Predicting Citywide Crowd Flows in Irregular Regions Using Multi-View Graph Convolutional Networks

Junkai Sun, Junbo Zhang, *Member, IEEE*, Qiaofei Li, Xiuwen Yi, Yuxuan Liang, and Yu Zheng, *Senior Member, IEEE*

Abstract—Being able to predict the crowd flows in each and every part of a city, especially in *irregular regions*, is strategically important for traffic control, risk assessment, and public safety. However, it is very challenging because of interactions and spatial correlations between different regions. In addition, it is affected by many factors: i) multiple *temporal correlations* among different time intervals: closeness, period, trend; ii) complex *external* influential factors: weather, events; iii) *meta* features: time of the day, day of the week, and so on. In this paper, we formulate crowd flow forecasting in irregular regions as a *spatio-temporal graph* (STG) prediction problem in which each node represents a region with time-varying flows. By extending *graph convolution* to handle the spatial information, we propose using *spatial graph convolution* to build a *multi-view graph convolutional network* (MVGCN) for the crowd flow forecasting problem, where different views can capture different factors as mentioned above. We evaluate MVGCN using four real-world datasets (taxicabs and bikes) and extensive experimental results show that our approach outperforms the adaptations of state-of-the-art methods. And we have developed a crowd flow forecasting system for irregular regions that can now be used internally.

Index Terms—Multi-view learning, neural network, spatio-temporal prediction

1 Introduction

Forecasting crowd flows in each and every part of a city, especially in *irregular regions*, plays an important role in traffic control, risk assessment, and public safety. For example, when vast amounts of people streamed into a strip region at the 2015 New Year's Eve celebrations in Shanghai, this resulted in a catastrophic stampede that killed 36 people. Such tragedies can be mitigated or prevented by utilizing emergency mechanisms, like sending out warnings or evacuating people in advance, if we can accurately forecast the crowd flow in a region ahead of time.

Prior works mainly focused on predicting the crowd flows in *regular gridded regions* [37], [41], [43]. Although partitioning a city into grids is more easily and effectively handled by the subsequent data mining [46] and machine learning approaches [43], the regions in a city are actually separated by road networks and therefore extremely *irregular*. There are also some existing literature that models the non-euclidean correlation using graph techniques to in the forecasting problems [9], [23], [42]. Different from these previous attempts, our work consists of three tasks: data preprocessing, map segmentation and traffic forecasting. It first takes the raw trajectories and road networks as inputs ans then simultaneously considers multi-view temporal features as well as external views. In this study, our goal is to collectively predict inflow and outflow of crowds in each and every *irregular* region of a city. Fig. 1 shows an illustration. *Inflow* is the total flow of crowds entering a region from other regions during a given time interval and *outflow* denotes the total flow of crowds leaving a region for other regions during a given time interval, both of which track the transition of crowds between regions. Knowing them is very beneficial for traffic control.

We can measure crowd flows by the number of cars/bikes running on the roads, the number of pedestrians, the number of people traveling on public transportation systems (e.g. metro, bus), or all of them together if the data is available. We can use the GPS trajectories of vehicles to measure the traffic flow, showing that the inflow and outflow of v_1 are (0, 2) respectively. Similarly, using mobile phone signals of pedestrians, the two types of flows are (3, 2) respectively.

- *Junkai Sun is with the JD Intelligent Cities Research, Beijing, China, also with the JD Intelligent Cities Business Unit, JD Digits, Beijing 100176, China, and also with the Xidian University, Xi'an 710071, China. E-mail: junkaisun@outlook.com.*
- *Junbo Zhang is with the JD Intelligent Cities Research, Beijing 100176, China, also with the JD Intelligent Cities Business Unit, JD Digits, Beijing 100176, China, and also with the Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China. E-mail: msjunbozhang@outlook.com.*
- *Qiaofei Li is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China. E-mail: qjliqiaofei@gmail.com.*
- *Xiuwen Yi is with the JD Intelligent Cities Research, Beijing 100176, China , and also with the JD Intelligent Cities Business Unit, JD Digits, Beijing 100176, China. E-mail: xiuwenyi@foxmail.com.*
- *Yuxuan Liang is with the School of Computing, National University of Singapore, 119077, Singapore. E-mail: yuxliang@outlook.com.*
- *Yu Zheng is with JD Intelligent Cities Research, Beijing 100176, China, also with the JD Intelligent Cities Business Unit, JD Digits, Beijing 100176, China, also with the Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China, and also affiliated with Xidian University, Xi'an 710071, China. E-mail: msyuzheng@outlook.com.*

Manuscript received 15 Mar. 2019; revised 19 June 2020; accepted 25 June 2020. Date of publication 13 July 2020; date of current version 1 Apr. 2022. Corresponding authors: Junbo Zhang and Yu Zheng. Recommended for acceptance by L. Xiong. Digital Object Identifier no. 10.1109/TKDE.2020.3008774

(a) Inflow and outflow (b) Measurement of flows

Fig. 1. Crowd flows in an irregular region.

We formulate the crowd flow forecasting problem as a spatio-temporal graph (STG) prediction problem in which an *irregular region* can be viewed as a *node* that is associated with time-varying inflow and outflow, and transition flow between regions can be used to construct the edges. However, forecasting these two kinds of flows in each and every node of an STG is very challenging because of the following three complex factors:

1) *Interactions and Spatial Correlations Between Different Vertices of an STG*. The inflow of the node v_1 (Fig. 1b) is affected by outflows of *adjacent* (1-hop) neighbors (v_2 and v_3) as well as *multi-hop* neighbors (other nodes, like v_4). Likewise, a node's outflow would affect its neighbors' inflows. Moreover, a node's inflow and outflow interact with each other.
2) *Multiple Types of Temporal Correlations Among Different Time Intervals*. closeness, period, and trend. i) *Closeness*: the flows of a node are affected by recent time intervals. Taking traffic flow as an example, a congestion occurring at 5pm will affect traffic flow at 6pm. ii) Many types of *Periods*: daily, weekly, etc. Traffic conditions during rush hours may be similar on consecutive workdays (daily period) and consecutive weekends (weekly period). iii) Many types of *Trends*: monthly, quarterly, etc. Morning peak hours may gradually happen earlier as summer comes, with people getting up earlier as the temperature gradually increases and the sun rises earlier.
3) *Complex External Factors and Meta Features*. Holidays can influence the flow of crowds for consecutive days, and extreme weather always changes the crowd flows tremendously in different regions of a city. Besides, crowd flows are also affected by meta data, like time of day, weekend/weekday. For example, the flow patterns on rush hours may differ from non-rush hours.

To tackle all aforementioned challenges, we propose a general multi-view learning framework for crowd flow prediction in all the irregular regions of a city, as shown in Fig. 2. The framework is composed of two stages: data preparation and model learning. The data preparation stage involves fetching global information based on the target time and selecting the dependent crowd flow matrices from key timesteps according to different temporal properties. Based on the collected multiple view data, we present a new model for learning, which we refer to as a *multi-view graph convolutional network* (MVGCN), consisting of several GCNs and fully-connected neural networks (FNNs). The contributions of this research lie in the following four aspects:

- We propose a variant of GCN, which can capture spatial correlations between different nodes. We design a multi-view fusion module, to fuse multiple latent representations from different views.The module is designed based on two fusion methods: gating and sum fusion, which are used to capture sudden and slight changes, respectively.
- We propose a comprehensive framework that consists of data preprocessing, map segmentation and map clustering by road networks, graph construction via transition flows, crowd flow prediction using graph convolutional networks. Besides, we also design a demo system to visualize the crowd flow forecasting results in citywide irregular regions.
- We evaluate our MVGCN using four real-world mobility datasets, including taxicab data in Beijing and New York City (NYC), and bike data in NYC and Washington D.C. The extensive results demonstrate advantages of our MVGCN beyond the adaptations of several state-of-the-art approaches,

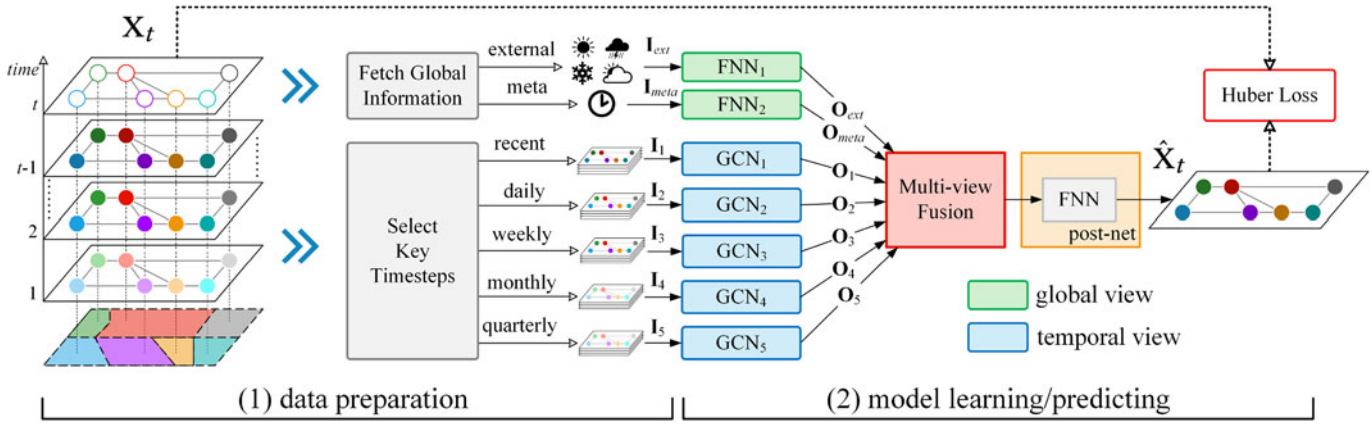


Fig. 2. Multi-view deep learning framework. (1) Data preparation stage: fetching global information based on the predicted target time and selecting key timesteps based on temporal dependencies. (2) Model learning stage: a) Graph convolutional net (GCN) is used to learn the spatial correlations and interactions using the structural information of the STG; b) Fully-connected nueral net (FNN) is employed to capture global information, like external factors and meta features (time of the day). c) Multi-view fusion can effectively integrate the outputs of GCNs and FNNs. d) Post-net, namely a FNN here, is used to project the latent representation to the output using an activation function (*e.g.* $\tanh$).

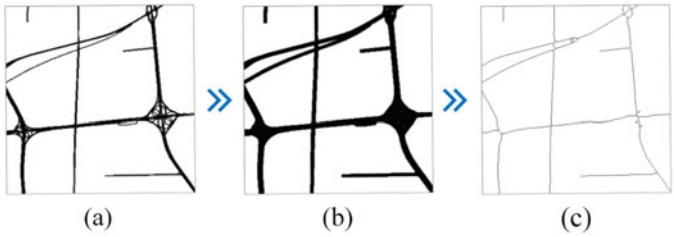


Fig. 3. (a) Before dilation; (b) After dilation; (c) After thinning.

like diffusion convolutional recurrent neural networks [22] and Gaussian Markov random field based model [13].

2 Problem Definition

2.1 Irregular Regions

Urban areas are naturally divided into different irregular regions by road network. These regions may have different functions, such as education and business function [39]. Different functional areas usually have different traffic flow patterns. For example, most people usually commute from residential areas to work places in the morning and return home after work. So it is actually more rational and insightful to perform the task of traffic flow prediction on these irregular regions.

Region Partition. The task of *region partition* consists of two main operations: map segmentation and map clustering. For example, the road network in Beijing is composed of multi-level roads, such as level 0, 1, 2, etc., which represent different functional road categories. As shown in Fig. 4a, the red segments denote highways and city express ways, and the blue segments represent urban arterial roads in Beijing.

Referring to [40], we utilize morphological image processing techniques to tackle the region partition task. Specifically, we partition the map into 2400×2400 small grid-cells, and map each road point to its corresponding grid-cells, thereby obtaining a binary image, in which 1 and 0 stand for road segments and blank areas respectively. Then we apply the dilation operation and thinning operation to get the skeleton of the road network. The dilation operation can help thicken the roads, fill the small holes and smooth out unnecessary details. Then the thinning operation is used to recover the size of a region while keeping the connectivity between region, as shown in Fig. 3. Finally, we can obtain all labeled irregular regions' locations using the connected component labeling algorithm (CCL) that finds individual regions by clustering "1"-labeled grids.

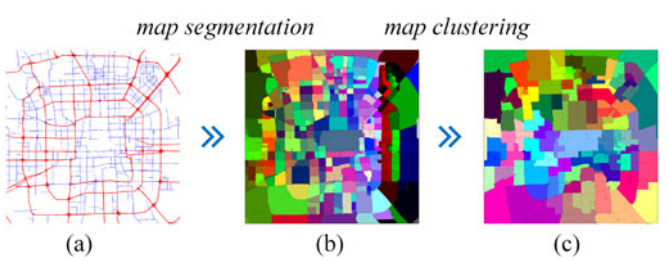


Fig. 4. (a) Road network in Beijing; (b) Regions after map segmentation; (c) Regions after map clustering.

TABLE 1
Notation

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	spatio-temporal graph
$\mathcal{V} = \{v_i\}$	a set of N nodes, $i = 1, \ldots, N$
$\mathbf{A} \in \mathbb{R}^{N \times N}$	an adjacency matrix
$\mathbf{S} \in \mathbb{R}^{N \times N}$	a modified adjacency matrix
$P = \{p_i\}$	geospatial position of node v_i
$\mathcal{T}$	available time interval set
$\mathbf{X}_t \in \mathbb{R}^{N \times C}$	a matrix of node feature vectors at $t \in \mathcal{T}$
$\mathbf{X}_t[i, :]$	vector of node n_i
$\mathbf{X}_t[:, c]$	vector of c-th channel in all nodes

After the map segmentation, we obtain a large number of low-level irregular regions, and many of them are too small to collect or predict traffic flows at the city scale. Therefore, we apply a clustering operation [16] on these regions. Specifically, we define the edge weight between two low-level regions as the Spearman's rank correlation coefficient between the average crowd flows within a time period (e.g. one day). After this operation, the small intractable regions are clustered into some high-level regions, as shown in Figs. 4b and 4c. *Graph Construction*. To capture the spatial dependency of traffic flow between different irregular regions, we construct a topological graph using historical region-wise transition flow. The intuition is that adjacent regions in geo-space are usually closely correlated, besides that, regions that are distant can also influence each other due to the convenient transportation such as subway, taxi and so on. Transition flow can reflect the traffic interaction between close or distant regions. Specifically, we select a period of time from the traffic data, such as one or two months. Then we can statistic the valid time slices between pair-wise regions. Valid time slice means when the region-wise transition flow is greater than a threshold α considering the noise of trajectories data. When the ratio of valid time slices for region-wise transition is greater than a threshold β, we place an binary value undirected edge to connect them. In our paper, the thresholds α is set as 3, β is set as 0.1.

2.2 Prediction Problem on Spatio-Temporal Graphs

The goal in this research is to collectively predict the future inflows/outflows in each and every node of an STG based on historical observations. Table 1 lists the mathematical notation used in the paper.

Definition 1 (STG). *A spatio-temporal graph (STG), denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ and $\mathcal{E}$ respectively denote the set of $|\mathcal{V}| = N$ vertices and edges, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a binary unweighted adjacency matrix. Specifically, each vertex $v_i \in \mathcal{V}$ has a geospatial position p_i and time-varying attributes. These attributes over an STG at time t can be viewed as a graph signal $\mathbf{X}_t \in \mathbb{R}^{N \times C}$, where $\mathbf{X}_t[i, :] \in \mathbb{R}^C$ represents C attributes in the node v_i, e.g., the inflow and outflow [43] ($C = 2$). The edges between two regions are constructed from region-wise transition flows and the binary entry value in $\mathbf{A}$ indicates whether two regions are correlated in traffic flow.*

Problem 1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ and observed attributes of nodes $\{\mathbf{X}_t | t = 1, 2, \ldots, T\}$, predict the attributes at the next time step*, i.e., $\mathbf{X}_{T+1}$.

3 METHODOLOGY

In this section, we present our new model for crowd flow forecasting. We first present a multi-view deep learning framework [35], [45], then we review the graph convolutional network and present our new spatial graph convolutional network. Finally, we present the multi-view fusion method and loss function used in our model.

3.1 Multi-View Deep Learning Framework

Fig. 2 provides an overview of our proposed deep learning framework to predict the crowd flows in an STG. We adopt the multi-view framework that is an effective mechanism to learn latent representations from cross domain data [32]. The framework proposed is composed of two stages: data preparation and model learning/predicting. The first stage is used to fetch global information and select the key timesteps, then we feed all of them to the second stage to perform model training. We provide concrete details in the following sections.

Data Preparation Stage. "What factors should be considered when forecasting the crowd flow in a region?" a) weather, b) time of the day, c) period, etc. Different people may have different answers that highlights different views on this problem. We summarize these views into two categories: *global view* and *temporal view*. (1) the global view is composed of external and meta views. According to the time of the predicted target, we fetch different external data, like meteorological data in previous timesteps and weather forecasting. We can also construct the meta features: time of the day, day of the week, and so on. The external and meta features are represented as $\mathbf{I}_{ext}$ and $\mathbf{I}_{meta}$, respectively. (2) the temporal view contains multiple views according to the temporal closeness, period, trend. Considering two types of periods (daily and weekly), and two types of trends (monthly and quarterly),[1] we select the corresponding recent, daily, weekly, monthly, and quarterly timesteps as the key timesteps, to construct five views. For each of the different temporal views, we fetch a list of key timesteps' flow matrices and concatenated them, to construct five inputs as follows,

$$\begin{aligned}
\mathbf{I}_1 &= \text{concat}[\mathbf{X}_{t-1}, \mathbf{X}_{t-2}, \ldots, \mathbf{X}_{t-l_r}] \in \mathbb{R}^{N\times C\times l_r} \\
\mathbf{I}_2 &= \text{concat}[\mathbf{X}_{t-p_d}, \mathbf{X}_{t-2p_d}, \ldots, \mathbf{X}_{t-l_d * p_d}] \in \mathbb{R}^{N\times C\times l_d} \\
\mathbf{I}_3 &= \text{concat}[\mathbf{X}_{t-p_w}, \mathbf{X}_{t-2p_w}, \ldots, \mathbf{X}_{t-l_w * p_w}] \in \mathbb{R}^{N\times C\times l_w} \\
\mathbf{I}_4 &= \text{concat}[\mathbf{X}_{t-p_m}, \mathbf{X}_{t-2p_m}, \ldots, \mathbf{X}_{t-l_m * p_m}] \in \mathbb{R}^{N\times C\times l_m} \\
\mathbf{I}_5 &= \text{concat}[\mathbf{X}_{t-p_q}, \mathbf{X}_{t-2p_q}, \ldots, \mathbf{X}_{t-l_q * p_q}] \in \mathbb{R}^{N\times C\times l_q},
\end{aligned}$$

where l_r, l_d, l_w, l_m, and l_q are input lengths of recent, daily, weekly, monthly, and quarterly lists, respectively. p_d and p_w are daily and weekly periods; p_m and p_q are monthly and quarterly trend spans.

By selecting these key timesteps, our approach can capture multiple types of temporal properties. The complexity of the input data of our approach is $l_r + l_d + l_w + l_m + l_q$, and these views can be modeled in parallel. If one uses a sequence neural network model (like recurrent neural networks, RNNs) to capture all these temporal dependencies automatically, the complexity would be $O(\max(l_r, l_d * p_d, l_w * p_w,\ l_m * p_m,\ l_q * p_q)) = O(l_q * p_q)$, while RNNs maintain a hidden state of the entire past that prevents parallel computation within a sequence. Assuming lengths of recent, daily, weekly, monthly, and quarterly lists are all equal 3, our architecture only needs $3 \times 5 = 15$ key frames. In contrast, RNNs needs 3 quarters of data, approximately $24\ \text{frames/day} \times\ 30\ \text{days/month} \times 3\ \text{months/quarter} \times 3\ \text{quarters} = 6480$ frames. Such a long-range sequence tremendously raises the training complexity for RNNs, making them infeasible in real-world applications.

Model Learning/Predicting Stage. We employ graph convolutional networks (GCNs, see Section 3.2) and fully-connected neural networks (FNNs) to model the temporal and global views, respectively. For each temporal view, GCN is used to learn the time-varying spatial correlations and interactions using the structural information of the STG, and The corresponding outputs of five GCNs are denoted $\mathbf{O}_1, \ldots, \mathbf{O}_5 \in \mathbb{R}^{N\times C}$. Two FNNs are employed to capture the influences from external and meta data, respectively, and the outputs are $\mathbf{O}_{ext}$ and $\mathbf{O}_{meta}$. All these outputs are then fed into the *multi-view fusion module* (see Section 3.3) followed by a post-net (e.g. FNN), to obtain the final prediction $\hat{\mathbf{X}}_t$. The multi-view fusion can effectively employ the outputs of different views based on their characteristics. Finally, we apply the Huber loss [15] for robust regression.

3.2 Graph Convolutional Network for STG

Convolutional Networks Over Graphs. Recently, generalizing convolutional networks to graphs have become an area of interest. In this paper, we mainly consider spectral convolutions [3], [7] on arbitrary graphs. As it is difficult to express a meaningful translation operator in the node domain [3], [7] presented a spectral formulation for the convolution operator on the graph, denoted as $*_{\mathcal{G}}$. By this definition, the graph signal $\mathbf{X} \in \mathbb{R}^{N\times C}$ with a filter $g_{\mathbf{w}} = \text{diag}(\mathbf{w})$ parameterized by $\mathbf{w} \in \mathbb{R}^N$ in the Fourier domain,

$$g_{\mathbf{w}} *_{\mathcal{G}} \mathbf{X} = g_{\mathbf{w}}(\mathbf{L})\mathbf{X} = g_{\mathbf{w}}(\mathbf{U}\Lambda\mathbf{U}^\top)\mathbf{X}, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{N\times N}$ is the matrix of eigenvectors, and $\Lambda \in \mathbb{R}^{N\times N}$ is the diagonal matrix of eigenvalues of the normalized graph Laplacian $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\Lambda\mathbf{U}^\top \in \mathbb{R}^{N\times N}$, where $\mathbf{I}_N$ is the identity matrix and $\mathbf{D} \in \mathbb{R}^{N\times N}$ is the diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. We can understand $g_{\mathbf{w}}$ as a function of the eigenvalues of $\mathbf{L}$. However, evaluating Eq. 1 is computationally expensive, as the multiplication with $\mathbf{U}$ is $\mathcal{O}(N^2)$. To circumvent this problem, the Chebyshev polynomial expansion (up to K^{th} order) [7] was applied to obtain an efficient approximation, as

$$g_{\mathbf{w}}(\mathbf{L})\mathbf{X} \approx \sum_{k=0}^{K-1} \mathbf{w}_k(\mathbf{L}^k)\mathbf{X} = \sum_{k=0}^{K-1} \mathbf{w}'_k T_k(\tilde{\mathbf{L}})\mathbf{X}, \tag{2}$$

where $T_k(\tilde{\mathbf{L}})$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}}\mathbf{L} - \mathbf{I}_N$, $\lambda_{\max}$ denotes the largest eigenvalue of $\mathbf{L}$, $\mathbf{w}' \in \mathbb{R}^{\mathbf{K}}$ is now a vector of Chebyshev coefficients. The details of this approximation can be found in [7], [11].

Furthermore, [18] proposed a fast approximation of the spectral filter by setting $K = 1$ and successfully used it for

1. One can set different periods and trends in practice, like yearly period, based on the characteristics of the data.

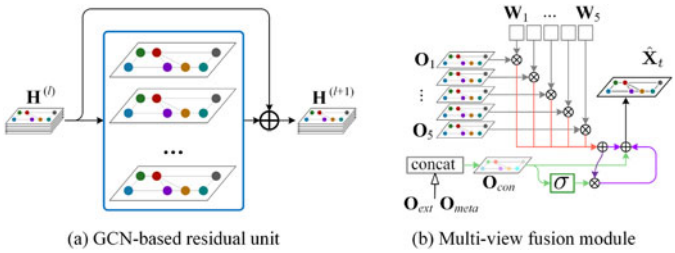


Fig. 5. Main components of MVGCN .

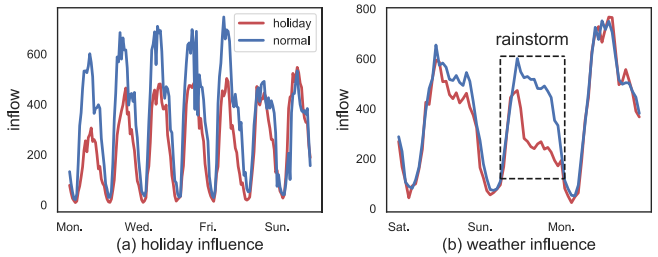


Fig. 6. Different influences with external factors. (a) 2016 Chinese Spring Festival. (b) A stormy day versus sunny day. The data is collected from TaxiBJ, as shown in Table 2.

semi-supervised classification of nodes, as

$$\mathbf{Y} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}, \tag{3}$$

where $\mathbf{Y} \in \mathbb{R}^{N\times F}$ is the signal convolved matrix. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix of $\mathcal{G}$ with added self-connections, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ and $\mathbf{W} \in \mathbb{R}^{C\times F}$ is a trainable matrix of filter parameters in a graph convolutional layer. The filtering operation has complexity $\mathcal{O}(|\mathcal{E}|FC)$ as $\tilde{\mathbf{A}}\mathbf{X}$ [18] and can be efficiently implemented as the product of a sparse matrix with a dense matrix.

Spatial Graph Convolutional Network. We present a variant of fast approximate graph convolution (Eq. (3)) that also considers the geospatial positions of vectices in an STG. Here we explore an approach to integrate such geospatial positions based on the First Law of Geography [30], *i.e.*, everything is related to everything else, but near things are more related than distant things.

Given an adjacency matrix $\mathbf{A}$, we assign spatial weights for existing edges based on the spatial distance, as

$$\mathbf{S} = \mathbf{A} \odot \omega, \tag{4}$$

where $\mathbf{S} \in \mathbb{R}^{N\times N}$ is the modified adjacency matrix, $\odot$ is the Hadamard product (*i.e.* element-wise multiplication). $\omega \in \mathbb{R}^{N\times N}$ is the spatial weighted adjacency matrix that is calculated via a thresholded Gaussian kernel weighting function [24], as

$$\omega_{ij} = \begin{cases} \exp\left(-\frac{[dist(p_i,p_j)]^2}{2\theta^2}\right) & \text{if } dist(p_i,p_j) \le \kappa \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

Here $dist(p_i, p_j)$ means the geographical distance between nodes v_i and v_j; θ and κ are two parameters to control the scale and sparsity of the adjacency matrix. With the modified matrix $\mathbf{S}$, we consider multiple graph convolutional layers with the following layer-wise propagation rule:

$$\mathbf{H}^{(l+1)} = f\left(\mathbf{Q}^{-\frac{1}{2}}\tilde{\mathbf{S}}\mathbf{Q}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{6}$$

where $\mathbf{H}^{(l+1)} \in \mathbb{R}^{N\times F_{l+1}}$ and $\mathbf{H}^{(l)} \in \mathbb{R}^{N\times F_l}$ are the output and input of the l^{th} layer. $\tilde{\mathbf{S}} = \mathbf{S} + \mathbf{I}_N$ is the modified adjacency matrix with added self-connections, $\mathbf{Q}_{ii} = \sum_j \mathbf{S}_{ij}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{F_l\times F_{l+1}}$ is a trainable matrix of filter parameters in a graph convolutional layer, f denotes an activation function, e.g. the rectifier $f(z) := \max(0, z)$ [19]. The filtering operation has complexity $\mathcal{O}(|\mathcal{E}|F_lF_{l+1})$ as $\tilde{\mathbf{S}}\mathbf{H}^{(l)}$ can be efficiently implemented as a product of a sparse matrix with a dense matrix.

GCN-Based Residual Unit. To capture M-hop spatial correlations and interactions, we stack M spatial graph convolutional layers, inspired by graph convolutions [18]. When M is large, we need a very deep network. Residual learning [12] allows neural networks to have a super deep structure of 100 layers. Here we propose a GCN-based residual unit that integrates the graph convolutional layer into the residual framework (Fig. 5a). Formally, the residual unit is defined as:

$$\mathbf{H}^{(l+1)} = \mathbf{H}^{(l)} + f\left(\mathbf{Q}^{-\frac{1}{2}}\tilde{\mathbf{S}}\mathbf{Q}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{7}$$

where f is an activation function.

By stacking multiple GCN-based residual units, we can build very deep neural networks to capture multi-hop spatial dependencies.

3.3 Multi-View Fusion

We propose a *multi-view fusion* (see Fig. 5b) method to fuse the latent representations of many flow views with two global views (external and meta data). In our previous crowd flow prediction task [43], we show that different regions have different temporal properties, but the degrees of influence may be different. Inspired by this, we here also employ the parametric-matrix-based fusion method [43] to fuse the outputs of five GCNs for temporal views as below

$$\mathbf{O} = \mathbf{W}_1 \odot \mathbf{O}_1 + \mathbf{W}_2 \odot \mathbf{O}_2 + \ldots + \mathbf{W}_5 \odot \mathbf{O}_5 \tag{8}$$

where $\mathbf{W}_1, \ldots, \mathbf{W}_5$ are the learnable parameters that adjust the degrees affected by closeness, daily period, weekly period, monthly trend, and quarterly trend, respectively.

For the external factor $\mathbf{I}_{ext}$ (like weather and holiday) and meta data $\mathbf{I}_{meta}$ (e.g. time of the day), we separately feed them into different fully-connected (FC) layers to obtain different latent representations $\mathbf{O}_{ext}$ and $\mathbf{O}_{meta}$. Then we simply concatenate all the outputs of the *embed* module and add a FC layer following by reshaping, thereby obtaining $\mathbf{O}_{con} \in \mathbb{R}^{N\times C}$.

Different factors may change the flows in different ways. For example, holidays may moderate the crowd flows, as shown in Fig. 6a, while rainstorms may sharply and dramatically reduce the flows (Fig. 6b). Specifically, the latter is just like a switch, changing flows tremendously change when it happens. On account of these insights, we leverage two different fusion methods to deal with these two types of situations. For the gradual changes, we propose employing a sum-fusion method, e.g., $\mathbf{O}_{con}$ + $\mathbf{O}$. For the sudden changes, we propose employing a gating-mechanism-based

fusion, e.g., $\sigma(\mathbf{O}_{con}) \odot \mathbf{O}$, where σ is an approximated gating function such as $sigmoid$. When the concatenated representation of $\mathbf{O}_{con}$ captures some special external information such as rainstorm weather, the term $\sigma(\mathbf{O}_{con}) \odot \mathbf{O}$ will suddenly increase and become a much larger value due to the property of sigmoid function compared with $\mathbf{O}_{con}$. And in most common cases, this term should be close to zero without sudden changes.

Based on two fusion methods, the final output is calculated as

$$\hat{\mathbf{X}}_t = f_o(\mathbf{O} + \mathbf{O}_{con} + \sigma(\mathbf{O}_{con}) \odot \mathbf{O}), \tag{9}$$

where f_o is the activation function, e.g., $\tanh$, $sigmoid$.

3.4 Loss and Algorithm

Let x and $\hat{x}$ be the observed and predicted values. The objective function we employ here is the Huber loss, which is an elegant compromise between squared-error loss $(x - \hat{x})^2$ and absolute-error loss $|x - \hat{x}|$, and has been verified as a robust loss function for regression [15].

The Huber loss, denoted $\mathcal{L}(x, \hat{x})$, is defined by

$$\mathcal{L}(x, \hat{x}) = \begin{cases} \frac{1}{2}(x - \hat{x})^2 & for\ |x - \hat{x}| \le \delta, \\ \delta|x - \hat{x}| - \frac{1}{2}\delta^2, & \text{otherwise}, \end{cases} \tag{10}$$

where δ is a threshold (1 by default). The Huber loss combines the desirable properties of squared-error loss near zero and absolute error loss when $|x - \hat{x}|$ is greater than δ (Fig. 12 shows the empirical comparison).

Let Θ be all the trainable parameters in MVGCN. For the Huber loss it yields the following optimization problem,

$$\arg\min_{\Theta} \sum_{t \in \mathcal{T}} \sum_{i=1}^{N} \sum_{c=1}^{C} \mathcal{L}(\mathbf{X}_t[i, c], \hat{\mathbf{X}}_t[i, c]), \tag{11}$$

where $\mathbf{X}_t[i, c]$ means the element of the i^{th} row and c^{th} column of $\mathbf{X}_t$.

4 Experiments

4.1 Settings

Datasets. We use four different datasets as shown in Table 2. The details are described as follows:

TaxiNYC[2]: The trajectory data is taxi GPS data for New York City (NYC) from 1st Jan. 2011 to 30th Jun. 2016. We partition NYC into 100 irregular regions based on the map segmentation method (Section 2.1), and build the graph according to transition flow and geographical distance between regions, then we calculate crowd flows like [13].

TaxiBJ: Trajectory data is the taxicab GPS data in Beijing from four time intervals: 1st Jul. 2013-30th Oct. 2013, 1st Mar. 2014-30th Jun. 2014, 1st Mar. 2015-30th Jun. 2015, 1st Nov. 2015-10th Apr. 2016. The graph construction and crowd flow calculation method in Beijing is the same as that of NYC.

BikeDC[3]: The data is taken from the Washington D.C. Bike System. Trip data includes: trip duration, start and end station IDs, start and end times. There are 472 stations in total. For each station, we get two types of flows, where the inflow is the number of checked-in bikes, and the outflow is the number of checked-out bikes. Since many stations have no data or very few records, we remove these stations and apply a cluster operation [16] to the remaining stations using the average flow of historical observations, to get 120 irregular regions. We construct the graph with transition flow and geographical distance between these regions.

TABLE 2
Datasets. Holidays Include Adjacent Weekends. WS: Wind Speed. Temp.: Temperature

Dataset	TaxiNYC	TaxiBJ	BikeDC	BikeNYC
Data type	Taxi trip	Taxi GPS	Bike rent	Bike rent
Location	NYC	Beijing	D.C.	NYC
Start time	1/1/2011	7/1/2013	1/1/2011	7/1/2013
End time	6/30/ 2016	4/10/ 2016	12/31/ 2016	12/31/ 2016
Time interval	1 hour	1 hour	1 hour	1 hour
# timesteps	48192	12336	52608	30720
# regions (stations)	100	100	120 (472)	120 (416)
# holidays	627	105	686	401
Weather	\	16 types	\	\
Temp. / °C	\	[-24.6,41]	\	\
WS / mph	\	[0,48.6]	\	\

BikeNYC[4]: The data is taken from the NYC Bike system from 1st Jul. 2013 to 31th Dec. 2016. There are 416 stations in total. We also remove unavailable bike stations, and cluster the remaining stations into 120 regions. The graph construction and the bike flow calculation method in NYC is same as that of DC.

For all aforementioned four datasets, we choose data from the last four weeks as the *test set*, all data before that as the *training set*. We build the commuting network (i.e. graph) via the geographical distance between stations or regions, which can be viewed as nodes in the graph. The stations each have geospatial positions. For the regions, we approximate using the geospatial position of the central location of the region.

Baselines. We compare MVGCN with the following 9 models:

- *HA*: *Historical Average*, which models crowd flows as a seasonal process, and uses the average of previous seasons as the prediction with a period of one week. For example, the prediction for this Tuesday is the averaged crowd flows from all historical Tuesdays.
- *VAR*: *Vector Auto-Regressive* is a more advanced spatio-temporal model, which is implemented using the *statsmodel* python package.[5] The number of lags is set as 3, 5, 10, or 30. The best result is reported.
- *GBRT*: *Gradient Boosting Decision Tree* [8]. It uses the same features as the input of ANN. The optimal parameters are achieved by the grid search.
- *FC-LSTM*: Encoder-decoder framework using LSTM [28]. Both encoder and decoder have two recurrent layers with 128 or 64 LSTM units.

2. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
3. https://www.capitalbikeshare.com/system-data
4. https://www.citibikenyc.com/system-data
5. http://www.statsmodels.org

TABLE 3
Comparisons With Baselines on Four Datasets Based Two Metrics

Dataset	Metric	HA	VAR	GBRT	FC-LSTM	GCN	DCRNN	FCCFnoTrans	FCCF	ST-MGCN	MVGCN
TaxiNYC	RMSE	101.54	30.78	83.71	27.82	26.52	25.50	26.02	26.00	23.53	**23.15**
	MAE	33.02	11.21	23.46	11.25	11.12	11.20	9.25	**9.24**	9.52	9.40
TaxiBJ	RMSE	38.77	18.79	33.89	19.04	17.38	16.44	18.70	18.42	16.30	**14.37**
	MAE	22.89	11.38	20.34	11.86	10.60	9.68	10.74	10.44	10.18	**9.11**
BikeDC	RMSE	2.61	1.95	3.46	1.88	1.88	1.90	2.22	2.14	-	**1.72**
	MAE	1.48	1.20	1.98	1.10	1.08	1.20	1.34	1.27	-	**1.00**
BikeNYC	RMSE	6.77	4.21	8.57	4.66	5.06	4.35	4.41	4.19	-	**4.15**
	MAE	4.00	2.71	5.17	2.78	2.85	2.90	2.79	2.65	-	**2.60**

RMSE and MAE (the Smaller the Better). HA and VAR are time-series models; GBRT use the spatial and temporal features; FC-LSTM/GCN/DCRNN/ST-MGCN are neural networks. FCCF/FCCFnoTran are based on Gaussian Markov random fields.

- *GCN*: We build a 3-layer supervised *graph convolutional network* where the graph convolution [18] is employed. The inputs are the previous 6 timesteps and the output is the target timestep.
- *DCRNN*: We build a 2-layer supervised *diffusion convolutional recurrent neural network* [22], which achieves state-of-the-art results on predicting traffic speed on roads. The inputs are the previous 6 timesteps and the output is the target timestep or timesteps.
- *FCCF*: Forecasting Citywide Crowd Flow model based on Gaussian Markov random fields [13], that leverages flows in all individual regions and transitions between regions as well as external factors. As other baselines did not use the transition features, we remove the transition to get a new baseline, named *FCCFnoTrans*.
- *ST-MGCN*: Forecasting ride-hailing demand with spatiotemporal multi-graph convolution network. [9]. We reproduce the model referring the paper and using the recommended model settings in the paper.

The neural network based models are implemented using TensorFlow and trained via backpropagation and Adam [17] optimization.

Preprocessing. The Min-Max normalization method is used to scale the data into the range $[-1, 1]$ or [0,1]. In the evaluation, we re-scale the predicted value back to the normal values, and compare them with ground truth data. For external factors, we use one-hot encoding to transform metadata (*i.e.*, the day of the week, the time of the day), holidays and weather conditions into binary vectors, and use Min-Max normalization to scale the Temperature and Wind speed into the range $[0, 1]$.

Environmental Settings & Hyperparameters. Our model as well as most baselines are implemented using TensorFlow and the model training process is performed on two Tesla V100 GPUs with 64GB RAM and 16GB GPU memory. The training time varies from 30 minutes to 3 hours on different datasets. The detailed hyperparameter settings about our model are as follows: (1) For lengths of the five dependent sequences, we set them as: l_r, l_d, l_w, l_m, $l_q \in \{0, \ldots, 6\}$. (2) The number of graph convolutional layers is set as $\{3, \ldots, 7\}$, no regularization is used. (3) The hidden unit is set as 10 for each embed layer by default. (4) The training data is split into three parts: the last four weeks' data is used as the test set, adjacent previous four weeks' data is used as validation set and the rest of the data is used to train the models. The validation set is used to control the training process by early stopping and choose our final model parameters for each model based on the best validation score. (5) The batch size is set as 32. (6) The learning rate is set as 0.0003. (7) The training epoch is set as 1000, early stopping patience is set as 50. For all trained models, we only select the model which has the best score on the validation set, and evaluate it on the test set.

Evaluation Metrics. For the evaluation of ST-prediction, we employ two metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), both of which are widely used in the regression tasks. Given predicted values $\{\hat{x}_i\}$ and ground-truth values $\{x_i\}$, the RMSE and MAE are respectively calculated as below

$$RMSE = \sqrt{\frac{1}{N}\sum_i (x_i - \hat{x}_i)^2}, \quad MAE = \frac{1}{N}\sum_i |x_i - \hat{x}_i|,$$

where N is the total number of all predicted values.

4.2 Comprehensive Results

Table 3 presents a comprehensive comparison with all 9 baselines. In general, it indicates that our MVGCN performs best on all datasets based on two metrics except MAE on TaxiNYC. Comparing our MVGCN with the state-of-the-art model ST-MGCN, both exploit the graph convolution technique. But our model aims to model different data views using the same graph, but ST-MGCN attempts to build multiple semantic graphs to make more accurate predictions. Due to the lack of graph data for ST-MGCN for bike datasets, we only report the performances of ST-MGCN on taxi datasets. We can find that our model performs better than ST-MGCN in two datasets on RMSE and MAE metrics.

Among four datasets, we can observe that our MVGCN achieves the greatest improvement on the dataset TaxiBJ. This is because the TaxiBJ dataset contains more external information, like weather, temperature, and wind speed. We find that FCCF performs very well because it also considers the period and trend as well as external information, even the transitions between regions. When transition features are removed, FCCF is degraded into the model FCCFnoTrans, resulting in a

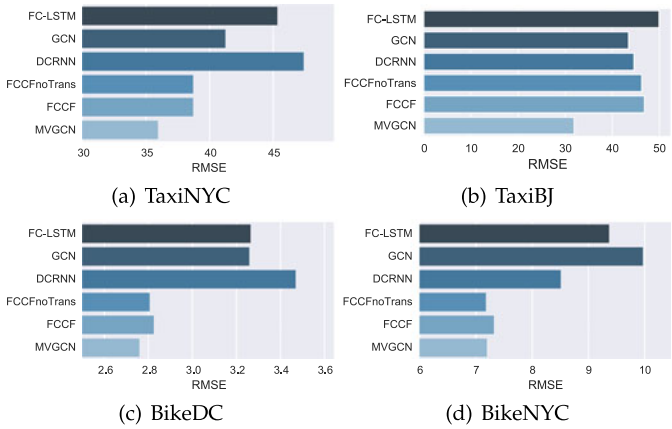


Fig. 7. RMSE comparisons on *sudden changes* in the four datasets.

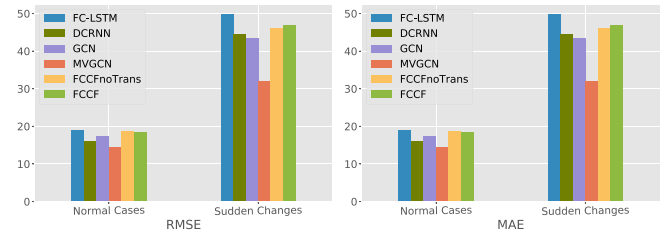


Fig. 8. Performance comparison on normal cases and sudden changes in TaxiBJ dataset.

small increase in both RMSE and MAE, which shows the effectiveness of transition features. FC-LSTM and DCRNN perform worse than FCCF and MVGCN because they are used to model sequences and do not consider period and trend in the crowd flow data.

4.2.1 Results on Sudden Changes

Fig. 7 presents the comparisons between MVGCN and the five baselines on sudden changes cases, which may be caused by anomalous weather or traffic events. For each timeslot t, we calculate the traffic flow difference with previous timeslot $t-1$ of all regions. Then we sort the absolute values of all differences in descending order and define the top 5 percent as the timeslots where sudden changes happen. And the left 95 percent timeslots are as normal cases. We obverse that our MVGCN greatly outperforms all other baselines, especially on TaxiBJ. As shown in Fig. 8, our model performs better than baselines on both normal cases and sudden changes, besides, achieves more improvements on the latter. One reason may be that our MVGCN can effectively model weather data that is more complete in TaxiBJ.

4.2.2 Results on Multi-Step Prediction

For further analysis, we present the multi-step prediction results based on RMSE and MAE over the dataset BikeDC in Fig. 9. For the single-step prediction models, e.g., our MVGCN, we train different models for different timesteps. For the multi-step prediction models, including FC-LSTM and DCRNN, we use the previous 6 timesteps as the input sequence and the next 6 times as the target sequence, to train the model. Our MVGCN is robust as the step number varies from 1 to 6, *i.e.* small increase in both RMSE and

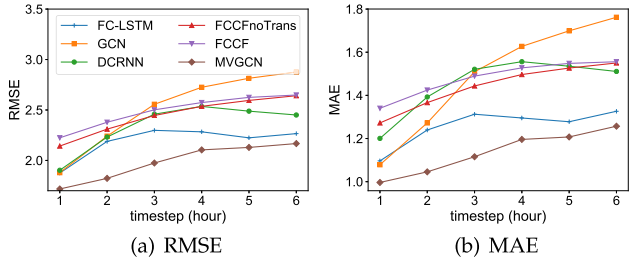


Fig. 9. Step-wise comparisons on the BikeDC test set.

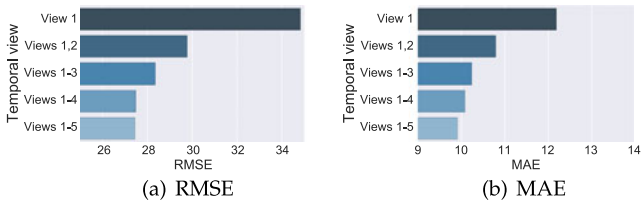


Fig. 10. Effect of temporal views using TaxiNYC.

TABLE 4
Effect of Different Components on TaxiNYC Test Set

Setting	RMSE	MAE
MVGCN	23.15	9.40
w/o geospatial position	23.64	9.85
w/o external	24.41	10.25
w/o metadata	23.23	9.59

MAE, achieving the best for all the 6 steps. We can observe that the original graph convolutional network (GCN) is not robust as the timestep increases, demonstrating that it does not work if we apply the existing models to the crowd flow prediction in a straightforward way. DCRNN performs less well because it also only use the sequence from the *recent* timesteps, resulting in that it cannot capture period, trend, and external factors.

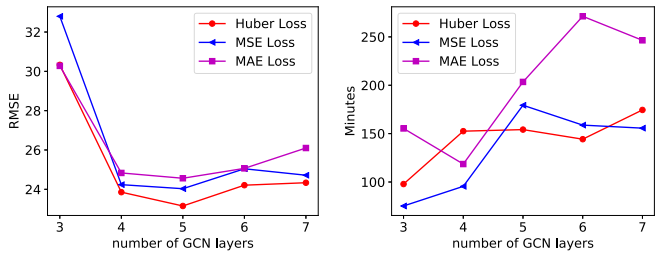
4.3 Effects of Different Components

4.3.1 Temporal View

Fig. 10 demonstrates the different experiment effects of different combinations of temporal views based on RMSE and MAE, including recent (view 1), daily (view 2), weekly (view 3), monthly (view 4), and quarterly views (view 5). With only the recent view considered, we get a terrible result. When taking daily view into consideration, the result is greatly improved, indicating the periodicity is an important feature of traffic flow pattern. Also, the result becomes better and better with more temporal views considered.

4.3.2 Geospatial Position

Recall that in our model, we introduce a spatial graph convolution (see Eq. (6)), which integrates the geospatial position into the graph convolution. After eliminating such geospatial information, the layer is degraded as a graph convolution (Eq. (3)). From Table 4, we observe that RMSE increases from 23.15 to 23.64 without the geospatial position, and MAE also becomes worse, demonstrating the effectiveness of the spatial graph convolution.



(a) RMSE (b) Training time

Fig. 11. Model performance with varying number of GCN layers.

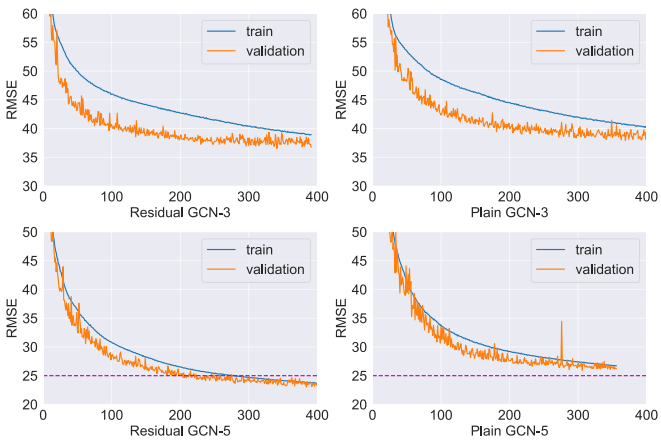


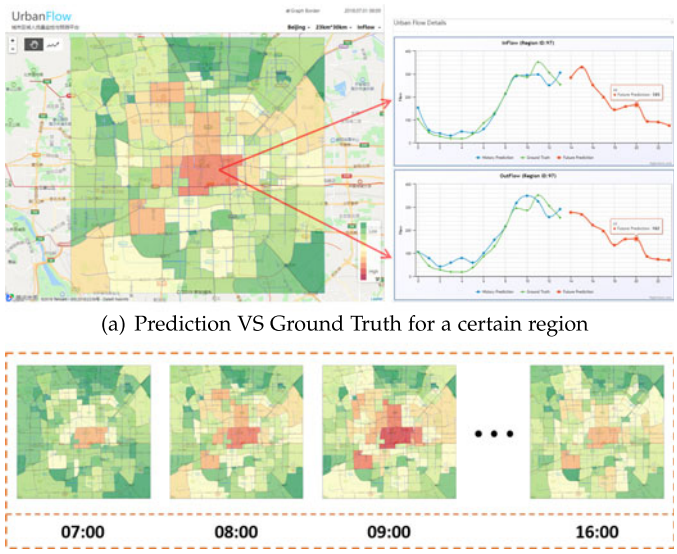
Fig. 12. Performance curve on training and validation sets using plain or residual GCNs. The x-axis represents training epoch.

4.3.3 Global Information

To show the effects of the *embed* component, we compare the performance of MVGCN under two settings: removing external factors or meta data, as shown in Table 4. By eliminating the external factors, the RMSE increases from 23.15 to 24.41. Similarly, without the meta data, RMSE increases to 23.23. The results demonstrate that the external factors/meta data affect the prediction in an STG.

4.3.4 Huber Loss and Number of GCN Layers

To further investigate the effects of different loss functions and number of GCN layers. We perform some ablation studies and report results on TaxiNYC dataset with varying spatial graph convolutional layers or loss functions, as shown in Fig. 11. We plot RMSE metric and can observe that the performance using RMSE, MAE or Huber as loss functions all first decrease and then increase as *the number of GCN layers* increases. Best results occur when the number of GCN layers is 5. The figure demonstrates that deep networks yield better results but much deeper networks still cause the common problem of higher prediction error. And the training time when early stopping happens increases with the model depth on the whole. To validate the effects of residual GCN layers, we compare the model with residual connections in GCN units with plain layers without residual connections. As shown in Fig. 12, we can observe that both of them perform similarly in shallow networks. But when the number of GCN layers is increased to 5, residual networks can achieve much better results, and both of them perform better than shallow networks while set at an appropriate depth.



(a) Prediction VS Ground Truth for a certain region

(b) Flow heatmap over time for the whole city

Fig. 13. Web user interface overview of our UrbanFlow system.

5 Crowd Flow Forecasting System in Irregular Regions

We have developed a crowd flow forecasting demo (called UrbanFlow) in irregular regions internally, which can be accessed now,[6] as shown in Fig. 13a. We have deployed it in the city area of Beijing, China, similar to that of our previous system for gridded regions. The detailed system architecture can be found in our previous work (Section 3 of [44]). Fig. 13a shows the inflow and outflow results for a certain region in the system, where the green line represents the ground truth inflow or outflow in the previous 14 hours, the blue line denotes the prediction results in the 14 hours, and the orange line points the forecasting values in the next 10 hours. We can see the green and blue lines have very close values and similar trend, meaning that our MVGCN can work effectively and well in the traffic flow forecasting system. Fig. 13b displays another function view of overall flow changes of different time stamps for the whole city. We can observe the overall flow distribution varying with time. As the figure shows, in the morning rush hours, most regions have larger crowd flows because people are travelling from home, and the flows decrease in the mid-afternoon during which most people are working or resting indoors.

6 Related Work

6.1 Spatio-Temporal Prediction

There have been a lot of works about spatio-temporal prediction. Such as predicting travel speed and traffic volume on the road [25], [33]. Most of them making predictions concerning single or multiple road segments, rather than citywide ones [5], [36]. Recently, researchers have started to focus on city-scale traffic flow prediction [13], [38].

6. http://101.124.0.58/urbanflow_graph

Specifically, [13] proposed a Gaussian Markov random field based model (called FCCF) that achieves state-of-the-art results on the crowd flow forecasting problem, which can be formulated as a prediction problem on an STG. [38] proposes a multi-view framework for citywide crowd flows prediction, but it is targeted for regular regions' flow prediction using of traditional convolutional neural networks. And most spatiotemporal prediction works for raster-based data have been surveyed in [31]. Work in [9] attempts to use multi-graph graph convolution to capture non-euclidean correlation between regions, so they actually still perform their experiments in regular grid-based regions. Compared with this work, ours is targeted at the real problem of traffic prediction in irregular urban areas, and we also propose the method to process the traffic data and perform map segmentation with road networks.

6.2 Classical Models for Time Series Prediction

Forecasting flow in a spatio-temporal network can be viewed as a time series prediction problem. Existing time-series models, like the auto-regressive integrated moving average model (ARIMA, [2]), seasonal ARIMA [27], and the vector autoregressive model [4] can capture temporal dependencies very well, yet it fails to capture spatial correlations.

6.3 Neural Networks for Sequence Prediction

Neural networks and deep learning [21] have achieved numerous successes in fields such as compute vision [19], [26], speech recognition [10], and natural language understanding [20]. Recurrent neural networks (RNNs) have been used successfully for sequence learning tasks [1], [29]. The incorporation of long short-term memory (LSTM) [14] or gated recurrent unit (GRU) [6] enables RNNs to learn long-term temporal dependency. However, these neural network models can only capture spatial or temporal dependencies. Recently, researchers have combined the above networks and proposed a convolutional LSTM network [34] that learns spatial and temporal dependencies simultaneously but cannot be operated on spatio-temporal graphs. [43] proposed a spatio-temporal residual network, which is capable of capturing spatio-temporal dependencies as well as external factors in regular regions, yet it cannot be adapted to deal with graphs.

7 Conclusion

We propose a novel multi-view deep learning model MVGCN, consisting of several graph convolutional networks, to predict the inflow and outflow in each and every irregular region of a city. MVGCN can not only capture spatial *adjacent* and *multi-hop* correlations as well as interactions, but also integrate the geospatial position via spatial graph convolutions. In addition, MVGCN can capture many types of temporal properties, including closeness, periods (daily, weekly, etc), and trends (e.g. monthly, quarterly), as well as various external factors (like weather and event) and meta information (e.g. time of the day). We evaluate our MVGCN on four real-world datasets in different cities, achieving a performance which is significantly better than 8 baselines, including recurrent neural networks, and Gaussian Markov random field-based models.

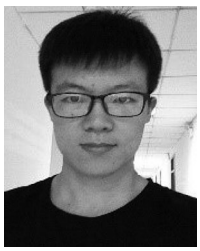
Acknowledgments

This work was supported by the National Key R&D Program of China (2019YFB2101805), Beijing Academy of Artificial Intelligence (BAAI), and the National Natural Science Foundation of China (Grant No. 61672399). Junbo Zhang and Junkai Sun are contributed equally to this work.

References

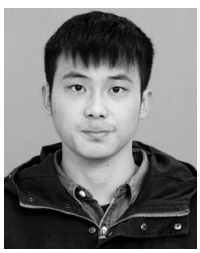
[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *Proc. ICLR*, 2015, pp. 1–15.
[2] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.
[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Representations*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6203
[4] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *J. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 53–72, 2009.
[5] P.-T. Chen, F. Chen, and Z. Qian, "Road traffic congestion monitoring in social media with hinge-loss markov random fields," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 80–89.
[6] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.
[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[8] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, 2001.
[9] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. AAAI*, 2019, pp. 3656–3663.
[10] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6645–6649.
[11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
[13] M. X. Hoang, Y. Zheng, and A. K. Singh, "FCCF: Forecasting citywide crowd flows based on big data," in *Proc. 24th ACM SIGSPATIAL*, 2016, pp. 1–10.
[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[15] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, 1964.
[16] G. Karypis and V. Kumar, "Parallel multilevel series k-way partitioning scheme for irregular graphs," *SIAM Rev.*, vol. 41, no. 2, pp. 278–300, 1999.
[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017, pp. 1–14.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[20] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Int. Conf. Mach. Learn.*, 2014, vol. 14, pp. 1188–1196.
[21] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
[22] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *Proc. ICLR*, 2018, pp. 1–16.

[23] Y. Lin *et al.*, "Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2018, pp. 359–368. [Online]. Available: http://doi.acm.org/10.1145/3274895.3274907
[24] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
[25] R. Silva, S. M. Kang, and E. M. Airoldi, "Predicting traffic volumes and estimating the effects of shocks in massive transportation systems," *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 18, pp. 5643–5648, 2015.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. ICLR*, 2015, pp. 1–14.
[27] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transp. Res. Part C: Emerg. Technol.*, vol. 10, no. 4, pp. 303–321, 2002.
[28] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *Proc. 32nd Int. Conf. Mach. Learn.*, F. R. Bach and D. M. Blei, Eds., 2015, vol. 37, pp. 843–852. [Online]. Available: http://proceedings.mlr.press/v37/srivastava15.html
[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Int. Conf Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[30] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Econ. Geography*, vol. 46, no. sup1, pp. 234–240, 1970.
[31] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *CoRR*, vol. abs/1906.04928, 2019. [Online]. Available: http://arxiv.org/abs/1906.04928
[32] W. Wang, R. Arora, K. Livescu, and J. Bilmes, "On deep multi-view representation learning," in *Int. Conf. Mach. Learn.*, 2015, pp. 1083–1092.
[33] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 25–34.
[34] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[35] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *CoRR*, vol. abs/1304.5634, 2013. [Online]. Available: http://arxiv.org/abs/1304.5634
[36] Y. Xu, Q.-J. Kong, R. Klette, and Y. Liu, "Accurate and interpretable bayesian mars for traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2457–2469, Dec. 2014.
[37] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5668–5675.
[38] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2588–2595.
[39] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and pois," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 186–194.
[40] N. J. Yuan, Y. Zheng, and X. Xie, "Segmentation of urban areas using road networks," *MSR-TR-2012–65, Techn. Rep.*, Microsoft Research Asia, Beijing, China, 2012.
[41] J. Zhang, Y. Zheng, J. Sun, and D. Qi, "Flow prediction in spatio-temporal networks based on multitask deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 3, pp. 468–478, Mar. 2019.
[42] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," in *Proc. 34th Conf. Uncertainty Artif. Intell.*, A. Globerson and R. Silva, Eds., 2018, pp. 339–349. [Online]. Available: http://auai.org/uai2018/proceedings/papers/139.pdf
[43] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Arti. Intell.*, 2017, pp. 1655–1661.
[44] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting city-wide crowd flows using deep spatio-temporal residual networks," *Artif. Intell.*, vol. 259, pp. 147–166, 2018. [Online]. Available: https://doi.org/10.1016/j.artint.2018.03.002
[45] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Inf. Fusion*, vol. 38, pp. 43–54, 2017.
[46] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, 2014, Art. no. 38.



Junkai Sun is currently working toward the master's degree in Xidian University, majoring in computer science and technology. His research interests mainly include spatio-temporal data mining with deep learning, and urban computing. He is also an intern at JD Intelligent Cities Business Unit, JD Digits.

Dr. Junbo Zhang (Member, IEEE) is a senior researcher of JD Intelligent Cities Research and the head of AI Department, JD Intelligent Cities Business Unit, JD Digits. Prior to that, he was a researcher at MSRA from 2015–2018. His research interests include urban computing, machine learning, data mining, and big data analytics. He currently serves as associate editor of *ACM Transactions on Intelligent Systems and Technology*. He has published more than 30 research papers (e.g., *AI Journal*, *IEEE Transactions on Knowledge and Data Engineering*, KDD, AAAI, IJCAI) in refereed journals and conferences, among which one paper was selected as the ESI Hot Paper, three as the ESI Highly Cited Paper. He received the ACM Chengdu Doctoral Dissertation Award, in 2016, the Chinese Association for Artificial Intelligence (CAAI) Excellent Doctoral Dissertation Nomination Award, in 2016, the Si Shi Yang Hua Medal (Top $1/1000$) of SWJTU. in 2012, and the Outstanding PhD Graduate of Sichuan Province, in 2013. He is a member of the ACM, CAAI, and China Computer Federation.



Qiaofei Li is currently working toward the master's degree in Xidian University, majoring in computer science and technology. His research interests focus on deep learning and spatio-temporal data mining.

Dr. Xiuwen Yi received the PhD degree in computer science and technology from Southwest Jiaotong University, in 2018. He is currently a data scientist of JD Intelligent Cities Business Unit, JD Digits, focuses on using big data and AI technology to build real-world applications for tackling urban challenges. He was an intern in Urban Computing Group at MSR Asia from 2014 to 2017. His research interests include: spatiotemporal data mining, deep learning, and urban computing. He has published more than 15 research papers in refereed conferences (e.g., KDD, IJCAI) and journals (e.g., *IEEE Transactions on Knowledge and Data Engineering*, *Artificial intelligence*).

Liang is currently working toward the PhD degree in the School of Computing, National University of Singapore. He has published several papers in refereed conferences, such as KDD, IJCAI, and AAAI. His research interests mainly lie in machine learning, deep learning and their applications in urban areas.

Dr. Yu Zheng (Senior Member, IEEE) is a vice president of JD.COM and the chief data scientist of JD Digits. He also leads the JD Intelligent Cities Business Unit as the president and serves as the managing director of JD Intelligent Cities Research. He is also a chair professor with Shanghai Jiao Tong University and an adjunct professor with the Hong Kong University of Science and Technology and Nanjing University. Before joining JD Digits, he was a senior research manager at Microsoft Research. He currently serves as the editor-in-chief of *ACM Transactions on Intelligent Systems and Technology*. He has served as chair on more than 10 prestigious international conferences, e.g., as the program co-chair of ICDE 2014 (Industrial Track) and CIKM 2017 (Industrial Track). In 2013, he was named one of the Top Innovators under 35 by MIT Technology Review (TR35) and featured by Time Magazine for his research on urban computing. In 2014, he was named one of the Top 40 Business Elites under 40 in China by Fortune Magazine. In 2017, he was named an ACM distinguished scientist.