Neurocomputing 446 (2021) 95–105

Contents lists available at ScienceDirect

Neurocomputing

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

Fine-grained predicting urban crowd flows with adaptive spatio-temporal graph convolutional network

Xu Yang [a], Qiang Zhu [a], Peihao Li [a], Pengpeng Chen [a,b], Qiang Niu [a,b,*]

[a] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China
[b] Mine Digitization Engineering Research Center of Minstry of Education of the People's Republic of China, Xuzhou, China

ARTICLE INFO

Article history:
Received 25 May 2020
Revised 20 December 2020
Accepted 27 February 2021
Available online 10 March 2021
Communicated by Zidong Wang

Keywords:
Predicting urban crowd flows
Spatio-temporal graph convolutional network
Modeling crowd flows
Flow direction

ABSTRACT

Predicting crowd flows is important for traffic management and public safety, which is very challenging as it is affected by many complex factors. In this paper, we propose a novel fine-grained predicting urban crowd flows approach with an adaptive spatio-temporal graph convolutional network, called ASTGCN. This approach can simultaneously predict the inflow, outflow, and flow direction. We first design a method for modeling crowd flow in irregular urban regions based on urban bus line data. Then, we design an end-to-end structure of the adaptive spatio-temporal graph convolutional network with unique properties of spatio-temporal data. Finally, extensive experiments on GAIA open dataset are constructed to evaluate the performance of ASTGCN. Results show that our approach outperforms four well-known methods, the average absolute error is reduced by 28.7 %, and the root mean square error is reduced by 37.9 %.

1. Introduction

Predicting crowd flows can provide scientific insights into the distribution of urban population, patterns in resident activities, optimization of public resource allocation, location choices for commercial facilities, and contingency plans for public safety. In the era of big data, the emergence of spatio-temporal big data and deep learning technology has brought new opportunities and challenges to the prediction of urban population mobility [1–3].

Prior work mainly focused on predicting the flows of crowds (inflow and outflow) [4–7]. Unlike previous work, this paper proposes a fine-grained crowd flows prediction approach based on an adaptive graph convolutional network, which can simultaneously predict inflow, outflow, and flow direction. Inflow is the total traffic of crowds entering a region from other places during a given time interval [8]. Outflow denotes the total traffic of crowds leaving a region for other places during a given time interval [8]. Flow direction is the moving orientation of people between different regions. However, utilizing spatio-temporal data to predict the above three types is very challenging because of the following three complex factors:

1. **How to divide irregular urban regions based on data?** Prior work mainly focused on predicting the crowd flows in regular grid regions. However, urban regions are very irregular as those regions in a city are separated by road networks. Therefore, we need to design an irregular urban region division method to maintain the urban spatial integrity and fine-grained information of the divided regions.
2. **How to depict crowd flows with trajectory data?** Prior work mainly counted the total inflow and outflow in a region, which is not suitable for the prediction of fine-grained crowd flows. Therefore, we need to propose a method that can utilize trajectory data to depict fine-grained crowd flows.
3. **How to make full use of the spatio-temporal features of crowd movement?** The urban crowd flows data contains rich spatio-temporal features. Temporal characteristics include proximity, periodicity, and trend in time. Spatial characteristics include regional, distance, and hierarchy.

To address these challenges, we propose an adaptive spatio-temporal graph convolutional network for crowd flow prediction, called ASTGCN. ASTGCN can predict fine-grained urban crowd moving information (including inflow, outflow, and flow direction). The main contributions of this paper are as follows.

1. We present a novel method for modeling crowd flow in irregular urban regions. With the urban bus route line, we merge the

* Corresponding author at: School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China.

E-mail addresses: yang_xu@cumt.edu.cn (X. Yang), niuqiangkd@163.com (Q. Niu).

https://doi.org/10.1016/j.neucom.2021.02.089
0925-2312/© 2021 Elsevier B.V. All rights reserved.

morphological geometry algorithm and K-means ++ clustering algorithm to divide the city into many regions in arbitrary shapes. Besides, a flow-field generation algorithm is proposed to generate low-volume, high-value density spatio-temporal flow-field data.

2. We propose a novel adaptive spatio-temporal graph convolutional network for predicting crowd flow. To the best of our knowledge, ASTGCN is the first to predict fine-grained crowd flows, including inflow, outflow, and flow direction. Spatio-temporal characteristics of crowd flows are largely utilized by the adaptive spatio-temporal graph convolutional network.
3. We establish extensive experiments on GAIA open dataset [9] to evaluate the performance of ASTGCN. Our proposed flow-field generation algorithm generates a total of 275 MB of urban crowd flow-field data, which is only 0.13% of the original data volume, significantly reducing data storage consumption. Experimental results show that the performance of ASTGCN is better than the four baseline models. The average absolute error is reduced by 28.7 %, and the root mean square error is reduced by 37.9 %.

2. Related Work

2.1. Spatio-Temporal Prediction based on Deep Leaning

Many works predict an individual's movement based on their location history [10–13], and they mainly predict the trajectory of millions or even billions of people, not the total population flow in the area. Such tasks may require large amounts of computing resources and are not necessary for public safety situations. Predicting travel speed and traffic volume on the road are the current research hotspots, where [14–16] have achieved excellent results. Most of them make predictions concerning single or multiple road segments, rather than citywide ones [17,18]. Recently, researchers have started to focus on city-scale traffic flow prediction [19–21]. Specifically, [6] proposes a deep spatio-temporal residual convolutional network to realize the urban traffic forecasting model, which divides the city into regular grid areas by latitude and longitude, and aggregates the trajectories into corresponding regions. [22] designs a spatio-temporal dynamic network (STDN) that uses a convolutional neural network and a long-short-term memory network with an attention mechanism to predict traffic flow. This method ingeniously converts urban crowd flow data into standard 2D or 3D grid data at the early stage of modeling and uses convolutional neural networks to extract spatial attributes. However, to the best of our knowledge, there is few work that can forecast the fine-grained flow information at the same time, including inflow, outflow, and flow direction.

2.2. Graph Convolutional Network

The graph convolutional network(GCN) is a key part of our paper. GCNs are deep learning based methods that operate on graph domain [23–25]. Due to its convincing performance and high interpretability, GCN has been a widely applied graph analysis method. GCNs are widely used in social network prediction [26,27], recommender systems [28,29], graph representation [30], image classification [31], and semantic segmentation [32]. Specially, [33] designs an extension of the spectral network to test large-scale classification problems. Some researchers apply the convolution to directly filter graph nodes and their neighbors [34,23]. [35] constructs graphs using a statistical variance measure dependent on joint distances. The temporal pyramid covariance descriptors are adopted to represent joint locations. Recently, [36] constructs a skeleton graph and applies graph convolutional networks (GCN) to extract correlated features to capture joint dependencies. In this paper, We propose a novel adaptive spatio-temporal graph convolutional network for predicting urban crowd flows.

3. Preliminary

3.1. Related Definitions

Definition 1 (*crowd flows*). $\mathbb{S}^t$ is the set of trajectory data in time period *t*. For irregular regions *i* and *j*, the flow of people from region *i* to region *j* is defined as follows:

$$x_t^{i\to j} = \sum_{Tr\in\mathbb{S}^t} |\{k > 1|c_{k-1} \in i \wedge c_k \in j\}|, i \neq j \tag{1}$$

where $Tr : c_1 \to c_2 \to \cdots \to c_{|Tr|}$ is a trajectory in $\mathbb{S}^t$, C_k is the *kth* spatial coordinate of the trajectory sampling point of $Tr, c_k \in i$ indicates that the sampling track point C_k is within the region i, $|\cdot|$ is the cardinality of the set, and $i \to j$ is the direction of crowd flows.

Definition 2 (*spatio-temporal graph*). A spatio-temporal graph is denoted as a directional graph $G_t^f = (\Re, E^t)$ where $\Re$ and E^t denote the set of irregular regions and edges, respectively. If within the time period *t*, the number of people starting from r_i to r_j is greater than the threshold ξ, then there would exist a directional edge between vertices r_i and r_j with a weight of $x_t^{i\to j}$.

Problem 1(urban crowd flow prediction): Given a spatio-temporal graph G and observed attributes of nodes $\{X_t|t = 1, 2, \ldots, \tau\} \in R^{\tau\times N\times C}$, where N is number of regions and C is number of channels, predicting the attributes is calculated as: $Y = \left\{X_{\widehat{T}+1}, X_{\widehat{T}+2}, \ldots, X_{\widehat{T}+n} | \widehat{T} \geqslant \tau, n \geqslant 1\right\} \in R^{n\times N\times C}$. When $\widehat{T} = \tau$ and $n = 1$, the single step prediction is $Y = \{X_{\tau+1}\}$.
Table 1 shows the description of relevant notations of important variables in this paper.

3.2. Analysis of Crowd Flow Spatio-Temporal Characteristics

This section analyzes the two-month trajectory data of Chengdu Didi's car-hailing, which is from GAIA open dataset [9]. We generate a crowd flow dataset of 8784-time segments at 10-min intervals. Experimental analysis proves that urban crowed flows have strong temporal and spatial characteristics. The specific analysis process and results are as follows:

(1) Temporal characteristics. The morning and evening peaks of the day are the most active periods of urban crowd flow. Moreover, the mobility of the urban population is low at midnight. The overall trend is first rising and falling in a day, so we can conclude that the regional crowd flows have a strong periodicity in the time dimension.

Table 1
Notation.

symbol	description
X	input data
Y, X_P	predicted target
lc, ld, lw	time series length of different samples
τ	total length of time series
$G_t^f = (\Re, E)$	spatio-temporal graph
A, D, L, S	adjacency matrix, degree matrix, Laplacian matrix, adaptive adjacency matrix
$\Re = \{r_1, r_2, \ldots, r_{\|\Re\|}\}$	city region set
Tw, Td, Tc	time sampling point
w, d, c	sampling time interval

(2) Spatial characteristics: The outflow and inflow of a region are approximately positively correlated, and an increase in the inflow of an area will inevitably lead to an increase in the outflow. However, its positive correlation coefficient will fluctuate with time and regional functionality. For example, in the morning peak of work day, the outflow of people in the residential area is larger than the inflow of its crowd, while the flow direction is opposite the evening peak hours of off work. To verify this characteristic, we calculate the Pearson correlation coefficient of the number of people moving in and the number of people moving out of each area within ten minutes a day. The Pearson correlation coefficient varies from 0.85 to 1. Therefore, we can conclude that outflow and inflow of a region are approximately positively correlated.

4. Modeling the Urban Crowd Flow in Irregular Regions

4.1. Irregular Region Division based on Bus Lines

The road network separates the regions in the city, so they are too irregular. Therefore, we propose a novel urban irregular division algorithm based on bus lines, which includes two stages: urban area division and urban area aggregation.

(1) Urban area division: First, we divide 2048×2048 fine-grained grid regions on the city map based on latitude and longitude coordinates. Data is then extracted from the city bus line data, which consists of one city bus line record, each with an orderly arrangement of GPS coordinates of the bus line's essential route points. Each point is mapped into the grid area and connects the two adjacent points to get a $0,1$ binary image as shown in Fig. 1, where 0 represents city bus lines, and 1 represents the blank area.

To reduce the impact of the bus line ends that do not extend to the image boundary, the original 2048×2048 image is cropped to obtain a new 2000×2000 size image. Then, the binary image is expanded and eroded. The operation results are shown in Fig. 1, Fig. 2 and Fig. 3.

The boundaries of each blank region are extracted using an algorithm proposed in the literature [37]. By calculating the region's centroid of the boundary and filling the region surrounded by the boundary point with a unique color, the grid-cell with the same pixel value belongs to the same irregular region. Fig. 4 shows an urban regional segmentation result, where a total of 293 of urban regions are divided based on the urban bus lines.

(2) Urban region aggregation: The result obtained by the above urban division method contains many fine-grained irregular areas. Most of them are too small to study urban-scale crowd movement. Therefore, we use the K-means ++ clustering algorithm [38,39] to cluster the centroid coordinates of low-level regions. First, the centroids of all regions are calculated as shown in Fig. 5. Then centroid coordinates are aggregated into 64 clusters based on the k-means ++ algorithm, as shown in Fig. 6. Finally, each line coordinate is assigned to its belonging cluster, avoiding the impact of the line network's existence on mapping the track points to the designated area in the subsequent process. After the above three steps, the city is divided into 64 sub-regions as shown in Fig. 7.

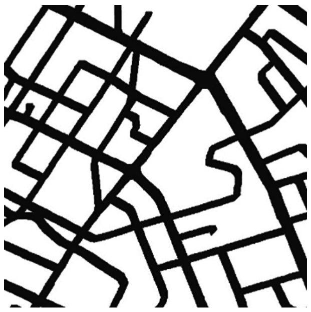
Fig. 2. Dilated bus network.

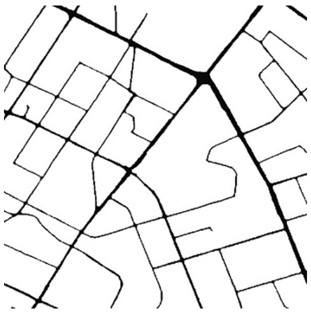
Fig. 3. Eroded bus network.

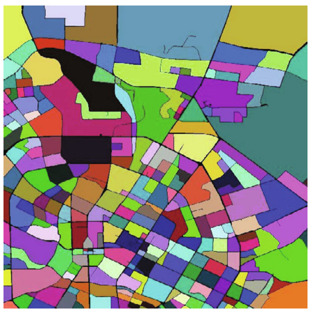
Fig. 4. Regional segmentation results of Second Ring Road, Chengdu.

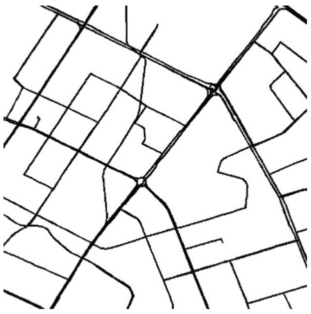
Fig. 1. Original bus network.

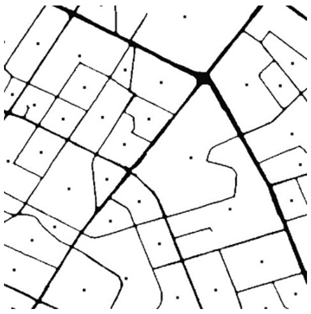
Fig. 5. Regional centroids.

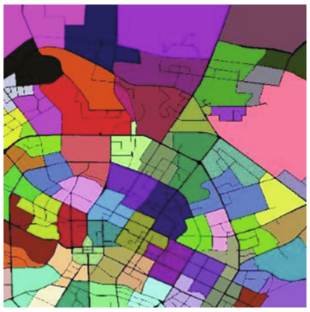

Fig. 6. Merged Region.

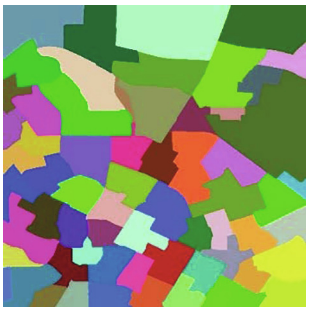

Fig. 7. 64 sub-regions.

The summarized algorithm flow is shown in Algorithm 1.

```
Algorithm 1: Region division algorithm
Input: Bus line dataset $B$, southwest coordinates of target area $< lat1, lng1 >$, northeast coordinates of the target area $< lat2, lng2 >$, number of grids $N_1, N_2, N_1 \times N_2$ zero-value base map $M$, and number of regions $K$.
Output: Urban region map after segmentation $M$.
    Step1:Partitioning the map into $N_1 \times N_2$ small grid-cells, each grid area represents a geographic region.
    Step2:Bus line is denoted as $L : c_1 \to c_2 \to \cdots \to c_{|Tr|}$, where $c_i$ is the critical point coordinate. Mapping it to the corresponding grid area according to its latitude and longitude coordinates to get its area coordinates $(a_i, a_j)$; connecting two adjacent points $(a_i, a_j), (a_{i+1}, a_{j+1})$; setting the pixel connected by the two bus stations in the base map $M$ to 1.
    Step3:Performing tailoring and expansion corrosion operations for the processed segmentation map $M$ in order to get irregular regions based on the bus line network.
    Step4:Performing 0 and 1 interchange operations on the segmentation map $M$; obtaining the boundary $C_i$ and the centroid coordinates of each region; using K-means ++ clustering algorithm to obtain $K$ clustering centers of the centroid coordinates of all regions; seting each pixel value in the boundary $C_i$ to the cluster center number.
    Step5:For the pixel with a value of 0 in the base image $M$, setting the value to the cluster center number, which is closest to it.
```

4.2. Flow-Field Data Generation Algorithm

We propose a flow-field generation algorithm that can generate urban crowd flow-field data from trajectory data. The flow-field data records the flow and direction of crowd flow between each urban area and adjacent areas. The specific description is shown in Algorithm 2.

```
Algorithm 2: flow-field generation algorithm
Input: Require trajectory dataset $\mathbb{S}$, time interval $t_d$, City region segmentation map $M$;
Output: Urban crowd flow-field dataset $\mathbb{F}$;
    Step1:According to the $t_d$ time interval, the dataset is divided into multiple time slices $t \in \{t_0, t_1, \ldots, t_n\}$, select the trajectory subset $\mathbb{S}^t$ within the range of the current t time slice from the trajectory dataset $\mathbb{S}$.
    Step2:For trajectory in $Tr : c_1 \to c_2 \to \ldots \to c_{|Tr|}$ in $\mathbb{S}^t$
    Step3:Calculate the crowd flow-field vector between any two regions according to Definition 1.
    Step4:For each area in the city, the main vectors that flow out to other areas in the corresponding time slice t are generated to obtain the crowd flow-field.
    Step5:Set a threshold value to filter out the main vector of the region with low flow rate.
```

We use the above flow-field data generation algorithm to process 195GB of trajectory data and set the time interval to 10 min. The experiment results prove that the generated urban crowd flow-field data's volume is only 0.13% of the original data, which is 275MB.

5. Prediction of crowd flow based on adaptive spatio-temporal graph convolutional network

5.1. Model Structure

Fig. 8 shows the architecture of the adaptive spatio-temporal map convolutional urban crowd flow prediction network (ASTGCN), including the input module, adaptive Laplacian matrix generation module, spatio-temporal graph convolutional component, and output module.

5.2. Spatio-temporal Graph Convolutional Component

The adaptive spatio-temporal graph convolutional network comprises several spatio-temporal graph convolutional components stacked to realize the extraction of spatio-temporal correlation in the spatio-temporal graph sequence data of urban population flow. Each spatio-temporal graph convolutional component comprises a spatial dimension graph convolutional module and a temporal dimension multi-view sequential convolutional module. This section introduces the design principles and structural details of the spatio-temporal convolutional model.

5.2.1. Spatial dimension graph convolutional module

In our model, the spatio-temporal graph data of crowed flows is based on the directed graph structure data. The standard convolutional neural network uses a standardized convolutional kernel and cannot well extract the spatial information of non-Euclidean data, so we use frequency-domain graph convolution for spatial feature extraction, which is denoted as $*_G$. According to the definition of convolution in the frequency domain, the diagonalized convolutional kernel is used in the Fourier domain $g_\theta = \mathrm{diag}(\theta), \theta \in R^N$. For the graph signal $X \in R^{N\times C}$, the formal definition of graph convolutional operation is as follows:

$$g_\theta * GX = g_\theta(L)X = g_\theta\left(U\Lambda U^T\right)X = Ug_\theta(\Lambda)U^T X \tag{2}$$

where $L = I_N - D^{-1/2}AD^{-1/2} = U\Lambda U^T \in R^{N\times N}$ is normalized form of Laplacian matrix for graph. A is the adjacency matrix and $D_{ii} = \sum_j A_{ij}$ the degree matrix of the graph. $U \in R^{N\times N}$ is eigenvector matrix of normalized Laplacian matrix, and Λ is eigenvalue matrix.

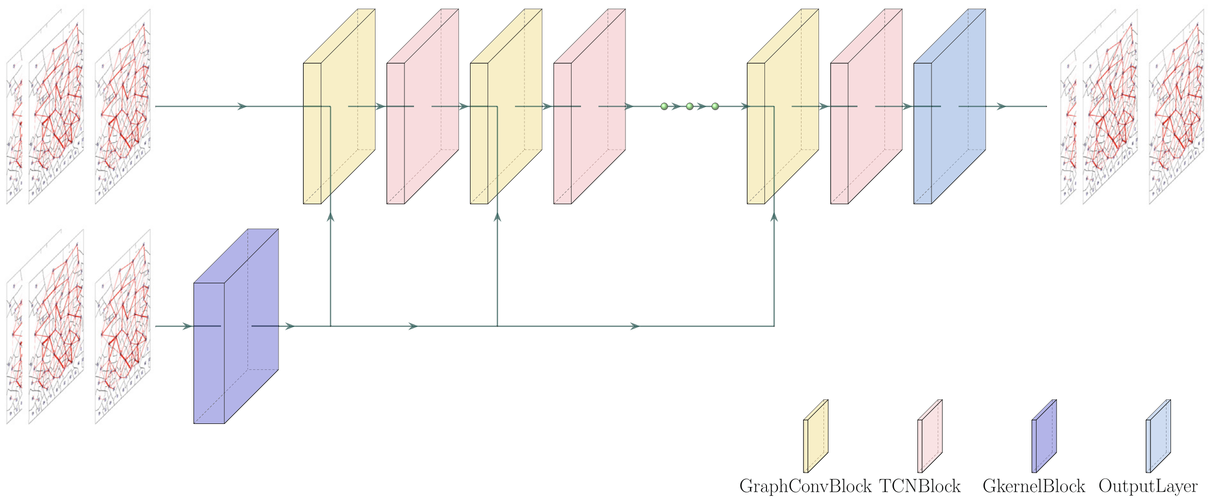

Fig. 8. Architecture of adaptive spatio-temporal convolutional model.

The above formula can be understood as the simultaneous Fourier transform of the convolutional kernel g_θ and the graph signal X to the frequency domain, the multiplication of the change results, and then the inverse Fourier transform to obtain the final convolutional operation result.

It is worth noting that the eigenvalue decomposition directly on the Laplacian matrix is computationally expensive, and its time complexity is $O(N)$. In particular, when the scale of the graph is larger, the consumption of computing resources is more obvious. To solve this problem, the literature [40] is proposed to use Chebyshev K-order polynomial expansion to approximate the solution. The specific details are as follows:

$$g_\theta * GX = g_\theta(L)X \approx \sum_{k=0}^{K-1}\theta_k T_k\left(\widetilde{L}\right)X \tag{3}$$

where $\theta_k \in R^k$ is the Chebyshev polynomial coefficient, $T_k\left(\widetilde{L}\right)$ is the order Chebyshev polynomial and its recursion is defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), T_0(x) = 1, T_1(x) = x, L = 2L/\lambda_{\max} - I_N, \lambda_{\max}$ is the maximum eigenvalue. Using Chebyshev polynomial solving is equivalent to updating the current vertex information using $0 \sim$ (K-1) order neighbor node information for current node in the graph.

To further simplify the calculation, literature [36] proposes a 1st-order Approximation method, which simplifies the graph convolutional operation by setting $K = 2$. The specific definition is as follows:

$$g_\theta * GX = g_\theta(L)X = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}X\theta \tag{4}$$

where $\widetilde{A} = A + I_N$, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, $\theta \in R^{C_\in \times C_{\text{out}}}$ is trainable matrix of filter parameters in graph convolutional layer. We use this method for fast graph convolutional calculation, and achieve the similar effect of K-localized algorithm by superimposing multiple same graph convolutional operations.

We propose a method to dynamically generate a Laplacian matrix based on the connected state of the input spatio-temporal graph sequence to generate a mean adjacency matrix suitable for the local input spatio-temporal graph sequence. The process is described formally as follows:

$$\overline{A} = \frac{1}{T}\sum_{i=1}^{T} A_i \tag{5}$$

where A_i is the region connected adjacency matrix corresponding to each spatio-temporal graph in the input spatio-temporal graph sequence. We need to transform the asymmetric adjacency matrix of the directional graph into a symmetric adjacency matrix. Therefore, we perform a secondary treatment on a $\overline{A}$:

$$S = 1/2\left(\overline{A} + \overline{A}^T\right) \tag{6}$$

Symmetric adjacency matrix S is used to construct Laplacian matrix.

The spatial graph convolutional module design is shown in Fig. 9. The graph convolutional module includes a graph convolutional network layer and a bottleneck convolutional module. The output of the convolution of each layer is described formally as follows:

$$H^{(l+1)} = \widetilde{Q}^{-1/2}\widetilde{S}\widetilde{Q}^{-1/2}H^l\theta^l \tag{7}$$

where $H^{(l+1)} \in R^{N\times C_{l+1}}, H^l \in R^{N\times C_l}, \theta^{(l)} \in R^{C_l\times C_{l-1}}$ represent the input, output and learnable parameters of the convolutional layer of the l layer, respectively. $\widetilde{S} = I_N + S$ is modified adjacency matrix with added self-connection and $\widetilde{Q}_{it} = \sum_j \widetilde{S}_{i,j}$ is degree matrix.

In order to prevent the gradient disappearance and gradient drop generated by the superimposed network module, we use the identity connection in the module, and use bottleneck convolution to maintain the consistency of the input and output.

Finally, the output formula of the convolution of the $l + 1$ space graph is described as follows:

$$H^{l+1} = f\left(f_{1\times 1\_\text{conv}}\left(H^l\right) + \widetilde{Q}^{-1/2}\widetilde{S}\widetilde{Q}^{-1/2}H^l\theta^l\right) \tag{8}$$

where f is an activation function, and this paper uses *relu* as an activation function.

5.2.2. Time dimension convolutional module

To expand the receptive-field of time-dimensional convolution, we design a multi-view timing convolutional module. The core of the module is to replace the standard convolutional kernel with a variety of $1 \times K$ size hollow convolutions with different expansion ratios in the time dimension. The example of the convolutional process acting on the one-dimensional time series is shown in the Fig. 10.

As shown in Fig. 11, for the hidden state $H_G^l \in R^{t\times N\times C}$ output by the convolutional module of the previous layer, the formula of the

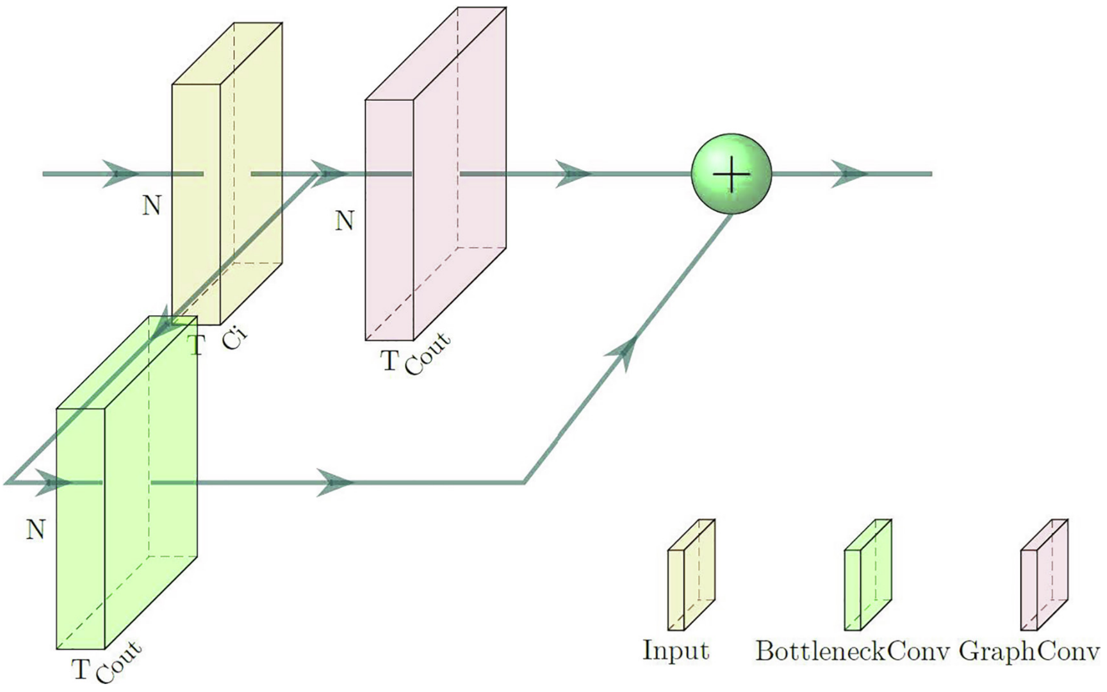

Fig. 9. Spatial graph convolutional module.

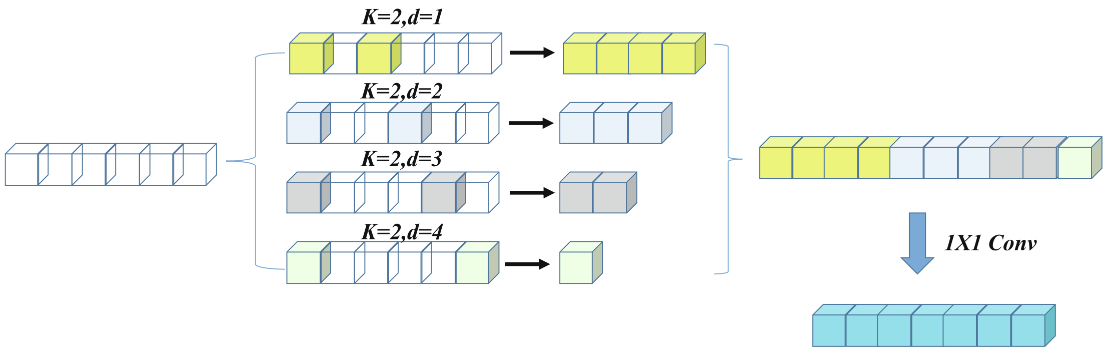

Fig. 10. 1-D multi-view temporal convolutional module.

hollow convolutional output of the i layer is described as follows the hidden state of the output of the convolutional module for the previous $H_G^l \in R^{t \times N \times C}$:

$$H_T^{l,K_i,r_i} = \mathrm{relu}\left(f_{\mathrm{atrous\_conv}}\left(H_G^l, K_i, r_i\right)\right) \tag{9}$$

where $f_{\mathrm{atrous\_conv}}$ represents convolution, K_i, r_i are convolutional kernel size and void rate, respectively. After connecting the empty convolutional outputs of different K, r combinations to each other, 1×1 convolution is used for feature fusion. The formula description of this process is as follows:

$$H_T^{l,K,r} = f_{1\times1\_\mathrm{conv}}\left(\mathrm{concat}\left\{H_T^{l,K_i,r_i} | i = 1,2,3,\ldots\right\}\right) \tag{10}$$

where $f_{1\times1\_\mathrm{conv}}$ represents 1×1 convolution, $H_T^{l,K_i,r_i} \in R^{t_i \times N \times C}$ is output, the final output of the *LTH* sequence convolutional module is formulated as follows after the addition of the output of the convolution with the bottleneck and the activation by the *relu* activation function:

$$H_T^l = relu\left(f_{1\times1\_conv}\left(H_G^l\right) + H_T^{l,K,r}\right) \tag{11}$$

where $H_T^l \in R^{\mathrm{lout} \times N \times C}$.

5.3. Output Layer and Loss Function

We use a time-dimensional convolutional module combined with the *tanh* activation function as the final output module. The output module takes the output $H_T^l \in R^{T \times N \times N}$ of the adjacent sequential convolutional module as input and uses a sequential convolutional module to get the hidden state $H_T^{l+1} \in R^{\mathrm{lout} \times N \times N}$. The *tanh* function is used as the output activation function of the output layer to map the output to [0,1]. The final prediction result of the model on the input time series is obtained, denoted as $X_P = \tanh\left(H_T^{l+1}\right) \in R^{\mathrm{tout} \times N \times N}$, *tout* represents the length of the predicted time series and N represents the number of regions.

Mean square error (MSE) is used as the loss function. We divide the calculation into 0-position MSE and non-0-position MSE, and finally use the sum of the two as a loss function. The loss function is defined as follows:

$$L(\theta) = \left\|X_P^0 - X_T^0\right\|_2^2 + \left\|X_P^{\bar{0}} - X_T^{\bar{0}}\right\|_2^2 \tag{12}$$

where θ is the learnable parameter of ASTGCN.

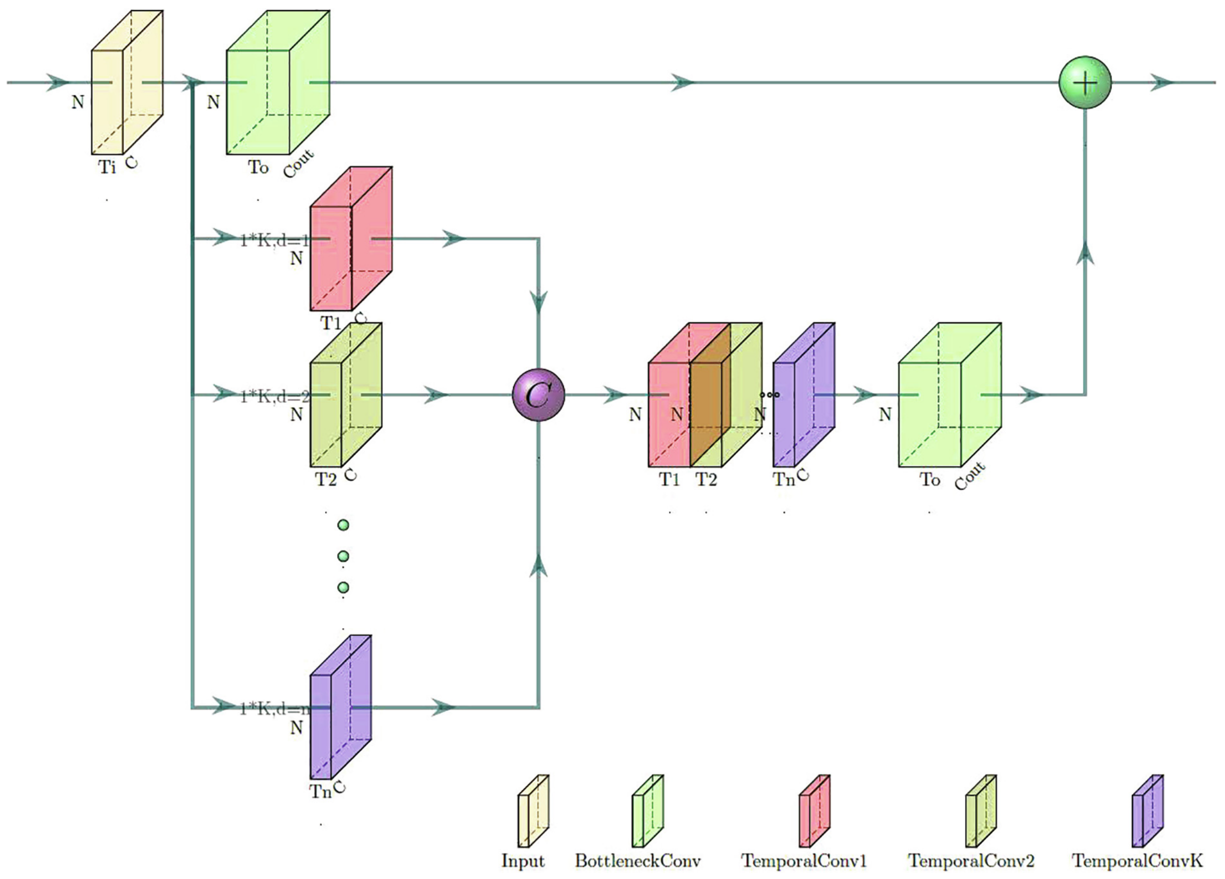

Fig. 11. 2-D Multi-view temporal convolutional module.

6. Experiments

6.1. Setting

Dataset: GAIA open dataset is Didi's car-hailing trajectory data. Its collection locations are Chengdu, Sichuan Province, China. The collection time was October-November 2016, and the collection interval was 2-4s. The city bus line data was the bus line data of Chengdu in 2016. The specific details of GAIA open dataset are shown in Table 2.

Data preprocessing: First, the urban region division method (proposed in Section 3.1) is used to divide the local area of Chengdu Second Ring Road into 64 spatially independent irregular regions using bus line data. Then, we filter the missing data from the original data of the trajectory to record the missing data and use the flow generation algorithm (proposed in Section 3.2) to generate the crowd flow-field in the urban area at 10 -minute intervals and store each time slice in the form of an adjacency matrix. Finally, the min–max normalization method is used to normalize the data to $[0,1]$ as the network input. This paper takes full account of the temporal properties of the data that are proximity to trend and periodicity when constructing the input time series, setting up three different sampling methods Tc, Td, Tw to obtain larger temporal dimensional fields, and modeling the temporal dependence of urban population flows. A sampling sequence of the observed values from the input spatio-temporal map is shown in Fig. 12.

Table 2
Data description.

	Type	Examples	Remarks
Driver ID	String	glox. jrrlltBMvCh8nxqktdr2dtopmlH	Desensitized
Order ID	String	jkkt8kxniovIFuns9qrrlvst@iqnpkwz	Desensitized
Timestamp	String	1501584540	Unix timestamp
Longitude	String	104.04393	GCJ-02 coordinate system
Latitude	String	30.726491	GCJ-02 coordinate system

Three different time dimension sampling is near time sampling Tc, daily cycle time sampling *Td* and weekly cycle time sampling *Tw*. Each sampling method corresponds to a different sampling time, and sequence lengths are respectively set to $lc \in \{4,5,6\}, ld \in \{0,1,2,3\}, lw \in \{0,1,2\}$. The corresponding sampling interval units are c, d, w, respectively. Each sampling method follows the following rules to generate the input space–time graph sequences:

$$I_c = \text{concat}\left[X_{t-lc^*c}, X_{t-(lc-1)^*c}, X_{t-(lc-2)*c}, \ldots X_{t-l^*c}\right] \in R^{lc\times N\times C} \tag{13}$$

$$I_d = \text{concat}\left[X_{t-ld^*d}, X_{t-(ld1)^*d}, X_{t-(ld-2)*d}, \ldots X_{t-ld^*d}\right] \in R^{ld\times N\times C} \tag{14}$$

$$I_w = \text{concat}\left[X_{t-lw^*w}, X_{t-(lw1)^*w}, X_{t-(lw-2)*w}, \ldots X_{t-lw^*w}\right] \in R^{lw\times N\times C} \tag{15}$$

The format of the final input sequence is as follows:

$$\text{Input} = \text{concat}[I_w, I_d, I_c] \in R^{(lw+ld+lc)\times N\times C} \tag{16}$$

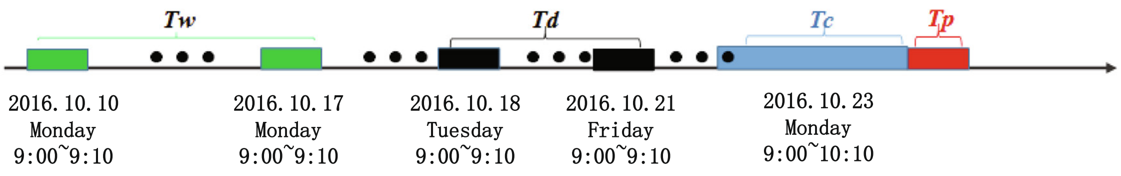

Fig. 12. Time dimension sampling.

Experimental environment: We have implemented an adaptive spatio-temporal graph convolutional network architecture (ASTGCN) model based on Tensorflow. Table 3 describes the experimental environment in detail.

Hyperparameter: First, the test and training data sets were constructed from the original data, and then the training and test sets were proportionally divided. Then the back propagation algorithm and Adam optimizer were used to set the learning rate as $le-5$ and the attenuation rate as 0.7. Model weights were loaded after model training, and the test data set was used to evaluate the model.

Baseline model: We compare our algorithm with the following four space–time prediction methods: (1) historical mean (HA): it uses the average value of the flow of crowds in recent historical cities as the predicted value; (2) autoregressive integral moving average method (ARIMA) [41]: this method requires time series data to be stable, its advantage lies in the prediction of short-term time series; (3) Long Short-Term Memory (LSTM) [42]: it uses the memory gate and the forget gate mechanism to realize the modeling of long and short-term time series; (4) STGCN: it is a spatio-temporal graph convolutional network with gated convolutional mechanism [43], where the frequency domain graph is used to extract the spatio-temporal characteristics of vehicle flow and achieve traffic flow prediction.

Metrics: We use root mean square error (RMSE) and mean absolute error (MAE) as evaluation indicators. The standard MAE and RMSE calculation formulas are as follows:

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|x_i - \widehat{x}_i| \tag{17}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \widehat{x}_i)^2} \tag{18}$$

When calculating RMSE and MAE, it is necessary to denormalize the data to the true value of the data. The denormalization process and its formula are described as follows:

Table 3
Experiment configuration.

	Release notes
Operating system	Ubuntu 16.04.1
RAM	256 GB
CPU	Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00 GHz
GPU	Tesla P100
Tensorflow	1.9.0
Python	3.6.5

$$X_P^{\mathrm{Real}} = X_p^{[0,1]} * (\mathrm{Max} - \mathrm{Min}) + \mathrm{Min} \tag{19}$$

where X_P^{Real} is the predicted value of the denormalized urban population flow, $X_p^{[0,1]}$ is the normalized predicted value of urban crowed flows, Max, Min are the maximum and minimum values in the dataset, respectively. In order to measure the performance of the model more accurately, we use a two-stage error calculation algorithm to extract the predicted values of 0-value tags and non-zero-value tags and their corresponding positions in the tag data to calculate the error. The error calculation formulas are described as follows:

$$RMSE = \sqrt{\frac{1}{n^0}\sum_{i=1}^{n^0}(x_i^0 - \widehat{x}_i^0)} + \sqrt{\frac{1}{n^{\bar{0}}}\sum_{i=1}^{n^{\bar{0}}}\left(x_i^{\bar{0}} - \widehat{x}_i^{\bar{0}}\right)} \tag{20}$$

$$MAE = \frac{1}{n^0}\sum_{i=1}^{n^0}|x_i^0 - \widehat{x}_i^0| + \frac{1}{n^{\bar{0}}}\sum_{i=1}^{n^{\bar{0}}}|x_i^{\bar{0}} - \widehat{x}_i^{\bar{0}}| \tag{21}$$

6.2. Experimental Results

In order to evaluate the performance of ASTGCN, we conducted various experiments on the GAIA open dataset [9], including the effect of the number of spatio-temporal convolutional components and the effect of different input data lengths. In addition, we also compared with other state-of-art methods.

Table 4 shows the effect of the number of spatio-temporal graph convolutional component. When $lc = 6, ld = 3$, and $lw = 1$, it can be found that the model performance increases first and then decreases with the growing number of spatio-temporal graph convolutional component. When the convolutional kernel of each module is set to 64 and the stack of convolutional network (CN) is 3, ASTGCN achieves optimal performance. MAE is 4.42, which is 51.4% lower than the average of other MAE. RMSE is 6.89, which is 41.3% lower than the average of other RMSE. In addition, we doubled the number of convolutional kernels so that the number of convolutional kernels was 128. There was no obvious abnormality in the model performance change trend, but the performance did not increase significantly. Therefore, the number of convolutional kernels of each module should be set to 64.

We evaluated the effect of different input lengths on ASTGCN. The number of convolutional components of the space–time graph and the number of convolutional kernels are set to 3and 64, respectively. Table 5 and Table 6 show that when $lc = 4$, the optimal performance of the model is obtained. The average MAE is 5.13 and the variance is 2.24. Comparing with $lc = 5$ and $lc = 6$, the average MAE decreases by 20.2% and 37.1% respectively, and the

Table 4
Effect of the number of spatio-temporal graph convolutional component.

	CN = 1		CN = 2		CN = 3		CN = 4		CN = 5		CN = 6	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
f = 64	11.90	18.98	6.04	10.92	4.42	6.89	6.72	11.81	10.71	17.85	10.08	17.02
f = 128	-	-	4.95	9.33	4.49	8.59	4.59	8.63	5.10	9.14	9.15	15.39

Table 5
RMSE with different input lengths on GAIA open dataset.

	ld = 0 lw = 0	*ld = 0 lw = 1*	*ld = 0 lw = 2*	*ld = 1 lw = 0*	*ld = 1 lw = 1*	*ld = 1 lw = 2*	*ld = 2 lw = 0*	*ld = 2 lw = 1*	*ld = 2 lw = 2*	*ld = 3 lw = 0*	*ld = 3 lw = 1*	*ld = 3 lw = 2*
lc = 4	-	7.90	14.90	9.90	6.22	8.67	6.43	5.99	6.36	6.36	9.47	15.30
lc = 5	10.71	6.95	11.53	6.84	15.98	19.20	20.03	8.17	9.26	9.37	19.98	21.36
lc = 6	7.33	6.92	16.32	7.08	21.24	9.75	10.08	9.84	13.51	6.80	6.89	14.99

Table 6
MAE with different input lengths on GAIA open dataset.

	ld = 0 lw = 0	*ld = 0 lw = 1*	*ld = 0 lw = 2*	*ld = 1 lw = 0*	*ld = 1 lw = 1*	*ld = 1 lw = 2*	*ld = 2 lw = 0*	*ld = 2 lw = 1*	*ld = 2 lw = 2*	*ld = 3 lw = 0*	*ld = 3 lw = 1*	*ld = 3 lw = 2*
lc = 4	-	4.70	7.85	4.98	3.99	5.04	4.12	3.91	4.08	3.93	5.42	8.38
lc = 5	5.85	4.35	6.51	4.28	9.30	12.12	13.64	5.00	5.27	5.33	12.52	13.76
lc = 6	4.57	4.39	9.31	4.47	13.92	5.45	5.64	5.45	7.37	4.32	4.42	7.88

Table 7
Comparison of different methods.

Method	*MAE*	RMSE
ASTGCN	**3.91**	**5.99**
HA	9.25	15.65
ARIMA	8.88	12.70
LSTM	7.69	13.44
GLU-STGCN	5.49	9.65

variance decreases by 83.3% and 70.1% respectively. We can find that the change of the length of the daily sampling and the weekly sampling has relatively little effect on the model. The RMSE and MAE increase with *ld* and *lw* without a significant upward or downward trend. However, after enumeration combination, when $ld = 2, lw = 1$, the model performance is the best. Thus, when $lc = 4, ld = 2, lw = 1$ are the input of the model, our model achieves the best performance.

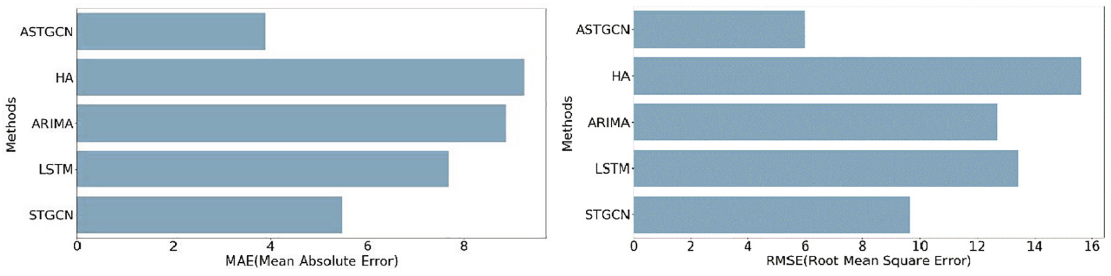

Fig. 13. MAE and RMSE with different methods on GAIA open dataset.

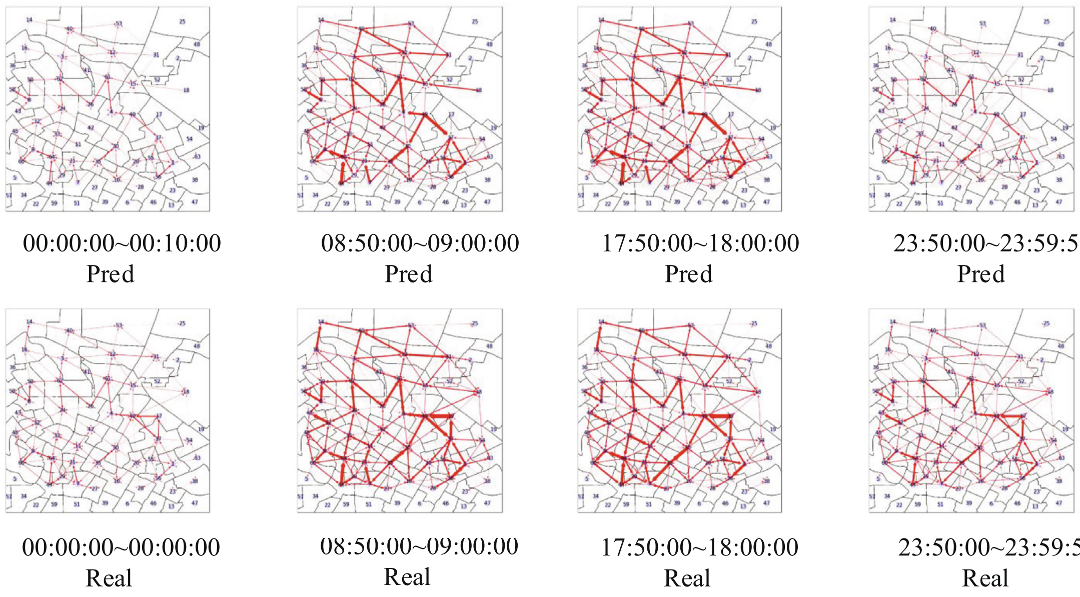

Fig. 14. Prediction VS Ground Truth.

Table 7 shows the comparison of the performance of different methods on the GAIA open dataset. Compared with other baseline methods, the ASTGCN model achieves the best performance in root RMSE and MAE. In order to show intuitively, we draw a comparison chart of different methods, as shown in Fig. 13. It can be obviously seen that ASTGCN is lower than the other four benchmark methods in both MAE and RMSE. MAE is reduced by 28.7% and RMSE is reduced by 37.9%.

In order to show the performance of the ASTGCN algorithm more intuitively, Fig. 14 visualizes the predicted and true values of urban crowd mobility at different times during the day on November 20, 2016. We can find that urban crowd movement would fluctuate greatly with time, urban crowd movement is more active during the day, and crowd movement is relatively rare at night. ASTGCN can achieve good prediction performance for different time periods of the day.

7. Conclusion

This paper proposes a fine-grained predicting urban crowd flows approach with an adaptive spatio-temporal graph convolutional network, called ASTGCN. This approach can predict inflow, outflow, and flow direction. A method for modeling crowd flow in irregular urban regions based on urban bus route data is proposed. Besides, we design an end-to-end structure of adaptive spatio-temporal graph convolutional network based on unique properties of spatio-temporal data. Finally, extensive experiments on GAIA open dataset are built to test the performance of our approach. The experimental comparison find that the model's performance in this paper is better than that of the four baseline models, with the mean absolute error reduction of 28.7% and the root mean square error reduction of 37.9%. In this paper, we only utilize trajectory data of car-hailing. It also leaves a challenging future work that we could leverage other types of flows (e.g., taxi/truck/bus trajectory data, phone signals data, metro card swiping data) to design a multi-source data fusion crowd flows prediction framework.

CRediT authorship contribution statement

Xu Yang: Conceptualization, Methodology, Formal analysis, Writing - original draft. **Qiang Zhu:** Data curation, Writing - original draft. **Peihao Li:** Visualization, Investigation. **Pengpeng Chen:** Supervision. **Qiang Niu:** Writing - review & editing.

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.

Acknowledgement

This work was supported by the Special Funds for Innovation of Graduate Students Double Top University Plan in China University of Mining and Technology (2018ZZCX14)

References

[1] Q. Wang, J. Gao, W. Lin, X. Li, Nwpu-crowd: A large-scale benchmark for crowd counting and localization, IEEE Transactions on Pattern Analysis and Machine Intelligence..
[2] Q. Wang, M. Chen, F. Nie, X. Li, Detecting coherent groups in crowd scenes by multiview clustering, IEEE Trans. Pattern Analysis Mach. Intell. 42 (1) (2018) 46–58.
[3] D. Cheng, F. Nie, J. Sun, Y. Gong, A weight-adaptive laplacian embedding for graph-based clustering, Neural Comput. 29 (7) (2017) 1902–1918.
[4] J. Zhang, Y. Zheng, J. Sun, D. Qi, Flow prediction in spatio-temporal networks based on multitask deep learning, IEEE Trans. Knowl. Data Eng. 32 (3) (2019) 468–478.
[5] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, arXiv preprint arXiv:1610.00081..
[6] J. Sun, J. Zhang, Q. Li, X. Yi, Y. Liang, Y. Zheng, Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks, IEEE Transactions on Knowledge and Data Engineering..
[7] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, T. Li, Predicting citywide crowd flows using deep spatio-temporal residual networks, Artif. Intell. 259 (2018) 147–166.
[8] Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: concepts, methodologies, and applications, ACM Trans. Intell. Syst. Technol. (TIST) 5 (3) (2014) 1–55.
[9] https://outreach.didichuxing.com/research/opendata/..
[10] Z. Fan, X. Song, R. Shibasaki, R. Adachi, Citymomentum: an online approach for crowd behavior prediction at a citywide level, in, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2015, pp. 559–569.
[11] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, A.T. Campbell, Nextplace: a spatio-temporal prediction framework for pervasive systems, in, in: International Conference on Pervasive Computing, Springer, 2011, pp. 152–169.
[12] X. Song, Q. Zhang, Y. Sekimoto, R. Shibasaki, Prediction of human emergency behavior and their mobility following large-scale disaster, in, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 5–14.
[13] Q. Wang, J. Gao, W. Lin, Y. Yuan, Pixel-wise crowd understanding via synthetic data, Int. J. Comput. Vision (2020) 1–21.
[14] A. Abadi, T. Rajabioun, P.A. Ioannou, Traffic flow prediction for road transportation networks with limited traffic data, IEEE Trans. Intell. Transp. Syst. 16 (2) (2014) 653–662.
[15] R. Silva, S.M. Kang, E.M. Airoldi, Predicting traffic volumes and estimating the effects of shocks in massive transportation systems, Proc. Nat. Acad. Sci. 112 (18) (2015) 5643–5648.
[16] Y. Wang, Y. Zheng, Y. Xue, Travel time estimation of a path using sparse trajectories, in, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 25–34.
[17] P.-T. Chen, F. Chen, Z. Qian, Road traffic congestion monitoring in social media with hinge-loss markov random fields, in: 2014 IEEE international conference on data mining, IEEE, 2014, pp. 80–89.
[18] Y. Xu, Q.-J. Kong, R. Klette, Y. Liu, Accurate and interpretable bayesian mars for traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 15 (6) (2014) 2457–2469.
[19] M.X. Hoang, Y. Zheng, A.K. Singh, Fccf: forecasting citywide crowd flows based on big data, in, in: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2016, pp. 1–10.
[20] Y. Li, Y. Zheng, H. Zhang, L. Chen, Traffic prediction in a bike-sharing system, in, in: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2015, pp. 1–10.
[21] 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, arXiv preprint arXiv:1802.08714..
[22] 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, 2019, pp. 5668–5675..
[23] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs (2016) 2014–2023..
[24] M. Zhang, H. Zhang, J. Li, L. Wang, Y. Fang, J. Sun, Supervised graph regularization based cross media retrieval with intra and inter-class correlation, J. Vis. Commun. Image Represent. 58 (2019) 1–11.
[25] D. Shi, L. Zhu, Z. Cheng, Z. Li, H. Zhang, Unsupervised multi-view feature extraction with dynamic graph learning, J. Vis. Commun. Image Represent. 56 (2018) 256–264.
[26] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034..
[27] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907..
[28] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.
[29] R. v. d. Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263..
[30] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: Advances in neural information processing systems, 2018, pp. 4800–4810..
[31] V. Garcia, J. Bruna, Few-shot learning with graph neural networks, arXiv preprint arXiv:1711.04043..
[32] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4558–4567.

[33] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, arXiv preprint arXiv:1506.05163..
[34] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, Adv. Neural Inform. Process. Syst. 28 (2015) 2224–2232.
[35] M. Li, H. Leung, Graph-based approach for 3d human skeletal action recognition, Pattern Recogn. Lett. 87 (87) (2017) 195–202.
[36] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907..
[37] S. Suzuki et al., Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, Image Processing 30 (1) (1985) 32–46.
[38] E. Zhu, Y. Zhang, P. Wen, F. Liu, Fast and stable clustering analysis based on grid-mapping k-means algorithm and new clustering validity index, Neurocomputing 363 (2019) 149–170.
[39] C. Wang, Z. Chen, K. Shang, H. Wu, Label-removed generative adversarial networks incorporating with k-means, Neurocomputing 361 (2019) 126–136.
[40] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inform. Process. Syst. 29 (2016) 3844–3852.
[41] B.M. Williams, L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results, J. Transp. Eng.-asce 129 (6) (2003) 664–672.
[42] J. Cheng, L. Dong, M. Lapata, Long short-term memory-networks for machine reading, arXiv preprint arXiv:1601.06733..
[43] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18, 2018, pp. 3634–3640.

Xu Yang was born in China in 1995. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China, where he is researching the distributed measurement systems, big data and sensor networks.

Qiang Zhu is currently working toward the M.D. degree in the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China, where he is researching the indoor localization and sensor networks.

Peihao Li was born in China in 1994. He is currently working toward the M.D. degree in the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China, where he is researching the indoor localization and sensor networks.

Pengpeng Chen was born in China in 1983. He received the Ph.D. degree in 2011 from the Department of Computer Science and Technology, Ocean University of China, Qingdao, China. He is currently an Professor with the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China. His main research interests are sensor networks, distributed measurement systems, Ocean observation network and data modeling. He has published more than 20 academic papers in important academic journals at home and abroad. He has presided over the National Natural Science Foundation, China Postdoctoral Science Foundation, the State Key Laboratory Open Fund and other projects.

Qiang Niu was born in 1974 in China. He received the Ph.D. degree from the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China. He is currently a Professor and the associate dean with the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China. He is also the deputy director of the Mine Information Engineering Research Center in Chinese Coal Industry. His main research interests are intelligent information processing, artificial intelligence and pattern recognition, machine learning and data mining and other aspects of research. He has published more than 20 academic papers retrieved by SCI.