compared with the traditional RNN models, the parallel self-attention model has the advantages of low cumulative error and efficient training in capturing long-term time dependencies. In summary, our contributions are summarized as follow:

- We propose a novel graph-based deep learning framework to estimate the travel time in urban area. To the best of our knowledge, it is the first exploration to involve the multi-scale spatial–temporal convolutional network in TTE.
- In ST-GCN cell, we not only consider the spatial correlations in fixed topology but also consider the dynamic spatial correlations in real world.
- In our model, we replace the RNN-based models with Transformer and Gated CNN model. This improve the efficiency of model training and inference.
- We evaluate STGNN-TTE by several experiments on three real-world datasets. The experimental results reveal that our proposed framework is superior to other baseline models.

In the remainder of this paper, we begin with the related work in Section 2. Then we give some definition in Section 3. The dataset processing and methodologies proposed in this research are presented in Section 4, followed by the experimental results and analysis in Section 5. The final section concludes the achievements of this research and proposes future direction.

2. Related work

2.1. Travel time estimation

There is a long research line in travel time estimation. At the very beginning, many classic time series mathematical model were employed in this field. ARIMA is a classical but fruitful mathematical model for time series prediction, which can be also introduced into travel time estimation. Angshuman [4] first proposed a ARIMA-based to predict future travel time using historical travel time data. Similarly, ARIMA-based travel time estimation model was deployed in Minnesota State Highway in [3]. Also, some improved versions of ARIMA were adopted in the same task. For example, ARIMA be combined with context of spatiotemporal incidents [24] and spatial features of the road network [25]. Although ARIMA-based model is simple and practical for most time series modeling tasks, it still suffers from the stationary limitation of the data itself. In other words, the mathematical model is difficult to process non-linear data. In order to estimate travel time from some non-linear data, some traditional machine learning methods were presented. Gradient boosting regression tree is a scalable method for this task, which can automatically process the feature from original data and optimize the objective function by steepest descent method [8,26–28]. Many previous work also integrated multiple regression trees to capture the non-linear and diversity features, which has been applied in travel time estimation problem [6,7,9]. However, these methods rely on experienced data feature engineering, and it is difficult to learn effective features from complex multi-source heterogeneous data. With the practical application of artificial neural network methods on big data, their powerful feature extraction and learning capabilities can also be transferred to travel time estimation task. In [29], LSTM, the variant of RNN model has been applied in GPS data to predict travel time. In [30], a CNN-based multi-task representation learning model was proposed for citywide travel time forecasting. To capture spatial and temporal correlations simultaneously, some CNN-RNN hybrid models were presented. Yibin Shen et al. [16] employed a CNN-LSTM for travel time prediction with sparse trajectory data. Dong Wang et al. [12] presented an end-to-end hybrid deep model to fuse multiple features to improve the accuracy of estimation. In the work [13] and [14], the urban regions were organized into grid maps and the CNN-RNN hybrid model can efficiently capture spatial–temporal representation. In order to make the estimation more refined and have a higher degree of matching with the road network, the deep learning models based on the graph neural network began to be gradually developed. In [18], the graph-based deep framework was first adopted in travel time estimation. In [19], graph attention network and RNN model are integrated to capture the dynamics in road network. Different from these previous works, we propose a spatial–temporal representation learning module that considers multi-scale static and dynamic structures synchronously. Meanwhile, we also adopts the efficient self-attention model Transformer to perform multi-task learning on the travel time estimation of the long-range trajectories.

2.2. Deep learning for traffic prediction

Travel time estimation is actually the sub-research topic of traffic prediction. In recent years, deep learning models have been widely used in another sub-research topic of traffic prediction: traffic flow prediction. In the beginning, most of previous works were based on the grid maps for exploration. Yisheng et al. [31] first adopted deep auto-encoder architecture in traffic flow prediction. Junbo Zhang et al. [32] proposed a hybrid CNN-based deep learning model for crowd flow prediction that considers multiple temporal modes. To integrate spatial–temporal and some semantic information, Huaxiu Yao et al. [33] proposed a multi-view deep fusion model for taxi demand prediction. Based on these models, more and more improved variants were presented. For instance, many works involved attention mechanism [34–38], adversarial loss [39–41] and some external information [36,42–46] to enhance the capability of spatial–temporal representation learning. In recent years, with the development of graph convolutional networks (GCN), deep learning methods based on spatial–temporal graphs began to be involved into traffic prediction. This structure makes up for the limitations of the grid map modeling method in non-Euclidean spaces. Bing Yu et al. [21] proposed ST-GCN for traffic flow prediction, the first deep learning framework based on GCN and temporal CNN.

Yaguang et al. proposed DCRNN [47], a novel deep model that integrates RNN and GCN. Based on these two classic spatial–temporal graph deep learning framework, many stronger and more efficient models were proposed. For instance, some models were enhanced by multiple attention or self-attention [48–50], some models were improved by adaptive graph learning [20,51–54] and some models integrate multi-graph information [55–59]. Although most of previous works did not adopt the deep learning methods directly in travel time estimation, we can still explore the potential of powerful deep models for spatial–temporal representation learning to capture complex patterns in traffic systems. In other words, the deep learning model employed in traffic flow/demand prediction can also be used in travel time estimation to some extent.

2.3. Graph convolution network

Graph convolution network (GCN) is the promotion of normal convolution network, which can handle the spatial representation learning in non-Euclidean structure. The first generation graph convolution was proposed in [60], which was defined in spectral domain. Based on this work, Defferrard et al. [61] improve the design approach of convolution kernel to reduce the complexity of the original graph convolution. Kipf et al. [62] further simplified the computation process of GCN and improves the efficiency of the model. To make graph convolution more convenient to operate on directed graphs and dynamic graphs, it is necessary to get rid of the dependence on the Laplacian matrices. Many spatial-based GCN were proposed to deal with this challenge. Petar et al. [63] proposed Graph Attention Network (GAT) to involve attention mechanism into graph representation learning. GraphSAGE model [64] was proposed to learn large graph by sampling strategy. However, most GCN models have encountered over-smooth problem, which makes it difficult to deepen the graph networks [65]. To alleviate this problem, Luan et al. proposed a deeper and stronger GCN model, named Multi-scale GCN [66].

3. Preliminaries

We describe in detail some important preliminaries required for this research and the problems to be solved.

Trajectory: In raw data, a trajectory Y is a sequence of consecutive GPS points $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$. Each point contains two elements, the location (latitude ($y_i^{l_1}$) and longitude ($y_i^{l_2}$)) and the timestamp (y_i^t) respectively.

Road Network: Road network can be naturally defined as a directed graph $G = (V, E)$, where V and E represent vertices and edges in the graph respectively. Node $v \in V$ denotes a road segment in the road network. Edge $e_{i,j} \in E$ denotes the connection between two different road segments v_i and v_j. If i to j are passable, then $e_{i,j}$ is equal to 1, otherwise 0. For convenience, a road segment is also referred to as a link in the following paper.

Route: A route R is a sequence of links in a road network. One route is defined as: $R = \{v_o, v_1, \ldots, v_d\}$, where v_o and v_d denote the origin and destination respectively. The route data is collected from the trajectory data, which has been operated by the map matching algorithm [67].

Objective: Given a route R and departure time t_d, the aim in this research is to estimate the travel time t_ω by historical trajectory data D as well as the predefined road network G, which can be expressed as:

$$f(R, t_d|D, G) \rightarrow t_\omega, \tag{1}$$

where $\omega = \{R, v_R\}$ represents the set of the total routes and the corresponding links. $f(\cdot)$ is the non-linear function we need to design for travel time estimation.

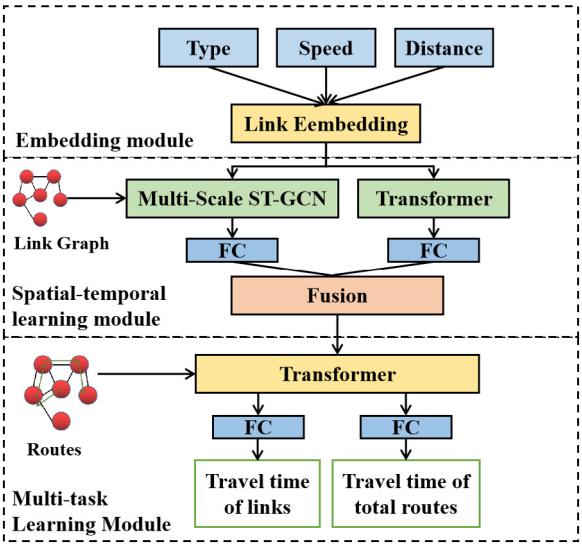

Fig. 1. The overview of our proposed deep learning framework for travel time estimation.

4. Methodologies

The overview of our proposed framework is displayed in Fig. 1. The framework consists of three main components, embedding module, spatial–temporal learning module and multi-task learning module respectively. The embedding module aims to map the static and dynamic properties of each link into the one-dimensional tensor to put into the neural networks. The spatial–temporal learning module needs the link graph and link embedding as the inputs for link representation learning. The outputs of the spatial–temporal learning module are the link embedding in next time slot. The multi-task learning module receive the inferred link embedding and cascade them based on the given routes. The outputs of this module is composed of two parts, travel time of links and travel time of total routes respectively. In the following subsections, we describe these main modules in detail.

4.1. Embedding module

Since the travel time of a route is impacted by many significant attributes such as traffic speed, the link type and the link distance, we obtain the embedding of links by merging these attributes into embedding module, as show in Fig. 2. To be specific, the embedding of links at time t can be operated by a fully connected layer, which is formulated as follows:

$$\boldsymbol{h}_e(t) = \tanh(\boldsymbol{W}_e \cdot [S_t \parallel D \parallel P \parallel O] + \boldsymbol{b}_e), \tag{2}$$

where W_e and b_e are the weight and bias of the fully-connected layer respectively, S_t, D, P, O denote the traffic speed at time t, the link type and the link distance respectively, the notation $\parallel$ means concentrate operation.

4.2. Spatial-temporal learning module

This module is the crucial module in our framework. It employs the link embedding at previous time slots to infer the next time slot's link embedding. The inferred embedding will be utilized in downstream tasks — travel time estimation. In other words, the capability of the spatial–temporal learning module can determine the accuracy of travel time estimation. The overview of the module is shown as Fig. 3.

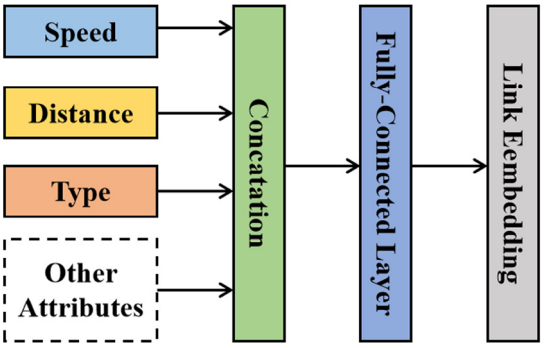

Fig. 2. The overview of the embedding module.

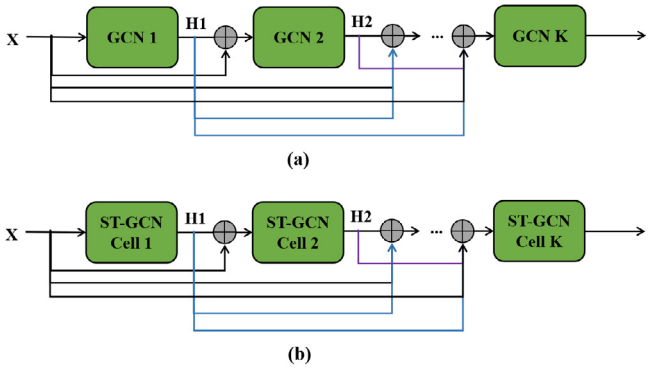

Fig. 4. The overview of the Multi-scale structure. Sub-figure(a) is Multi-scale GCN while Sub-figure(b) is Multi-scale ST-GCN.

From Fig. 3, the spatial–temporal learning module is composed of three main components, ST-GCN cell, Transformer and Gated fusion layer.

4.2.1. ST-GCN cell

ST-GCN cell is the basic unit to capture spatial–temporal dynamics in the module. As shown in Fig. 3, the ST-GCN cell contains two graph convolution models (GCN-a and GCN-b), an adjacency matrix generator and a temporal convolution model.

GCN-a is the model that capture the spatial correlations from the predefined link graph, which is formulated as follows:

$$h_{GCN-a} = ReLU(\hat{D}^{-\frac{1}{2}}\hat{A}_{link}\hat{D}^{-\frac{1}{2}} \cdot X \cdot W_a) \tag{3}$$

where A_{link} is the adjacency matrix of road network, X is the input feature matrix of GCN-a, W_a is the weight parameter of GCN-a. $\hat{D}^{-\frac{1}{2}}\hat{A}_{link}\hat{D}$ denotes the normalized adjacency matrix.

GCN-b is the model that capture the spatial correlations from the dynamically uncertain adjacency matrix. The computation process is similar to that of GCN-a model, which is defined as:

$$h_{GCN-b} = ReLU(A_{adp} \cdot X \cdot W_b) \tag{4}$$

A_{adp} is the adaptive adjacency matrix from adjacency matrix generator. The adjacency matrix generator is formulated as follows:

$$A_{adp} = ReLU(SoftMax(E_1 \cdot E_2^T)) \tag{5}$$

where $E_1 \in R^{N*d}$ and $E_2 \in R^{N*d}$ are randomly initialized node embedding.

Then the output of GCN-a and GCN-b are accumulated to obtain the final graph embedding:

$$h_g = h_{GCN-a} + h_{GCN-b} \tag{6}$$

GCN model can capture the spatial dependencies while the temporal dependencies can be captured by temporal convolution model in ST-GCN cell. The framework of the temporal convolution model is similar to [21], which is widely used in efficient sequence modeling.

$$h_{st-gcn} = tanh(h_g \star \Gamma_1(\theta_1) + b_1) \odot Sigmoid(h_g \star \Gamma_2(\theta_2) + b_2) \tag{7}$$

where θ_1, $\theta_2 b_1$ and b_2 are the model parameters, $\odot$ is element-wise product operation, Γ is the convolution filter. Note that, the temporal convolution operation is for each individual node and the parameters are shared by each node.

4.2.2. Multi-scale ST-GCN

To overcome the problem of over-smooth in graph convolution, Multi-scale GCN model was proposed [66]. Multi-scale GCN makes the GCN deeper and efficiently capture spatial multi-scale information by stacking shallow hidden information into the deeper layer. Motivated by this idea, we can design a novel Multi-scale ST-GCN model, integrating the multi-scale spatial–temporal in each layer. As shown in Fig. 4, the architecture of Multi-scale GCN and Multi-scale ST-GCN are displayed. The external connection structure of the two models is similar, but ST-GCN cell integrates not only spatial information but also temporal information.

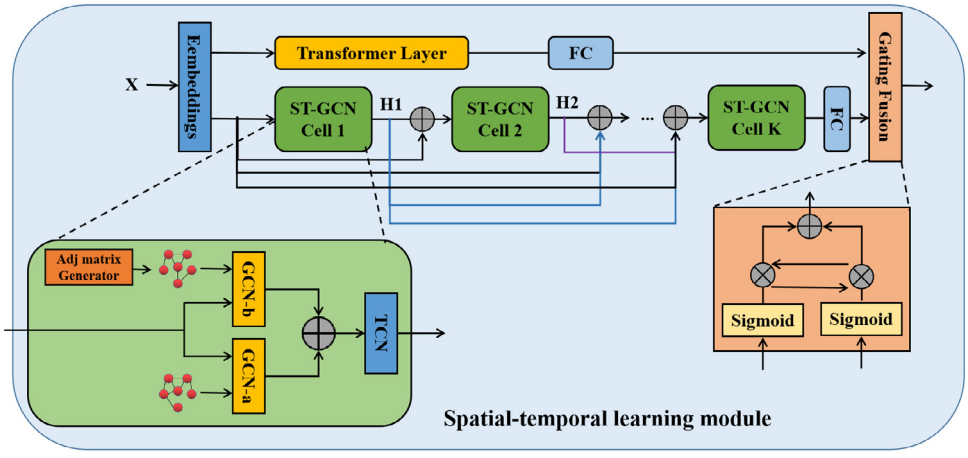

Fig. 3. The overview of the spatial–temporal learning module.

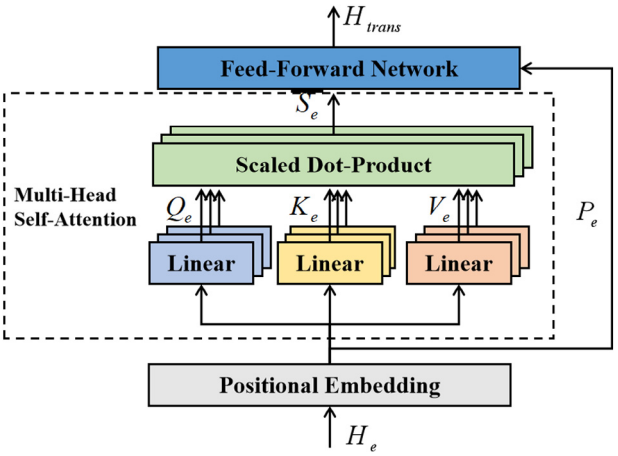

Fig. 5. The overview of Transformer layer.

The computation of this architecture can be expressed as:

$$H_{st-gcn}^{(l)} = f(W_m \cdot [X, H_{st-gcn}^{(1)}, H_{st-gcn}^{(2)}, \ldots, H_{st-gcn}^{(l-1)}]) \tag{8}$$

where f denotes the mapping operation by ST-GCN cell, l denotes the l_{th} ST-GCN layer, $[\cdot]$ denotes the concentrate operation. W_m aims to map the concentrate tensors into the fixed size embedding.

4.2.3. Transformer layer

Since Multi-scale ST-GCN model computes each link embedding by aggregating K-hop neighbor node, the features of the node itself also need to be emphasized and strengthened. Hence, we adopt the powerful long-range sequence modeling tool, named Transformer, to capture the dynamics of each independent node. The overview of Transformer layer is shown as Fig. 5. Transformer layer is composed of three main components, Positional Embedding, Multi-head self-attention and Feed-Forward Network respectively.

To enhance the positional information in long-range sequences, position embedding mechanism is involved in Transformer layer. The common position embedding method is to conduct sine encoding on even positions as well as cosine encoding on odd positions. The equation of position embedding is defined as:

$$pe_t = \begin{cases} sin(t/(10000^{2i/d})) & \text{if } t = 0, 2, 4 \cdots \\ cos(t/(10000^{2i/d})) & \text{if } t = 1, 3, 5 \cdots \end{cases} \tag{9}$$

where d denotes embedding size of link, i denotes the i_{th} time step. In the case of involving position embedding, the inputs of self-attention network need to add the positional encoding variable, which is formulated as:

$$\hat{X}_t[i, :] = X_t[i, :] + pe_t \tag{10}$$

where $X_t \in R^{T*d}$ denotes the individual node embedding.

For single head self-attention layer, Scaled Dot-Product operation is conducted. There are commonly three types of vectors, queries, keys and values for all the nodes, corresponding to the notation $Q, K, V \in R^{T*d}$, which is defined as:

$$SA(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d}}) \cdot V \tag{11}$$

where d is a normalization factor and its value is consistent with the feature dimension of Q. The total input of self-attention layer is $\hat{X} \in R^{N*T*d}$ from embedding layer but considering that self-attention layer is for individual node, we should divide $\hat{X}$ into $N \cdot \hat{x} \in R^{T*d}$ as input for each node. The three types of vectors are obtained by three different projection matrices, which is defined as:

$$Q_e = \hat{x} \cdot W^q, K_e = \hat{x} \cdot W^k, V_e = \hat{x} \cdot W^v \tag{12}$$

where W^q, W^k, W^v are there types of learnable weight parameters for Q, K and V. To enhance the representation learning capability of the model, we can also adopt multi-head self-attention (MHSA), which is defined as:

$$MHSA(Q_e, K_e, V_e) = \|_{i=1}^{h} (softmax((\frac{Q_e \cdot K_e^T}{\sqrt{d}}) \cdot V_e)) \tag{13}$$

where h denotes the number of attention heads and $\|$ denotes the concentrate operation. Then the MHSA output of each node is integrated, we can obtain $S_e \in R^{N*T*d}$ Following the multi-head self-attention layer is a two-layer feed-forward neural network, which is defined as:

$$H_{trans} = ReLU(W_{f2} \cdot (ReLU(BN(W_{f1} \cdot S_e + b_{f1})) + b_{f2})) \tag{14}$$

where H_{trans} is the final output of Transformer layer, BN denotes batch normalization operation.

4.2.4. Gated fusion layer

The size of H_{trans} and H_{st-gcn} are both [N, T, d], where N denotes the number of nodes, T denotes the sequence length and d denotes the dimension of the outputs. Since we only need to obtain the next time slot's link representation, we employ two fully connected layers to map the sequence length dimension from T to 1, which is formulated as:

$$H_{f1} = W_{T1} \cdot H_{trans} + b_{T1} \tag{15}$$

$$H_{f2} = W_{T2} \cdot H_{st-gcn} + b_{T2} \tag{16}$$

where $W_{T1} \in R^{T*1}$ and $W_{T2} \in R^{T*1}$ are weight parameters of these two fully connected layers, $b_{T1} \in R^1$ and $b_{T2} \in R^1$ are the basis.

Then, we need to fuse the two part of information. In this case, we adopt a dual gated fusion mechanism to control the retained information from these two channels, as shown in Fig. 3. The equation is defined as:

$$H_g = H_{f1} \cdot Sigmoid(W_{s2} \cdot H_{f2} + b_{s2}) + H_{f2} \cdot Sigmoid(W_{s1} \cdot H_{f1} + b_{s1}) \tag{17}$$

where $W_{s2} \in R^{d*d}$ and $W_{s1} \in R^{d*d}$ are weight parameters, $b_{s2} \in R^d$ and $b_{s1} \in R^d$ are basis, $H_g in R^{N*d}$ is the final representation from spatial–temporal learning module.

4.3. Multi-task learning module

In order to estimate the routes' travel time and the links' travel time synchronously by the representation from spatial–temporal learning module, an appropriate multi-task learning module need to be established.

In this case, we adopt another Transformer model as the crucial part of the multi-task learning module. The overview of the multi-task learning module is shown in Fig. 6. The input of the module contains two parts: given travel routes and the corresponding inferred link representation from spatial–temporal module. In the first step, the corresponding link representation need to be selected by the link sequences. Assuming that the representation dimension of each link is d and the number of links passed by one route is L, then the representations of all links can be merged together to obtain a tensor $H_r \in R^{L*d}$ as the input of Transformer model.

The computation process of Transformer model in multi-task learning module is similar with that in spatial–temporal learning module. We can obtain the output tensor $H_m \in R^{L*d}$ from Transformer model. Then we use two fully connected layers to estimate the links' travel time and the routes' travel time respectively.

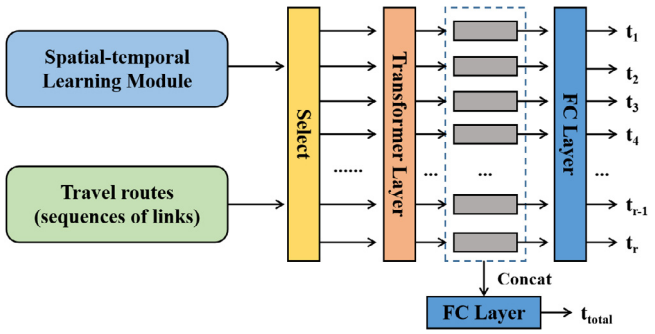

Fig. 6. The overview of the multi-task learning module.

For estimating the links' travel time, the equation is defined as:

$$t = W_{l2} \cdot (Relu(W_{l1} \cdot H_m + b_{l1})) + b_{l2} \tag{18}$$

For estimating the routes' travel time, we need to sum in the sequence length dimension L first. And then we pass the tensor into the fully connected layer, which is defined as:

$$t_{total} = W_{r2} \cdot (Relu(W_{r1} \cdot \sum_{L}(H_m) + b_{r1})) + b_{r2} \tag{19}$$

4.4. Loss function of STGNN-TTE

During the training process, mean absolute percentage error (MAPE) is selected as our loss functions for travel time estimation of the links and entire routes. The loss function is defined as:

$$L_r = \sum_{i=1}^{|D|} \frac{|T_i - \hat{T}_i|}{\hat{T}_i + \epsilon} \tag{20}$$

where $\hat{T}_i$ denotes the estimated entire routes' travel time, and T_i is the ground truth for ith route's travel time. Similar with [12], the parameter ϵ is involved to avoid exploded loss value in case that the denominator approaches 0. In this paper, ϵ is set as 5.

For the local links' travel time estimation, we define the corresponding loss as the average loss of all local paths,

$$L_l = \sum_{r \in D} \sum_{i=1}^{|r|} \frac{|t_{l_i} - \hat{t}_{l_i}|}{\hat{t}_{l_i} + \epsilon} \tag{21}$$

where $\hat{t}_{l_i}$ denotes the estimated travel time for ith link.

The training process of STGNN-TTE is to minimize the weighted combination of these two part loss terms, which is defined as:

$$L = \alpha L_r + (1 - \alpha) L_l \tag{22}$$

where α is used to balance L_r and L_l.

5. Experiment

In this section, we conduct sufficient experiments to evaluate our proposed model in different real-world datasets.

5.1. Experimental settings

We evaluate our proposed model on three real-world datasets, the three datasets are collected from two original public datasets, Chengdu dataset and Porto dataset respectively. Chengdu dataset contains taxi trajectories of over 14 thousand taxis in August 2014 in Chengdu, China.[2] while Porto dataset contains taxi trajectories of 442 taxis from July 2013 to June 2014 in Porto, Portugal[3] For these two raw datasets, we first obtain the topology information of road networks in Chengdu and Porto from OpenStreetMap.[4] Then we utilize the existing map matching approach [67] to project each trajectory into the road network, getting the corresponding routes. After that, we delete some significant mismatched routes (deviation≥ 10%) from their raw trajectories [14]. Finally, we get 310,507 and 1,788,723 trajectories from the raw datasets of Porto and Chengdu respectively. Due to the limitation of computation resources, we have difficulties to use all the trajectories in raw datasets. Hence, we filter trajectories according to a certain spatial range. Based on this approach, we collect one small dataset from Porto and two small dataset from Chengdu. Fig. 7 shows the spatial distribution of the trajectories on the map and Table 1 shows the statistics of the three generated small datasets and some basic attributes (average degree, density and clustering coefficient) of three road networks in subsequent experiments. In addition, the speed of each link at time interval t can be calculated from the average of the speeds of all trajectories passing through each link at t. However, this manner could cause speed data missing on some links with sparse trajectories. Some previous work provide many novel methods to deal with traffic sparsity like graph neural networks [68–71], but we adopt a simple time–space averaging approach in this case. If there are missing speed values on one specific link at t, we average the speed of the neighboring links from the last 12 time steps as the estimated speed at t.

Evaluation Metric

Same as some previous works, we choose three common metrics to evaluate all our proposed models and baseline models, including Mean Average Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). In detail, MAE and RMSE can well reflect the absolute errors between the predicted values and the real values, where RMSE is more sensitive for some large errors. MAPE is a percentage error to measure the estimation accuracy. For these three metrics, the lower RMSE, MAE and MAPE, the better the model perform. The definition of these three metrics as follows:

$$RMSE = \left(\frac{1}{n} \Sigma_{i=1}^{n} \left\| t_i - \hat{t}_i \right\|_2^2 \right)^{\frac{1}{2}} \tag{23}$$

$$MAE = \frac{1}{n} \Sigma_{i=1}^{n} \left| t_i - \hat{t}_i \right| \tag{24}$$

$$MAPE = \frac{1}{n} \Sigma_{i=1}^{n} \frac{\left| t_i - \hat{t}_i \right|}{|t_i|} \tag{25}$$

For our model, the embedding size of each link is set as 32. The number of ST-GCN Cell is set as 3, and kernel size of temporal convolution is set as 3. The number of attention head in two Transformer are both set as 4. We adopt the links' historical information at last 12 time steps with intervals of 5 min to capture their temporal features. The number of all the fully connected layers in this model are set as 2, and their hidden sizes are all set as 64. The Adam optimizer is applied in the training process with an initial learning rate of 0.001 in each dataset. We implement our model by Pytorch 1.4.0 and run all the deep learning models on NVIDIA GTX 2080 GPU.

[2] 2016. Taxi Travel Time Prediction Challenge. http://www.dcjingsai.com

[3] 2020. Kaggle. https://www.kaggle.com/crailtap/taxi-trajectory.

[4] Steve Coast. 2004. OpenStreetMap. https://www.openstreetmap.org/.

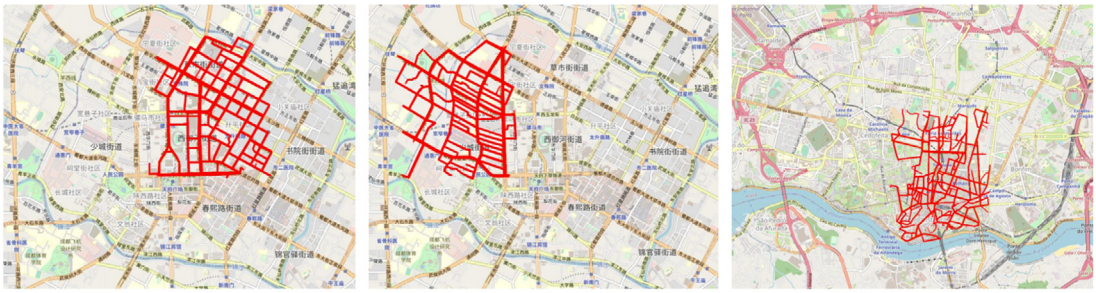

Fig. 7. The spatial distribution of the trajectories on three maps. Sub-figure(a)(b)(c) shows the spatial distribution of trajectories in Chengdu 1 dataset, Chengdu 2 dataset and Porto dataset respectively.

Table 1
The statistics and graph attributes of three datasets.

Dataset	Chengdu1	Chengdu2	Porto
Latitude range	[30.66, 30.68]	[30.66, 30.68]	[41.14, 41.16]
Longitude range	[104.06, 104.08]	[104.04, 104.06]	[−8.62, −8.60]
Number of trajectories	13442	14934	16858
Number of links	532	413	736
Average travel time (s)	246.54	242.12	247.16
Average moving distance (m)	1417.23	1289.67	1387.75
Average Degree	2.5582	2.2807	1.8329
Density	0.0048	0.0053	0.0025
Clustering coefficient	0.0125	0.0103	0.0017

5.2. Methods for comparison

To verify the performance of our model, nine baselines are presented to compare with STGNN-TTE. We divide the baselines into three categories. The first category is non-deep learning models, including AVG, TEMP [72] and LightGBM [73]. The second category is single deep learning models, including Mlp-TTE and Rnn-TTE. The third category is hybrid deep learning models, including DeepTTE [12], STGCN [21], DCRNN [47] and Graph WaveNet [20].

(1) **AVG:** This is a simple mathematical modeling approach. In this model, we need the historical speed in each link and calculate their average value. Then divide the distance of the links by the average speed to get the travel time.
(2) **TEMP:** This is an empirical modeling method that applies the neighboring historical trajectories with similar origin and destination to calculate the average travel time. We set the number of neighboring trajectories as 10 for each estimation.
(3) **LightGBM:** We use the Light Gradient Boosting Machine for regression to estimate the travel time. LightGBM is the state-of-art model in GBDT family, which has faster training speed, lower memory consumption, and better accuracy. Similar with [21], the GPS points and links' distance are the crucial features in regression. The max depth of this tree model is set as 10.
(4) **Mlp-TTE:** Multiple-layer perceptron is the most simple fully-connected neural network model. In our experiments, the number of layers is set as 4, and the activation function is ReLU. The hidden size is fixed as 64. The input of Mlp-TTE is the sequence of GPS points and link distances.
(5) **Rnn-TTE:** In this model, GRU is utilized as an implementation of the recurrent neural networks to map the raw GPS and the link distance sequence into a 64-dimensional feature vector. Then, it is put forward into a two-layer fully-connected network to obtain the estimated time.
(6) **DeepTTE:** This is a hybrid neural network model for travel time estimation, which captures the spatial and temporal dependencies based on the Geo-Conv layer and GRU layer. The hidden size of Geo-Conv layer and GRU layer are both set as 64. The input of the model is also the GPS sequence and link distances.
(7) **STGCN:** This is spatial–temporal graph deep learning model that combines the GCN and Gated CNN to simultaneously extract the spatial and temporal dependencies for traffic prediction. In our experiments, we use its core architecture in the spatial–temporal learning module for representation learning. The embedding size of GCN model and the dimension of Gated CNN are both set as 32, the kernel size of Gated CNN is set as 3.
(8) **DCRNN:** This is a spatial–temporal graph deep learning model for traffic flow prediction. It exploits GCN to capture the spatial dependencies, and then applies GRU to model the temporal dependencies. In our experiments, we leverage its core architecture in the spatial–temporal learning module for representation learning. The embedding size of GCN model is set as 32
(9) **Graph WaveNet:** This is also the spatial–temporal graph deep learning model that combines the GCN and Gated CNN. But in this model, adaptive graph modeling mechanism and stack skip connection are involved. Similarity, the core architecture of Graph WaveNet is utilized in the spatial–temporal learning module for representation learning. The setting of GCN model and Gated CNN model are the same as STGCN model in our experiments.

5.3. Experimental results

5.3.1. Overall performance

In Table 2, the experimental results on AVG, TEMP, LightGBM, Mlp-TTE, Rnn-TTE, DeepTTE, STGCN, DCRNN and Graph WaveNet are shown. We conducted five independent experiments and the best results of each metric are in boldface.

From Table 2, we can find that the non-deep learning methods AVG, TEMP and LightGBM are much weaker than deep learning methods in each metric. The reason may be that non-deep learning methods have limited capabilities to capture the complex and

Table 2
Performance of STGNN-TTE and other methods for estimating the travel time of the whole trajectories on three datasets, where the performance improvements of our model are compared with the best of these baseline methods, marked by the asterisk.

Method	Chengdu1			Chengdu2			Porto		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
AVG	147.32	91.69	0.5621	192.48	102.44	0.5485	92.43	64.75	0.4132
TEMP	116.38	77.50	0.3903	155.34	81.19	0.4057	72.81	53.14	0.2678
LightGBM	107.89	71.97	0.3518	142.56	74.10	0.3316	65.02	49.63	0.2065
MlpTTE	96.17	66.58	0.2963	128.52	68.31	0.2839	58.20	44.38	0.1692
RnnTTE	95.29	65.74	0.2915	128.79	68.43	0.2841	57.76	43.16	0.1683
DeepTTE	93.68	64.03	0.2864	127.24	67.39	0.2807	56.86	42.77	0.1658
STGCN	94.37	65.02	0.2787	127.89	67.58	0.2720	57.32	42.96	0.1562
DRCNN	96.78	66.34	0.2817	130.11	69.57	0.2786	58.41	43.78	0.1613
Graph WaveNet	92.65*	63.57*	0.2725*	126.58*	67.10*	0.2701*	55.21*	41.42*	0.1543*
STGNN-TTE (ours)	**88.03**	**60.71**	**0.2606**	**121.14**	**63.38**	**0.2568**	**52.35**	**39.25**	**0.1474**
Improvements	+4.98%	+4.50%	+4.36%	+4.30%	+4.92%	+5.23%	+5.18%	+5.24%	+4.47%

Table 3
Performance of STGNN-TTE and other graph-based deep learning models for estimating the travel times of each link on three datasets.

Method	Chengdu1			Chengdu2			Porto		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
STGCN	33.87	18.79	0.5301	52.33	21.13	0.5345	7.04	4.25	0.1711
DRCNN	35.75	19.42	0.5457	54.21	21.88	0.5519	7.93	4.72	0.2027
Graph WaveNet	33.12	18.23	0.5264	51.08	20.76	0.5330	6.86	4.03	0.1682
STGNN-TTE (ours)	**31.28**	**17.57**	**0.5152**	**49.77**	**20.01**	**0.5217**	**6.21**	**3.77**	**0.1479**

dynamic spatial–temporal features for travel time estimation. We can also find that the performance of Mlp-TTE and Rnn-TTE is significantly better than that of AVG, TEMP and LightGBM, but they are significantly weaker than hybrid deep learning models. This suggests that hybrid deep learning models have stronger capabilities for spatial–temporal representation learning, therefore single deep learning models are still difficult to capture entangled spatial–temporal dynamics.

Obviously, STGNN-TTE achieves better performance than all baseline models in terms of all three metrics. Specifically, our model outperforms at least 4.98%, 4.50%, 4.36% in RMSE, MAE and MAPE on Chengdu-1 dataset, the improvement ratios of RMSE, MAE and MAPE are at least 4.30%, 4.92%, 5.23% on Chengdu-2 dataset and the improvement ratios of RMSE, MAE and MAPE are at least 5.18%, 5.24%, 4.47% of improvements on Porto dataset, which further confirms that our proposed model STGNN-TTE is an effective model for travel time estimation. Such significant improvements can be explained in three perspectives. First, we adopt multi-scale STGCN structure in spatial–temporal learning module, which considers the coupled spatial–temporal multi-scale information at each level of ST-GCN cell, thereby enhancing the learning capability of complex spatial–temporal dynamics. Second, the dynamic adaptive mechanism is also involved in each ST-GCN cell. Compared with the model that only utilizes the road network topology, the diversity of spatial representation has been supplemented to a certain extent. Third, the Transformer model in spatial–temporal learning module retains the personalized representation of individual nodes and organically integrates with neighbor aggregation representation.

Since our proposed model can simultaneously predict the travel time for links and the whole trajectories, we also conduct experiments to compare STGNN-TTE with other graph-based deep learning methods on forecasting the travel times of each link. As the results revealed in Table 3, we find that for links travel time estimation, our model still achieves better prediction performance on three datasets.

5.3.2. Performance under different temporal scenarios

We have demonstrated that STGNN-TTE is superior to other models in different spatial scenarios. To further verify the performance of our proposed model in different temporal scenarios, we respectively divide the three datasets into rush hours and non-rush hours according to the time period. Specially, we define the rush hours as 7:00–9:00 AM and 5:00–7:00 PM in weekdays, and the rest time periods are set as non-rush hours.

Table 4
Performance of STGNN-TTE and other methods for estimating the travel times of the routes under different temporal scenarios on Chengdu-1 dataset.

Scenario	Rush Hours			Non-Rush Hours		
Method	RMSE	MAE	MAPE	RMSE	MAE	MAPE
AVG	132.58	93.46	0.5731	120.41	88.93	0.5510
TEMP	110.41	79.41	0.4022	102.33	75.44	0.3896
LightGBM	101.34	74.67	0.3582	93.47	70.10	0.3476
MlpTTE	90.84	67.01	0.2991	86.95	61.36	0.2931
RnnTTE	89.49	66.16	0.2967	84.09	60.14	0.2884
DeepTTE	87.24	64.93	0.2889	83.05	59.51	0.2851
STGCN	88.27	65.35	0.2810	83.56	59.43	0.2726
DRCNN	90.57	66.72	0.2883	86.83	61.05	0.2785
Graph WaveNet	86.49	64.10	0.2789	82.19	59.36	0.2692
STGNN-TTE (ours)	**82.65**	**61.45**	**0.2662**	**77.34**	**56.55**	**0.2581**

Table 5
Performance of STGNN-TTE and other methods for estimating the travel times of the routes under different temporal scenarios on Chengdu-2 dataset.

Scenario	Rush Hours			Non-Rush Hours		
Method	RMSE	MAE	MAPE	RMSE	MAE	MAPE
AVG	175.84	105.74	0.5733	158.32	100.65	0.5387
TEMP	144.66	83.52	0.4341	136.55	80.02	0.3982
LightGBM	131.94	76.63	0.3589	125.84	72.38	0.3224
MlpTTE	122.67	69.16	0.2908	117.15	68.02	0.2806
RnnTTE	122.12	69.21	0.2911	117.47	68.18	0.2810
DeepTTE	121.56	68.63	0.2885	117.10	67.07	0.2793
STGCN	121.35	68.41	0.2798	116.27	67.19	0.2703
DRCNN	125.22	70.14	0.2823	120.17	69.08	0.2759
Graph WaveNet	120.62	68.16	0.2759	115.86	65.91	0.2684
STGNN-TTE (ours)	**114.25**	**64.45**	**0.2611**	**110.83**	**62.77**	**0.2551**

Tables 4–6 shows the experimental results. As we can observe, although traffic conditions are more unpredictable during the rush hours because of traffic congestion , our model can achieve at least 4.43%, 4.13% and 4.55% improvement in RMSE, MAE and MAPE on Chengdu 1 datasets, at least 5.28%, 5.44% and 5.36% improvement in RMSE, MAE and MAPE on Chengdu 2 datasets, at least 5.76%, 5.73% and 4.56% improvement in RMSE, MAE and

Table 6
Performance of STGNN-TTE and other methods for estimating the travel times of the paths under different temporal scenarios on Porto dataset.

Scenario	Rush Hours			Non-Rush Hours		
Method	RMSE	MAE	MAPE	RMSE	MAE	MAPE
AVG	88.69	65.95	0.4286	85.25	63.57	0.4101
TEMP	71.88	54.26	0.2719	69.01	52.97	0.2653
LightGBM	64.17	50.20	0.2174	63.24	48.69	0.2037
MlpTTE	56.86	44.45	0.1707	59.46	44.31	0.1683
RnnTTE	55.34	43.75	0.1692	53.18	43.06	0.1679
DeepTTE	54.76	43.17	0.1687	52.78	40.01	0.1649
STGCN	55.10	43.24	0.1577	53.03	42.81	0.1549
DRCNN	57.14	44.12	0.1625	55.61	43.48	0.1604
Graph WaveNet	54.32	42.56	0.1557	52.64	39.86	0.1531
STGNN-TTE (ours)	**51.19**	**40.12**	**0.1486**	**49.78**	**38.15**	**0.1463**

MAPE on Porto datasets. However, in non-rush hours, our model achieves at least 5.90%, 4.73% and 4.12% improvement in RMSE, MAE and MAPE on Chengdu 1 dataset, at least 4.32%, 4.76% and 4.95% improvement in RMSE, MAE and MAPE on Chengdu 2 dataset, at least 5.43%, 4.29% and 4.44% improvement in RMSE, MAE and MAPE on Porto dataset.

5.3.3. Ablation study

To verify the effectiveness of spatial–temporal learning components in our proposed model, we conduct experiments with STGNN-TTE using four different variants. The details of these four variants are described as below:

(1) **Ours-STGCN:** This variant only reserves Transformer model in spatial–temporal learning module. This means that the topology of road network on spatial correlation is not considered, but only the variant is expected to learn the temporal dynamics from individual links.
(2) **Ours-Transformer:** This variant reserves STGCN model and removes Transformer model in spatial–temporal learning module. We expect to explore the gains of the approach to enhance the temporal dynamics of individual links to the overall performance.
(3) **Ours-Adaptive:** This variant removes the adaptive GCN model in ST-GCN cell. We expect to explore whether adding the dynamic adaptive mechanism can improve the performance of travel time estimation.
(4) **Ours-Multi-scale:** This variant removes the multi-scale mechanism in ST-GCN. This means that the integration manner of ST-GCN cells is similarity to that in normal STGCN model. We expect to explore whether adding the multi-scale mechanism can improve the capabilities spatial–temporal representation learning.

Table 7 shows the result scores of different variants. We find that our complete model performs better than Ours-STGCN model and Ours-Transformer. This means that both aggregate feature representation and individual feature representation can improve the overall spatial–temporal representation learning. Our complete model performs better than Ours-Adaptive model suggests that only the road network topology is not enough to reflect the diversity of spatial characteristics. Also, Ours-Multi scale model is weaker than our complete model, which indicates that the importance of multi-scale spatial–temporal information in representation learning.

5.3.4. Parameter analysis

To further show the effectiveness of our proposed model, we conduct the experiments under different parameter setting, including the number of the ST-GCN cell in spatial–temporal learning module (denoted by k) and the embedding size of GCN model (denoted by d).

First, we fix d as the default value and change k in the range of [1–6]. The results are displayed in Fig. 8. When k is equal to 4, the model obtains the best value on the Chengdu2 dataset. When k is equal to 5, the model obtains the best value on Chengdu1 and Porto dataset. Obviously, we can find that the performance of our model does not necessarily get better as k increases and the optimal k is different for different dataset. The reason could be that over-fitting is caused when k increases beyond a certain threshold. The optimal k on Chengdu1 and Porto dataset is larger than that on Chengdu2. This indicate that the spatial and temporal long-term dependencies of the Chengdu1 and Porto dataset are more significant than that of Chengdu2.

Second, we fix k as the default value and change d in the range of [8,16,32,64]. The results are displayed in Fig. 9. When d is equal to 32, the model obtains the best value on all these three datasets. The changing trend of the performance of our model illustrates that the value of d also needs to be selected appropriately. Too small d could lead to insufficient learning capability of our model while too large d could lead to over-fitting.

To sum up, we can conclude that the crucial hype-parameters need to be set appropriately in our model. Otherwise, the performance of the model may be impaired by inappropriate parameters.

6. Conclusion

In this paper, we propose a novel graph-based deep learning framework named STGNN-TTE for travel time estimation. We have evaluated the model under different spatial and temporal scenarios. The experimental results of STGNN-TTE are remarkable compared with other baselines, which demonstrates the powerful capability of structured spatial–temporal data representation learning in our model. However, there are still some limitations in this work: (a) Due to the limitation of computing resource, we only select small spatial range to evaluate our model, resulting in relatively short trajectories. Therefore, there is still a gap between our model and the real application. (b) Due to the lack of external data, our model does not consider the impact of weather conditions, spatial–temporal events, etc. on travel time estimation. In the future, we will improve and compress our model so that the model can be deployed efficiently in a larger spatial range. Meanwhile, we will also involve some spatial–temporal variables that have a greater impact on the travel time estimation, such as weather conditions, emergencies, and traffic accidents.

Table 7
Performance of STGNN-TTE and its variants for estimating the travel times of the whole trajectories on three datasets.

Method	Chengdu1			Chengdu2			Porto		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Ours-STGCN	94.56	64.13	0.2775	127.78	66.21	0.2712	56.76	42.10	0.1579
Ours-Transformer	90.22	61.32	0.2632	122.59	64.06	0.2584	53.24	40.01	0.1494
Ours-Adaptive	89.34	61.20	0.2635	122.86	64.12	0.2597	53.10	39.92	0.1492
Ours-Multi scale	91.12	62.05	0.2668	123.57	64.61	0.2613	53.46	40.38	0.1516
STGNN-TTE (ours)	**88.03**	**60.71**	**0.2606**	**121.14**	**63.38**	**0.2568**	**52.35**	**39.25**	**0.1474**

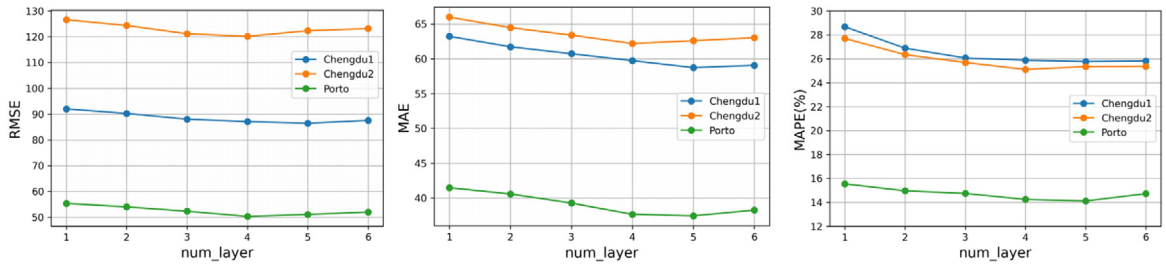

Fig. 8. The change trend of the model's evaluation metrics with the parameter *k*.

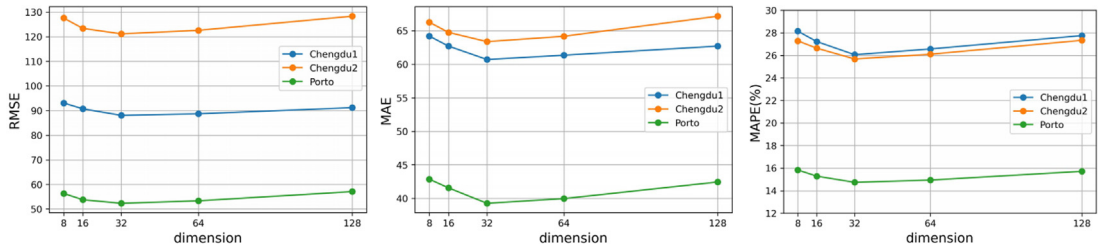

Fig. 9. The change trend of the model's evaluation metrics with the parameter *d*.

CRediT authorship contribution statement

Guangyin Jin: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **Min Wang:** Conceptualization, Software, Methodology. **Jinlei Zhang:** Conceptualization, Software. **Hengyu Sha:** Writing – original draft, Writing – review & editing. **Jincai Huang:** Supervision.

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.

References

[1] J.Y. Chen, J.L. Qiu, Digital utility: Datafication, regulation, labor, and DiDi's platformization of urban transport in China, Chin. J. Commun. 12 (3) (2019) 274–289.
[2] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Trans. Intell. Transp. Syst. 12 (4) (2011) 1624–1639.
[3] D. Billings, J.-S. Yang, Application of the ARIMA models to urban roadway travel time prediction-a case study, in: 2006 IEEE International Conference on Systems, Man and Cybernetics, 3, IEEE, 2006, pp. 2529–2534.
[4] A. Guin, Travel time prediction using a seasonal autoregressive integrated moving average time series model, in: 2006 IEEE Intelligent Transportation Systems Conference, IEEE, 2006, pp. 493–498.
[5] J.-S. Yang, A study of travel time modeling via time series analysis, in: Proceedings of 2005 IEEE Conference on Control Applications, 2005, CCA 2005, IEEE, 2005, pp. 855–860.
[6] B. Yu, H. Wang, W. Shan, B. Yao, Prediction of bus travel time using random forests based on near neighbors, Comput.-Aided Civ. Infrastruct. Eng. 33 (4) (2018) 333–350.
[7] B. Gupta, S. Awasthi, R. Gupta, L. Ram, P. Kumar, B.R. Prasad, S. Agarwal, Taxi travel time prediction using ensemble-based random forest and gradient boosting model, in: Advances in Big Data and Cloud Computing, Springer, 2018, pp. 63–78.
[8] Y. Zhang, A. Haghani, A gradient boosting method to improve travel time prediction, Transp. Res. C 58 (2015) 308–324.
[9] M. Elhenawy, H. Chen, H. Rakha, Random forest travel time prediction algorithm using spatiotemporal speed measurements, in: 21st World Congress on Intelligent Transport Systems, ITSWC, 2014.
[10] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, in: The Handbook of Brain Theory and Neural Networks, Vol. 3361, (10) 1995, p. 1995.
[11] T. Robinson, M. Hochberg, S. Renals, The use of recurrent neural networks in continuous speech recognition, in: Automatic Speech and Speaker Recognition, Springer, 1996, pp. 233–258.
[12] D. Wang, J. Zhang, W. Cao, J. Li, Y. Zheng, When will you arrive? estimating travel time based on deep neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, No. 1, 2018.
[13] H. Zhang, H. Wu, W. Sun, B. Zheng, Deeptravel: a neural network based travel time estimation model with auxiliary supervision, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3655–3661.
[14] T.-y. Fu, W.-C. Lee, Deepist: Deep image-based spatio-temporal network for travel time estimation, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 69–78.
[15] L. Fu, J. Li, Z. Lv, Y. Li, Q. Lin, Estimation of short-term online taxi travel time based on neural network, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2020, pp. 20–29.
[16] Y. Shen, J. Hua, C. Jin, D. Huang, TCL: Tensor-CNN-LSTM for travel time prediction with sparse trajectory data, in: International Conference on Database Systems for Advanced Applications, Springer, 2019, pp. 329–333.
[17] J. Qiu, L. Du, D. Zhang, S. Su, Z. Tian, Nei-TTE: intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city, IEEE Trans. Ind. Inf. 16 (4) (2019) 2659–2666.
[18] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, H. Wang, ConSTGAT: Contextual spatial-temporal graph attention network for travel time estimation at Baidu Maps, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2697–2705.
[19] Q. Wang, C. Xu, W. Zhang, J. Li, GraphTTE: Travel time estimation based on attention-spatiotemporal graphs, IEEE Signal Process. Lett. (2021).
[20] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: The 28th International Joint Conference on Artificial Intelligence, IJCAI, International Joint Conferences on Artificial Intelligence Organization, 2019.
[21] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640.
[22] G. Jin, H. Sha, Y. Feng, Q. Cheng, J. Huang, GSEN: An ensemble deep learning benchmark model for urban hotspots spatiotemporal prediction, Neurocomputing (2021).
[23] G. Jin, Q. Wang, C. Zhu, Y. Feng, J. Huang, J. Zhou, Addressing crime situation forecasting task with temporal graph convolutional neural network approach, in: 2020 12th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA, IEEE, 2020, pp. 474–478.

[24] R. Reza, S.S. Pulugurtha, V.R. Duddu, Arima Model for Forecasting Short-Term Travel Time Due to Incidents in Spatio-Temporal Context, Technical Report, 2015.
[25] Q.Y. Ding, X.F. Wang, X.Y. Zhang, Z.Q. Sun, Forecasting traffic volume with space-time ARIMA model, in: Advanced Materials Research, Vol. 156, Trans Tech Publ, 2011, pp. 979–983.
[26] R. Logendran, L. Wang, et al., Dynamic Travel Time Estimation Using Regression Trees, Technical Report, Oregon. Dept. of Transportation. Research Unit, 2008.
[27] X. Li, R. Bai, Freight vehicle travel time prediction using gradient boosting regression tree, in: 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA, IEEE, 2016, pp. 1010–1015.
[28] F. Zhang, X. Zhu, T. Hu, W. Guo, C. Chen, L. Liu, Urban link travel time prediction based on a gradient boosting method considering spatiotemporal correlations, ISPRS Int. J. Geo-Inf. 5 (11) (2016) 201.
[29] Y. Duan, L. Yisheng, F.-Y. Wang, Travel time prediction with LSTM neural network, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2016, pp. 1053–1058.
[30] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, Y. Liu, Multi-task representation learning for travel time estimation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1695–1704.
[31] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: a deep learning approach, IEEE Trans. Intell. Transp. Syst. 16 (2) (2014) 865–873.
[32] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31, No. 1, 2017.
[33] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, Z. Li, Deep multi-view spatial-temporal network for taxi demand prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, No. 1, 2018.
[34] H. Yao, X. Tang, H. Wei, G. Zheng, Z. Li, Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 5668–5675.
[35] H. Lin, R. Bai, W. Jia, X. Yang, Y. You, Preserving dynamic attention for long-term spatial-temporal prediction, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 36–46.
[36] G. Jin, Q. Wang, C. Zhu, Y. Feng, J. Huang, X. Hu, Urban fire situation forecasting: Deep sequence learning with spatio-temporal dynamics, Appl. Soft Comput. 97 (2020) 106730.
[37] J. Zhang, H. Che, F. Chen, W. Ma, Z. He, Short-term origin-destination demand prediction in urban rail transit systems: A channel-wise attentive split-convolutional neural network method, Transp. Res. C 124 (2021) 102928.
[38] G. Jin, Y. Cui, L. Zeng, H. Tang, Y. Feng, J. Huang, Urban ride-hailing demand prediction with multiple spatio-temporal information fusion network, Transp. Res. C 117 (2020) 102665.
[39] G. Jin, Q. Wang, X. Zhao, Y. Feng, Q. Cheng, J. Huang, Crime-gan: A context-based sequence generative network for crime forecasting with adversarial loss, in: 2019 IEEE International Conference on Big Data, Big Data, IEEE, 2019, pp. 1460–1469.
[40] Y. Zhang, S. Wang, B. Chen, J. Cao, Z. Huang, Trafficgan: Network-scale deep traffic prediction with generative adversarial nets, IEEE Trans. Intell. Transp. Syst. (2019).
[41] S. Wang, J. Cao, H. Chen, H. Peng, Z. Huang, SeqST-GAN: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction, ACM Trans. Spatial Algorithms Syst. 6 (4) (2020) 1–24.
[42] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, F. Wu, Deep sequence learning with auxiliary information for traffic prediction, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 537–546.
[43] A. Koesdwiady, R. Soua, F. Karray, Improving traffic flow prediction with weather information in connected cars: A deep learning approach, IEEE Trans. Veh. Technol. 65 (12) (2016) 9508–9517.
[44] C. Li, L. Bai, W. Liu, L. Yao, S.T. Waller, Urban mobility analytics: A deep spatial–temporal product neural network for traveler attributes inference, Transp. Res. C 124 (2021) 102921.
[45] J. Tang, J. Liang, F. Liu, J. Hao, Y. Wang, Multi-community passenger demand prediction at region level based on spatio-temporal graph convolutional network, Transp. Res. C 124 (2021) 102951.
[46] G. Jin, C. Zhu, X. Chen, H. Sha, X. Hu, J. Huang, UFSP-NEt: a neural network with spatio-temporal information fusion for urban fire situation prediction, in: IOP Conference Series: Materials Science and Engineering, Vol. 853, (1) IOP Publishing, 2020, 012050.
[47] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, 2018.
[48] C. Zheng, X. Fan, C. Wang, J. Qi, Gman: A graph multi-attention network for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 1234–1241.
[49] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, J. Yu, Traffic flow prediction via spatial temporal graph neural network, in: Proceedings of the Web Conference 2020, 2020, pp. 1082–1092.
[50] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04, 2020, pp. 3529–3536.
[51] Q. Zhang, J. Chang, G. Meng, S. Xiang, C. Pan, Spatio-temporal graph structure learning for traffic forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 1177–1185.
[52] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, in: 34th Conference on Neural Information Processing Systems, 2020.
[53] D. Zhang, F. Xiao, M. Shen, S. Zhong, DNEAT: A novel dynamic node-edge attention network for origin-destination demand prediction, Transp. Res. C 122 (2021) 102851.
[54] F. Li, J. Feng, H. Yan, G. Jin, D. Jin, Y. Li, Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution, 2021, arXiv preprint arXiv:2104.14917.
[55] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 3656–3663.
[56] D. Chai, L. Wang, Q. Yang, Bike flow prediction with multi-graph convolutional networks, in: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2018, pp. 397–400.
[57] J. Zhang, F. Chen, Y. Guo, X. Li, Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit, IET Intelligent Transport Systems 14 (10) (2020) 1210–1217.
[58] G. Jin, Z. Xi, H. Sha, Y. Feng, J. Huang, Deep multi-view spatiotemporal virtual graph neural network for significant citywide ride-hailing demand prediction, 2020, arXiv preprint arXiv:2007.15189.
[59] J. Ke, X. Qin, H. Yang, Z. Zheng, Z. Zhu, J. Ye, Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network, Transp. Res. C 122 (2021) 102858.
[60] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.
[61] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 3844–3852.
[62] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[63] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[64] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.
[65] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, 2018, arXiv preprint arXiv:1812.08434.
[66] S. Luan, M. Zhao, X.-W. Chang, D. Precup, Break the ceiling: Stronger multi-scale deep graph convolutional networks, 2019, arXiv preprint arXiv:1906.02174.
[67] C. Yang, G. Gidofalvi, Fast map matching, an algorithm integrating hidden Markov model with precomputation, Int. J. Geogr. Inf. Sci. 32 (3) (2018) 547–570.
[68] B. Yang, M. Kaul, C.S. Jensen, Using incomplete information for complete weight annotation of road networks, IEEE Trans. Knowl. Data Eng. 26 (5) (2013) 1267–1279.
[69] T. Idé, M. Sugiyama, Trajectory regression on road networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 25, No. 1, 2011.
[70] J. Zheng, L.M. Ni, Time-dependent trajectory regression on road networks via multi-task learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 27, No. 1, 2013.
[71] J. Hu, C. Guo, B. Yang, C.S. Jensen, Stochastic weight completion for road networks using graph convolutional networks, in: 2019 IEEE 35th International Conference on Data Engineering, ICDE, IEEE, 2019, pp. 1274–1285.

[72] H. Wang, X. Tang, Y.-H. Kuo, D. Kifer, Z. Li, A simple baseline for travel time estimation using large-scale trip data, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–22.
[73] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017) 3146–3154.

Guangyin Jin: He is a Ph.D. student in Systems Engineering College from National University of Defense Technology. He worked as a member at the Intelligent System Engineering Group. His research interests include Spatiotemporal Data Mining, Urban Computing, Graph Neural Network and Real-world Machine Learning.

Min Wang: She is a master student in Systems Engineering College from National University of Defense Technology. He worked as a member at the Intelligent System Engineering Group. His research interests include Graph Neural Network and Data Mining.

Jinlei Zhang: He is a Ph.D. student in Beijing Jiaotong University. He is also a visiting Ph.D. student at the University of Washington. His research interests include machine learning, deep learning, traffic data mining and applications, as well as dynamic traffic modeling and management.

Hengyu Sha: He is a master student in Systems Engineering College from National University of Defense Technology. He worked as a member at the Intelligent System Engineering Group. His research interests include Spatiotemporal Data Mining, and Graph Neural Network.

Jincai Huang: He is a professor of the National University of Defense Technology, Changsha, Hunan, China, and a researcher of Science and Technology on Information Systems Engineering Laboratory. His main research interests include artificial general intelligence, deep reinforcement learning, and multi-agent systems.