Improving Urban Crowd Flow Prediction on Flexible Region Partition

Xu Wang, *Student Member, IEEE,* Zimu Zhou, *Student Member, IEEE,* Yi Zhao, *Student Member, IEEE,* Xinglin Zhang, *Student Member, IEEE,* Kai Xing, *Member, IEEE,* Fu Xiao, *Member, IEEE,* Zheng Yang, *Member, IEEE,* Yunhao Liu, *Fellow, IEEE,*

Abstract—Accurate forecast of citywide crowd flows on flexible region partition benefits urban planning, traffic management, and public safety. Previous research either fails to capture the complex spatiotemporal dependencies of crowd flows or is restricted on grid region partition that loses semantic context. In this paper, we propose DeepFlowFlex, a graph-based model to jointly predict inflows and outflows for each region of arbitrary shape and size in a city. Analysis on cellular datasets covering 2.4 million users in China reveals dependencies and distinctive patterns of crowd flows in not only the conventional space and time domains, but also the speed domain, due to the diverse transportation modes in the mobility data. DeepFlowFlex explicitly groups crowd flows with respect to speed and time, and combines graph convolutional long short-term memory networks and graph convolutional neural networks to extract complex spatiotemporal dependencies, especially long-term and long-distance inter-region dependencies. Evaluations on two big cellular datasets and public GPS trace datasets show that DeepFlowFlex outperforms the state-of-the-art deep learning and big-data-based methods on both grid and non-grid city map partition.

1 Introduction

Predicting citywide crowd flows is of great importance for urban planning, traffic management, and public safety. Crowd flows can be specified by *inflows* and *outflows*, which refer to the total amount of crowds entering or leaving a region within a given time interval [1]. With a macroscopic view of crowd flows in and out of every region, city authorities can make strategies on traffic control and power supply to reduce environment pollution and energy waste [2]. Forecasts of commuter volumes facilitate urban transit developers to schedule feeder bus routes and minibus services at traffic hotspots [3]. Taxi-calling platforms benefit from crowd flow prediction by pre-dispatching taxis to regions with large numbers of potential passengers [4]. Emergency mechanisms can also be enhanced if knowledge of overloaded regions is known in advance [5].

The deep penetration of mobile phones in everyday life has made cellular data ideal to estimate citywide crowd flows. While GPS dominates for outdoor localization and tracking, GPS data may be unavailable (*e.g.*, underground metros). Pedestrians, who account for a considerable portion of crowd flows, are often reluctant to turn on the GPS continuously and share their locations due to privacy concerns. In contrast, the anonymous cellular data collected and aggregated at cell towers can be easily associated with user locations. Due to the pervasive usage of mobile phones, cellular data cover a wide geographical range, a large population and diverse transportation modes, thus characterizing human mobility more comprehensively. Although cellular data yield larger location error than GPS, they suffice to derive fundamental laws in human mobility [6] [7] and aggregated mobility models [8] [9] [10].

Macroscopic prediction of crowd flows is promising but challenging. Large-scale studies have revealed the predictability of human mobility [7] and extensive mathematical models have been proposed in the past decade [8]. However, the estimations of mathematical models often yield low accuracies and notably differ from the actual crowd flows [11]. Some researchers predict user-specific movements based on historical trajectories [12]. Yet it involves substantial computation overhead to aggregate crowd flows using predictions on individual mobility. General time-series approaches [13] yield limited accuracy because they fail to model the complex spatiotemporal dependencies of crowd flows.

In addition to the challenges in modeling the spatiotemporal dependencies, the citywide crowd flow prediction problem is further complicated by the cellular dataset and the need for flexible region partition. Take Fig. 1 as an example. *(i)* Since a mobile phone regularly reports its location to the associated cell tower [2], the location of a mobile phone user is available regardless of his/her transportation mode (*e.g.*, walk, bike, vehicle, underground metro). The diverse transportation modes lead to complex dependencies of crowd flows among regions. For example, crowd flows of one region (r_1) may not only depend on those of adjacent regions (*e.g.*, r_2), but also those of distant regions (*e.g.*, r_3

- *X. Wang, Y. Zhao, Z. Yang and Y. Liu are with the School of Software and TNLIST, Tsinghua University. Email: xu-wang15@mails.thu.edu.cn, zhaoyi17@mails.thu.edu.cn, yangzheng@tsinghua.edu.cn, yunhao@tsinghua.edu.cn*
- *Z. Zhou is with Computer Engineering and Networks Laboratory, ETH Zurich. E-mail: zzhou@tik.ee.ethz.ch*
- *X. Zhang is with School of Computer Science and Engineering, South China University of Technology, China. E-mail: zhxlinse@gmail.com*
- *F. Xiao is with College of Computers, Nanjing University of Posts and Telecommunications, China. E-mail: xiaof@njupt.edu.cn*
- *K. Xing is with the School of Computer Science and Technology, University of Science and Technology of China. Email: kxing@ustc.edu.cn*

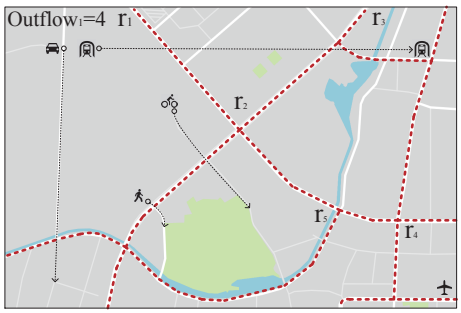


Fig. 1: An illustration of crowd flows on flexible region partition. Region partition is denoted by red dotted lines, which can be of irregular shapes and different sizes. As an example, the outflow of region r_1 is 4 (see the 4 black dotted arrows).

due to fast, underground metros). *(ii)* Urban planning and traffic management applications are usually performed on semantic, geographical or administrative regions of a city. For instance, it is reasonable to segment region r_5 following the directions of the river banks, which results in *irregular* region shapes. It is also desired to cluster areas around a transportation hub (*e.g.*, an airport in region r_4) into a small region due to its high volume of mobility. Pioneer research on crowd flow prediction either works with only grids of the same size [1] or fails to characterize the distant dependencies among regions [5]. Furthermore, existing solutions [1] [5] [14] are only tested on mobility datasets of a single transportation mode, leaving validation with heterogeneous transportation modes unexplored.

In this work, we propose DeepFlowFlex, a *Deep* graph learning enabled citywide crowd *Flow* prediction scheme on *Flex*ible region partition. In addition to the conventional space and time domain, DeepFlowFlex decomposes crowd flows in the new speed domain to characterize the distinctive patterns due to diverse transportation modes. DeepFlowFlex utilizes graph-based deep neural networks to capture complex spatial dependencies among regions and predict inflows and outflows for every region in the whole city. Specifically, DeepFlowFlex utilizes graph convolutional long short-term memory networks to model spatiotemporal (especially long-term) dependencies. It also employs residual learning to model spatial dependencies, particularly long-distance dependencies. The graph-based structures enable DeepFlowFlex to operate even on irregular-shaped regions. Evaluations on large-scale cellular datasets covering 2.4 million users and 7.7 thousand cell towers in two major cities of China show that DeepFlowFlex improves the prediction accuracy by 8.7% (RMSE), 12.9(MAPE) and 11.4% (RMSE), 9.3% (MAPE) than the state-of-the-art grid-based deep model [1], and the spatiotemporal model for crowd flows [5], respectively. Trained on mobility data alone, DeepFlowFlex also outperforms the state-of-the-art data fusion schemes [1] [5] on both grid and non-grid region partition.

The contributions of this work are summarized as follows.

- To the best of our knowledge, DeepFlowFlex is the first work that predicts citywide crowd flows with cellular data via deep graph learning on flexible region partition. It will benefit various urban computing applications that involve regions of irregular shapes and diverse sizes.
- In addition to the conventional time and space domains, we further decompose crowd flows in the speed domain. The speed domain accounts for the diverse transportation modes of crowd flows, which is neglected in previous models designed for a single transportation mode.
- Extensive evaluations on two large-scale cellular datasets show that DeepFlowFlex outperforms the state-of-the-arts on both grid and non-grid region partition. DeepFlowFlex is also applicable to other mobility datasets such as GPS traces and yields lower prediction errors than data fusion based approaches.

In the rest of the paper, we review related work in Sec. 2 and describe our dataset and the problem formulation in Sec. 3. We present the detailed design in Sec. 4 and evaluate the performance of DeepFlowFlex in Sec. 5. Sec. 6 finally concludes this study.

2 Related Work

The development of communication technologies [15], [16] has made it possible to collect large-scale and fine-grained users' traces (*i.e.*, from cellular networks [2], [8], [17] or by crowdsourcing [18], [19]) to study human dynamics. Particularly, there has been active research on modeling and predicting human mobility.

With nation-wide cellular logs, researchers derive the regularity [6] and predicability [7] of human mobility. Based on these fundamental laws of human mobility, extensive mathematical models have been proposed at macro and micro levels [8]. For instance, Simini *et al.* [10] propose a macroscopic model for migration flows in and out or a city. To derive finer-grained mobility, additional knowledge and assumptions are needed [8]. Song *et al.* [20] build a self-consistent model for individual human mobility based on mobile phone traces. However, it requires individual-specific information such as the number of unique locations visited. Other individual mobility models are effective among people sharing similar social contexts [9]. They involve extensive computation to aggregate citywide crowd flows using individual mobility models. While these analytical models are able to simulate mobility patterns to an acceptable accuracy, the generated trajectories still often deviate from the reality [11].

The development of mobile big data and urban computing brings in a promising paradigm in human mobility prediction [21]. Instead of deriving synthetic models, these research efforts directly predict human mobility via data-driven approaches. CityMomentum [12] predicts short-term movements especially during rare events such as New Year's Eve countdown by tracking personal historical GPS logs. Other researchers focus on vehicular traffic flow prediction on road networks for intelligent transportation management. Lv *et al.* [22] introduce a deep learning based model to learn genetic traffic flow features. Calabrese *et al.* [23] estimate the origin-destination flows in the Boston metropolitan area. These schemes aggregate

crowd flows by road segments [22] or points of interest [23]. In contrast, we include crowd flows from both vehicles and pedestrians, and aim to predict flows for every region citywide.

Our work is most related to FCCF [5], DeepST [14] and ST-ResNet [1]. FCCF partitions a city into regions according to road networks and clusters regions based on historical human mobility. It then predicts new-flows and end-flows for each region using taxi trajectories and weather data. DeepST designs a deep neural network framework to predict inflows and outflows for each grid region in a city combining taxi GPS logs and meta data (dayofweeek, weekday/weekend). ST-ResNet employs the residual neural network to model temporal properties of crowd traffic and absorb more external factor (*i.e.*, weather) compared with DeepST. Unlike the aforementioned works [5] [14] [1] that leverage the sparse taxi trajectories to sample human mobility, we harness cellular data from mobile devices, which covers significantly wider range, population, and activities. In contrast to FCCF [5], we carefully design a deep spatiotemporal model to make full use of the big cellular data and characterize the complex (especially long-term and long-distance) inter-region dependencies of crowd flows. Our graph-based model is applicable to both grid and irregular city map partitions, whereas ST-ResNet [1] is restricted to grids due to the CNN models.

3 Dataset and Preliminaries

This section presents an overview of our dataset and formally defines the crowd flow prediction problem.

3.1 Cellular Dataset

A mobile phone regularly notifies its rough location in case of events (*e.g.*, call, Internet usage) or network updates (*e.g.*, switching among 2G/3G/4G) [2]. A large-scale cellular dataset provides information about the macroscopic mobility regardless of the location and the transportation mode of mobile users.

3.1.1 Data Collection

Our work is grounded upon two citywide cellular usage datasets collected by a major cellular carrier in CityA and CityB of China. The datasets monitor every packet exchanged between mobile-to-mobile, mobile-to-Internet and Internet-to-mobile sources and destinations. Each mobile record consists of a unique anonymous user ID, the flow create time, the flow connected cell tower ID, App ID, device type ID, uplink traffic and downlink traffic. Table 1 summarizes the basic statistics for the datasets. To the best of our knowledge, the datasets are the largest urban-scale cellular traffic datasets in terms of the number of mobile users and cell towers. The wide coverage in mobile users and cell towers promises to capture various transportation modes and comprehensive spatiotemporal dynamics in crowd flows.

TABLE 1: Cellular dataset description.

Statistics	CityA	CityB
Mobile Records	3.1×10^9	2.5×10^9
Cell Towers	5.2×10^3	2.5×10^3
Covered Users	1.5×10^6	9.4×10^5
Covered Apps	7.0×10^2	7.0×10^2
Covered Area	$3.5 \times 10^3 km^2$	$2.4 \times 10^3 km^2$
Date	12/20/2016-02/04/2017	

3.1.2 Data Preprocessing

The raw mobile records are preprocessed in three steps.

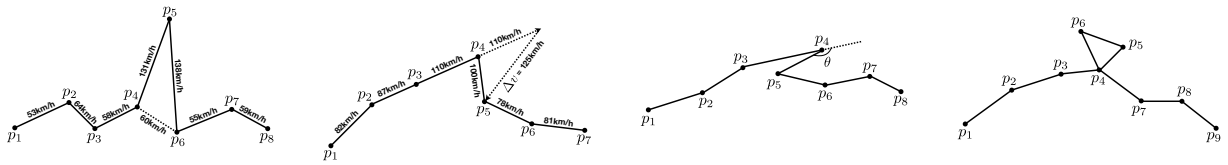
(i) User Filtering. In this work we mainly focus on crowd flows that likely come from residents in the city. Therefore we filter short-lived mobile records that fail to connect to the Internet for at least three consecutive days. Note that devices outside the city may also be recorded in the dataset due to the ISP's mobile roaming policy. This portion of mobile records is also removed by checking the coordinates of the cell towers.

(ii) Localization Estimation. Since our aim is to predict citywide crowd flows, we use the location of the cell tower as an estimate for each mobile device (user). Specifically, the cell tower ID field in each mobile record is mapped to the geographical coordinates of the corresponding cell tower. Note that a mobile device may not be associated to the nearest cell tower [2] and the flow-level records only observe the cell sector where the user initiates his session [24]. However, a location accuracy of 70m is achievable with the coverage information of cell towers [25].

(iii) Trajectory Denoising. To further reduce the location errors, we harness trajectory information to avoid large single location errors. Specifically, we first extract the trajectory T of user u as $T_u = \{p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n\}$ where $p_i = (L_i^u, t_i^u, d_i^u)$, L_i^u is the location of the connected cell tower, and $L_1^u \neq L_2^u, L_2^u \neq L_3^u, \cdots, L_{n-1}^u \neq L_n^u$, t_i^u is the first create time of a flow record meeting $L_i \neq L_{i-1}$, $t_1 < t_2 < \cdots < t_n$. $d_i^u = t_{i+1}^u - t_i^u$ is the travel time from L_i^u to L_{i+1}^u. Then we apply rule-based filters (speed filter, speed variation filter, angle filter and loop filter) inspired by [26], [27] to denoise the trajectories. The speed filter and speed variation filter estimate the immediate speed and speed variation at p_i and identify p_i as an outlier if its immediate speed or speed variation exceeds the threshold $speed_{max} = 120km/h$ or $variation_{max} = 120km/h$, respectively. Angle filter is designed to remove outliers with sharp angles. A point p_i forms a sharp angle when $\angle p_{i-1}p_ip_{i+1}$ exceeds $\theta_{max} = 120^\circ$ and both $\angle p_{i-2}p_{i-1}p_i$ and $\angle p_ip_{i+1}p_{i+2}$ are smaller than θ_{max}. Since a mobile device may switch between several nearby towers frequently even though the user stays in the same place, we apply loop filter to find loops within a given small area $s_{max} = 2km \times 2km$ and compress them to a weighted centroid. Fig. 2 shows four instances applying the filters, respectively. Fig. 2a removes p_5 since the immediate speed of $p_4 \rightarrow p_5$ and $p_5 \rightarrow p_6$ exceeds $speed_{max}$. Fig. 2b detects p_4 as an outlier since its speed variation is abnormal. Fig. 2c deletes p_4 in the trajectory because p_4 forms a sharp angle in the trajectory. Fig. 2d views p_4, p_5, p_6 as a loop.

3.2 Problem Statement

Similar to [1], we define two types of crowd flows, *i.e.*, inflow and outflow for each region. Inflow is the total movement of crowds entering a region from other places



(a) Speed filter (b) Speed variation filter (c) Angle filter (d) Loop filter

Fig. 2: Examples of trajectory denoising via (a) speed filter, (b) speed variation filter, (c) angle filter and (d) loop filter.

during a given time interval, and outflow denotes the total movement of crowds leaving a region for other places.

Definition 1 (Region Partition)**.** *Region partition for a city is to geographically divide the city map into non-overlapping regions. Let* $R = \{r_i | i = 1, 2, \cdots, N_r\}$ *denote the set of partitioned regions, where* N_r *is the count of regions. The spatial relationship is represented by* $G = (R, E)$ *where* E *is the set of all pairs of* (r_i, r_j) *meeting* r_i *shares one or more boundary point with* r_j*.*

Definition 2 (Crowd Flow)**.** *Let* $\mathcal{T} = \{T_u | u \in U\}$ *denote the trajectories of all users* U*.* $P_t(r)$ *denotes the set of mobile devices located at region* r *at time* t*.*

$$P_t(r) = \{u | T_u(t) \in r\} \tag{1}$$

where $T_u(t) = L_i^u$ *when* $t \in [t_i^u, t_{i+1}^u)$ *is the location of* u *at time* t*.*

The inflow of the region r *is defined as people coming from other regions.*

$$x_t^{in}(r) = P_t(r) \setminus P_{t-1}(r) \tag{2}$$

The outflow of r *is defined as people going to leave* r *and reach to other regions.*

$$x_t^{out}(r) = P_t(r) \setminus P_{t+1}(r) \tag{3}$$

We use $x_t^{in} = \{x_t^{in}(r_i) | i = 1, \cdots, N_r\}$ *to represent the set of inflow of all regions at time interval* t*, and* $x_t^{out} = \{x_t^{out}(r_i) | i = 1, \cdots, N_r\}$ *to denote the set of outflow of all regions at time interval* t*. We use* $x_t = \{x_t^{in}, x_t^{out}\}$ *to represent total crowd flow for clear expression.*

Definition 3 (Citywide Crowd Flow Prediction)**.** *Given* $\{x_1, \cdots, x_{t-1}\}$*, the citywide crowd flow prediction problem aims to predict* x_t*.*

Compared with prior studies on crowd flow prediction [1] [5] that optimize for GPS traces, our problem try to predict crowd flows covering various transportation modes (on foot, bikes, vehicles, underground metros, etc.), large populations (2.4 million), and complex spatiotemporal dependencies. As will be discussed in the next section, our model simultaneously characterizes the *deep spatiotemporal* dependencies and works with city partition of *irregular* shaped regions, which is impossible with previous works [1] [5] [14].

4 Citywide Crowd Flow Prediction

This section presents the design of DeepFlowFlex.

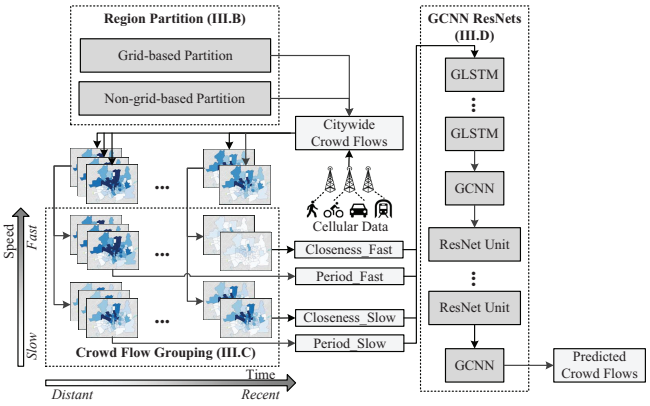


Fig. 3: The framework of DeepFlowFlex.

4.1 Overview

Fig. 3 illustrates the work flow of DeepFlowFlex. We first segment a city map into grid-based regions or non-grid-based regions based on its applications (Sec. 4.2). We characterize crowd flows in the time and space domains, as well as a new speed domain. The speed domain explicitly accounts for the heterogeneous transportation modes of mobility recorded by cellular data. We extract temporal features from crowd flows in each region and group them by speed (Sec. 4.3). The feature tensors are input into a graph-based deep spatiotemporal model to capture both temporal and spatial dependencies (Sec. 4.4). We first utilize graph convolutional long short-term memory networks (GLSTM) to encode the spatial time series into fix-sized tensors and model the spatiotemporal dependencies, especially long-term temporal dependencies. Then we employ a residual learning method to further model the spatial dependencies, particularly long-distance dependencies. The residual function is learned with graph convolutional neural networks (GCNN). The graph-based structures enable DeepFlowFlex to operate on regions of irregular shapes and different sizes. We describe each of the functional modules in sequel.

4.2 Region Partition

Region partition methods can be categorized into two classes *i.e.*, the grid-based partition and the non-grid-based partition. The grid-based partition segments a city map into $M \times N$ grids according to the longitude and the latitude. Each cell denotes a unique region in the scheme. This

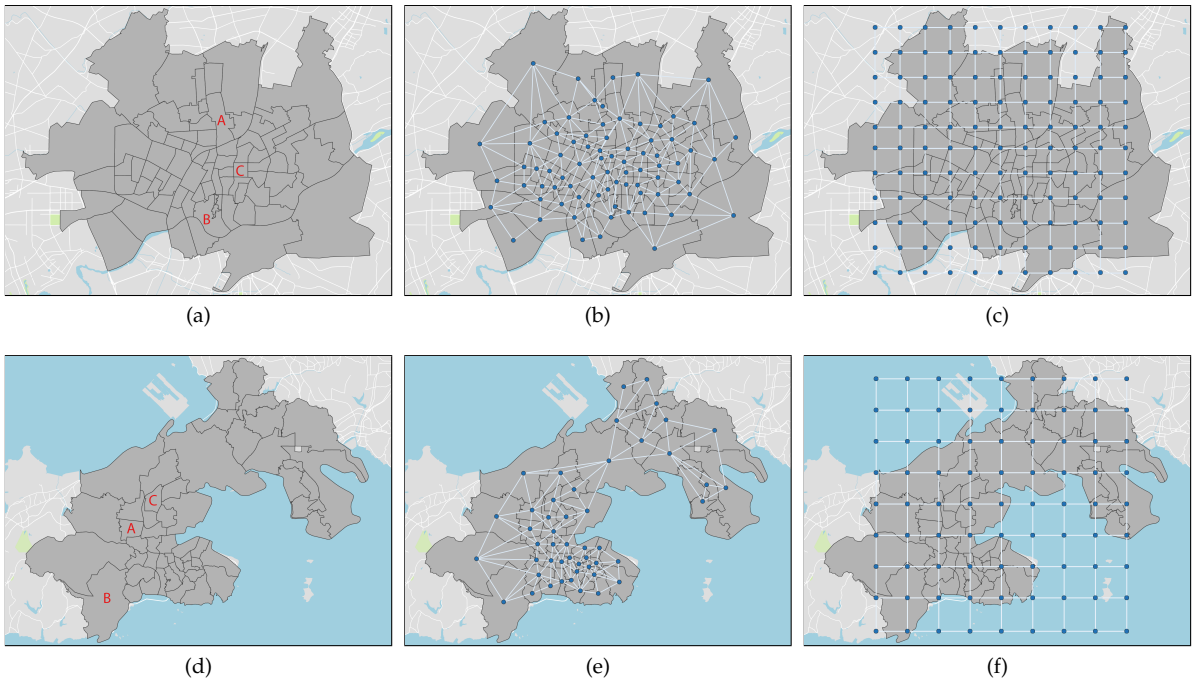


Fig. 4: Region partitions of CityA and CityB. (a) and (d) are the city maps of CityA and CityB. A, B, C on the maps are sample regions of transportation hub, school and business area, respectively. (b) and (e) are administrative division-based partitions. (c) and (f) are grid-based partitions.

method is easy to implement but can be inconvenient for crowd flow prediction. The reasons are two-fold.

(i) Many urban planning and traffic management decisions are made based on semantic, geographical or administrative regions of a city, which may be represented by an irregular shape rather than a rectangle.

(ii) Due to the uneven distribution of citizens, regions of different sizes should be considered. For example, in the rural area, a large region is preferable to characterize the small volume of crowd flows; while in the urban area, a small and dense region is desirable to characterize the high dynamics and large volume of crowd flows.

Previous works have explored efficient non grid-base map partitioning methods such as road network-based [5], voronoi-based [28] and administrative division based [29]. Note that for any irregular shapes, their spatial relationship can be model as a graph which can be handled by our proposed graph learning model. The key challenges for the problem are to model the spatial dependencies and temporal dependencies. We design a deep learning scheme to model the complex spatio-temporal dependencies. The scheme can both learn long-term dependencies and long-distance dependencies to improve the prediction performance. Due that different partition methods are incomparable, for convenience, in the rest of this paper, we use administrative division-based partition for crowd flow pattern analysis. We evaluate our model on both grid-based and administrative division-based partitions. Fig. 4 shows the two region partition methods for CityA and CityB, each region is a subdistrict or town of the city.

4.3 Crowd Flow Characterization and Grouping

In this subsection we investigate the characteristics of the crowd flows defined on the administrative division-based partition. We show distinctive crowd flow patterns in time, space and speed domains, which we leverage to design effective inputs for our graph-based deep spatiotemporal model.

4.3.1 Time Domain Characteristics

Fig. 5a shows the time dynamics of crowd flows of a sample region in CityA in one day. The crowd flows before dawn (06:00) are low and peak at around 08:00 and 18:00, which are usually the start and end for work hours. Fig. 5b shows that crowd flows have strong periodicity for days. However, we do not investigate the periodicity in weeks for the limited dataset. Fig. 5c shows the difference between crowd flows of weekdays and weekends in the sample region, that is crowd flows of weekdays have larger volumes and variances than crowd flows of weekends.

We further utilize autocorrelation to explore the temporal properties of crowd flows. The sample Auto-correlation Function (ACF) of the inflow of region r is defined as follow:

$$\rho_h^{in}(r) = \frac{\sum_{t=1}^{T-|h|}(x_{t+|h|}^{in}(r) - \overline{x^{in}(r)})(x_t^{in}(r) - \overline{(x^{in}(r))})}{\sum_{t=1}^{T}(x_t^{in}(r) - \overline{x^{in}(r)})^2} \quad (4)$$

where $h \in [-T, T]$ and T is the total time interval, $\overline{x^{in}(r)}$ averages $x_t^{in}(r)$ for $t = 1, 2, \cdots, T$. The ACF of the outflow can be defined similarly.

Fig. 6a shows the ACF of the inflow of a sample region. We observe that the autocorrelation peaks at time lags of one or multiple of 48 (hours). This result indicates a rough

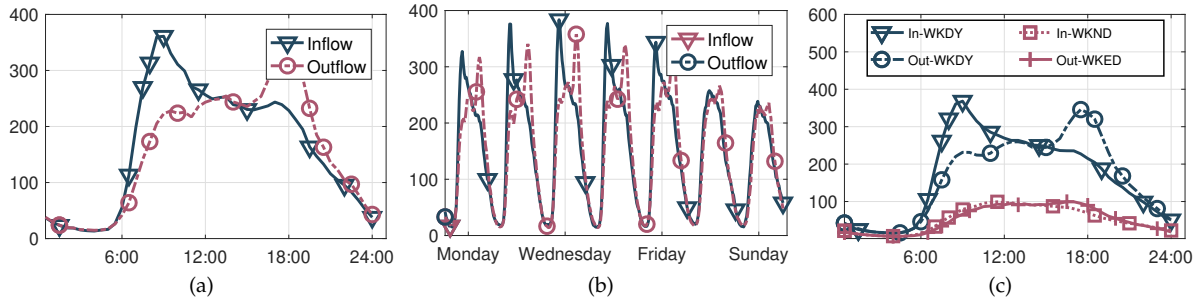


Fig. 5: Crowd flow patterns of a sample region: (a) inflow/outflow of one day; (b) inflow/outflow of one week; (c) inflow/outflow of weekdays and weekends.

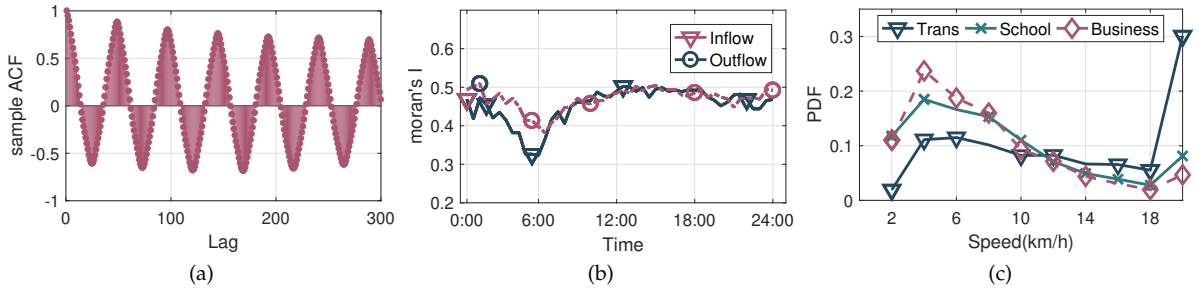


Fig. 6: Characterizing crowd flows in the time, space and speed domains: (a) ACF of inflow in a region; (b) Morans'I of inflow and outflow; (c) speed distribution of citywide crowd flows.

periodic pattern of the crowd flows. In addition, the peaks nearer to zero are higher than those further away. Thus the current crowd flows are strongly related to the most recent historical ones. These insights motivate us to utilize *period* and *closeness* to depict such temporal patterns. Specifically, the *closeness* feature is a tensor of crowd flows in the recent S time intervals.

$$x_t^c = [x_{t-S}, x_{t-S+1} \cdots, x_{t-1}] \in \mathbb{R}^{S \times N_r \times 2} \tag{5}$$

The *period* feature is a tensor of crowd flows of S time intervals in the last day ranging from $t - T - \lfloor \frac{S}{2} \rfloor$ to $t - T + \lfloor \frac{S-1}{2} \rfloor$.

$$x_t^p = [x_{t-T-\lfloor \frac{S}{2} \rfloor}, x_{t-T-\lfloor \frac{S}{2} \rfloor+1}, \cdots, x_{t-T+\lfloor \frac{S-1}{2} \rfloor}] \in \mathbb{R}^{S \times N_r \times 2} \tag{6}$$

where T is the period of the time series (*e.g.*, for half-hour time interval, $T = 48$). The time range in the period feature can be interpreted as the past and future sequences of $t - T$ can benefit to predicting at t.

4.3.2 Space Domain Characteristics

To explore the spatial characteristic of crowd flows, we visualize the crowd flows of all regions of CityA at different times in Fig. 7. We make the following observations. *(i)* The crowd flows in transportation area (*i.e.*, A) and some business area exhibit heavy cellular traffic throughout the day. *(ii)* Adjacent regions mostly have similar crowd flow scale at all times in a day. *(iii)* In the morning, rural areas have higher outflows than inflows while in the evening, rural areas have higher inflows than outflows. It can be explained that some people living in rural areas go to the central areas for work and return home in the evenings.

To further study the spatial dependencies of crowd flows, we examine the correlations among regions using the Moran's I [30]. Moran's I is a measure of spatial autocorrelation, which is defined as:

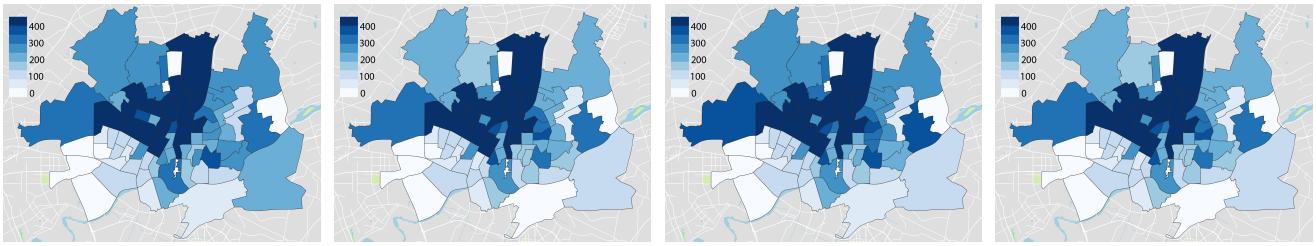
$$I(t) = \frac{n}{\sum_{i,j} w_{ij}} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(x_t(r_i) - \overline{x_t})(x_t(r_j) - \overline{x_t})}{\sum_{i=1}^{n}(x_t(r_i) - \overline{x_t})^2} \tag{7}$$

where $w_{ij} = \frac{1}{d_i}$ when r_i and r_j are neighbors, d_i is the degree of r_i. For a normalized weight matrix (*i.e.*,$\sum_j w_{ij} = 1$), values of I range from -1 to $+1$. Values significantly below $\frac{-1}{N-1}$ indicate negative spatial autocorrelation and values significantly above $\frac{-1}{N-1}$ indicate positive spatial autocorrelation.

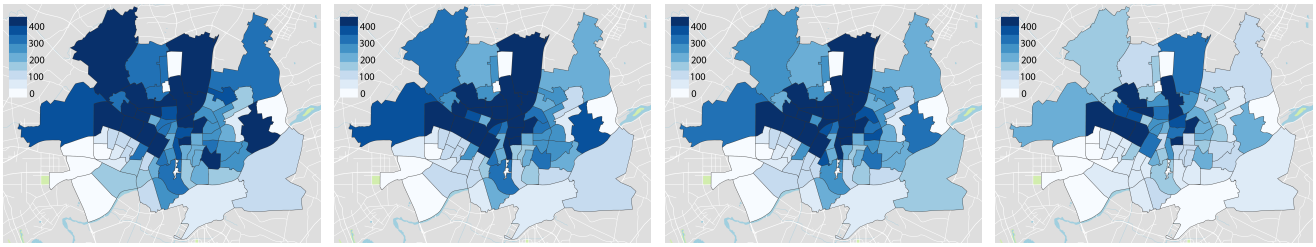
Fig. 6b shows the values of Moran's I for crowd flows at the granularity of half an hour in December 20, 2016 of CityA. As shown, all the Moran's I values are greater than 0.3 throughout the day, indicating positive spatial autocorrelation. The spatial dependencies of crowd flows among regions suggest the necessity to apply convolutions in the prediction model, which is validated in [1]. However, the CNNs in [1] are restricted to grids, while we utilize GCNNs to operate on flexible region partition.

4.3.3 Speed Domain Characteristics

Since the crowd flows estimated by cellular data cover not only vehicles but also pedestrians, we explore whether crowd flows of different speed groups exhibit distinctive patterns. We estimate the speed of users as follows. For $u \in x_t^{in}(r)$, the speed of u is the immediate speed at t of T_u, that is, $s = \frac{D(L_{i-1}, L_i)}{d_{i-1}}$, where $t_{i-1} < t \leq t_i < t + \Delta t$ and for $u \in x_t^{out}(r)$, the speed of u is defined as $s = \frac{D(L_i, L_{i+1})}{d_i}$,



(a) 08:00 (b) 12:00 (c) 16:00 (d) 20:00



(e) 08:00 (f) 12:00 (g) 16:00 (h) 20:00

Fig. 7: Spatial dynamics of crowd flows at different times of a day. (a)-(d) are inflows and (e)-(h) are outflows.

where $t_i < t \leq t_{i+1} < t + \Delta t$. $D(x, y)$ is the distance function and can be defined as Euclidean distance or Manhattan distance. In our work, we use Euclidean distance to estimate the speed of user. Although the definition of speed is coarse-grained and tends to underestimate the actual speed, it suffices to show the speed distribution of crowd flows.

Fig. 6c plots the speed distribution of crowd flows in three different regions labeled in Fig. 4a. We aggregate different speed ranges into 10 bins, including $[0, 2), [2, 4), \cdots, [18, +\infty)$. As shown in Fig. 6c, the speed distribution of the three regions notably differ. The region with a transportation hub contains a large portion of fast crowd flows ($\geq$ 14 km/h) while in the business area and school area slow crowd flows dominate ($<$ 14km/h). This observation indicates that many citizens take metros or ground vehicles from and to the transportation hub, while pedestrians dominate in school and business area. Compared to the school area, the business area has a higher portion of speed in $[0, 2)$ bin. The reason might lie in the fact that people prefer to walk in the business area for shopping.

To explore the speed dynamic, we propose to group crowd flows according to their speed range, which is neglected in previous crowd flow prediction studies with mobility data of single transportation mode [1] [5] [14].

4.3.4 *Feature Summary*

Based on the above characteristics of crowd flows, we propose to decompose the crowd flow of each region into fast flows ($\geq$ 14 km/h) and slow flows ($<$ 14km/h), and summarize the crowd flows using *closeness* and *period*.

Note that despite that our speed feature only includes two groups (*i.e.*, fast and slow), it can be decomposed in a more fine-grained way with more accurate location estimation and much bigger data for training.

To predict x_t, historical crowd flows are represented as four feature tensors. We then reshape the feature tensors into $x_t^F \in \mathbb{R}^{S \times N_r \times 8}$ by concatenating flow type (*i.e.*, inflow and outflow), speed type (*i.e.*, fast speed and slow speed) and temporal type (*i.e.*, closeness and period). The feature tensor x_t^F is fed into our graph-based spatiotemporal model to jointly learn the spatiotemporal dependencies, which are described as below.

4.4 Graph-based Deep Spatiotemporal Model

For crowd flows $x = \{x_1, \cdots, x_N\}$ on a region partition $G = (V, E)$ defined in Sec. 4.2, x can be taken as graph signals whose response at time t is x_t. The features of x are grouped as $\{x_1^F, x_2^F, \cdots, x_N^F\}$ as in Sec. 4.3. Our graph-based deep spatiotemporal model DeepFlowFlex is a function f^*, which is formulated as:

$$\hat{x}_t = f^*(G, x_t^F) \tag{8}$$

Our DeepFlowFlex model consists of two modules (*i.e.*, the encoding module and the residual learning module).

4.4.1 *Encoding Module*

The encoding module identifies spatial structures and finds temporal dynamic patterns simultaneously. We use GLSTM [31] as our encoding module. GLSTM is a generalization of convLSTM [32], which is an efficient method for modeling spatiotemporal sequences. GLSTM is a combination of GCNN [33] and LSTM [34]. The former is an efficient generalization of CNN and could learn local, stationary and compositional features on graphs. The latter provides a good way to learn long-term dependencies avoiding explosion and vanishing of the gradient problem [35].

Like LSTM, GLSTM learns temporal correlations stably by maintaining a memory cell c_t which acts as an accumulator of the state information. Every time a new graph signal x_t comes, its information will be accumulated to the cell if the input gate i is activated. And the past cell status c_{t-1} will be "forgotten" if the forget gate f is on. The output gate o controls the output of the memory cell. All input-to-state and state-to-state transitions in GLSTM are implemented with a one-layer GCNN which applies graph convolution

operator $*_{\mathcal{G}}$ on the input. The convolution operator $*_{\mathcal{G}}$ for the graph signal x on G is defined by applying a non-parametric kernel $g_\theta(\Lambda) = \text{diag}(\theta)$, where $\theta \in \mathbb{R}^{|V|}$ is a vector of Fourier coefficients as:

$$y = g_\theta *_{\mathcal{G}} x = g_\theta(L)x = g_\theta(U\Lambda U^T)x_t = U g_\theta(\Lambda)U^T x \quad (9)$$

where $U \in \mathbb{R}^{|V|\times|V|}$ is the matrix of eigenvectors. $\Lambda \in \mathbb{R}^{|V|\times|V|}$ is the diagonal matrix of eigenvalues of the normalized graph Laplacian $L = I_{|V|} - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = U\Lambda U^T \in \mathbb{R}^{|V|\times|V|}$. $I_{|V|}$ is the identity matrix, W is the weight matrix defined as $w_{ij} = 1$ when r_i and r_j are neighbors otherwise 0 in this work. $D \in \mathbb{R}^{|V|\times|V|}$ is the diagonal degree matrix with $D_{ii} = \sum_j w_{ij}$.

The key equations of GLSTM are shown in Eq. (10) where $\odot$ denotes the Hadamard product.

$$\begin{aligned} i &= \sigma(W_{xi} *_{\mathcal{G}} x_t + W_{hi} *_{\mathcal{G}} h_{t-1} + w_{ci} \odot c_{t-1} + b_i) \\ f &= \sigma(W_{xf} *_{\mathcal{G}} x_t + W_{hf} *_{\mathcal{G}} h_{t-1} + w_{cf} \odot c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{xc} *_{\mathcal{G}} x_t + W_{hc} *_{\mathcal{G}} h_{t-1} + b_c) \\ o &= \sigma(W_{xo} *_{\mathcal{G}} x_t + W_{ho} *_{\mathcal{G}} h_{t-1} + w_{co} \odot c_t + b_o) \\ h_t &= o \odot tanh(c_t) \end{aligned} \quad (10)$$

In DeepFlowFlex, the feature tensor introduced in Sec. 4.3 is fed into the multi-layer GLSTM to encode the spatiotemporal dependencies and the encoding output is further fed into a deep residual networks implemented by GCNN to further model the long-distance spatial dependencies, which is explained as follows.

4.4.2 Residual Learning Module

In this module, we implement deep residual networks with GCNN to learn the far-away spatiotemporal dependencies induced by the modern and convenient transportation. Residual learning is proposed to build a much deeper structure of neural networks and has shown excellent performance on many visual recognition tasks [36]. It utilizes "shortcut connections" [37] to learn the residual mapping instead of learning the desired underlying mapping directly, making the training process easier to optimize. Despite of its popularity, the effectiveness has not been validated on GCNNs.

In our module, the residual unit or ResGCNN is implemented by two stacked "GCNN + ReLU" as Fig. 8 illustrates. The input for the l^{th} residual unit is denoted as $x^{(l)}$ and the output of l^{th} residual unit is $x^{(l+1)}$. For the l^{th} residual unit, the residual function $\mathcal{F}$ for optimizing is defined as:

$$x^{(l+1)} = x^{(l)} + \mathcal{F}(x^{(l)}) \quad (11)$$

Putting it together, the encoding output as Sec. 4.4.1 describes, is denoted as $x^{(0)} \in \mathbb{R}^{|V|\times f_L}$, where f_L is the count of hidden units in the GLSTM layers. $x^{(0)}$ will first be fed into a "GCNN+ReLU" layer with filter number as f_C and the output is $x^{(1)} \in \mathcal{R}^{|V|\times f_C}$. The aim of the first layer is to ensure the dimension of input and output of residual unit equal. Then the output $x^{(1)}$ will be fed into L-stacked residual units with f_C as the filter number. Finally, $x^{(L+1)}$ is input into another "GCNN+ReLU" with filter number as 2 to output the final prediction for inflow and outflow.

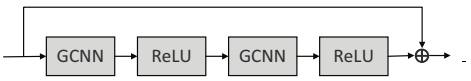


Fig. 8: Structure of a residual unit.

TABLE 2: The statistics of unregular partition of cities.

Statistics	CityA	CityB
Regions Count	72	49
Max Area of Regions (km^2)	49.1	81.0
Min Area of Regions (km^2)	0.6	1.4
Average Area of Regions (km^2)	6.3	16.0
Max Covered Cell Tower Count of Regions	270	153
Min Covered Cell Tower Count of Regions	3	6
Average Covered Cell Tower Count of Regions	53.0	53.3

5 Evaluation

In this section, we present the evaluations of DeepFlowFlex with both grid-based and non-grid-based partitions.

5.1 Experiment Setting

5.1.1 Datasets

- **CityA Cellular Dataset (CityA).** CityA is one metropolis in northern China. The CityA cellular dataset contains cellular data usage traces of 1.5 million users monitored at 5.2 thousand cell towers from Dec 5th, 2016-Feb 4th, 2017.
- **CityB Cellular Dataset (CityB).** CityB is one big industrial city in China. The CityB cellular dataset contains cellular data usage traces of 0.9 million users monitored at 2.5 thousand cell towers from Dec 5th, 2016-Feb 4th, 2017.

Each data record is preprocessed as in Sec. 3.1. For grid-based partition, CityA is partitioned into 10×10 regions and CityB into 8×8 regions. For non-grid-based partition, we use the township-level administrative divisions to partition CityA into 72 districts and CityB into 49 districts as Fig. 4b and Fig. 4e shows. The inflows and outflows of each region are aggregated as in Definition 2. We normalize the crowd flows for all regions to [0, 1] by applying Max-Min normalization on the dataset. As with previous works on crowd flow prediction [1] [5] [14], we predict crowd flows at a time resolution of half an hour. In the evaluation, the prediction values are re-scaled to normal values and compared with the ground truth. We take 80% of the data as Train Set and the rest 20% as Test Set. Validation Set accounts for a proportion of 20% of Train Set if needed.

To validate the applicability of DeepFlowFlex on other data sources, we also evaluate the performance of DeepFlowFlex on two public GPS trace datasets used for grid-based partition [1] and non-grid-based partition [5].

5.1.2 Metric

We use Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) to quantify the performance of each method as below:

$$RMSE = \sqrt{\frac{1}{|\Theta| \times |T| \times |R|} \sum_{\theta,t,r} (x_t^\theta(r) - \hat{x}_t^\theta(r))^2} \quad (12)$$

$$MAPE = \frac{1}{|\Theta| \times |T| \times |R|} \sum_{\theta,t,r} |\frac{\hat{x}_t^\theta(r) - x_t^\theta(r)}{x_t^\theta(r)}| \quad (13)$$

where $\theta \in \Theta = \{Inflow, Outflow\}$ is the flow types, $t \in T = \{1, 2, \cdots, N\}$ is timestamp in test data, $r \in R = \{r_1, r_2, \cdots, r_{N_r}\}$ is a region, x and $\hat{x}$ are the ground truth and prediction value, respectively.

5.1.3 Baselines

We compare our method with the following baselines.

- **HA.** Historical Average (HA) predicts the crowd flows at a certain time by averaging the volumes of crowd flows at the same time of past days.
- **ARIMA.** The Auto-Regressive Integrated Moving Average (ARIMA) model [13] is commonly used for modeling time series behaviors and has been widely adopted in time series prediction [38].
- **LSTM.** Long-Short Term Memory (LSTM) [34] is a Recurrent Neural Network (RNN) architecture. Unlike traditional RNNs, LSTM uses "gates" instead of activation functions, making it suitable to learn from experience to classify, process and predict time series when there are very long time lags of unknown sizes between important events.
- **Prophet.** Prophet [39] is an open-source tool released by Facebook in 2017. It is based on an additive model where non-linear trends are fit with yearly and weekly seasonality, plus holidays.
- **STARMA.** Space Time Auto-Regressive Moving Average (STARMA) [40] is a more advanced spatiotemporal model, which can capture the spatial dependency among regions and their k^th-order neighbors.
- **ST-ResNet.** ST-ResNet [1] is the state-of-the-art deep-learning based crowd flow prediction scheme. It uses residual neural networks to model the spatiotemporal pattern of crowd flows on *grid* partition.
- **FCCF.** FCCF [5] is the state-of-the-art model for crowd flow prediction on *non-grid* partition. It applies Intrinsic Gaussian Markov Random Fields (IGMRF) to capture the temporal information in the crowd flows and utilizes Bayesian network transit model to capture inter-region dependence.

To demonstrate the effectiveness of each module in our DeepFlowFlex model, we also compare the performance of the following variants of DeepFlowFlex.

- **GLSTM-FC.** This variant feeds closeness features into GLSTM and the output of GLSTM is input to Fully Connected (FC) layers. It does not take period and speed features into consideration.
- **GLSTM-GCNN.** This variant feeds closeness features into GLSTM and the output of GLSTM is input to GCNN layers.
- **GLSTM-ResGCNN.** This variant feeds closeness features into GLSTM and the output of GLSTM is input to ResGCNN layers.
- **GLSTM-ResGCNN-PERIOD.** This variant feeds closeness and period features into GLSTM and the output of GLSTM is input to ResGCNN layers.
- **DeepFlowFlex.** DeepFlowFlex is our final model that feeds closeness, period and speed features into GLSTM, whose output is the input to ResGCNN layers.

5.1.4 Implementation

We implement ARIMA using the "forecast" R package [41]. The package automatically selects the best model parameters based on the given order constraints. We utilize the "starma" R package [42] to implement STARMA model and take a list of lagged neighbors lists as input. The first-order neighbors of each region is defined as the regions sharing one or more boundary point with it. We implement Prophet with the "prophet" open-source library and consider daily seasonality and weekly seasonality in the model. We build a two-layer LSTM model with "linear" activation functions. The parameters are set as hidden units = 1024, time step=4, dropout=0.8. We apply Adam Optimizer [43] to minimize the MSE criterion.

We directly use the open-source code[1] for ST-ResNet [1] and FCCF [5]. Since ST-ResNet only works on grid-based partition and FCCF mainly focus on non-grid-based partition, we only evaluate ST-ResNet for grid-based partition and FCCF for non-grid-based partition. For ST-ResNet, we train the model with 3×3 filters with the filter number, layer number, learning rate best tuned. FCCF is trained in the same way as [14]. Since our dataset does not contain weather information, we set weather to zero for any time. Our DeepFlowFlex model and its variants are implemented on TensorFlow [44] with time step S, the layer numbers of GLSTM and ResGCNN, the hidden unit number of GLSTM layer, and the filter number of ResGCNN as hyper-parameters to be tuned.

All experiments use the same Train Set and Test Set settings and are run on a Centos machine with Intel Xeon E5-2620@2.10GHz CPU and K40C 12GB GPU.

5.2 Performance Comparison

5.2.1 Overall accuracy

We evaluate different prediction methods on both CityA and the CityB datasets. Fig. 9 shows the predicted inflows and outflows by DeepFlowFlex and the ground truth at a randomly chosen region of grid-based partition or non-grid-based partition on both datasets. In can be observed that the predictions match with the trend of the actual crowd flows in all cases. The predictions are also close to the actual values in transition points.

Table 3 summarizes the performance of different prediction schemes on the two datasets with grid-based partition and non-grid-based partition. All variants of DeepFlowFlex significantly outperform HA, ARIMA, Prophet, LSTM, STARMA. One variant, GLSTM-FC, is slightly worse than the state-of-art methods (*i.e.*, ST-ResNet on grid-based partition, FCCF on non-grid-based partition). However, other variants of DeepFlowFlex perform significantly better than the state-of-art methods. Among the baselines, HA and ARIMA perform poorly as they rely purely on historical

1. Note that ST-ResNet [1] and FCCF [5] rely on data fusion *e.g.*, weather and event data for optimal performance. Since such datasets are accessible during the period of our cellular data collection, we implement these two models without the parts for integrating weather and event information.

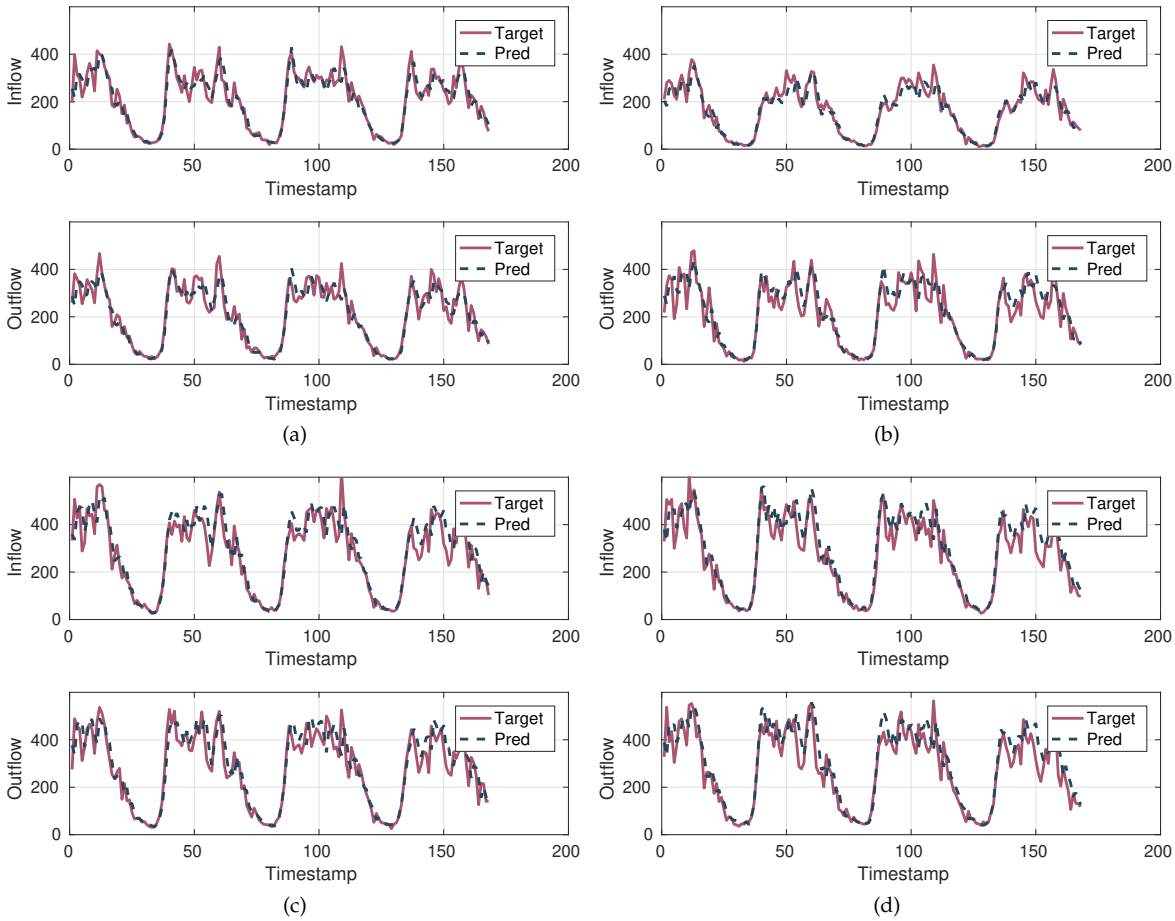


Fig. 9: Predicted and actual crowd flows over time of (a) grid-base partition in CityA, (b) non-grid-based partition in CityA, (c) grid-based partition in CityB, (d) non-grid-based partition in CityB.

values for prediction. LSTM and Prophet are more advanced time series models. STARMA considers spatial dependencies. All of them achieve a better performance than HA and ARIMA. ST-ResNet achieves the best performance among baselines on grid-based partition. FCCF achieves best performance among baselines on non-grid-based partition.

Among the variants of DeepFlowFlex, GLSTM-GCNN achieves a lower RMSE (6.6%, 8.5%, 6.7%, 4.2%) and MAPE (9.4%, 5.3%, 2.5%, 6.4%) than GLSTM-FC. This demonstrates the effectiveness of learning the spatial dependency with GCNN. There is a higher performance gain with ResGCNN, indicating the necessity to leverage residual learning to model long-distance spatial dependencies. By accounting for the influence of period, we obtain an additional improvement of 2.7%, 2.8%, 2.4%, 2.1% in RMSE and 2.4%, 2.3%, 1.6%, 6.8% over GLSTM-ResGCNN. And our final model DeepFlowFlex achieves the best result for explicitly considering the speed domain, which is overlooked in existing works [1] [5] [14].

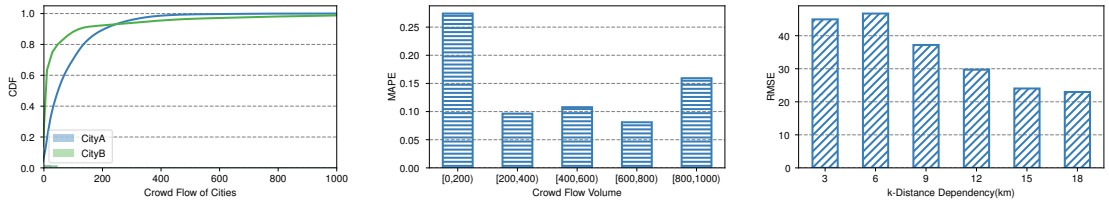
Furthermore, DeepFlowFlex achieves a 13.6% (RMSE) and 12.9% (MAPE), 8.7% (RMSE) and 15.1% (MAPE) improvement over ST-ResNet on CityA and CityB with grid-based partition, respectively. This performance improvement is because DeepFlowFlex utilizes GLSTM to encode the temporal dependencies and extract speed information. For non-grid-based partition, DeepFlowFlex achieves 12.9% (RMSE) and 20.7 %(MAPE), 11.8% (RMSE) and 9.3% (MAPE) improvement over FCCF on CityA and CityB, respectively. Compared with FCCF, DeepFlowFlex utilizes GLSTM and ResGCNN to better characterize the complex spatiotemporal dependencies of crowd flows.

Since we mainly focus on crowd flows defined on non-grid-based partitions in this work, we evaluate our works on crowd flows with non-grid-based partitions in CityA and CityB in next sections. For the convenience of evaluations, we ignore HA and ARIMA for their poor performance. Since ST-ResNet does not work on non-grid-based partitions, we ignore ST-ResNet, too.

Mention that due to the uneven distribution of crowd flows in different regions as shown in Fig. 10a, it is imperfect to evaluate the performance just from RMSE and MAPE. As a result, most works[1][5] don't report MAPE metric. For example, Fig. 10b shows the different performance with crowd flow volume increasing for CityA/Non-grid from MAPE perspective. We find that for huge crowd flow (*i.e.*, $[600, 800)$) which is much more important for city planning and traffic management, our method can achieve 8.10% MAPE, smaller than 18.03% of MAPE for all test data. Therefore, although more than 20% MAPE seems imperfect result, we believe our method will benefit city planning and

TABLE 3: Performance of baselines, DeepFlowFlex and its variants.

Method	CityA/Non-grid		CityA/Grid		CityB/Non-grid		CityB/Grid	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
HA	29.02	23.03%	36.08	25.82%	42.89	23.6%	46.98	23.6%
ARIMA	28.02	32.08%	33.30	56.32%	40.09	22.20%	44.49	27.12%
Prophet	27.65	23.43%	33.14	22.20%	38.08	23.45%	43.01	25.11%
LSTM	27.29	23.85%	33.28	26.78%	38.53	24.37%	43.02	24.96%
STARMA	27.33	23.06%	32.72	25.62%	38.74	23.40%	43.45	23.31%
ST-ResNet	-	-	31.18	24.49%	-	-	38.76	24.25%
FCCF	25.35	22.74%	-	-	35.29	23.63%	-	-
GLSTM-FC	25.86	22.83%	32.09	25.08%	36.47	23.68%	39.81	24.67%
GLSTM-GCNN	24.15	20.69%	29.36	23.75%	34.01	23.09%	38.12	23.09%
GLSTM-ResGCNN	23.33	19.35%	28.79	23.15%	32.71	22.73%	37.46	22.80%
GLSTM-ResGCNN-PERIOD	22.68	18.88%	27.99	22.62%	31.93	22.40%	36.66	21.24%
DeepFlowFlex	**22.06**	**18.03%**	**26.95**	**21.33%**	**31.11**	**21.44%**	**35.35**	**20.60%**



(a) (b) (c)

Fig. 10: (a) Average prediction errors at different times in CityA and CityB. (b) MAPE of different crowd flow volume for CityA/Non-grid. (c) DeepFlowFlex's performance for k-distance dependencies of CityA/Non-grid.

traffic management.

5.2.2 Performance at Different Times

In this subsection we investigate the prediction performance at different time of DeepFlowFlex and the selected baselines for crowd flows defined on non-grid-based partition in CityA and CityB. Fig. 11a and Fig. 11b show the temporal trend of the prediction errors of DeepFlowFlex and the baselines. During the entire day, DeepFlowFlex mostly performs better than all the baselines in both CityA and CityB. Particularly, DeepFlowFlex shows large improvement at 8:00 and 18:00, which is 25.3% (RMSE), 23.2% (MAPE) at 8:00 and 21.3% (RMSE), 17.6% at 18:00 in CityA and 29.9% (RMSE), 6.2% (MAPE) at 8:00 and 18.9 %(RMSE), 7.4% (MAPE) at 18:00 in CityB. This may be because in modern big cities, many citizens travel long distances to work (at 8:00) and return home (at 18:00). Hence spatial dependencies, especially those between regions far-away are crucial to accurately model the patterns in crowd flows.

5.2.3 Performance in Different Regions

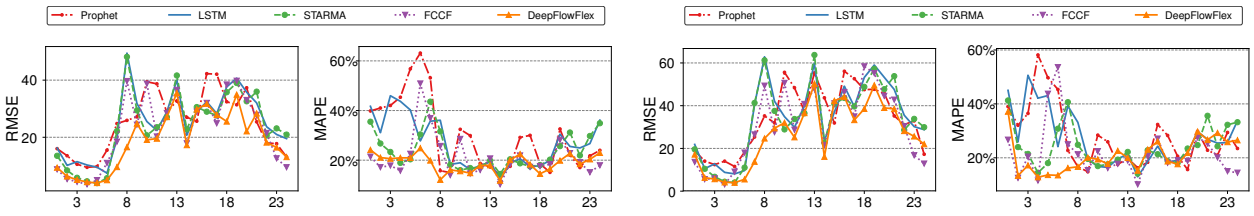
We evaluate the performance of DeepFlowFlex, the variant GLSTM-ResGCNN-PERIOD and selected baselines in three representative regions both in CityA and CityB, which are labeled in Fig. 4a and Fig. 4d as A (transportation hub), B (school), C (business). We compare GLSTM-ResGCNN-PERIOD with DeepFlowFlex to understand the effectiveness of speed domains. The results are shown in Fig. 12a and Fig. 12b where GRP in the legends is the abbreviation of GLSTM-ResGCNN-PERIOD. We observe that both DeepFlowFlex and the variant GLSTM-ResGCNN-PERIOD achieve lower errors in the transport hub, school and business regions. We notice one interesting difference between GLSTM-ResGCNN-PERIOD and DeepFlowFlex. In the transportation hub region in both CityA and CityB, DeepFlowFlex has a bigger improvement than GLSTM-ResGCNN-PERIOD of 10.0% (RMSE), 10.7% (MAPE) in CityA and 6.1% (RMSE), 13.3% (MAPE) in CityB while in the school or business regions, DeepFlowFlex has only a marginal improvement (*i.e.*, 0.4% (RMSE), 1.2% (MAPE) for the business region of CityA and 3.1% (RMSE) and 2.6% (MAPE) for the business region of CityB). The results indicate the effectiveness of speed domain features and regions with a larger portion of high speed crowd flows (*e.g.*, transportation hubs) will benefit more in DeepFlowFlex.

5.2.4 Impact of Hyper-Parameters

There are two important hyper-parameters in DeepFlowFlex, *i.e.*, the time step S and the layer count of ResGCNN.

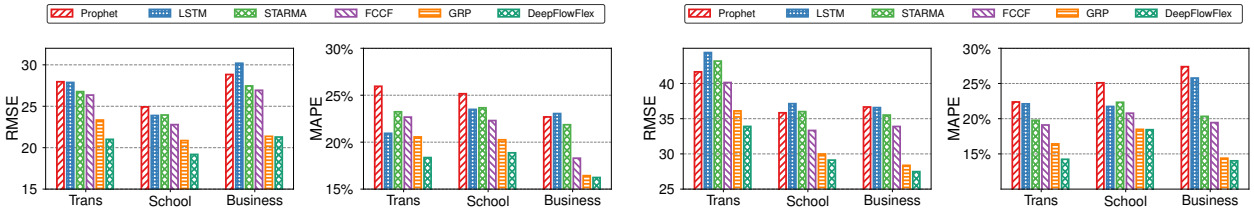
Fig. 13a shows the prediction error with respect to the time step S. When the time step $S = 6$, DeepFlowFlex yields the lowest prediction error. When S increases from 1 to 6, the prediction error decreases monotonously, which indicates the importance of learning long-term temporal dependencies. However, the prediction error starts to increase when S is larger than 6. The potential reason is that the limited dataset size causes over-fitting.

Fig. 13b shows the influence of the count of ResGCNN layers on the prediction error. We also consider DeepFlowFlex implemented with GCNN to verify the effectiveness of ResGCNN. In the experiment, the only difference between ResGCNN and GCNN in DeepFlowFlex is



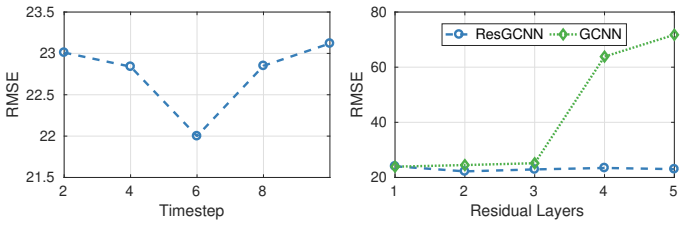
(a) CityA (b) CityB

Fig. 11: Average prediction errors at different times in CityA and CityB.



(a) CityA (b) CityB

Fig. 12: Prediction errors in regions of different functions in CityA and CityB.



(a) (b)

Fig. 13: Impact of hyper-parameters.

TABLE 4: Performance on GPS Datasets .

Methods	BJ-1	BJ-2
ST-ResNet	16.69	-
FCCF	-	14.17
GLSTM-ResGCNN-Period	14.58	12.20

whether the model has short-cut connections. From the results, DeepFlowFlex achieves the best performance with two layer ResGCNN and the prediction error keeps stable when stacking more ResGCNN layer. However, a deeper structure implemented with GCNN will induce significantly large errors. Therefore it is necessary to exploit residual learning in training deep GCNN models.

5.2.5 *Applicability on GPS Datasets*

We further evaluate a variant of our DeepFlowFlex model, GLSTM-ResGCNN-Period on the GPS datasets used in the state-of-art works [1] [5] since the datasets have no speed information. The Beijing taxi GPS dataset (BJ-1) used by ST-ResNet [1] covers 12 months of Beijing taxi GPS data and the city map is segmented into 32×32 grids. The Beijing taxi GPS dataset (BJ-2) used by FCCF [5] covers the Beijing taxi GPS data from March 1th, 2015 to June 28th, 2015 and the city map is partitioned into 26 high-level irregular regions.

We train our proposed GLSTM-ResGCNN-Period only with GPS data and the graph extracted from region partition. Table 4 summarizes the RMSE of prediction results and the best reported result in ST-ResNet and FCCF. Our proposed model achieves 12.6% (RMSE) improvement than ST-ResNet on BJ-1 and 13.9% improvement than FCCF on BJ-2. The results indicate that our proposed hybrid model with GLSTM to capture long-term temporal dependencies and ResGCNN to capture long-distance spatial dependencies is able to better model the complex spatiotemporal dependencies of crowd flows than the state-of-the-art models. Furthermore, our model is not only applicable in cellular data, but is also effective on GPS data.

5.2.6 *Impact of Long-distance Dependencies*

In this subsection we investigate the effectiveness of DeepFlowFlex to model long-distance dependencies, which is totally necessary in modern cities where quantities of citizens travel long distances to commute. In practice, DeepFlowFlex predicts the crowd flows of each regions with k-distance dependencies, that is, we only utilize the information of regions (*i.e.*, other regions' crowd flows are set to zero) which have smaller distance than k *km* from the target region, the distance of two region is estimated as the distance of their centroids. Fig. 10c shows the prediction error of DeepFlowFlex with different k for CityA/Non-grid. We find that if only considering the close-distance dependencies, it is hard to predict highly precisely, for example, the case of 3-distance dependencies is more than twice worse than 18-distance dependencies. Our model models the spatial dependencies among the whole city achives the best performance.

6 Conclusion

In this work, we propose DeepFlowFlex, a graph-based deep spatiotemporal model for citywide crowd flow predic-

tion on flexible region partition. DeepFlowFlex is devised to accurately predict crowd flows covering a large population and travelling in various transportation modes. Analysis with large-scale urban cellular datasets demonstrates strong dependencies of crowd flows in the space, time and speed domains. DeepFlowFlex exploits graph-based models to operate on irregular shaped regions. Specifically, DeepFlowFlex utilizes graph convolutional long short-term memory networks to model spatiotemporal (especially long-term) dependencies. It also employs residual learning to model spatial dependencies, particularly long-distance dependencies.

Experimental results validate the effectiveness of DeepFlowFlex on both grid-based and non-grid-based region partition. We envision DeepFlowFlex will offer an accurate, adaptive, and scalable macroscopic view of citywide mobility for various urban computing applications. In the future, we plan to predict both regular and abnormal crowd flows by incorporating heterogenous data sources.

7 Acknowledgement

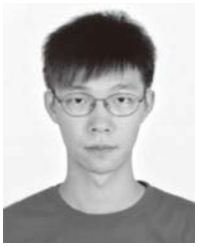
This work is supported in part by the National Key Research Plan under grant No. 2016YFC0700100, NSFC under grant 61832010, 61632008, 61672319, 61872081, Microsoft Research Asia, and Tsinghua University Initiative Scientific Research Program.

References

[1] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. AAAI*, 2017, pp. 1655–1661.
[2] F. Calabrese, L. Ferrari, and V. D. Blondel, "Urban sensing using mobile phone network data: a survey of research," *ACM Computing Surveys*, vol. 47, no. 2, p. 25, 2015.
[3] H. Poonawala, V. Kolar, S. Blandin, L. Wynter, and S. Sahu, "Singapore in motion: Insights on public transport service level through farecard and mobile data analytics," in *Proc. KDD*, 2016, pp. 589–598.
[4] Y. Tong and Y. e. Chen, "The simpler the better: A unified approach to predicting original taxi demands on large-scale online platforms," in *Proc. KDD*, 2017.
[5] M. X. Hoang, Y. Zheng, and A. K. Singh, "Fccf: Forecasting citywide crowd flows based on big data," in *Proc. SIGSPATIAL*, 2016, pp. 6:1–6:10.
[6] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.
[7] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.
[8] A. Hess, K. A. Hummel, W. N. Gansterer, and G. Haring, "Data-driven human mobility modeling: a survey and engineering guidance for mobile networking," *ACM Computing Surveys*, vol. 48, no. 3, p. 38, 2016.
[9] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "Slaw: A new mobility model for human walks," in *Proc. INFOCOM*, 2009, pp. 855–863.
[10] F. Simini, M. C. González, A. Maritan, and A.-L. Barabási, "A universal model for mobility and migration patterns," *Nature*, vol. 484, no. 7392, pp. 96–100, 2012.
[11] Y. Qiao, Y. Cheng, J. Yang, J. Liu, and N. Kato, "A mobility analytical framework for big mobile data in densely populated area," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1443–1455, 2017.
[12] Z. Fan, X. Song, R. Shibasaki, and R. Adachi, "Citymomentum: an online approach for crowd behavior prediction at a citywide level," in *Proc. UbiComp*, 2015, pp. 559–569.
[13] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[14] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *Proc. SIGSPATIAL*, 2016, pp. 92:1–92:4.
[15] F. Khan, *LTE for 4G mobile broadband: air interface technologies and performance*. Cambridge university press, 2009.
[16] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of lora: Experiences from a large-scale measurement study," *ACM Transactions on Sensor Networks (TOSN)*, vol. 15, no. 2, p. 16, 2019.
[17] T. Liu, Z. Yang, Y. Zhao, C. Wu, Z. Zhou, and Y. Liu, "Temporal understanding of human mobility: A multi-time scale analysis," *PloS one*, vol. 13, no. 11, p. e0207697, 2018.
[18] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2015.
[19] X. Zhang, Z. Yang, Y. Liu, and S. Tang, "On reliable task assignment for spatial crowdsourcing," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 174–186, 2016.
[20] C. Song, T. Koren, P. Wang, and A.-L. Barabási, "Modelling the scaling properties of human mobility," *Nature Physics*, vol. 6, no. 10, pp. 818–823, 2010.
[21] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, p. 38, 2014.
[22] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
[23] F. Calabrese, G. Di Lorenzo, L. Liu, and C. Ratti, "Estimating origin-destination flows using mobile phone location data," *IEEE Pervasive Computing*, vol. 10, no. 4, pp. 36–44, 2011.
[24] Q. Xu, A. Gerber, Z. M. Mao, and J. Pang, "Acculoc: practical localization of performance measurements in 3g networks," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 183–196.
[25] I. Leontiadis, A. Lima, H. Kwak, R. Stanojevic, D. Wetherall, and K. Papagiannaki, "From cells to streets: Estimating mobile paths with cellular-side data," in *Proc. ACM CoNEXT*, 2014, pp. 121–132.
[26] H. Wu, W. Sun, B. Zheng, L. Yang, and W. Zhou, "Clsters: A general system for reducing errors of trajectories under challenging localization situations," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 115, 2017.
[27] R. Mohamed, H. Aly, and M. Youssef, "Accurate real-time map matching for challenging environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 847–857, 2017.
[28] Y. Ding, Y. Li, K. Deng, H. Tan, M. Yuan, and L. M. Ni, "Detecting and analyzing urban regions with high impact of weather change on transport," *IEEE Transactions on Big Data*, vol. 3, no. 2, pp. 126–139, 2017.
[29] D. Zhang, J. Huang, Y. Li, F. Zhang, C. Xu, and T. He, "Exploring human mobility with multi-source data at extremely large metropolitan scales," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 201–212.
[30] H. Li, C. A. Calder, and N. Cressie, "Beyond moran's i: testing for spatial dependence based on the spatial autoregressive model," *Geographical Analysis*, vol. 39, no. 4, pp. 357–375, 2007.
[31] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv preprint arXiv:1612.07659*, 2016.
[32] 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 *Advances in neural information processing systems*, 2015, pp. 802–810.
[33] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, 2016, pp. 3837–3845.
[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[35] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.
[37] C. M. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

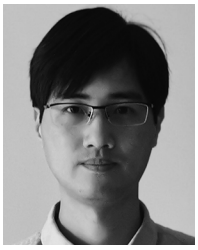
[38] J. D. Hamilton, *Time series analysis*. Princeton University Press, 1994, vol. 2.
[39] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, no. just-accepted, 2017.
[40] P. E. Pfeifer and S. J. Deutrch, "A three-stage iterative procedure for space-time modeling phillip," *Technometrics*, vol. 22, no. 1, pp. 35–47, 1980.
[41] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software*, vol. 26, no. 3, pp. 1–22, 2008.
[42] F. Cheysson, "Modelling space time autoregressive moving average (starma) processes," 2016. [Online]. Available: https://cran.r-project.org/web/packages/starma/
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[44] M. Abadi and A. A. *et al.*., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

Xu Wang received his B.E. degree in School of Software from Tsinghua University in 2015. He is currently a PhD student in School of Software at Tsinghua University. He is a member of the Tsinghua National Lab for Information Science and Technology. His research interests include mobile computing and machine learning.



Zimu Zhou is currently a post-doc researcher at ETH Zurich. He received his Ph.D. degree from Hong Kong University of Science and Technology in 2015. His research interests include ubiquitous computing and mobile systems.

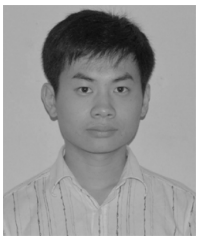
Yi Zhao received his B.E. degree in School of Software from Tsinghua University in 2017. He is currently a PhD student in School of Software at Tsinghua University. He is a member of the Tsinghua National Lab for Information Science and Technology. His research interests include mobile computing and human mobility.



Xinglin Zhang received the B.E. degree from the School of Software, Sun Yat-sen University, in 2010 and the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology in 2014. He is currently an Associate Research Fellow with the School of Computer Science and Engineering, South China University of Technology, China. His research interests include wireless ad-hoc/sensor networks, mobile computing and crowdsourcing.

Kai Xing received the M.S. and Ph.D. degree in computer science from The George Washington University in 2006 and 2009, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, University of Science and Technology of China. His current research interests include cyber-physical networking systems, mobile computing, in-network information processing, and network security.

Fu Xiao received the Ph.D. degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2007. He is currently a Professor and a Ph.D. Supervisor with the School of Computers, Nanjing University of Posts and Telecommunications. His main research interests are wireless sensor networks and mobile computing.



Zheng Yang received a B.E. degree in computer science from Tsinghua University in 2006 and a Ph.D. degree in computer science from Hong Kong University of Science and Technology in 2010. He is currently a professor in Tsinghua University. His main research interests include wireless ad-hoc/sensor networks and mobile computing.

Yunhao Liu received the BS degree from the Automation Department, Tsinghua University, and the MA degree from Beijing Foreign Studies University, China. He received the MS and PhD degrees from Computer Science and Engineering, Michigan State University. He is now the ChangJiang professor at Tsinghua University. His research interests include sensor network and IoT, localization, RFID, distributed systems, and cloud computing. He is fellow of the ACM and IEEE.