DeepUrbanEvent: A System for Predicting Citywide Crowd Dynamics at Big Events

Renhe Jiang
Xuan Song
The University of Tokyo
National Institute of Advanced Industrial Science and Technology
[jiangrh,songxuan]@csis.u-tokyo.ac.jp

Dou Huang
Xiaoya Song*
The University of Tokyo
[huangd,song.xy]@csis.u-tokyo.ac.jp

Tianqi Xia†
Zekun Cai
The University of Tokyo
[xiatianqi,caizekun]@csis.u-tokyo.ac.jp

Zhaonan Wang
National Institute of Advanced Industrial Science and Technology
zn.wang@aist.go.jp

Kyoung-Sook Kim
National Institute of Advanced Industrial Science and Technology
ks.kim@aist.go.jp

Ryosuke Shibasaki
The University of Tokyo
shiba@csis.u-tokyo.ac.jp

ABSTRACT

Event crowd management has been a significant research topic with high social impact. When some big events happen such as an earthquake, typhoon, and national festival, crowd management becomes the first priority for governments (e.g. police) and public service operators (e.g. subway/bus operator) to protect people's safety or maintain the operation of public infrastructures. However, under such event situations, human behavior will become very different from daily routines, which makes prediction of crowd dynamics at big events become highly challenging, especially at a citywide level. Therefore in this study, we aim to extract the "deep" trend only from the current momentary observations and generate an accurate prediction for the trend in the short future, which is considered to be an effective way to deal with the event situations. Motivated by these, we build an online system called DeepUrbanEvent which can iteratively take citywide crowd dynamics from the current one hour as input and report the prediction results for the next one hour as output. A novel deep learning architecture built with recurrent neural networks is designed to effectively model these highly-complex sequential data in an analogous manner to video prediction tasks. Experimental results demonstrate the superior performance of our proposed methodology to the existing approaches. Lastly, we apply our prototype system to multiple big real-world events and show that it is highly deployable as an online crowd management system.

CCS CONCEPTS

• **Information systems → Information systems applications**; • **Human-centered computing → Ubiquitous and mobile computing**; • **Computing methodologies → Artificial intelligence**.

*Also with Harbin Institute of Technology.
†Also with National Institute of Advanced Industrial Science and Technology.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330654

KEYWORDS

crowd management; ubiquitous and mobile computing; deep learning; application and system

ACM Reference Format:
Renhe Jiang, Xuan Song, Dou Huang, Xiaoya Song, Tianqi Xia, Zekun Cai, Zhaonan Wang, Kyoung-Sook Kim, and Ryosuke Shibasaki. 2019. DeepUrbanEvent: A System for Predicting Citywide Crowd Dynamics at Big Events. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330654

1 INTRODUCTION

Event crowd management has been a significant research topic with highly social impact. When some big events happen such as an earthquake, typhoon, and national festival, crowd management becomes the first priority for governments (e.g. police) and public service operators (e.g. subway/bus operator) to protect people's safety or maintain the operation of public infrastructures. Especially for a large urban area such as Tokyo and Shanghai, the population density is very high, which naturally leads to high risk for various accidents and emergency situations. Recall the tragedy on New Year's Eve in Shanghai, around 300,000 people gathered to celebrate the arrival of 2015 near Chen Yi Square on the Bund. However, the large crowd was not well controlled, and a stampede occurred where 36 people died and 47 were injured in the tragedy. Meanwhile, AI technology is rapidly developing and the 5G mobile Internet technology is forthcoming. Big human mobility data are being continuously generated through a variety of sources, some of which can be treated and utilized as streaming data for understanding and predicting crowd dynamics. All these stimulate us to take new efforts and achieve new success on this social issue by using such streaming mobility data and advanced AI technologies. However, when big events or disasters happen, urban human mobility may dramatically change from normal situations. It means people's movements will almost be uncorrelated with their daily routines. As shown in Fig.2, the big earthquake occurred at 14:46 JST 11th March 2011. Citywide human mobility in Tokyo area was greatly impacted since the transportation network was suddenly shut down

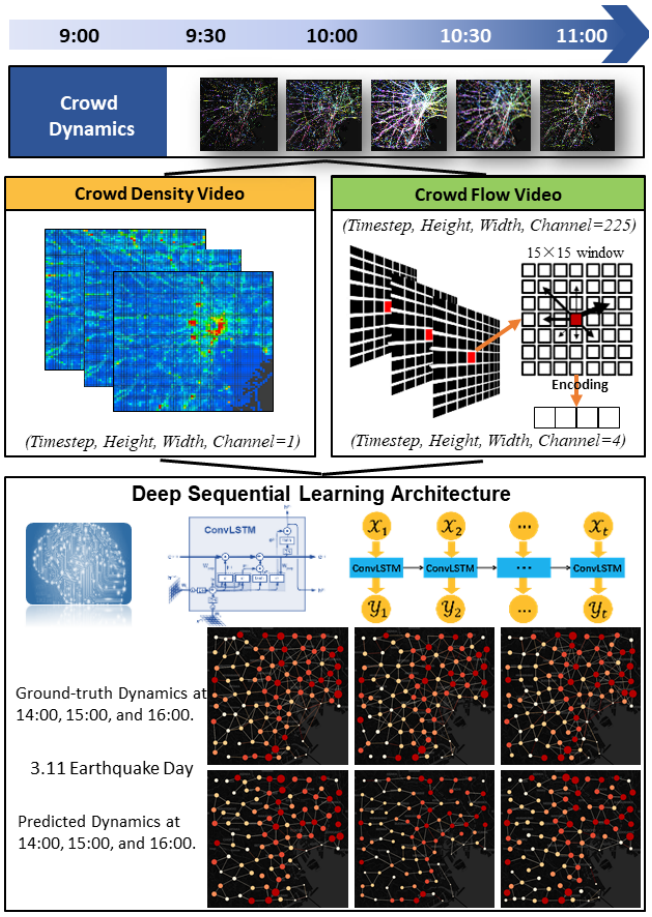

Figure 1: DeepUrbanEvent is designed as an effective real-world system for predicting citywide crowd dynamics at big events. Crowd dynamics graph for management can be built based on the predicted density (nodes) and flow (edges).

by the earthquake. Due to this big event, an abnormal pattern of crowd density can be observed in both Tokyo station area and Shinjuku station area. All these demonstrate that predicting crowd dynamics under event situations is of high social impact, but very challenging, especially at a citywide level.

To address this challenge, we aim to extract the "deep" trend only from the current momentary observations and generate an accurate prediction for the trend in the short future, which is considered to be an effective way to handle the event situations[7]. We build an intelligent system called DeepUrbanEvent based on collected big human mobility data and a unique deep-learning architecture. It is designed to be deployed as an online system for crowd management at big events, which can continuously take limited steps of currently observed crowd dynamics as input and report multiple steps of prediction results for a short time period in the future as output. With such multiple steps of prediction, it can help us understand how the crowd dynamics are evolving with more details.

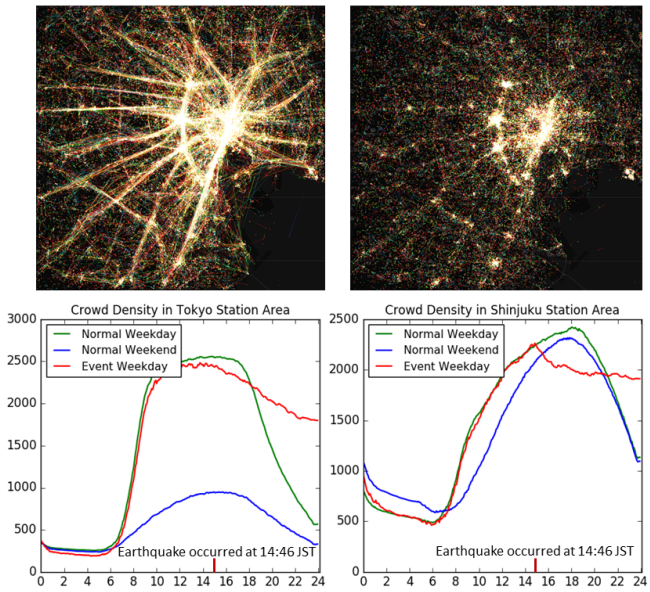

Figure 2: Citywide human mobility in Tokyo before and after the Great East Japan Earthquake (top). Crowd density in Tokyo station area and Shinjuku station area (bottom).

Specifically, in this study, citywide crowd dynamics are first decomposed into two parts: crowd density and crowd flow. By meshing a large urban area into fine-grained grids, they can both be represented by a four-dimensional tensor (*Timestep*, *Height*, *Width*, *Channel*) analogously to a short video, where *Timestep* represents the number of observation/prediction steps, and *Height*, *Width* is determined by mesh size. Crowd density/flow video represents a time series of density/flow value for each mesh-grid, therefore *Channel* for density is equal to 1, whereas *Channel* for flow is equal to the size of flow kernel window $\eta \times \eta$. The stored value indicates how many people inside a central mesh-grid will transit to each of $\eta \times \eta$ neighboring mesh-grids in a given time interval. A Multitask ConvLSTM Encoder-Decoder architecture is designed to simultaneously model these two kinds of high-dimensional sequential data to gain concurrent enhancement. Based on this architecture, our system works as an online system that can continuously take limited steps of currently observed crowd density and crowd flow as input, and report multiple steps of predictions results as output for the future time period. Finally, we validate our system on four big real-world events that happened in Tokyo area, namely 3.11 Japan Earthquake, Typhoon Roke(2011), New Year's Day(2012), and Tokyo Marathon(2011), and demonstrate the superior performance to baseline models. The overview of our system has been shown in Fig.1. In summary, our work has the following key characteristics that make it unique:

- For predicting crowd dynamics at citywide-level big events, we build an online deployable system that need only limited steps of current observations as input.
- Citywide crowd dynamics are decomposed into two kinds of artificial videos, namely crowd density video and crowd flow video, and a Multitask ConvLSTM Encoder-Decoder is

designed to simultaneously predict multiple steps of crowd density and flow for the future time period.
- We validate our system on four big real-world events with big human mobility data source and verify it as a highly deployable prototype system.

2 RELATED WORK

Forecasting the citywide crowd flow [12, 37, 38] are related works, which build a time-series prediction model based on inflow and outflow, which can only indicate how many people will flow into or out from a certain mesh-grid, and can't answer where the people flow come or transit. Their models also can't give out the crowd density prediction in a straight-forward way, which is very crucial for event crowd management.

CityProphet[14] and [40] utilize query data of Smartphone APP to forecast only crowd density other than crowd flow. [2, 27] conduct transition estimation from aggregated population data, and [29] estimates the transition populations using inflow and outflow defined by [12]. Some researchers tried to detect the urban anomalies from mobility data based on statistical methodologies [10, 36]. Modeling human mobility for very large populations [7, 25], predicting human mobility from the sparse and lengthy trajectories[9], and simulating human emergency mobility following disasters [26] are similar problems to ours, however, their models are built based on millions of individuals' mobility.

Many recent studies have analyzed human mobility data. For example, [6, 17, 18, 39] utilized the tensor factorization approach to decompose urban human mobility, which aimed to understand the basic life patterns of people or recommend location-based services. [23] conducted next place prediction in location-based services based on user features. Using population-scale data, [32] detected popular temporal modes and [8] modeled urban population of multiple cellphone networks. Moreover, some studies also applied deep learning to predict the traffic flow, traffic speed, congestion, taxi demand, and traffic accident[4, 13, 19–21, 33–35].

3 DATA SOURCE

"Konzatsu-Tokei (R)" from ZENRIN DataCom Co., Ltd. was used. It refers to people flow data collected by individual location data sent from mobile phones with an enabled AUTO-GPS function under the users' consent, through the "docomo map navi" service provided by NTT DoCoMo, Inc. Those data are processed collectively and statistically in order to conceal private information. The original location data is GPS data (latitude, longitude) sent at a minimum period of about 5 min, and does not include information (such as gender or age) to specify individuals. In this study, the proposed methodology is applied to raw GPS data from NTT DoCoMo, Inc. The raw GPS log dataset was collected anonymously from approximately 1.6 million mobile phone users in Japan over a three-year period (August 1, 2010, to July 31, 2013). Each record contains user ID, latitude, longitude, altitude, timestamp and accuracy level.

4 CITYWIDE CROWD DYNAMICS MODELING

Definition 1 (Calibrated human trajectory database): Human trajectory is stored and indexed by day (i) and userid (u) in the trajectory database Γ. Given a mesh M of an area, namely a set of mesh-grids

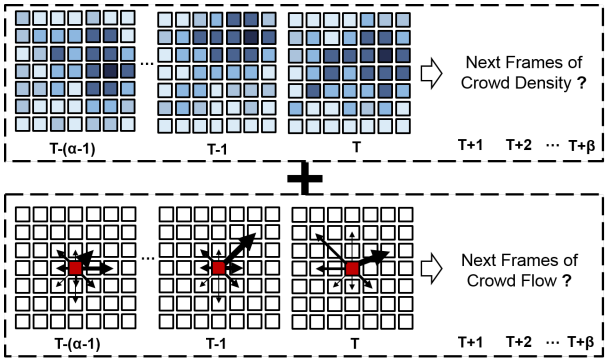

Figure 3: Citywide Crowd Dynamics Prediction.

$\{g_1, g_2, ..., g_m, ..., g_{Height*Width}\}$, and a time interval Δt, each user's trajectory on each day Γ_{iu} is mapped onto mesh-grids and then calibrated to obtain constant sampling rate as follows:

$$\Gamma_{iu} = (t_1, g_1), ..., (t_k, g_k) \wedge \forall j \in [1, k), |t_{j+1} - t_j| = \Delta t,$$

which means that the time interval between any two consecutive timeslots is calibrated into Δt. For simplicity, from now on we only consider one-day slice of the trajectory database Γ, then the day index (i) can be omitted when refer to Γ.

Definition 2 (Crowd density): Given Γ, M, crowd density at timeslot t on mesh-grid g_m is defined as follows:

$$d_{tm} = |\{u|\Gamma_u.g_t = g_m\}|,$$

which intuitively indicates how many people inside g_m at t.

Definition 3 (Crowd flow): To capture the crowd flow starting from a certain mesh-grid, we utilized a kernel window denoted as $\eta \times \eta$ w.r.t g_m, which represents a square area made up of $\eta \times \eta$ neighboring mesh-grids with g_m as the centroid mesh-grid. Given Γ, M, and a kernel window $\eta \times \eta$ w.r.t each g, crowd flow at timeslot t on mesh-grid g_m is defined as follows:

$$f_{tmw} = |\{u|\Gamma_u.g_{t-1} = g_m \wedge \Gamma_u.g_t = g_w\}|,$$

which intuitively indicates how many people transit from mesh-grid g_m at timeslot t-1 to a neighboring mesh-grid g_w inside a kernel window at timeslot t. After calculating the crowd density/flow for each mesh-grid over the entire mesh, citywide crowd density/flow can be obtained for each timeslot.

Definition 4 (Crowd density/flow video): As the mesh is represented in a 2-dimensional format, a crowd density/flow video containing a couple of consecutive frames can be represented by a 4-dimensional tensor $\mathbb{R}^{Timestep \times Height \times Width \times Channel}$, where $Timestep$ represents the number of video frames, $Channel$ for density is equal to 1, and $Channel$ for flow is equal to the size of the given kernel window namely η^2. An illustration for crowd density/flow video has been shown in Fig.1.

Definition 5 (Crowd density/flow video prediction): Given currently observed a-step crowd density/flow video $x_d = d_{t-(\alpha-1)}, ..., d_t$, $x_f = f_{t-(\alpha-1)}, ..., f_t$ at timeslot t, prediction for the next β-step density/flow video $\hat{y}_d = \hat{d}_{t+1}, ..., \hat{d}_{t+\beta}$, $\hat{y}_f = \hat{f}_{t+1}, ..., \hat{f}_{t+\beta}$ is modeled as

follows:

$$\hat{y}_d = \hat{d}_{t+1}, \hat{d}_{t+2}, ..., \hat{d}_{t+\beta} = \underset{d_{t+1}, d_{t+2}, ..., d_{t+\beta}}{\text{argmax}} P(d_{t+1}, d_{t+2}, ..., d_{t+\beta} \mid d_{t-(\alpha-1)}, ..., d_t),$$

$$\hat{y}_f = \hat{f}_{t+1}, \hat{f}_{t+2}, ..., \hat{f}_{t+\beta} = \underset{f_{t+1}, f_{t+2}, ..., f_{t+\beta}}{\text{argmax}} P(f_{t+1}, f_{t+2}, ..., f_{t+\beta} \mid f_{t-(\alpha-1)}, ..., f_t).$$

Definition 6 (Citywide crowd dynamics prediction): Given currently observed a-step crowd density/flow video, citywide crowd dynamics prediction aims to simultaneously generate next β-step density/flow video, which is modeled as follows:

$$\hat{y}_d, \hat{y}_f = \underset{y_d, y_f}{\text{argmax}} P(y_d, y_f \mid x_d, x_f).$$

Moreover, by jointly modeling these two highly correlated tasks, concurrent enhancement for both can be expected. It should be noted that crowd density video and crowd flow video are summarized and proposed as a new concept called crowd dynamics here, which aims to not only reflect the crowd density for each mesh-grid but also depict how a crowd of people move/transit among the mesh-grids. Fig.3 demonstrates the overall problem definition mentioned above.

5 DEEP SEQUENTIAL LEARNING ARCHITECTURE

As shown above, citywide crowd dynamics problem has been defined in an analogous manner to a video prediction task. However, citywide crowd dynamics are highly complex phenomenon especially when big events happen, which makes it very difficult for handling these high-dimensional sequential data with some classical methodologies. This naturally motivates us to employ the most advanced deep video learning model as the basic component of our system.

Convolutional LSTM. ConvLSTM[31] has been proposed to build an end-to-end trainable model for the precipitation nowcasting problem. It extends the fully connected LSTM (FC-LSTM) to have convolutional structures in both the input-to-state and state-to-state transitions and achieves new success on video modeling tasks. Thus, ConvLSTM is utilized as the core component of our system for the density and flow video prediction task. As shown in Fig.1, a ConvLSTM has three gates comprising an input gate i, an output gate o, and a forget gate f as same as an ordinary LSTM. Hidden state h_t in a ConvLSTM is calculated iteratively from t=1 to T for an input sequence of frames $(x_1, x_2, ..., x_T)$ as follows:

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

where W is weight, b bias vector, $*$ denotes the convolution operator and $\odot$ represents Hadamard product. All of these weight parameters are determined by applying the standard "backpropagation through time" (BPTT) algorithm, which starts by unfolding the recurrent neural networks through time and it then generalizes the backpropagation for feed-forward networks to minimize the defined loss function, which will be Mean Squared Error (*MSE*) for our problem. The full details of the algorithm are omitted here.

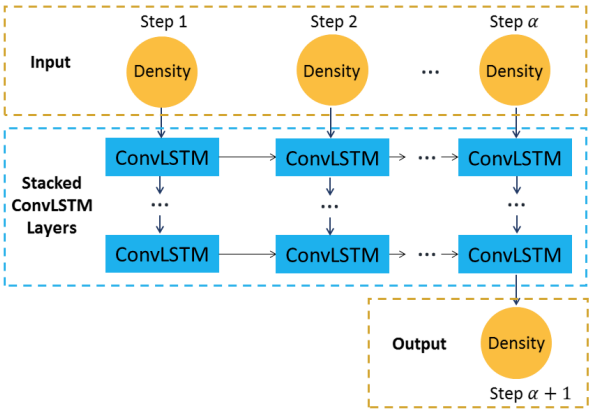

Figure 4: Stacked ConvLSTM for One-Step Prediction.

5.1 Stacked ConvLSTM Architecture

As a comparative modeling approach, we would like to verify how the performance of our system would be like if we use a one-step-by-one-step prediction model and obtain multiple steps of predictions by iterating in an autoregressive manner. Then one-step-by-one-step crowd density prediction model can be defined as follows:

$$\hat{d}_{\alpha+1}, \hat{d}_{\alpha+2}, ..., \hat{d}_{\alpha+\beta} = \prod_{i=1}^{\beta} \underset{d_{\alpha+i}}{\text{argmax}} P(d_{\alpha+i} \mid d_i, d_{i+1}, ..., d_{i+\alpha-1}).$$

The definition given above can be regarded as a typical application of the n-gram language model except that each item d is a 3D tensor. Crowd flow prediction could also be modeled in a similar formula, which will be omitted in this paper for simplicity.

Moreover, the use of multiple stacked layers of neural networks can also be considered to boost the performance in difficult time-series modeling tasks according to [11]. Thus, a deep architecture constructed with multiple stacked ConvLSTM layers has been shown in Fig.4 for one-step prediction. It has strong representational power which makes it suitable for giving predictions in complex phenomenons like the citywide crowd dynamics. Note that the same network architecture can also be applied to one-step crowd flow prediction, but a special AutoEncoder component is first necessary due to the uniqueness of crowd flow video which will be explained in the following.

5.2 CNN AutoEncoder for Crowd Flow

Crowd density and flow video are both represented as 4D tensor $\mathbb{R}^{Timestep \times Height \times Width \times Channel}$, however, the *Channel* for flow is much larger than density. In our system, each grid-cell is set to 500m×500m, by taking into account all the possible transportation modes such as WALK, BUS, CAR and TRAIN, the transition distance from one grid-cell to another neighboring one can be up to 4km within 5 minutes time interval (approximately 48km/h at most).

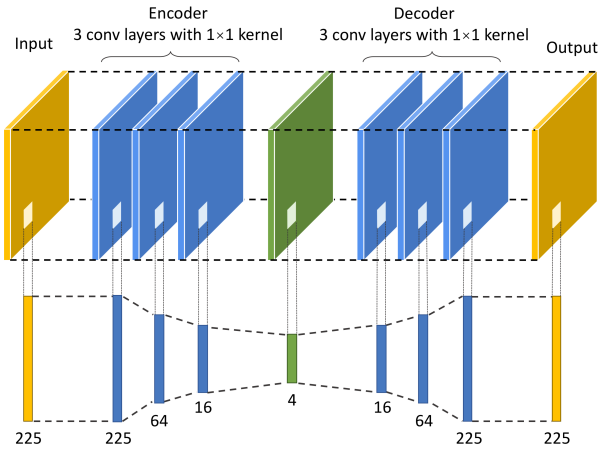

Figure 5: CNN AutoEncoder for Crowd Flow.

Thus kernel window needs to be 15×15 to capture all the possible crowd flow within 5 minutes. *Channel* for flow is then equal to 225, which is just too large for most of the state-of-the-art video learning models to handle. Thus, we build a special CNN AutoEncoder [3, 30] to obtain a low-dimension representation of *Channel* for crowd flow.

Compared with traditional neural networks, CNNs were designed specifically for analyzing visual imagery [16], where the neurons in a layer are only connected to a small region of the previous layer instead of all of the neurons in a fully-connected manner. CNNs are the state-of-the-art method for image recognition or classification tasks [15, 24]. For a typical CNN layer, the convolutional feature value at location (i, j) in the k-th feature map, feature $c_{i,j,k}$, is calculated as:

$$c_{i,j,k} = ReLU(w_k x_{i,j} + b_k),$$

where w_k and b_k are the weight and bias of the k-th filter, respectively, and $x_{i,j}$ is the input patch centered at location (i, j). *ReLU* is often used as the activation function.

The details of the special CNN AutoEncoder has been proposed in Fig.5. An original crowd flow image is represented with a 3D tensor (15, 15, 225), an encoder is constructed with 3 convolutional layers to encode the image into a small 3D tensor (15, 15, 4), and then an decoder is constructed with 3 convolutional layers to decode the compressed tensor back to the original 3D tensor (15, 15, 225). The end-to-end model parameters can be optimized by minimizing reconstruction error (*MSE*) between the original flow image and decoded flow image. In our system, we aim to obtain a compressed *Channel* (from 225 to 4) but keep the spatial structural information of the flow image at (15, 15). Thus, only convolutional layer with 1×1 kernel window is utilized. At the last layer of the encoder, a unique ReLU(MAX=1.0) function was utilized to ensure that the values are all scaled into [0,1], which can help the value range of crowd flow approximately same to the value range of crow density. Without this, the multitask learning mechanism introduced in the following couldn't function well.

5.3 Multitask ConvLSTM Encoder and Decoder

With such a CNN AutoEncoder, citywide crowd flow can be modeled and computed with the same architecture for crowd density shown in Fig.4. The prediction can be performed in an iterative one-by-one manner, but one major limitation of this model is to predict a relatively long short-term crowd dynamics. With the iteration going on, the accumulated iteration error will become large, which can result in terrible performance on the last several predicted steps. To tackle this problem, we improve the one-step-by-one-step modeling with multi-step-to-multi-step modeling (Definition 5) aimed at achieving better performance on "long" short-term predictions. To deliver this idea, a sequential encoder and decoder architecture [28, 31] has been built with four ConvLSTM layers in this study. It works in the following steps: (1) the first two hidden layers of ConvLSTM (encoder) map the α steps of the inputted crowd density or flow video into a single latent vector, which contains information about the entire video sequence; (2) this latent vector is repeated β times to a constant sequence; and (3) the other two hidden layers of ConvLSTM (decoder) are used to turn this constant sequence into the β steps of the output video sequence. Batch normalization layer is added between two consecutive ConvLSTM layers. *ReLU* is used as the activation function in the final decoding layer. The ConvLSTM Enc.-Dec. model for crowd density and flow can be separately trained by minimizing the prediction error $L(\theta_d)$ and $L(\theta_f)$, described as follows:

$$L(\theta_d) = ||\hat{\mathbb{Y}}_D - \mathbb{Y}_D||^2, L(\theta_f) = ||\hat{\mathbb{Y}}_F - \mathbb{Y}_F||^2$$

Crowd density video and crowd flow video share important information and are highly correlated with each other. The insights behind this are two folds: (1) People flow tend to follow the trend, especially under the event/emergency situations, which may make crowded places attract more and more people; (2) Higher inflow will lead to higher density for each grid-cell, and similarly higher outflow will reduce the crowd density. Moreover, as mentioned above, an online crowd management system needs to predict not only the crowd density but also the crowd flow for each grid-cell. Thus, we jointly model these two highly correlated tasks defined as Definition 6, and propose a Multitask ConvLSTM Encoder and Decoder architecture as shown in Fig.6. The key concept of multi-task learning [22] is to learn multiple tasks simultaneously with the aim of gaining mutual benefits; thus, learning performance can be improved through parallel learning while using a shared latent representation. Therefore, it is reasonable to expect better performances from this learning framework for our system. Our Multitask ConvLSTM Encoder and Decoder architecture first takes $\mathbb{X}_D$ and $\mathbb{X}_F$ as two separate inputs. The separate input encoders first encode the crowd density and crowd flow video respectively. Then, the shared encoder maps the encoded crowd density and flow into a joint latent representation z_α, which can be taken as the auto-extracted features for the entire crowd dynamics; z_α is then repeated β times to be passed to the shared decoder, and finally the output decoders give the multiple steps of prediction results for crowd density video $\hat{\mathbb{Y}}_D$ and flow video $\hat{\mathbb{Y}}_F$ respectively. The entire model can be trained by minimizing the total prediction error $L(\theta)$ of crowd density and flow, described as follows:

$$L(\theta) = \lambda_d||\hat{\mathbb{Y}}_D - \mathbb{Y}_D||^2 + \lambda_f||\hat{\mathbb{Y}}_F - \mathbb{Y}_F||^2.$$

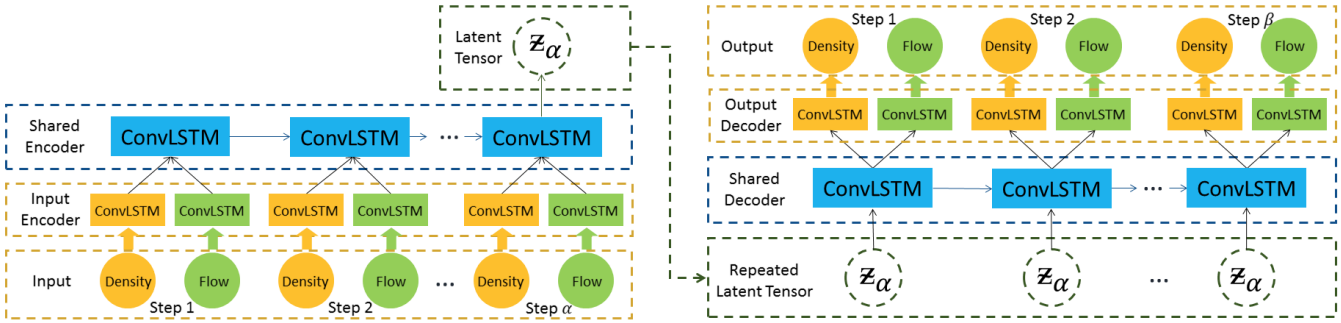

Figure 6: Multitask ConvLSTM Enocder-Decoder for Simultaneous Multi-Step Prediction of Crowd Density and Crowd Flow.

where λ_d and λ_f are set equally to 0.5 in our final system. CNN AutoEncoder still needs to be applied first to original crowd flow.

6 EXPERIMENT

6.1 Settings

Experimental Setup: We selected the Greater Tokyo Area ($Long. \in [139.50, 139.90]$, $Lat. \in [35.50, 35.82]$) as our target urban area. Four citywide-level events happened in this area were selected as the testing events as follows. (1) 3.11 Earthquake (2011/03/11), a magnitude 9.0-9.1 earthquake off the coast of Japan that occurred at 14:46 JST, which caused a great impact on people's behaviors in the Great Tokyo Area. (2) Typhoon Roke (2011/09/21), recorded as one of the strongest typhoon in Japan's history, which made subway operators shut down part of their services. (3) New Year's Day (2012/01/01). There are a number of New Year celebrations in Tokyo area, especially, for "Hatsumode" (the first visit in Buddhist temple or shrine), most of the railway lines operate overnight on the New Year's Eve for this. (4) Tokyo Marathon(2011/02/27). The number of people attending this event was 2.16 million (the number of people along the road was 1.53 million, and the number of visitors to the Tokyo Marathon Festival was 0.63 million). Also traffic regulation was strictly conducted along the Marathon route. These four event days were used as testing dates, and 10 consecutive days before the event day were utilized as training and validation dataset, which means 2011/03/01-2011/03/10, 2011/09/11-2011/09/20, 2011/12/22-2011/12/31, and 2011/02/17-2011/02/26 were the selected periods for the four events respectively. Our data source contained approximately 100,000~130,000 users' GPS logs on each day within the target urban area. After conducting data cleaning and noise reduction to the raw dataset, we did linear interpolation to make sure each user's 24-hour (00:00~23:59) GPS log has a constant 5-minute sampling rate. Then by mapping each coordinate onto mesh-grid, crowd density video and crowd flow video can be generated based the definitions listed in Section 4.

Parameter Settings: We meshed the entire area with $\Delta Long.$=0.005, $\Delta Lat.$=0.004 (approximately $500m \times 500m$), which resulted an 80×80 mesh-grid map. As mentioned above, the time interval Δt of our system was set to 5 minutes. Therefore, we got 2880 timeslots (288 * 10 days) as training dataset and 288 timeslots as testing dataset, and crowd density frame and crowd flow frame were generated for each timeslot. Kernel window was set to 15×15 for crowd flow, which could capture enough transit distance of crowd flow within 5 minutes. We set the observation step α and the prediction step β both to 6 to generate length-6 crowd/flow video as inputs and their corresponding next length-6 videos as outputs. This means our system could predict the crowd dynamics for the next 30 minutes. In each report, it contained a 6 steps of prediction results for each 5 minutes, and the result at 6th step gave us the maximum lead time 30 minutes. Similarly, an evaluation for the prediction with 60 minutes lead time is also conducted by setting α and β to 12. Finally we could get 2,868 sample pairs from training dataset, and randomly selected 80% of them (2,294) as the training samples and 20% of them (574) as the validation samples. The Adam algorithm was employed to control the overall training process, where the batch size was set to 4 and the learning rate to 0.0001 for all deep learning models except that the learning rate of CNN AutoEncoder was tuned as 0.001. The training algorithm would be stopped after 200 epochs and only the best model would be saved. In addition, we used 500 as the scaling factor for crowd density to scale the data down to relatively small values, and 100 as the scaling factor for crowd flow value. In the evaluation, we rescaled the predicted value back to the normal values, and compared them with the ground-truth. The parameter settings were kept the same for each event. Python and some Python libraries such as Keras[5] and TensorFlow[1] were used in this study. The experiments were performed on a GPU server with four GeForce GTX 1080Ti graphics cards.

Baseline models: We implemented the following models as baseline models for comparison. (1) HistoricalAverage. Crowd density/flow for each timeslot were estimated by averaging last 10 days' corresponding values. (2) CopyYesterday. We directly used yesterday's value as the predicted value on event days. (3) CopyLastFrame. We directly copied the last/latest observation as the predicted value, which can be a simple but very effective method for event situation. (4) ARIMA. It is a classical time-series prediction model designed for one dimensional data. For each mesh-grid, we build one ARIMA model to predict the time-series density prediction. However, for flow tensor (80,80,225) at each timeslot, the dimension was just too high for ARIMA to handle. (5) VectorAutoRegressive. It is an advance time-series prediction model designed for high dimensional data. By flattening density tensor (80,80,1) at each timeslot into 6400-dimension vector, the model could handle the crowd density prediction task. For flow tensor (80,80,225), the dimension was also just too high for VAR to deal with. (6)

Table 1: Performance Evaluation of 30 Minutes Ahead Prediction on Four Events

Model	3.11 Earthquake		Typhoon Roke		New Year's Day		Tokyo Marathon	
	Density	Flow	Density	Flow	Density	Flow	Density	Flow
HistoricalAverage	106.032	0.726	75.402	0.519	176.013	1.099	33.381	0.223
CopyYesterday	129.436	0.912	85.641	0.592	110.444	0.660	65.765	0.437
CopyLastFrame	7.824	0.116	9.756	0.186	5.498	0.079	6.496	0.107
ARIMA	10.430	NA	13.376	NA	8.343	NA	7.808	NA
VectorAutoRegressive	10.843	NA	13.377	NA	9.511	NA	9.380	NA
CityMomentum [7]	27.670	0.653	29.305	0.962	23.058	0.235	25.774	0.475
ST-ResNet [37]	6.542	0.113	7.802	0.183	4.544	0.080	5.548	0.103
CNN	8.698	0.178	10.245	0.196	6.178	0.083	6.614	0.100
CNN Enc.-Dec.	7.115	0.117	8.571	0.187	5.216	0.079	6.004	0.095
MultiTask CNN Enc.-Dec.	6.802	0.119	8.226	0.197	5.158	0.084	5.953	0.097
ConvLSTM	6.737	0.124	7.959	0.195	4.679	0.077	5.675	0.094
ConvLSTM Enc.-Dec.	6.281	0.102	7.508	0.171	4.500	0.074	5.372	0.089
MultiTask ConvLSTM Enc.-Dec.	**5.549**	**0.102**	**6.753**	**0.170**	**4.117**	**0.074**	**5.012**	**0.086**

Table 2: Performance Evaluation of 60 Minutes Ahead Prediction on Four Events

Model	3.11 Earthquake		Typhoon Roke		New Year's Day		Tokyo Marathon	
	Density	Flow	Density	Flow	Density	Flow	Density	Flow
HistoricalAverage	104.604	0.731	75.927	0.529	175.344	1.102	33.422	0.225
CopyYesterday	128.133	0.920	86.069	0.601	106.991	0.645	65.725	0.440
CopyLastFrame	13.020	0.168	16.607	0.252	8.650	0.096	10.004	0.128
ARIMA	24.296	NA	32.933	NA	15.411	NA	15.259	NA
VectorAutoRegressive	22.355	NA	29.872	NA	21.072	NA	17.261	NA
CityMomentum [7]	32.034	0.570	35.090	0.821	25.867	0.207	28.825	0.400
ST-ResNet [37]	11.899	0.157	13.418	0.238	7.633	0.103	8.501	0.124
CNN	12.247	0.189	17.670	0.360	10.469	0.119	12.114	0.223
CNN Enc.-Dec.	11.372	0.164	13.876	0.245	8.311	0.097	9.127	0.119
MultiTask CNN Enc.-Dec.	10.812	0.177	13.800	0.247	8.153	0.101	9.004	0.124
ConvLSTM	11.355	0.139	12.285	0.228	7.615	0.118	9.511	0.140
ConvLSTM Enc.-Dec.	9.309	0.122	11.186	0.197	6.885	0.086	7.843	0.103
MultiTask ConvLSTM Enc.-Dec.	**8.094**	**0.122**	**9.900**	**0.196**	**6.496**	**0.085**	**7.483**	**0.101**

CityMomentum[7]. It was firstly proposed for momentary mobility prediction at the citywide level for big events. Although the model was build from a perspective of individual's mobility, the predicted/simulated trajectory of each individual could be used for generating aggregated crowd density and flow, which makes it comparable with our system. (7) ST-ResNet[37]. This deep residual learning-based method shows a state-of-the-art performance on citywide crowd flow prediction. To compare its performance under the same problem definition with us, we adapt ST-ResNet to take in limited steps of latest observations on crowd density/flow right and perform one-step-by-one-step autoregression to obtain multiple steps of predictions. Here, we also found that 1-residual-unit ST-ResNet without external features could achieve the best performances on our event situations. (8) CNN. It is a one-step predictor constructed with four Conv layers. Note that the 4D tensor would be converted to 3D tensor ($Height, Width, Timestep * Channel$) by concatenating the channels at each timestep just as the way [37] did, so that CNN could take our 4D tensors as inputs. The first three Conv layers used 32 filters of 3×3 kernel window, and the final Conv layer used a ReLU activation function to output single step of video frame. (9) CNN Enc.-Dec.. It is a multi-step predictor also constructed with four Conv layers. It shares the same parameter settings with (5). The only difference is the final Conv layer outputs a 3D tensor ($Height, Width, Timestep * Channel$) as multiple steps of predictions. (10) Multitask CNN Enc.-Dec. It has 4 Conv layers sharing a similar multitask architecture as illustrated in Fig.6, namely separate input encoding Conv layer, shared encoding and decoding layer, and separate output Conv layer. All the parameters were kept same with (6). (11) ConvLSTM (one-step-by-one-step) and (12) ConvLSTM Enc.-Dec (multi-step-to-multi-step) are the proposed comparison models constructed with four ConvLSTM layers in Section 5. Each ConvLSTM layer uses a 32 filters of 3×3 kernel window and the ReLU activation is used in the final layer. BatchNormalization was added between two consecutive CNN/ConvLSTM layers for all the models. Note that for all of the crowd flow parts, as shown in Fig. 5, CNN AutoEncoder will be first applied to encode the original flow tensor and then decode the (predicted) encoded flow back to the original format. Our final system is implemented using **MultiTask ConvLSTM Enc.-Dec.**. The source codes for these models have been released at github.com/deepkashiwa/DeepUrbanEvent.

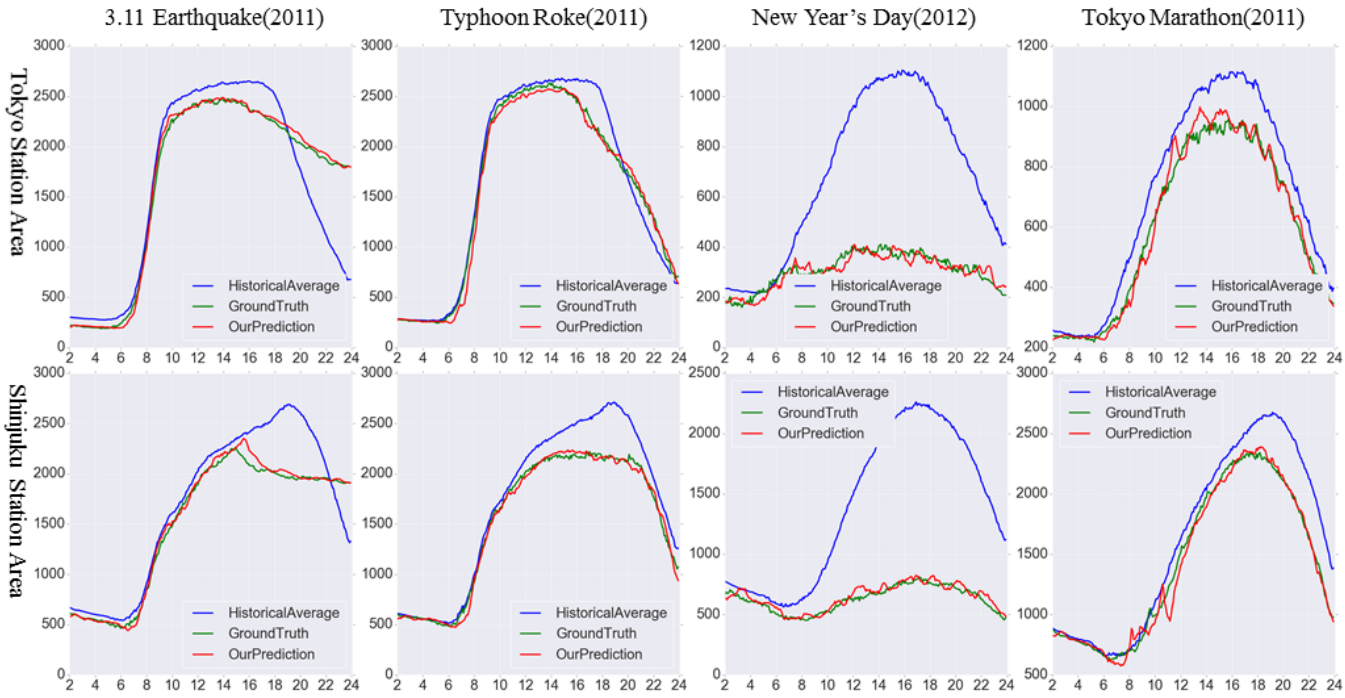

Figure 7: Visualization of the ground-truth crowd density, the prediction result of Historical Average (seen as the normal situation), and the prediction result of our model (MultiTask ConvLSTM Enc.-Dec.) at four events (60 minutes ahead).

6.2 Performance Evaluation

Evaluation metric: We evaluated the performances of the models with Mean Squared Error (*MSE*) as follows:

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

where n is the number of samples, $\mathbb{Y}$ and $\hat{\mathbb{Y}}$ are the ground-truth value and predicted value in 4D tensor format, namely, (*Timestep*, *Height*, *Width*, *Channel*). Density tensor and flow tensor differ at the *Channel*.

Overall performance: We compared the performance of the baseline models and our proposed model **Multitask ConvLSTM Enc.-Dec.** on four events. The overall evaluation results are summarized in Table 1 for 30 minutes ahead prediction and Table 2 for 60 minutes ahead prediction, which both show that based on all four events: (1) our model performed better than the others; and (2) all deep learning models had advantages compared with existing methodologies (CityMomentum and VAR). In particular, we could also find that (1) the superiority of ConvLSTM to CNN on video-like modeling tasks; (2) Encoder-Decoder architecture had the advantage on multi-step sequential prediction task; and (3) the effectiveness of mutlitask learning on enhancing the correlated tasks.

Performance on density: We also verified the performance of our system by using a times-series evaluation over the event day to show the ground-truth and predicted density for selected areas (Tokyo Station Area and Shinjuku Station Area) in the city. Each area consist of 3×3 neighboring mesh-grids, with Tokyo Station and Shinjuku Station locating at the central mesh-grid respectively.

From Fig.7, we can straightforwardly confirm the effectiveness of our model for 60 minutes ahead prediction and its high deployability for a real-world online event crowd management system. Referring to the normal situation (prediction result of HistoricalAverage) shown in the figure, we can find that the densities on event days differ a lot from normal situations. Furthermore, even comparing these four events, the density patterns are quite different with each other. This could further demonstrate the crowd management at event situations is really challenging and our online prediction system can be so indispensable for these special cases.

7 CONCLUSION

In this study, we built a data-driven intelligent system called DeepUrbanEvent to predict citywide crowd dynamics at big events in an analogous manner to a video prediction task. We proposed to decompose crowd dynamics into crowd density and crowd flow and designed a Multitask ConvLSTM Encoder-Decoder architecture to simultaneously predict multiple steps of crowd density and crowd flow for the future time period. The experimental results based on four big real-world events demonstrated the superior performance of our proposed model compared with the baseline methods. However, our method can still be improved in the following aspects. (1) Transportation network data and other types of heterogeneous data such as the census data can be utilized to improve the performance. (2) Current dataset contained approximately 1% of the total population of Japan. We need to collect more trajectory data from other sources and design reasonable scaling factors in order to predict crowd dynamics closer to the reality.

ACKNOWLEDGMENTS

This work was supported by Leading Initiative for Excellent Young Researchers (LEADER) Program and Grant in-Aid for Scientific Research B (17H01784) of Japans Ministry of Education, Culture, Sports, Science, and Technology(MEXT); JST, Strategic International Collaborative Research Program (SICORP); the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

[2] Yasunori Akagi, Takuya Nishimura, Takeshi Kurashima, and Hiroyuki Toda. 2018. A Fast and Accurate Method for Estimating People Flow from Spatiotemporal Population Data.. In *IJCAI*. 3293–3300.

[3] Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.

[4] Pablo Samuel Castro, Daqing Zhang, and Shijian Li. 2012. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Pervasive Computing*. Springer, 57–72.

[5] François Chollet. 2015. keras. https://github.com/fchollet/keras.

[6] Zipei Fan, Xuan Song, and Ryosuke Shibasaki. 2014. CitySpectrum: a non-negative tensor factorization approach. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 213–223.

[7] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 2015. CityMomentum: an online approach for crowd behavior prediction at a citywide level. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 559–569.

[8] Zhihan Fang, Fan Zhang, Ling Yin, and Desheng Zhang. 2018. MultiCell: Urban Population Modeling Based on Multiple Cellphone Networks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 106.

[9] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1459–1468.

[10] Takashi Fuse and Keita Kamiya. 2017. Statistical anomaly detection in human dynamics monitoring using a hierarchical dirichlet process hidden Markov model. *IEEE Transactions on Intelligent Transportation Systems* 18, 11 (2017), 3083–3092.

[11] Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*. 190–198.

[12] Minh X Hoang, Yu Zheng, and Ambuj K Singh. 2016. Forecasting citywide crowd flows based on big data. *ACM SIGSPATIAL* (2016).

[13] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. 2014. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *Intelligent Transportation Systems, IEEE Transactions on* 15, 5 (2014), 2191–2201.

[14] Tatsuya Konishi, Mikiya Maruyama, Kota Tsubouchi, and Masamichi Shimosaka. 2016. CityProphet: city-scale irregularity prediction using transit app logs. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 752–757.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[17] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 831–840.

[18] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1043–1051.

[19] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: a deep learning approach. *Intelligent Transportation Systems, IEEE Transactions on* 16, 2 (2015), 865–873.

[20] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies* 54 (2015), 187–197.

[21] Xiaolei Ma, Haiyang Yu, Yunpeng Wang, and Yinhai Wang. 2015. Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS one* 10, 3 (2015), e0119044.

[22] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 689–696.

[23] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. 2012. Mining user mobility features for next place prediction in location-based services. In *Data mining (ICDM), 2012 IEEE 12th international conference on*. IEEE, 1038–1043.

[24] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[25] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. 2010. Modelling the scaling properties of human mobility. *Nature Physics* 6, 10 (2010), 818–823.

[26] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Teerayut Horanont, Satoshi Ueyama, and Ryosuke Shibasaki. 2013. Modeling and probabilistic reasoning of population evacuation during large-scale disaster. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1231–1239.

[27] Akihito Sudo, Takehiro Kashiyama, Takahiro Yabe, Hiroshi Kanasugi, Xuan Song, Tomoyuki Higuchi, Shin'ya Nakano, Masaya Saito, and Yoshihide Sekimoto. 2016. Particle filter for real-time human mobility prediction following unprecedented disaster. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 5.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[29] Yusuke Tanaka, Tomoharu Iwata, Takeshi Kurashima, Hiroyuki Toda, and Naonori Ueda. 2018. Estimating Latent People Flow without Tracking Individuals.. In *IJCAI*. 3556–3563.

[30] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11, Dec (2010), 3371–3408.

[31] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*. 802–810.

[32] Fengli Xu, Tong Xia, Hancheng Cao, Yong Li, Funing Sun, and Fanchao Meng. 2018. Detecting Popular Temporal Modes in Population-scale Unlabelled Trajectory Data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 46.

[33] Yu Yang, Fan Zhang, and Desheng Zhang. 2018. SharedEdge: GPS-Free Fine-Grained Travel Time Estimation in State-Level Highway Systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 48.

[34] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[35] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-ConvLSTM: A Deep Learning Approach to Traffic Accident Prediction on Heterogeneous Spatio-Temporal Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 984–992.

[36] Huichu Zhang, Yu Zheng, and Yong Yu. 2018. Detecting Urban Anomalies Using Multiple Spatio-Temporal Data Sources. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 54.

[37] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.. In *AAAI*. 1655–1661.

[38] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 92.

[39] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the 19th international conference on World wide web*. ACM, 1029–1038.

[40] Jingbo Zhou, Hongbin Pei, and Haishan Wu. 2018. Early Warning of Human Crowds Based on Query Data from Baidu Maps: Analysis Based on Shanghai Stampede. In *Big Data Support of Urban Planning and Management*. Springer, 19–41.