DeepCrowd: A Deep Model for Large-Scale Citywide Crowd Density and Flow Prediction

Renhe Jiang, Zekun Cai, Zhaonan Wang, Chuang Yang, Zipei Fan, Quanjun Chen, Kota Tsubouchi, Xuan Song, and Ryosuke Shibasaki

Abstract—Predicting the density and flow of the crowd or traffic at a citywide level becomes possible by using the big data and cutting-edge AI technologies. It has been a very significant research topic with high social impact, which can be widely applied to emergency management, traffic regulation, and urban planning. In particular, by meshing a large urban area to a number of fine-grained mesh-grids, citywide crowd and traffic information in a continuous time period can be represented with 4D tensor (Timestep, Height, Width, Channel). Based on this idea, a series of methods have been proposed to address grid-based prediction for citywide crowd and traffic. In this study, we revisit the density and in-out flow prediction problem and publish a new aggregated human mobility dataset generated from a real-world smartphone application. Comparing with the existing ones, our dataset holds several advantages including large mesh-grid number, fine-grained mesh size, and high user sample. Towards this large-scale crowd dataset, we propose a novel deep learning model called DeepCrowd by designing pyramid architectures and high-dimensional attention mechanism based on Convolutional LSTM. Lastly, thorough and comprehensive performance evaluations are conducted to demonstrate the superiority of the proposed DeepCrowd comparing to multiple state-of-the-art methods.

Index Terms—Crowd density, crowd flow, urban computing, deep learning, ubiquitous and mobile computing

1 Introduction

Nowadays, massive urban human mobility data are being generated from mobile phones, car navigation systems, and traffic sensors. Many studies have analyzed these big mobility data with cutting-edge technologies, which have been summarized as urban computing. In particular, crowd or traffic prediction at a citywide level becomes an emerging topic in both academia and industry, as it can be of great importance for emergency management, traffic regulation, and urban planning. As illustrated in Fig. 1, by meshing a large urban area to a number of fine-grained mesh-grids, citywide crowd density in a continuous time period can be represented with a four-dimensional tensor $\mathbb{R}^{Timestep \times Height \times Width \times Channel}$ in an analogous manner to video data, where each $Timestep$ can be seen as one video frame, $Height$, $Width$ is two-dimensional index for mesh-grids, and each $Channel$ stores an aggregated scalar value for each mesh-grid. Following this representation, as shown in Table 1, a series of studies [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] have been conducted to address grid-based urban computing problems such as crowd in-out flow prediction, taxi demand prediction, and traffic accident prediction. These forecasts can be provided to governments (e.g., police) and public service operators (e.g., subway or bus company) to protect people's safety or maintain the operation of public infrastructures under event situation (e.g., New Year Countdown); to ride-sharing companies like Uber and Didi Chuxing to more effectively dispatch the taxis; to web mapping services like Yahoo Japan Map (https://map.yahoo.co.jp/congestion?lat=35.67299&lon=139.74330&zoom=12&maptype=basic) and Itsumo-Navi (https://lab.its-mo.com/densitymap/) to improve the functionality of crowd density map service.

Specifically, we focus on the grid-based prediction on citywide crowd density and in-out flow. The crowd density prediction is to predict how many people will be in each mesh-grid at the next timestamp, and the crowd in-out flow prediction is to predict how many people will flow into or out from each mesh-grid in next time interval. Both tasks take multiple steps of historical observations as input and output the next-step prediction result. Although the deep models in Table 1 have been proposed to address such tasks, their actual effects on citywide crowd density and in-out flow prediction at a citywide level are still not well validated on large-scale and high-quality datasets. As shown in Table 2, the datasets used in most of the works so far are originally generated based on taxi or bicycle Origin-Destination (O-D) data, which don't cover

- *Renhe Jiang is with the Center for Spatial Information Science, The University of Tokyo, Tokyo 113-8654, Japan, with the Information Technology Center, The University of Tokyo, Tokyo 113-8654, Japan, and also with SUSTech-UTokyo Joint Research Center on Super Smart City, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China. E-mail: jiangrh@csis.u-tokyo.ac.jp.*
- *Zekun Cai, Zhaonan Wang, Chuang Yang, Zipei Fan, Quanjun Chen, and Ryosuke Shibasaki are with the Center for Spatial Information Science, The University of Tokyo, Tokyo 113-8654, Japan. E-mail: {caizekun, znwang, chuang.yang, fanzipei0725, chen1990, shiba}@csis.u-tokyo.ac.jp.*
- *Kota Tsubouchi is with the Yahoo Japan Corporation, Tokyo 102-8282, Japan. E-mail: ktsubouc@yahoo-corp.jp.*
- *Xuan Song is with the SUSTech-UTokyo Joint Research Center on Super Smart City, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, and with the Center for Spatial Information Science, The University of Tokyo, Tokyo 113-8654, Japan. E-mail: songxuan@csis.u-tokyo.ac.jp.*

Manuscript received 16 April 2020; revised 18 February 2021; accepted 27 April 2021. Date of publication 3 May 2021; date of current version 7 December 2022. (Corresponding author: Xuan Song.)
Recommended for acceptance by X. Xiao.
Digital Object Identifier no. 10.1109/TKDE.2021.3077056

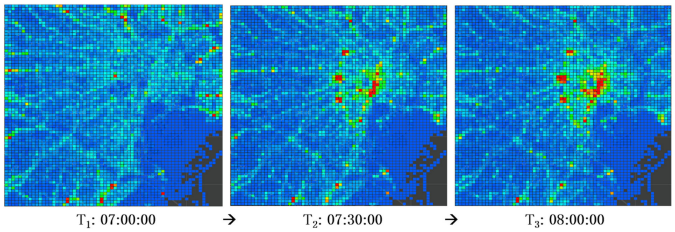


Fig. 1. Citywide crowd density in Tokyo within 1 hour.

and reflect the citywide crowd density and flow. Moreover, the scales of the datasets are small due to the limitations in terms of the number of mesh-grids, the granularity of mesh size, and the number of user samples. For example, TaxiNYC data is generated from bicycle O-D data, that only has 20×10 mesh-grids with 1km. A large-scale dataset for citywide crowd density and crowd in-out flow prediction tasks is needed to validate the actual performance of the deep spatiotemporal models.

Thus, in this study, we first publish a new crowd flow dataset at a citywide scale with high-quality. Our new dataset is created using the GPS log data collected from a popular smartphone app published by Yahoo Japan Corporation, which can well reflect the real-world crowd flow information. After doing interpolation to the raw GPS trajectory data and aggregation with citywide mesh grids, 4D tensors $\mathbb{R}^{Timestep \times Height \times Width \times Channel}$ representing citywide crowd density and in-out flow can be generated. The scale of our new dataset is large on the following aspects: (1) larger spatial area; (2) finer mesh size; (3) higher user sample. The existing models are found to deliver a weaker performance on our new dataset than the small-scale datasets, because as the spatial domain becomes 4∼16 times larger than before, the complexity of the problem also increases a lot. Then we design a new deep learning model called DeepCrowd to capture the large-scale citywide spatiotemporal dependencies from the 4D sequential data.

To pursue the computation efficiency for citywide prediction, instead of predicting the crowd density and in-out flow for each mesh-grid, we employ a "citywide to citywide" computation mechanism. Convolutional LSTM (ConvLSTM) [13] is utilized as the basic component of our prediction model to handle our high-dimensional data. In particular, to capture the large spatial dependency at a citywide scale, ConvLSTMs are stacked in a pyramid and hierarchical architecture for better utilizing low-resolution and high-resolution feature maps. To automatically capture the temporal dependency from historical observations in different stages, 4D attention block is implemented based on ConvLSTM to obtain an overall representation of the inputs. Moreover, we employ an early-fusion mechanism to better utilized the meta/external information. It should be noted that to validate the pure ability of grid-based modeling on spatiotemporal data, unified objective function is adopted for model training, and extra data source such as Point-Of-Interest(POI) data and the related processing module are excluded from the models. Finally, we evaluate our model with four metrics including MSE, RMSE, MAE, and MAPE, and do a thorough comparison with the state-of-the-arts listed in Table 1. In summary, our work has four-fold contributions as follows:

- We propose a complete framework from preprocessing to deep learning models for predicting citywide crowd density and crowd in-out flow.
- We generate and publish new datasets called BousaiTYO and BousaiOSA for crowd density and in-out flow prediction from a real-world smartphone app, which have larger scale and higher quality than the previous datasets. Our dataset will be officially published after this paper is accepted.
- We develop a novel deep learning model called DeepCrowd for large-scale citywide crowd density and in-out flow prediction, by designing pyramid ConvLSTMs, 4D high-dimensional attention block, and early-fusion mechanism.
- We implement multiple state-of-the-art methods, and conduct a thorough evaluation using a series of metrics. The experiment results demonstrate both the effectiveness and efficiency of our citywide crowd prediction model for two big cities Tokyo and Osaka.

The remainder of this paper is organized as follows. In Section 2, we introduce data preprocessing and give problem definition. In Section 3, we provide a description of the datasets. In Section 4, we explain the implemented models. In Section 5, we present the evaluation results. In Section 6, we briefly recap some other related works. In Section 7, we give our conclusion and discuss future work.

2 Preliminary

2.1 Data Preprocessing

Without loss of generality, in this study, we use object to refer to people, vehicle, and bicycle in different GPS data sources. An object's GPS trajectory is represented by a sequence of 3-tuple: ($timestamp$, $latitude$, $longitude$), which be further simplified as a sequence of (t, l)-pair. Object trajectory is stored and indexed by day (i) and object id (o) in the database Γ.

TABLE 1
Summary of the State-of-The-Arts

Model	Reference	Dataset (* means Open)	Prediction Task
ST-ResNet [1]	AAAI 2017	TaxiBJ*, BikeNYC*	Taxi In-Out Flow (Traffic)
DMVST-Net [2]	AAAI 2018	Didi Taxi Request	Taxi Demand (Traffic)
Periodic-CRN [3]	IJCAI 2018	TaxiBJ*, TaxiSG	Taxi Density&Taxi In-Out Flow (Traffic)
Hetero-ConvLSTM [4]	KDD 2018	Vehicle Crash Data*	Traffic Accident (Traffic)
STDN [5]	AAAI 2019	TaxiNYC*, BikeNYC-II*	Taxi&Bike O-D Number (Traffic)
DeepSTN+ [6]	AAAI 2019	MobileBJ, BikeNYC-I*	Crowd&Taxi In-Out Flow (Crowd&Traffic)
MDL [7]	TKDE 2019	TaxiBJ, BikeNYC	Taxi Transition&Taxi In-Out Flow (Traffic)
DeepUrbanEvent [8]	KDD 2019	Konzatsu Toukei	Crowd Density&Crowd Flow (Crowd)

TABLE 2
DataSet Summary

Dataset	Data Type	Mesh Size	H, W	Time Period, Interval	Maximum Value
BousaiTYO	Density, In-Out Flow	450m×450m	80, 80	2017/4/1/-2017/7/9, 0.5 hour	2300*, 1200*
BousaiOSA	Density, In-Out Flow	450m×450m	60, 60	2017/4/1/-2017/7/9, 0.5 hour	1800*, 770*
TaxiBJ	In-Out Flow	unknown	32, 32	inconsecutive 4 parts, 0.5 hour	1292
BikeNYC	In-Out Flow	unknown	16, 8	2014/4/1-2014/9/30, 1 hour	267
BikeNYC-I	In-Out Flow	unknown	21, 12	2014/4/1-2014/9/30, 1 hour	737
BikeNYC-II	In-Out Flow	1km×1km	10, 20	2016/7/1-2016/8/29, 0.5 hour	307
TaxiNYC	In-Out Flow	1km×1km	10, 20	2015/1/1-2015/3/1, 0.5 hour	1289

*According to the company policy of Yahoo Japan Corporation, the most significant two digits are recorded here.

Given a time interval $\Delta\tau$, each object's trajectory on each day Γ_{io} is calibrated to obtain constant sampling rate as follows:

$$\Gamma_{io} = (t_1, l_1), \ldots, (t_k, l_k), \forall j \in [1, k), |t_{j+1} - t_j| = \Delta\tau.$$

Then given the mesh-grids of an urban area $\{g_1, g_2, ..., g_{Height * Width}\}$, trajectory Γ_{io} is mapped onto mesh-grids as follows:

$$\Gamma_{io} = (t_1, g_1), \ldots, (t_k, g_k), \forall j \in [1, k], l_j \in g_j.$$

This preprocessing has been illustrated in Fig. 2. After this, density video and in-out flow video can be aggregated and generated with the processed trajectories with constant sampling rate according to the definitions below.

2.2 Problem Definition

Definition 1 (Density and In-Out Flow): Crowd density at timestamp t in mesh-grid g_m is defined as follows:

$$d_{tm} = |\{o|\Gamma_o.g_t = g_m\}|.$$

According to [9], [10], crowd in-out flow between consecutive timestamps t-1 and t in mesh-grid g_m is defined as follows:

$$f_{tm}^{(in)} = |\{o|\Gamma_o.g_{t-1} \neq g_m \wedge \Gamma_o.g_t = g_m\}|$$

$$f_{tm}^{(out)} = |\{o|\Gamma_o.g_{t-1} = g_m \wedge \Gamma_o.g_t \neq g_m\}|.$$

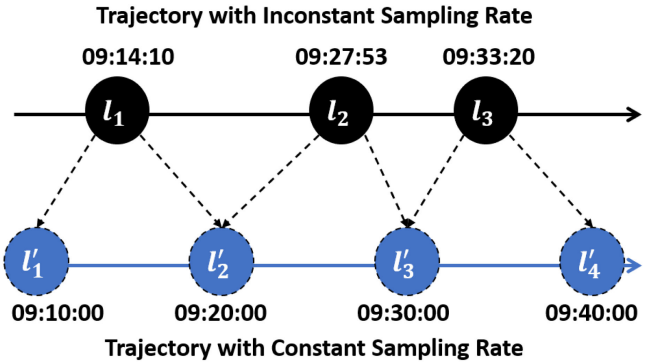


Fig. 2. The illustration of data preprocessing is listed, through which the raw trajectory data with inconstant sampling rate will be calibrated to the trajectories with constant sampling rate. For example, when $\Delta\tau$ is set to 10 minutes, the raw trajectory snippet $\{l_1, l_2, l_3\}$ timestamped at {09:14:10, 09:27:53, ..., 09:51:40} will be converted to a calibrated trajectory snippet $\{l'_1, l'_2, ..., l'_4\}$ constantly timestamped at {09:10:00, 09:20:00, ..., 09:40:00}. The coordinate value of l'_3 is calculated through linear interpolation based on l_2 and l_3.

Then, by representing the mesh-grids with a 2D index (H, W), crowd density and in-out flow video containing T consecutive frames can be represented by a 4D tensor $\mathbb{R}^{T\times H\times W\times C}$, where channel C for density and in-out flow are equal to 1 and 2 respectively. Here, Min-Max normalization is used to scale the scalar values into [0, 1].

Definition 2 (Density and In-Out Flow Prediction): Given historical observations of crowd density and in-out flow $x_d = d_1, \ldots, d_t$, $x_f = f_1, \ldots, f_t$ at timestamp t, building prediction models for the next-step density and in-out flow $y_d = d_{t+1}$, $y_f = f_{t+1}$ is to obtain such parameters θ_d, θ_f that can minimize the objective function $\mathcal{L}(\cdot)$ respectively as follows:

$$\theta_d = \arg\min_{\theta_d} \mathcal{L}(\hat{Y}_d, Y_d)$$

$$\theta_f = \arg\min_{\theta_f} \mathcal{L}(\hat{Y}_f, Y_f),$$

where Mean Squared Error (MSE) = $||Y\text{-}\hat{Y}||_2^2$ is uniformly utilized as $\mathcal{L}(\cdot)$ in our study to better validate the prediction capability of our proposed deep learning model.

3 Large-Scale Citywide Crowd Dataset

BousaiTYO and BousaiOSA. Yahoo Japan Corporation provides a smartphone application to give early information and warning towards different disasters such as earthquake, rain, snow, and tsunami. Users are required to provide their location information so that Bousai App can precisely send local disaster alerts to the users in relevant areas. Also, real-time GPS trajectory data are anonymously collected under users' consent for real-time notification and research purposes. GPS logs will be generated when the smartphone user stops staying at one location and starts moving by identifying the change of current location. Every day since 2017, the GPS logs are being generated from around 1 million users (approximately 1 percent of the total population of Japan). The file size of each day is about 18 GB, containing approximately 150 million GPS records. Each record includes user ID, timestamp, latitude, and longitude. The sampling rate of each user's GPS data is approximately 20 records per day. We select two big cities in Japan (Tokyo[1] and Osaka[2]) as target urban areas, 100 consecutive days from 2017/4/1/ to 2017/7/9 as target time period. We crop the raw GPS trajectory data in this

1. Tokyo: $Longitude \in [139.50, 139.90]$, $Latitude \in [35.50, 35.82]$
2. Osaka: $Longitude \in [135.35, 135.65]$, $Latitude \in [34.58, 34.82]$

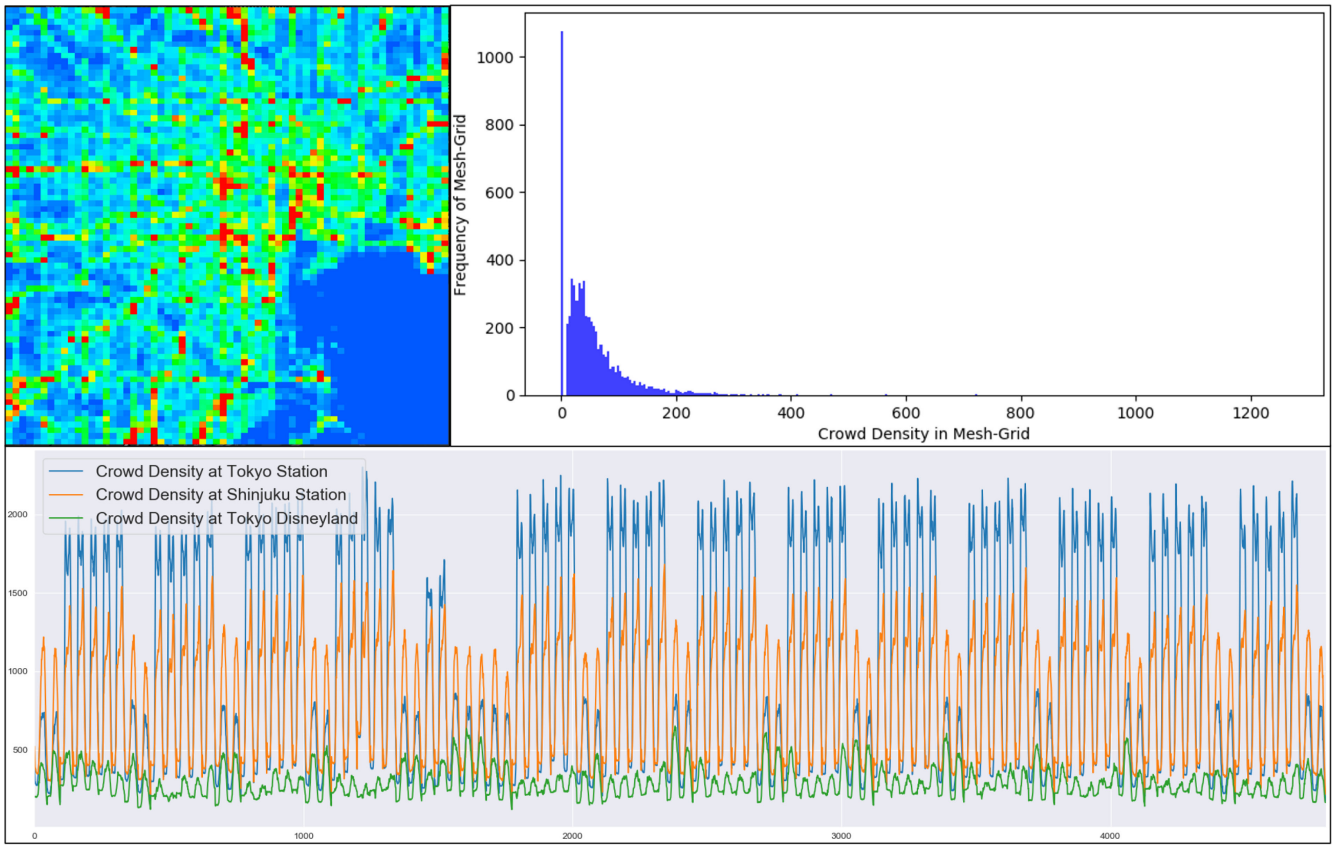


Fig. 3. One snapshot of BousaiTYO density at 2017/4/1 09:00:00 is listed on the upper left, where the red color represents the highest density and the blue color represents the lowest density. The crowd density distribution over the 6,400 mesh-grids is plotted on the upper right. Time-series of crowd density over 100 days (4,800 timestamps) in Tokyo station, Shinjuku station, and Tokyo Disneyland is plotted at the bottom.

spatiotemporal range out. As data preprocessing, we first conduct data cleaning and noise reduction to the raw GPS trajectory data, and then do linear interpolation to make sure each user's 24-hour (00:00∼23:59) GPS log has a constant sampling rate with $\Delta\tau$ equal to 5 minutes. Then we mesh each area with $\Delta Lon.$ = 0.005 $\Delta Lat.$ = 0.004 (approximately 450m×450m), which results 80 × 80 and 60 × 60 mesh-grids respectively for each city. We get one timestamp every 30 minutes, so there are 4800 timestamps (100 days × 48/day) in total. According to Definition 1, crowd density and in-out flow are generated for each timestamp. To further eliminate the concerns of user privacy problem, K-anonymization is conducted by setting the scalar values less than 10 to 0. Finally, for Tokyo dataset BousaiTYO, the density and in-out flow video can be represented by tensor (4800, 80, 80, 1) and (4800, 80, 80, 2) respectively; for Osaka dataset BousaiOSA, the density and in-out flow video can be represented by tensor (4800, 60, 60, 1) and (4800, 60, 60, 2) respectively.

TaxiBJ is taxi in-out flow data used by [1], [3], created based on the taxi GPS data in Beijing from four separate time periods: 2013/7/1-2013/10/30, 2014/3/1-2014/6/30, 2015/3/1-2015/6/30, and 2015/11/1-2016/4/10. *BikeNYC* is bike in-out flow data used by [1], taken from the NYC Bike system from 2014/4/1 to 2014/9/30. Similar datasets *BikeNYC-I*, *BikeNYC-II* were used by [6] and [5] respectively. *TaxiNYC* is taxi in-out flow data used by [5], created from the NYC Taxi data in 2015. The details of our new dataset and the existing ones are given in Table 2. Through it, we can see the following advantages of our new dataset:

1) Large mesh-grid number. The H, W is 80,80 for Tokyo and 60,60 for Osaka, which are much larger than the TaxiBJ, BikeNYC and TaxiNYC.
2) Fine-grained mesh size. Our mesh size is set to 450 meters, finer than TaxiNYC and BikeNYC, while the mesh size settings in some datasets are unknown.
3) High user sample. The maximum values of crowd density and crowd in-out flow are much bigger than the existing ones.
4) Density and in-out flow. Our datasets contain both crowd density and in-out flow while other datasets only contain taxi/bike in-out flow.

Most importantly, our dataset can better reflect the real-world citywide crowd as it is generated based on a real-world smartphone app with large active users rather than taxi or bike O-D data. A series of visualization results about BousaiTYO density data have been listed as Fig. 3. The visualization results on the upper side also demonstrate that our datasets have a high crowd density over fine-grained mesh-grids in a large urban area (there are many crowded regions colored with red). The three time-series of crowd density over 100 days (i.e., 4,800 timestamps) in Tokyo station, Shinjuku station, and Tokyo Disneyland all show clear periodical patterns.

4 Deep Learning Model

How to do the citywide crowd density and in-out flow prediction on such large-scale data becomes a new challenge for us. For big cities like Tokyo and Osaka, the crowd

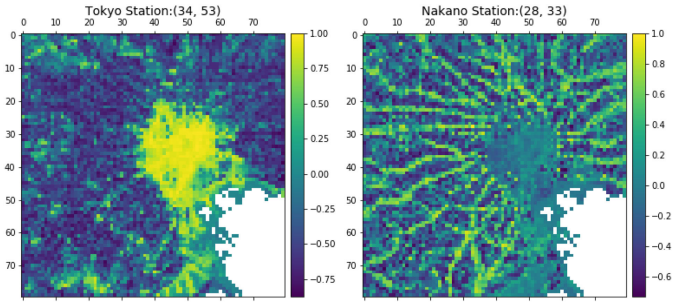


Fig. 4. Left is pearson correlation matrix between the crowd density in Tokyo Station grid (34, 53) and other mesh-grids; Right is pearson correlation matrix between the crowd density in Nakano Station grid (28, 33) and other mesh-grids.

density and in-out flow inside a certain mesh-grid could be highly correlated with the ones in other mesh-grids, as people can transit from one mesh-grid in north to another in south by taking a train within 30 minutes. We select two mesh-grids in Tokyo area to calculate the pearson correlation value between them and other mesh-grids, one is the Tokyo Station grid, the biggest station in Tokyo, another is the Nakano Station grid, a relatively small station in Tokyo. As depicted by Fig. 4, the highly correlated mesh-grids with yellow color distribute over almost the entire urban area (i.e., 80*80 mesh-grids). More importantly, the two stations demonstrate two very different correlation patterns, that is to say, for citywide crowd prediction over each mesh-grid, we need to take the complex and compound dependency over the entire spatial domain into account.

On the other hand, apart from the effectiveness, the computation efficiency is also an indispensable factor for us to consider. In terms of the computation schemes, the state-of-the-art prediction models can be categorized into two types. One is city-unit prediction including ST-ResNet [1], Periodic-CRN [3], DeepSTN+ [6], MDL [7], and DeepUrbanEvent [8]. These models output the prediction result for the entire urban area at a time. Another is grid-unit prediction like DMVST-Net [2] and STDN [5]. The models predict one specific mesh-grid at each time. When the spatial domain or the mesh-grid number of the dataset is small like TaxiNYC (H, W = 10,20) or BikeNYC (H, W = 16,8), the grid-unit models could somehow work under an acceptable efficiency. However, for crowd flow dataset with large spatial domain (huge number of mesh-grids) like BousaiTYO (H, W = 80,80) or BousaiOSA (H, W = 60,60), the models that take each mesh-grid as computation unit could be too cumbersome to work, as the computation complexity of grid-unit model is H*W times larger than city-unit model.

Based on the above, it drives us to design a real citywide model that can work effectively and efficiently to deal with the complex spatiotemporal dependencies on the large-scale crowd data. Thus, in this study, we propose a new deep learning model called DeepCrowd, the feature and structure of which will be described in the following.

4.1 Model Feature

4.1.1 Spatiotemporal Feature

As mentioned above, we take the entire spatial domain as the feature rather than a certain mesh-grid or a couple of

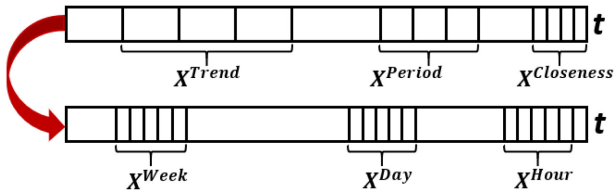


Fig. 5. From {Closeness, Period, Trend} to {Hour, Day, Week}.

mesh-grids. Temporally, the next step of citywide crowd density and in-out flow could be correlated with historical observations from different time periods. ST-ResNet [1] first designs a set of unique features namely *Closeness*, *Period*, and *Trend*, which correspond to *the recent time intervals*, *daily periodicity*, and *weekly trend* respectively. Intuitively, the three parts of the features can be represented by:

$$X^{Closeness} = [X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$$
$$X^{Period} = [X_{t-l_p \times s_p}, X_{t-(l_p-1)\times s_p}, ..., X_{t-s_p}]$$
$$X^{Trend} = [X_{t-l_q \times s_q}, X_{t-(l_q-1)\times s_q}, ..., X_{t-s_q}]$$

where l_c, l_p, l_q are the sequence length of *Closeness*, *Period*, *Trend*, s_p and s_q are the time span of *Period* and *Trend*, the *Closeness* span s_c is equal to 1 by default. This feature was also inherited by DeepSTN+ [6]. ST-ResNet[1] and DeepSTN+ [6] converted the 4D tensor (H, W, T, C) to 3D tensor (H, W, T*C) by concatenating the channels at each timestep, through which the features lose their temporal order and function like an image rather than a video. In our study, we aim to build a deep learning model to learn the temporal trends corresponding to different time periods. Each trend should be continuous and consistent in observation step (i.e., number of frames) as illustrated by Fig. 5.

Thus, we modify the *Closeness*, *Period*, and *Trend* features to the *Hour*, *Day*, and *Week* features as follows:

$$X^{Hour} = [X_{(t-\Delta_{\mathcal{H}})-T}, X_{(t-\Delta_{\mathcal{H}})-(T-1)}, ..., X_{(t-\Delta_{\mathcal{H}})-1}]$$
$$X^{Day} = [X_{(t-\Delta_{\mathcal{D}})-T}, X_{(t-\Delta_{\mathcal{D}})-(T-1)}, ..., X_{(t-\Delta_{\mathcal{D}})-1}]$$
$$X^{Week} = [X_{(t-\Delta_{\mathcal{W}})-T}, X_{(t-\Delta_{\mathcal{W}})-(T-1)}, ..., X_{(t-\Delta_{\mathcal{W}})-1}]$$

where X^{Hour}, X^{Day}, and X^{Week} represent the trends corresponding to the current (i.e., hourly trend), the previous day (i.e., daily trend), and the previous week (i.e., weekly trend), T is the observation step in each trend. The time interval of Bousai dataset is 30 minutes, thus $\Delta_{\mathcal{H}}$ is set to 0, $\Delta_{\mathcal{D}}$ is set to 48, and $\Delta_{\mathcal{W}}$ is set to 7×48.

4.1.2 External Feature

External information such as date, weather, and event also have significant influences on crowd density and crowd in-out flow. For example, as shown in Fig. 4, the crowd density during the weekday at Tokyo Station could be much higher than the weekend. And as mentioned in Section 1, to validate the pure ability of modeling spatiotemporal data, only date information for the next step (the prediction timestamp) will be utilized as external information and auxiliary input in our study. Specifically, V^E is a vector with 57 dimensions (57 features) containing time of day (48 features), day of week (7 features), weekday flag (1 feature), and holiday flag (1 feature).

4.2 Model Structure

Based on the spatiotemporal and external features, DeepCrowd is elaborately built to automatically capture the

TABLE 3
Notation of Operator

Operator	Meaning
$\cdot$	Dot Product
$\odot$	Hadamard Product
$*$	Convolution Operation
$[;]$	Concatenate Operation
$+$	Vector Addition
$\oplus$	Tensor Addition

citywide spatial dependencies and the multiple stages of temporal dependencies analogously to video modeling task. We explain each component of DeepCrowd neural network in the following. To be easy to understand, we summarize the operator notation as Table 3.

4.2.1 Convolutional LSTM

Convolutional LSTM (ConvLSTM) [13] extends the fully connected LSTM (FC-LSTM) to have convolutional structures in both the input-to-state and state-to-state transitions and achieves a lot of successes on video modeling tasks due to its superior performance in capturing both spatial and temporal dependency than CNN or LSTM. Thus, ConvLSTM is utilized as the basic component to handle our high-dimensional sequential data analogously to video modeling tasks. Hidden state h_t in a ConvLSTM is calculated iteratively from $t = 1$ to T for an input sequence of frames $X = [x_1, x_2, \ldots, x_T]$ as follows:

$$\begin{aligned} i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \odot c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \odot c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \odot c_t + b_o) \\ h_t &= o_t \odot tanh(c_t), \end{aligned} \tag{1}$$

where W is weight, b bias vector, * denotes the convolution operator and $\odot$ represents Hadamard product. The function $f_{ConvLSTM}(\cdot)$ refers to the set of operations in Eq. (1).

Early External-Info Fusion. ST-ResNet [1] and PeriodicCRN [3] adopt late-fusion mechanism to combine the external information at the end of the network by using fully connected layer or simple convolutional layer. Here, in our framework, we employ early-fusion mechanism to let ConvLSTM better explore the complex interactions between spatio-temporal data and meta data at a 4D level. Meta-info vector V^E is first sent to two fully connected (FC) layer and then reshaped to a 4D tensor $X^E = (Timestep, Height, Width, Channel = 1)$ as illustrated by Eq. (2). As Eq. (3) indicates, X^E will be concatenated with 4D "video" tensor X^{Hour}, X^{Day}, X^{Week} at the beginning of the network. This kind of early input mechanism for external information processing enables the model to deeper fuse multiple heterogeneous data and faster convergence.

$$X^E = ReLU[W_{fc_2} \cdot ReLU(W_{fc_1} \cdot V^E + b_{fc_1}) + b_{fc_2}] \tag{2}$$

$$X^{\mathcal{H}}, X^{\mathcal{D}}, X^{\mathcal{W}} \leftarrow [X^{Hour}; X^E], [X^{Day}; X^E], [X^{Week}; X^E]. \tag{3}$$

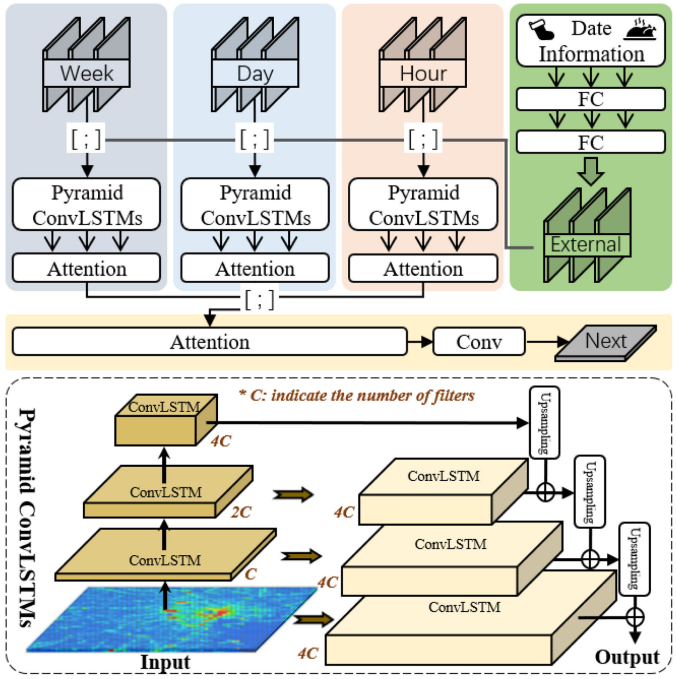


Fig. 6. The architecture of our deep learning model.

4.2.2 Pyramid ConvLSTMs

So far, we can simply stack multiple ConvLSTM layers to handle the video-shape 4D tensors $[X^{\mathcal{H}}, X^{\mathcal{D}}, X^{\mathcal{W}}]$. However, to capture the citywide spatial dependency for a large spatial domain like BousaiTYO (H, W = 80,80) or BousaiOSA (H, W = 60,60), stacking multiple layers of ConvLSTMs is not enough, even infeasible. Assuming we stack n convolution layers with (f, f) kernel size and no padding and striding, the region size that a neuron at the final convolution layer can see is $(f\text{-}1)\times n+1$, $(f\text{-}1)\times n+1$. Theoretically, it requires 40 convolution layers with (3,3) kernel size to model the (H, W = 80,80) spatial domain of BousaiTYO, which will definitely invoke the vanishing gradient problem. Therefore, we propose pyramid ConvLSTMs to extract and utilize pyramid and hierarchical features, which has shown the great success to reinforce the performance of the computer vision models such as Feature Pyramid Networks [14], Mask R-CNN [15], and Faster R-CNN [16]. In pyramid ConvLSTMs as shown in Fig. 6, the outputs from the lower ConvLSTM layers are concatenated with the upper layers to better combine information from both low-resolution and high-resolution feature maps. The latent representations derived from pyramid ConvLSTMs are considered to contain richer spatial information than the ones from plain ConvLSTMs. In addition, pyramid ConvLSTMs can shorten the distance from the input to output which functions similarly as deep residual networks, but pyramid networks do not require to build a very deep network to gather the upper-level features from the input. The pyramid ConvLSTM blocks for each X in $[X^{\mathcal{H}}, X^{\mathcal{D}}, X^{\mathcal{W}}]$ are the same, so we list the formulas for one $X = [x_1, x_2, \ldots, x_T]$ as follows:

$$[h_1, h_2, \ldots, h_T] = f_{ConvLSTM^{\#}}([x_1, x_2, \ldots, x_T]) \tag{4}$$

$$[h_1^{(1)}, h_2^{(1)}, \ldots, h_T^{(1)}] = f_{ConvLSTM\#}^{(1)}([h_1, h_2, \ldots, h_T])$$

$$\ldots$$

$$[h_1^{(l)}, h_2^{(l)}, \ldots, h_T^{(l)}] = f_{ConvLSTM\#}^{(l)}([h_1^{(l-1)}, h_2^{(l-1)}, \ldots, h_T^{(l-1)}]),$$

where $f_{ConvLSTM\#}$ denotes a downsampling ConvLSTM with stride equal to (2,2) and the h are the hidden states generated by the downsampling ConvLSTM layer.

$$[p_1^{(l)}, p_2^{(l)}, \ldots, p_T^{(l)}] = f_{UpSampling}^{(l)}([h_1^{(l)}, h_2^{(l)}, \ldots, h_T^{(l)}]) \tag{5}$$

$$[q_1^{(l)}, q_2^{(l)}, \ldots, q_T^{(l)}] = f_{ConvLSTM\$}^{(l)}([h_1^{(l-1)}, h_2^{(l-1)}, \ldots, h_T^{(l-1)}])$$

$$[s_1^{(l)}, s_2^{(l)}, \ldots, s_T^{(l)}] = [p_1^{(l)}, p_2^{(l)}, \ldots, p_T^{(l)}] \oplus [q_1^{(l)}, q_2^{(l)}, \ldots, q_T^{(l)}]$$

$$\ldots$$

$$[p_1^{(1)}, p_2^{(1)}, \ldots, p_T^{(1)}] = f_{UpSampling}^{(1)}([s_1^{(2)}, s_2^{(2)}, \ldots, s_T^{(2)}])$$

$$[q_1^{(1)}, q_2^{(1)}, \ldots, q_T^{(1)}] = f_{ConvLSTM\$}^{(1)}([h_1, h_2, \ldots, h_T])$$

$$[s_1^{(1)}, s_2^{(1)}, \ldots, s_T^{(1)}] = [p_1^{(1)}, p_2^{(1)}, \ldots, p_T^{(1)}] \oplus [q_1^{(1)}, q_2^{(1)}, \ldots, q_T^{(1)}]$$

$$[p_1, p_2, \ldots, p_T] = f_{UpSampling}([s_1^{(1)}, s_2^{(1)}, \ldots, s_T^{(1)}])$$

$$[q_1, q_2, \ldots, q_T] = f_{ConvLSTM\$}([x_1, x_2, \ldots, x_T])$$

$$[s_1, s_2, \ldots, s_T] = [p_1, p_2, \ldots, p_T] \oplus [q_1, q_2, \ldots, q_T]$$

where $f_{ConvLSTM\$}$ denotes a special ConvLSTM with stride equal to (1,1) and kernel equal to (1,1), $f_{UpSampling}$ denotes an upsampling operation with size equal to (2,2), p and q are intermediate hidden states, and s is the added hidden state generated by the upsampling ConvLSTM layer. By combining Eqs. (4) and (5), we could abstract the pyramid ConvLSTMs as follows:

$$[s_1, s_2, \ldots, s_T] = f_{PyramidConvLSTMs}([x_1, x_2, \ldots, x_T]). \tag{6}$$

4.2.3 Attention ConvLSTMs

Crowd density and in-out flow prediction is taken as high-dimensional sequential modeling task, which is non-trivial to capture the compound dependencies from the historical observations of different time periods, i.e., X_{Hour}, X_{Day}, X_{Week}, totally 3*T citywide crowd images. Mentioning the long-term sequential modeling, LSTM plus attention mechanism [17] has been seen as a state-of-the-art technique. Especially, in the urban computing filed, LSTM plus attention has achieved a lot of success on individual's next-location prediction (DeepMove [18]), time prediction for each road path (DeepTTE [19]), and traffic volume and flow prediction (STDN [5]). Thus, we employ attention mechanism to our ConvLSTM-based model, through which we aim to learn how much each hidden state generated by the pyramid ConvLSTMs (Eq. (6)) will match the next prediction frame. Originally, for LSTMs, an attention block takes a 2D tensor ($Timestep$, $Feature$) as input, and outputs a 1D attention $Feature$ vector. Now, for ConvLSTMs, we extend the original attention block to take a 4D tensor ($Timestep$, $Height$, $Width$, $Filter$) as input, and output a 3D attention tensor ($Height$, $Width$, $Filter$). The formulas for the attention block are listed as follows:

$$z_i = tanh(W_{att} \cdot s_i + b_{att}) \tag{7}$$

$$\alpha_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$$h_{att} = \sum_{i=1}^{T} \alpha_i \cdot s_i.$$

Here, W_{att} is the weight and b_{att} is the bias, s_i is the ith hidden state in $[s_1, s_2, \ldots, s_T]$ outputted by pyramid ConvLSTMs. The function $f_{Attention}(\cdot)$ refers to the set of operations in Eq. (7). Note that s_i is a 3D tensor ($Height$, $Width$, $Filter$), and W_{att} has ($Height \times Width \times Filter$) learnable parameters. Through $f_{Attention}(\cdot)$, the hidden states of pyramid ConvLSTMs $[s_1, s_2, \ldots, s_T]$ will be dynamically fused as one attention state denoted as h_{att}. For each S in $\{S^{\mathcal{H}}, S^{\mathcal{D}}, S^{\mathcal{W}}\}$, we utilize one independent attention block to generate an attention state, namely $h_{att}^{(\mathcal{H})}, h_{att}^{(\mathcal{D})}, h_{att}^{(\mathcal{W})}$. Then the list of attention states $[h_{att}^{(\mathcal{H})}, h_{att}^{(\mathcal{D})}, h_{att}^{(\mathcal{W})}]$ will be further fed into another attention block to fuse the three attention states as a final attention state denoted as $h_{att}^{(all)}$. Eq. (8) and (9) formulate the process above.

$$h_{att}^{(\mathcal{H})} = f_{Attention}^{(\mathcal{H})}([s_1^{(\mathcal{H})}, s_2^{(\mathcal{H})}, \ldots, s_T^{(\mathcal{H})}]) \tag{8}$$

$$h_{att}^{(\mathcal{D})} = f_{Attention}^{(\mathcal{D})}([s_1^{(\mathcal{D})}, s_2^{(\mathcal{D})}, \ldots, s_T^{(\mathcal{D})}])$$

$$h_{att}^{(\mathcal{W})} = f_{Attention}^{(\mathcal{W})}([s_1^{(\mathcal{W})}, s_2^{(\mathcal{W})}, \ldots, s_T^{(\mathcal{W})}])$$

$$h_{att}^{(all)} = f_{Attention}^{(all)}([h_{att}^{(\mathcal{H})}, h_{att}^{(\mathcal{D})}, h_{att}^{(\mathcal{W})}]). \tag{9}$$

After the two levels of attention blocks, we utilize one convolution layer with (1,1) kernel window to get the final prediction result as follows:

$$\widehat{y} = ReLU(W_{conv} * h_{att}^{(all)} + b_{conv}), \tag{10}$$

The overall architecture of DeepCrowd is shown as Fig. 6.

5 Experiment

5.1 Setup and Setting

We set the observation step T to 6 and $\{\Delta_{\mathcal{H}}, \Delta_{\mathcal{D}}, \Delta_{\mathcal{W}}\}$ to {0, 7, 7×48} respectively. It should be noted that these settings follow the ones widely used and well tuned in the state-of-the-arts listed in Table 1. Data from the first 80 percent were set as training data (20 percent of which were taken as validation data), and the other 20 percent were set as testing data. Adam was employed to control the overall training process, where the learning rate was set as one of {0.01, 0.001, 0.0005, 0.0001}. The training algorithm would either be early-stopped if the validation error converged within 10 epochs or be stopped after 200 epochs, and the best model on validation data would be saved. We rescaled the predicted value back to the original value. All models were run

multiple times on each dataset, and the best result would be recorded. Keras [20] and TensorFlow [21] are used to implement the deep learning models. The experiments were performed on 3 GPU servers, together with six GeForce GTX 1080Ti graphics cards and two Tesla P40 graphics cards. Pyramid ConvLSTMs in *DeepCrowd* consist of three bottom-up ConvLSTM layers with {32, 64, 128} filters of 3×3 kernel window and three top-down ConvLSTM layers with {128,128,128} filters of 1×1 kernel window.

5.2 Baseline

HistoricalAverage. Density and in-out flow for each timestamp are estimated by averaging the historical values from the corresponding timestamp in the training dataset, and weekday and weekend will be considered separately.

CopyYesterday. We directly copy the corresponding observation (frame) from the previous day (yesterday) as the result.

SimpleCNN. It is a basic deep learning predictor constructed with four CNN layers. The 4D tensor would be converted to 3D tensor $(H, W, T*C)$ by concatenating the channels at each timestep just like the way [1] did, so that CNN could take a 4D tensor as input. The *SimpleCNN* predictor utilizes four Conv layers to take the current observed t-step frames as input and predicts the next frame as output. The four Conv layers use a 32 filters of 3×3 kernel window, and the fourth Conv layer uses a ReLU activation function to output the next frame (step) of urban video. BatchNormalization is added between two consecutive layers.

SimpleConvLSTM. It is a simplified predictor of Hetero-ConvLSTM [4] constructed with four ConvLSTM layers. Here, heterogeneous data features were excluded from Hetero-ConvLSTM [4] and only the pure power for capturing spatiotemporal dependency remained. The Simple-ConvLSTM predictor is constructed with four ConvLSTM layers, which also takes current t-step observations as input and predicts the next step. The ConvLSTM layers use a 32 filters of 3×3 kernel window and the ReLU activation is used in the final layer. BatchNormalization is also added between two consecutive layers.

ST-ResNet. Deep Spatio-Temporal Residual Networks (ST-ResNet) [1] is a CNN-based deep learning method for traffic in-out flow prediction. To capture citywide spatial dependency, it employs residual learning to construct deep enough CNN networks; To capture temporal dependency, it first designs a set of unique features namely $Closeness, Period$, and $Trend$, which correspond to *the recent time intervals*, *daily periodicity*, and *weekly trend* respectively, then fuses them together through three learnable parametric matrices. Intuitively, the three sequences can be represented by $[X_{t-l_c}, X_{t-(l_c-1)}, ..., X_{t-1}]$, $[X_{t-l_p \cdot p}, X_{t-(l_p-1)\cdot p}, ..., X_{t-p}]$, and $[X_{t-l_q \cdot q}, X_{t-(l_q-1)\cdot q}, ..., X_{t-q}]$, where l_c, l_p, l_q are the sequence length of $Closeness, Period, Trend$, p and q are the span of $Period$ and $Trend$, the $Closeness$ span is equal to 1 by default. 4D tensor would be converted to 3D tensor $(H, W, T*C)$ by concatenating the channels at each timestep. The computation unit is the whole citywide image. Additionally, it further utilizes weather, holiday event information, and metadata(i.e., DayOfWeek, Weekday/Weekend) as external information. To verify the pure ability of capturing spatial and temporal dependency, only metadata will be utilized in our study.

DMVST-Net. Deep Multi-View Spatial-Temporal Network (DMVST-Net) [2] is a deep learning method for taxi demand prediction based on CNN and LSTM. It uses a local CNN to capture spatial dependency only among nearby grids; employs LSTM to capture temporal dependency only from the recent time intervals (i.e., $Closeness$). The local CNN takes one grid and its surrounding grids (i.e., $S \times S$ region) as the input, and a separate and unshared CNN is constructed for each timestamp. The input tensor is essentially $(T, S, S, 1)$, and the computation unit is grid (pixel) rather than citywide image. Furthermore, it constructs a weighted graph, where nodes are the grids, and each edge represents the similarity of two time-series values (i.e., historical taxi demand) between any two grids; then it embeds this graph into a feature vector and concatenates it with the main feature vector from LSTM layer. Through this, it can improve the ability to capture citywide spatial dependency. Similarly, in our study, only metadata will be utilized as external information.

PCRN. Convolutional Recurrent Network with Periodic Representation (PCRN) [3] is a ConvGRU-based deep learning model for taxi density and in-out flow prediction by fully making use of recurrent periodic patterns. To capture citywide spatial dependency, it builds a pyramidal architecture by stacking three convolutional RNN layers. To capture temporal dependency, it first learns a representation from the observations of $Closeness$ through the stacked pyramidal ConvGRUs; it divides the representations into two types of periodic patterns, namely daily and weekly pattern, each of them is a set of periodic representations corresponding to a specific time span (i.e., day or week); then it maintains a memory-based dictionary to reuse and update the two types of periodic patterns dynamically; lastly it employs a weighting based fusion to merge periodic representations with the current representation of input sequence. Thus, the input feature can be seen as $Closeness$, $Period$, and $Trend$. The computation unit is the whole citywide image. Also, only metadata will be utilized as external information. In our study, we replace ConvGRU with ConvLSTM and simplify the architecture.

STDN. Spatial-Temporal Dynamic Network (STDN) [5] is an improved version of DMVST-Net for taxi/bike Origin-Destination number (volume) prediction. To capture spatial dependency, it inherits the local CNN technique from DMVST-Net, and further designs a flow gating mechanism to fuse local flow information (i.e., flow from one central grid to its surrounding $S \times S$ grids) with the traffic volume information together. In terms of temporal dependency, it improves DMVST-Net by taking not only $Closeness$ information but also long-term daily periodicity (i.e., $Period$) into account. Moreover, it considers the temporal shifting problem about periodicity (i.e., traffic data is not strictly periodic) and designs a *Periodically Shifted Attention Mechanism* to solve the issue. Specifically, it sets a small time window to collect Q time intervals right before and after the currently-predicting one. And it uses an LSTM plus attention mechanism to obtain a weighted average representation h from the time intervals in each window. For previous P days to be considered as $Period$, it gets a sequence of

TABLE 4
Summary of Fine-Tuned Hyper-Parameters

ST-ResNet [1]		DMVST-Net [2]		PCRN [3]	
Residual unit	2	Layer	3	Layer	3
Filter	32	Filter	32	Filter	32
Kernel	3	Kernel	3	Kernel	3
		Neighborhood	9	Fusion kernel	1
		Graph embedding	32	Transform kernel (Tokyo)	80, 80
		Spatial output	64	Transform kernel (Osaka)	60, 60
		Temporal output	512		
		Semantic output	6		
Multitask-DF [8]		STDN [5]		DeepSTN+ [6]	
Layer	4	Layer	3	ResPlus unit	2
Filter	32	Filter	32	Filter	32
Kernel	3	Kernel	3	Kernel	1
		Neighborhood	9	ConvPlus channel	32
		Short term	6	Separated channel	8
		Long term	3	Pooling rate (Tokyo)	8
		Attention	3	Pooling rate (Osaka)	6
		Hidden output	128		

representations (h_1, h_2, ..., h_P), then it uses another LSTM layer to extract the final periodic representation from the sequence. The computation unit is grid (pixel) same with DMVST-Net. Lastly it jointly models inflow (start traffic volume) and outflow (end traffic volume) together. The flow gating mechanism will be pruned in our study since flow detail information are needed.

DeepSTN+. Context-aware Spatial-Temporal Neural Network (DeepSTN+) [6] is an improved version of ST-ResNet for crowd and traffic in-out flow prediction. It directly inherits the input features (i.e., *Closeness*, *Period*, and *Trend*), and enhances the ST-ResNet from the following aspects: (1) to capture longer-range spatial dependency, it designs a unique *ConvPlus* block, and replaces the ordinary *Conv*-based residual unit in ST-ResNet with *ResPlus* (i.e., *ConvPlus*-based residual unit). Furthermore, multi-scale fusion mechanism is employed to preserve the representation from each *ResPlus* layer; (2) in terms of temporal dependency, it applies early-fusion mechanism instead of end-fusion in ST-ResNet to get better interaction among *Closeness*, *Period*, and *Trend*; (3) additionally, it takes the influence of location function on the crowd/traffic flow into consideration by using POI data to gain a semantic plus. The computation unit is the whole citywide image. Similarly, we prune the POI processing component from DeepSTN + to verity the pure spatiotemporal modeling capability.

Multitask-DF. Multitask Learning of Crowd Density and In-Out Flow (Multitask-DF) is a ConvLSTM-based Multitask Learning model to jointly predict crowd density (D) and in-out flow (F). It can be seen as a simplified variant of Multitask Deep Learning (*MDL*) [7] and *DeepUrbanEvent* [8], where multitask learning [22] was employed to model two correlated tasks together and gain concurrent enhancement. The motivation comes from the following two points: (1) People tend to follow the trend. Crowded places may attract more and more people to visit; (2) Higher inflow will lead to higher density, higher outflow will lead to lower density. Multitask-DF first takes X_d (t-step observed density) and X_f (t-step observed in-out flow) with two separate ConvLSTM layers; then concatenates two separate latent representations and passes it to two consecutive ConvLSTM layers; finally outputs $\hat{Y}_d$ (next-step density) and $\hat{Y}_f$ (next-step in-out flow) with two separate ConvLSTM layers. The model parameters θ can be trained by minimizing the objective function $\mathcal{L}(\cdot)$ as follows:

$$\theta = \arg\min_{\theta} [\lambda_d \mathcal{L}(\hat{Y}_d, Y_d) + \lambda_f \mathcal{L}(\hat{Y}_f, Y_f)].$$

SimpleConvLSTM and *Multitask-DF* naturally rely on the superior performance of ConvLSTM to capture both spatial and temporal dependency.

For all approaches, we used gird search to tune the parameters based on the validation dataset and chose the best set of hyperparameters. We also considered the parameter settings recommended by the original study. The hyper-parameters are finely tuned as Table 4 by spending equal amounts of effort in each.

5.3 Evaluation Metric

We evaluate the effectivenesses of the models with MSE (Mean Squared Error), RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and MAPE (Mean Absolute Percentage Error):

$$MSE = \frac{1}{n}\sum_{i}^{n} ||\hat{Y}_i - Y_i||^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n} ||\hat{Y}_i - Y_i||^2}$$

$$MAE = \frac{1}{n}\sum_{i}^{n} |\hat{Y}_i - Y_i|$$

$$MAPE = \frac{1}{n}\sum_{i}^{n} |\frac{\hat{Y}_i - Y_i}{Y_i}|,$$

where n is the number of samples, Y and $\hat{Y}$ are the ground-truth tensor and predicted tensor.

5.4 Performance Evaluation

Effectiveness Evaluation. The overall evaluation results on effectiveness are summarized in Tables 5 and 6. Besides, we set the RMSE result of ST-ResNet as the standard and give out the relative increment ΔRMSE. Through them we can see that the state-of-the-art models including our proposed model had advantages compared with baselines (HistoricalAverage∼ConvLSTM). In particular, the state-of-the-arts (ST-ResNet∼Multitask-DF) had their own advantages on different cities, tasks, and metrics. Our proposed DeepCrowd model becomes a dominant one, and demonstrates the superiority to the-state-of-the-arts on all of the datasets and metrics. As the spatial domain (i.e., H, W=80,80/60,60) of the new published dataset is much larger than the existing datasets, the pyramid architecture plays a vital role in capture the spatial dependency. Next to DeepCrowd, STDN also achieved satisfactory performances in general. However, as the computation unit of STDN is pixel (mesh-grid), the big spatial domain will generate a huge amount of training samples, so the training process of STDN would take far more time than other models.

Efficiency Evaluation. Besides the comparison of prediction accuracy, we also provide an efficiency comparison in terms of computational time and neural network complexity among the different approaches, as these can play an important role in practice when deciding which approach

TABLE 5
Effectiveness Evaluation: Crowd Density and In-Out Flow Prediction on BousaiTYO

Model	Tokyo Density					Tokyo In-Out Flow				
	MSE	RMSE	MAE	MAPE	ΔRMSE	MSE	RMSE	MAE	MAPE	ΔRMSE
HistoricalAverage	221.501	14.883	7.175	3.89%	105.82%	47.433	6.887	3.09	6.66%	48.65%
CopyYesterday	1304.393	36.116	11.804	5.16%	399.46%	148.604	12.19	4.156	6.93%	163.11%
SimpleCNN	124.657	11.165	6.101	6.49%	54.40%	49.07	7.005	3.812	8.16%	51.20%
SimpleConvLSTM [13]	81.778	9.043	4.335	2.69%	25.06%	24.237	4.923	2.54	6.07%	6.26%
ST-ResNet [1]	52.288	7.231	4.336	2.83%	0.00%	21.469	4.633	2.467	5.80%	0.00%
DMVST-Net [2]	42.726	6.537	3.918	2.51%	-9.60%	34.795	5.899	2.985	6.87%	27.33%
PCRN [3]	55.676	7.462	4.653	3.10%	3.19%	22.71	4.765	2.491	6.10%	2.85%
STDN [5]	39.492	6.284	3.713	2.42%	-13.10%	19.654	4.433	2.468	5.86%	-4.32%
DeepSTN+ [6]	89.775	9.475	4.907	3.08%	31.03%	19.062	4.366	2.387	5.66%	-5.76%
Multitask-DF [8]	49.784	7.056	**3.043**	2.42%	-2.42%	22.185	4.710	2.451	5.97%	1.66%
DeepCrowd (Ours)	**33.138**	**5.757**	3.394	**2.25%**	**-20.38%**	**18.697**	**4.324**	**2.203**	**5.36%**	**-6.67%**

TABLE 6
Effectiveness Evaluation: Crowd Density and In-Out Flow Prediction on BousaiOSA

Model	Osaka Density					Osaka In-Out Flow				
	MSE	RMSE	MAE	MAPE	ΔRMSE	MSE	RMSE	MAE	MAPE	ΔRMSE
HistoricalAverage	79.557	8.919	4.864	4.35%	93.60%	20.267	4.502	2.011	5.88%	29.63%
CopyYesterday	274.857	16.579	7.264	5.55%	259.87%	42.034	6.483	2.443	5.87%	86.67%
SimpleCNN	35.325	5.944	3.203	2.94%	29.02%	131.318	11.459	3.443	7.87%	229.95%
SimpleConvLSTM [13]	26.939	5.19	2.94	3.00%	12.65%	13.955	3.736	1.824	5.68%	7.57%
ST-ResNet [1]	21.222	4.607	2.9	2.93%	0.00%	12.064	3.473	1.727	5.14%	0.00%
DMVST-Net [2]	17.852	4.225	2.613	2.72%	-8.29%	17.151	4.141	2.054	6.10%	19.23%
PCRN [3]	21.064	4.59	2.898	3.01%	-0.37%	18.073	4.251	1.817	5.44%	22.40%
STDN [5]	22.791	4.774	2.884	2.90%	3.62%	11.579	3.403	1.782	5.35%	-2.02%
DeepSTN+ [6]	32.962	5.741	3.23	3.24%	24.61%	11.449	3.384	1.646	4.96%	-2.56%
Multitask-DF [8]	21.151	4.599	2.517	2.79%	-0.17%	12.871	3.588	1.756	5.53%	3.31%
DeepCrowd (Ours)	**16.743**	**4.092**	**2.458**	**2.49%**	**-11.18%**	**11.382**	**3.374**	**1.565**	**4.96%**	**-2.85%**

to use in real-world application. First, we plot the total training time in minute for each model as Fig. 7. Through it we can see that: (1) the overall efficiency of our proposed DeepCrowd was controlled at an acceptable level; (2) the training time of DMVST-Net and STDN were far more than others as they took mesh-grid as the computation unit; (3) ST-ResNet holds a very clear advantage over other state-of-the-art models from the perspective of efficiency. Second, we study the total number of parameters for each model under the fine-tuned hyper-parameter setting. Due to space limitations, we only list the results on BousaiTYO Density and In-Out Flow as Table 7. Through it we can see that the parameter number of DeepSTN+ was far more than others as it utilized fully-connected layer to capture the citywide spatial dependency.

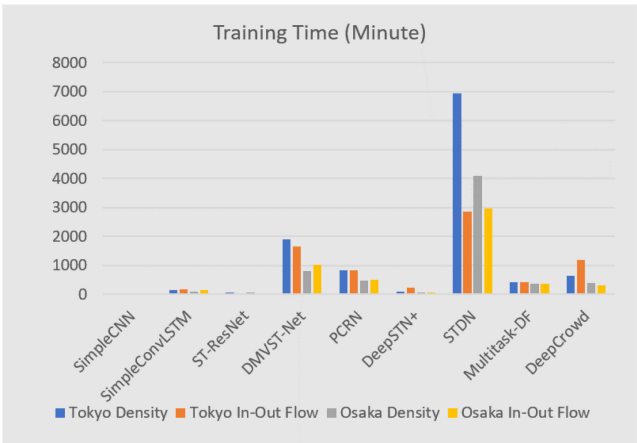


Fig. 7. Efficiency evaluation: Comparison of training time in minute.

Discussion. Through the comprehensive evaluations, we could find the main limitations of the state-of-the-art models as follows: (1) ST-ResNet converts the 4D input tensor (T, H, W, C) data to 3D tensor $(H, W, T * C)$ to apply CNN, thus it somehow fails to capture the real temporal dependency; (2) PCRN needs to dynamically save and load the learned representations as periodic patterns for each time-step, which is not so effective nor efficient; (3) DMVST-Net and STDN use local CNN to take mesh-grid (pixel) as computation unit, resulting the low efficiency on training time (nearly 1 week on BousaiTYO); (4) DeepSTN+ utilized a fully-connected layer in *ConvPlus* block, which would result in a huge number of parameters in Tokyo area (over 0.3 billion as shown in Table 7); (5) Multitask-DF needs both density and in-out flow data for computing. In summary, we can recommend DeepCrowd as a good solution to real-world crowd flow (density&in-out flow) prediction task with good balance of effectiveness and efficiency.

TABLE 7
Efficiency Evaluation on BousaiTYO Density and In-Out Flow

	BousaiTYO	
Model	Density	In-Out Flow
SimpleCNN	20,929	22,946
SimpleConvLSTM [13]	187,432	189,848
ST-ResNet [1]	205,095	297,002
DMVST-Net [2]	1,578,253	1,578,253
PCRN [3]	1,284,999	1,561,352
DeepSTN+ [6]	327,820,193	327,820,546
STDN [5]	6,181,505	6,349,922
Multitask-DF [8]	266,176	266,176
DeepCrowd (Ours)	7,882,842	8,315,355

TABLE 8
Performance Evaluation of Different DeepCrowd Structures on BousaiTYO Density

	BousaiTYO Density			
Model Variant	MSE	RMSE	MAE	MAPE
DeepCrowd-plain	33.784	5.812	3.454	2.30%
DeepCrowd-noConvLSTM	49.673	7.048	4.033	2.72%
DeepCrowd-noAttention	36.182	6.015	3.528	2.33%
DeepCrowd-oneAttention	34.638	5.885	3.450	2.27%
DeepCrowd-lateExternal	36.192	6.016	3.542	2.38%
DeepCrowd-noExternal	41.527	6.444	3.792	2.55%
DeepCrowd (Ours)	**33.138**	**5.757**	**3.394**	**2.25%**

5.5 Ablation Study

Effect of Model Components. We first give the comparison with 6 different model variants on BousaiTYO density, as shown in Table 8. We verify the effectiveness of our model from spatial, temporal and external view respectively. For the spatial view, the pyramid model structure was replaced with a stacked ConvLSTM structure named DeepCrowd-plain, and all ConvLSTM layers were removed named DeepCrowd-noConvLSTM. The models without the attention structure or only with one level attention in the temporal view have the suffix of noAttention and oneAttention respectively. LateExternal and noExternal mean fusing the external information at the end of the model and not using the external information respectively. We observe that our DeepCrowd model achieves best performance against others, which proves the effectiveness and reliability of each module in our model.

Effect of Network Depth. Fig. 8 demonstrates the influence of network depth on BousaiTYO density and in-out flow. Here, the depth of the network represents the height of the pyramid ConvLSTMs. We can observe that when the depth of the model gets deeper, the performance of the model also becomes better with a continuing tendency, demonstrating that a deeper structure can better help the model gather information from the data. However, when the network depth comes to 4, the model performance will get worse on both density and in-out flow. This is considered to be the typical vanishing gradient problem, and the very deep model could not be fully trained. Another interesting point is that deeper models could gain better performance on in-out flow prediction task comparing to density task. This is because crowd in-out flow is more dynamic and globally involved than crowd density, and deeper networks could sense crowd flow features in a wider range.

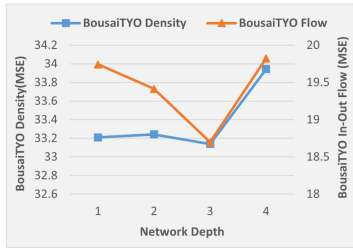


Fig. 8. Effect of Network Depth.

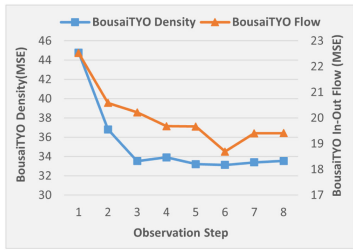


Fig. 9. Effect of observation step.

Effect of Observation Step. Fig. 9 presents the effect of the observation step on BousaiTYO density and in-out flow. As the length of the input sequence grows, the MSE of the model first decreases, showing that the longer input often gets a better result because they can provide the model more information on the dynamics of crowd flow. However, the model gets worse when the observation step is greater than 6, showing that the training process becomes more difficult and there is a balance between input information and the model processing capabilities.

5.6 Case Study

Prediction Performance. To further verify the prediction performance of our proposed model, we conduct multiple case studies on BousaiTYO Density dataset. We pick up two places, one is Tokyo Station, the biggest station in Tokyo, another is Tokyo Disneyland, the most famous theme park in Japan, to plot the ground-truth and our prediction for the last 7 days of our dataset, i.e., 2017-07-03 (Monday)∼2017-07-09 (Sunday). The reasons we choose these two places are as follows: (1) they are the typical representatives of two types of areas, i.e., office and recreation; (2) they have different levels of crowd density, i.e., Tokyo station is around five times higher than Tokyo Disneyland. Through the time-series plots in Fig. 10, we could find that our model gives satisfactory prediction results on both places for the entire week. Besides, we visualize the ground-truth and predicted crowd densities for the entire Tokyo area during the morning rush hour on weekday, i.e., 2017-07-03 07:00:00∼2017-07-03 09:00:00. Through Fig. 11, we can observe that our model gives a very accurate prediction over the entire 80*80 mesh-grids of Tokyo area.

Model Interpretation. We further do two case studies to derive a better interpretation of our new model architecture.

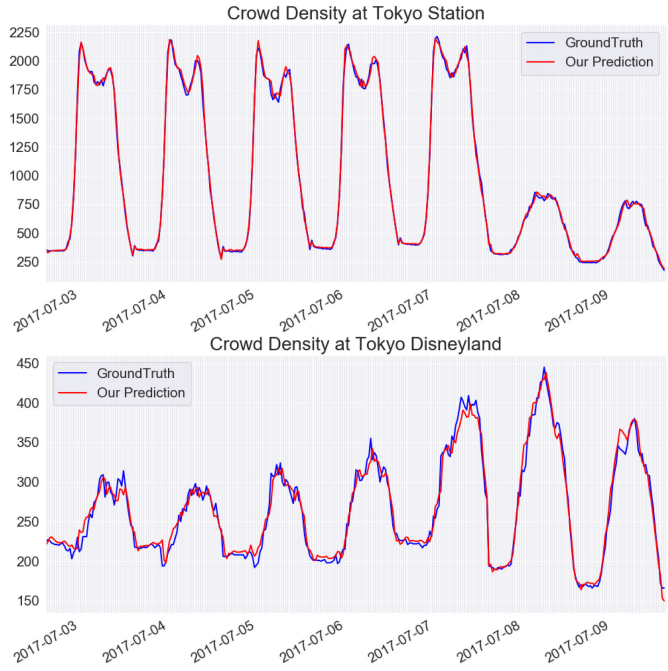


Fig. 10. Case study: Ground-truth and predicted crowd density at tokyo station and tokyo disneyland for one week.

Similar with the above case studies, we select the crowd density of Tokyo area at 2017-07-03 09:00:00 as the prediction target. Since the time interval is 30 minutes and observation step is 6, X_{Hour}: 2017-07-03 06:00:00∼2017-07-03 08:30:00, X_{Day}: 2017-07-02 06:00:00∼2017-07-02 08:30:00, X_{Week}: 2017-06-26 06:00:00∼2017-06-26 08:30:00 visualized in the upper part of Fig. 13 are taken as input features. By using these input features and the trained weights of DeepCrowd, we plot gradient-based activation feature maps from the last two layers of PyramidConvLSTMs as Fig. 12. Left is the Tth hidden state p_T of the last UpSampling layer, and right is the Tth hidden state q_T of the last ConvLSTM layer, which are respectively corresponding to the following in Eq. (5):

$$[p_1, p_2, \ldots, p_T] = f_{UpSampling}([s_1^{(1)}, s_2^{(1)}, \ldots, s_T^{(1)}])$$

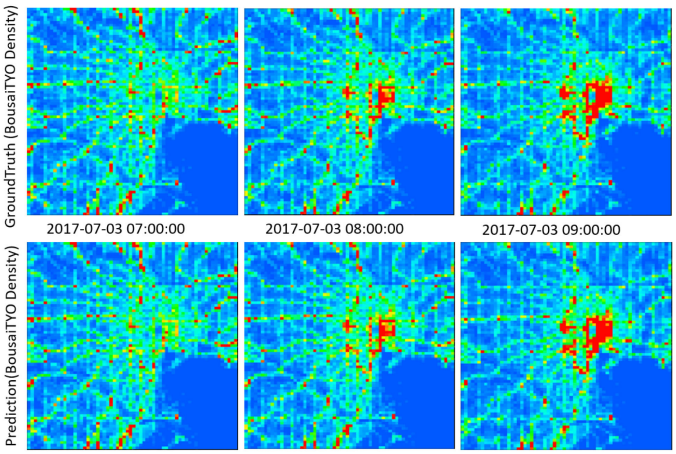


Fig. 11. Case study: Ground-truth and predicted crowd density for the tokyo area on a weekday morning.

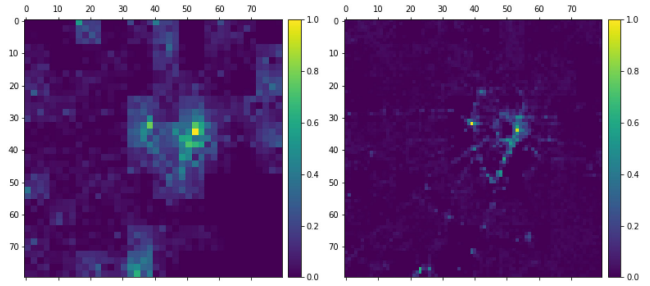


Fig. 12. Case study: Activation feature maps from the last two layers of PyramidConvLSTMs. Left is the last Upsampling layer, and right is the last ConvLSTM layer.

$$[q_1, q_2, \ldots, q_T] = f_{ConvLSTM^\$}([x_1, x_2, \ldots, x_T]).$$

From Fig. 12, we can find that the activation features of the final UpSampling layer are scattered in a wide range, and the final ConvLSTM layer captures the features in a relatively small range. In other words, both local and global features could be well preserved through the PyramidConvLSTMs architecture. Then, using the same prediction target and the input features, we examine the attention scores achieved by the two levels of attention blocks. Through Fig. 13, we find that the overall attention scores are X_{Hour}:X_{Day}:X_{Week} = 0.76:0.12:0.12, and the final step (i.e., 6th step) plays a vital role in each X, weighing around 60 percent among the total six observation steps. This demonstrates that our Attention-ConvLSTMs architecture can accurately pick up the temporal features from multiple time periods.

6 Related Work

In this section, we briefly discuss some existing researches concerning to crowd or traffic prediction problem in the field of urban computing [23]. Except for the deep learning approaches, there are also some statistical models proposed. Spatiality preservable factored Poisson regression [24] incorporates point of interest to overcome data sparsity and

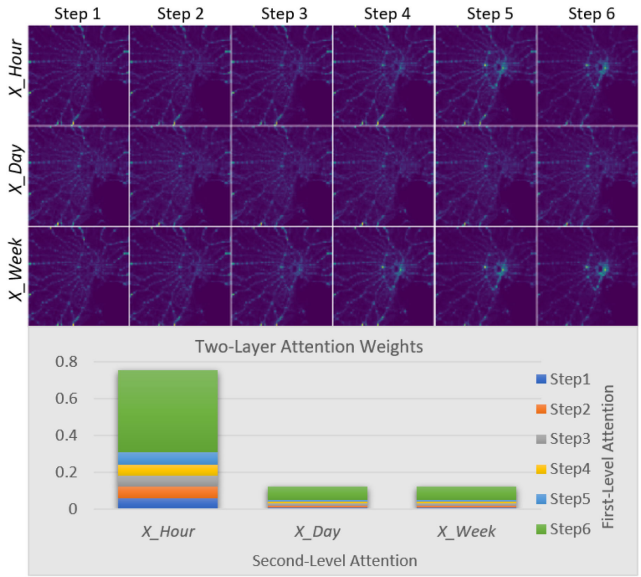


Fig. 13. Case study: Attention weights from the two levels of Attention-ConvLSTMs. Upper is the input features X_{Hour}, X_{Day}, X_{Week}, each with six observation steps, and bottom is the attention weights from the two levels of attention blocks.

degraded performance in even finer grains population prediction. CityProphet [25], [26] and [27] utilize query data of Smartphone App to forecast only crowd density other than crowd flow. [28], [29] conduct transition estimation from aggregated population data, and [30] estimates the transition populations using inflow and outflow defined by [9]. Based on road network, [31], [32], [33], [34], [35], [36], [37] were proposed to predict the traffic flow, speed, congestion, human mobility as well as transportation mode. In particular, DeepTTE [38] and DeepGTT [39] are proven as effective deep learning models to predict the travel time on each road segment. Leveraging on the latest techniques, a series of models have been proposed to address traffic-related problems, such as using graph neural networks for traffic forecasting [40] and ride-hailing demand prediction [41], multi-task learning for travel time estimation [42], or meta learning for traffic prediction [43].

Besides, many trajectory-based deep learning models were proposed to predict each individual's movement [44], [45], [46]. [45] extends a regular RNN by utilizing time and distance specific transition matrices to propose an ST-RNN model for predicting the next location. DeepMove [18], considered to be a state-of-the-art model for trajectory prediction, designed a historical attention module to capture periodicities and augment prediction accuracy. VANext [46] further enhanced DeepMove by proposing a novel variational attention mechanism. Modeling human mobility for very large populations [47], [48] and simulating human emergency mobility following disasters [49], [50] are similar problems to ours; however, their models are built based on millions of individuals' mobility. [51] propose tensor factorization approach to decompose urban human mobility, aiming to understand basic urban life patterns from city-scale human mobility data. Using mobility data from location-based social networks (LBSN), [52], [53] design matrix factorization-based models to conduct POI recommendation or location prediction. Also, [54] utilized latent factor models for POI recommendation using heterogeneous features. [55] conducted chain store site recommendation with transfer learning and multi-source data.

Last, other urban computing problems are also modeled based on citywide mesh-grids, and addressed through advanced deep learning technologies, including air quality prediction [56], [57], crop yield prediction [58], abnormal event prediction [59], and multiple transportation demand prediction [60].

7 Conclusion

In this study, we revisit grid-based urban computing on citywide traffic and crowd prediction by publishing new datasets called BousaiTYO and BousaiOSA. The scale of our new dataset is much larger than the previous datasets on the following aspects: (1) larger spatial area; (2) finer mesh size; (3) higher user sample. Most importantly, our dataset can better reflect the real-world citywide crowd as it is generated based on a real-world smartphone app with large active users rather than taxi or bike O-D data. To effectively capture spatial and temporal dependency from this large-scale crowd data, we propose a novel deep learning model called DeepCrowd by designing pyramid architectures and high-dimensional attention mechanism based on Convolutional LSTM. We implement the state-of-the-art works of literature including ST-ResNet, DMVST-Net, PCRN, STDN, DeepSTN+, and Multitask-DF as baselines, and conduct a thorough performance evaluation on crowd density and in-out flow prediction problems. The experimental results demonstrate the superior performances of DeepCrowd to the state-of-the-arts on new datasets. Our new dataset and model will be officially published if this paper is accepted. In the future, the effects of different settings on the use of heterogeneous data sources such as POI and event info, objective function, and scaling strategy could be further analyzed. Moreover, we consider to apply our framework to other urban computing problems, such as citywide air quality prediction, citywide electric power consumption, citywide transportation demand prediction, and citywide crime incident prediction. The data and code can be found at https://github.com/deepkashiwa20/DeepCrowd.

Acknowledgments

This work was supported by Grant-in-Aid for Early-Career Scientists under Grant 20K19859 of Japan Society for the Promotion of Science (JSPS). Also, the support given by Jiaqi Marry Chen has been a great help for me during the study period. Renhe Jiang and Zekun Cai contributed equally to this work.

References

[1] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.
[2] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," *Proc. 32nd AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[3] A. Zonoozi, J.-J. Kim, X.-L. Li, and G. Cong, "Periodic-CRN: A convolutional recurrent model for crowd density prediction with recurring periodic patterns," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3732–3738.
[4] Z. Yuan, X. Zhou, and T. Yang, "Hetero-convLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 984–992.
[5] 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.
[6] Z. Lin, J. Feng, Z. Lu, Y. Li, and D. Jin, "DeepSTN+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1020–1027.
[7] 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. 2020.
[8] R. Jiang *et al.*, "DeepUrbanEvent: A system for predicting citywide crowd dynamics at big events," in *Proc. 25th ACM SIGKDD Int. Conf.Knowl. Discov. Data Mining*, 2019, pp. 2114–2122.
[9] M. X. Hoang, Y. Zheng, and A. K. Singh, "Fccf: Forecastingcitywide crowd flows based on big data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, no. 6, pp. 1–10.
[10] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, Art. no. 92.
[11] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, Art. no. 92.
[12] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 243–254.

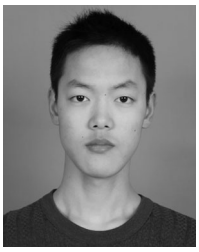
[13] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[14] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.
[15] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[17] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Advances Neural Inf. P Syst.*, 2017, pp. 5998–6008.
[18] J. Feng *et al.*, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf.*, 2018, pp. 1459–1468.
[19] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," *Proc. 32nd AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[20] F. Chollet, "keras," 2015. [Online]. Available: https://github.com/fchollet/keras
[21] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, *arXiv:1603.04467*.
[22] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.
[23] 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.
[24] M. Shimosaka, Y. Hayakawa, and K. Tsubouchi, "Spatiality preservable factored poisson regression for large-scale fine-grained GPS-based population analysis," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1142–1149.
[25] T. Konishi, M. Maruyama, K. Tsubouchi, and M. Shimosaka, "CityProphet: City-scale irregularity prediction using transit app logs," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 752–757.
[26] M. Shimosaka, K. Maeda, T. Tsukiji, and K. Tsubouchi, "Forecasting urban dynamics with mobility logs by bilinear poisson regression," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 535–546.
[27] J. Zhou, H. Pei, and H. Wu, "Early warning of human crowds based on query data from Baidu maps: Analysis based on Shanghai stampede," in *Proc. Big Data Support Urban Plan. Manage.*, 2018, pp. 19–41.
[28] Y. Akagi, T. Nishimura, T. Kurashima, and H. Toda, "A fast and accurate method for estimating people flow from spatiotemporal population data." in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3293–3300.
[29] A. Sudo *et al.*, "Particle filter for real-time human mobility prediction following unprecedented disaster," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2016, Art. no. 5.
[30] Y. Tanaka, T. Iwata, T. Kurashima, H. Toda, and N. Ueda, "Estimating latent people flow without tracking individuals." in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3556–3563.
[31] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi GPS traces," in *Proc. 10th Int. Conf. Pervasive Comput.*, 2012, pp. 57–72.
[32] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.
[33] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.
[34] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.
[35] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS One*, vol. 10, no. 3, 2015, Art. no. e0119044.
[36] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. Part C: Emerg. Technol.*, vol. 54, pp. 187–197, 2015.
[37] X. Song, H. Kanasugi, and R. Shibasaki, "DeepTransport: Prediction and simulation of human mobility and transportation mode at a citywide level," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 2618–2624.
[38] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov.Data Mining*, 2018, pp. 858–866.
[39] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *Proc. World Wide Web Conf.* 2019, pp. 1017–1027.
[40] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.
[41] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3656–3663.
[42] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proc.24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1695–1704.
[43] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1720–1730.
[44] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "GMove: Group-level mobility modeling using geo-tagged social media," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1305–1314.
[45] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 194–200.
[46] Q. Gao, F. Zhou, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Predicting human mobility via variational attention," in *Proc. World Wide Web Conf.*, 2019, pp. 2750–2756.
[47] C. Song, T. Koren, P. Wang, and A.-L. Barabási, "Modelling the scaling properties of human mobility," *Nat. Phys.*, vol. 6, no. 10, pp. 818–823, 2010.
[48] Z. Fan, X. Song, R. Shibasaki, and R. Adachi, "CityMomentum: An online approach for crowd behavior prediction at a citywide level," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 559–569.
[49] X. Song, Q. Zhang, Y. Sekimoto, T. Horanont, S. Ueyama, and R. Shibasaki, "Modeling and probabilistic reasoning of population evacuation during large-scale disaster," in *Proc.19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 1231–1239.
[50] X. Song, Q. Zhang, Y. Sekimoto, R. Shibasaki, N. J. Yuan, and X. Xie, "A simulator of human emergency mobility following disasters: Knowledge transfer from big disaster data," in Proc. *29th AAAI Conf. Artif. Intell.*, 2015, pp. 730–736.
[51] Z. Fan, X. Song, and R. Shibasaki, "CitySpectrum: A non-negative tensor factorization approach," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 213–223.
[52] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation," in *Proc.20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 831–840.
[53] Y. Wang *et al.*, "Regularity and conformity: Location prediction using heterogeneous mobility data," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1275–1284.
[54] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2537–2551, Nov. 2017.
[55] B. Guo, J. Li, V. W. Zheng, Z. Wang, and Z. Yu, "CityTransfer: Transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 4, 2018, Art. no. 135.
[56] Y. Lin *et al.*, "Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2018, pp. 359–368.
[57] X. Yi, J. Zhang, Z. Wang, T. Li, and Y. Zheng, "Deep distributed fusion network for air quality prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*. 2018, pp. 965–973.

[58] J. You, X. Li, M. Low, D. Lobell, and S. Ermon, "Deep Gaussian process for crop yield prediction based on remote sensing data," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4559–4565.
[59] C. Huang, C. Zhang, J. Zhao, X. Wu, N. Chawla, and D. Yin, "Mist: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting," in *Proc. World Wide Web Conf.*, 2019, pp. 717–728.
[60] J. Ye, L. Sun, B. Du, Y. Fu, X. Tong, and H. Xiong, "Co-prediction of multiple transportation demands based on deep spatio-temporal neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 305–313.

Renhe Jiang received the BS degree in software engineering from the Dalian University of Technology, China, in 2012, the MS degree in information science from Nagoya University, Japan, in 2015, and the PhD degree in civil engineering from The University of Tokyo, Japan, in 2019. Since 2019, he has been an assistant professor with Information Technology Center, The University of Tokyo. His research interests include ubiquitous computing, deep learning, and spatio-temporal data analysis.

Zekun Cai received the BS degree in computer science and technology from the University of Electronic Science and Technology of China in 2018. Since 2018, he has been a research student with the Department of Socio-Cultural Environmental Studies, The University of Tokyo. His research interests include artificial intelligence, deep learning, ubiquitous computing, and spatio-temporal data analysis.

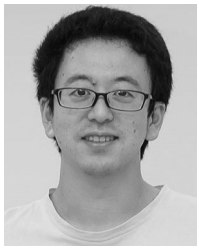
Zhaonan Wang received the BS degree in geographical information systems from Peking University, China, in 2014, and the MS degree in city planning from Boston University, USA, in 2017. In 2018, he joined the National Institute of Advanced Industrial Science and Technology as a technical staff. His research interests include ubiquitous computing and spatio-temporal data mining.



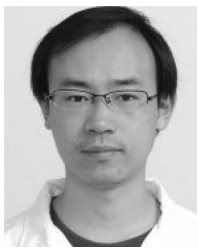
Chuang Yang received the BS degree in computer science and technology from the Southern University of Science and Technology in 2019. In 2019, he became a research student with the Department of Socio-Cultural Environmental Studies, The University of Tokyo. His current research interests include spatio-temporal data analysis, data visualization, and ubiquitous computing.

Zipei Fan received the BS degree in computer science from Beihang University, China, in 2012, and the MS and PhD degrees in civil engineering from The University of Tokyo, Japan, in 2014 and 2017, respectively. In 2017, he became a project assistant professor with the Center for Spatial Information Science, The University of Tokyo. His research interests include ubiquitous computing, machine learning, spatio-temporal data mining, and heterogeneous data fusion.

Quanjun Chen received the BS degree in automation from Hunan University, China, in 2011, the MS degree in pattern recognition and intelligent system from Beihang University, in 2014, and the PhD degree in civil engineering from The University of Tokyo, Japan, in 2017. In 2017, he became a post-doctoral researcher with the Center for Spatial Information Science, The University of Tokyo. His research interests include ubiquitous computing, machine learning, and traffic accident analysis.



Kota Tsubouchi received the PhD degree from the University of Tokyo, Japan, in 2010. Until March 2012, he did research about on-demand traffic systems with the University of Tokyo. Since April 2012, he has been a data scientist and a senior researcher with Yahoo JAPAN Research. His research interest focuses on data analysis including human activity logs, such as location information, search logs, shopping history, and sensor data.



Xuan Song received the BS degree in information engineering from Jilin University, China, in 2005, and the PhD degree in signal and information processing from Peking University, China, in 2010. He was promoted to the position of a project assistant professor and a project associate arofessor with the Center for Spatial Information Science, The University of Tokyo, in 2012 and 2015, respectively. His research interests include artificial intelligence, computer vision, and robotics, especially on smart city, intelligent system design, multitarget tracking, sensor fusion, and abnormality detection.

Ryosuke Shibasaki was born in Fukuoka, Japan. He received the BS, MS, and doctoral degrees in civil engineering from The University of Tokyo, Japan, in 1980, 1982, and 1987, respectively. From 1982 to 1988, he was with Public Works Research Institute, Ministry of Construction. From 1988 to 1991, he was an associate professor with Civil Engineering Department, The University of Tokyo. In 1991, he joined the Institute of Industrial Science, The University of Tokyo. In 1998, he was promoted to the position of a professor with the Center for Spatial Information Science, The University of Tokyo. His research interests include 3-D data acquisition for GIS, conceptual modeling for spatial objects, and agent-based microsimulation in a GIS environment.