THE UNIVERSITY OF TOKYO

博士論文

A Study on Modeling and Analyzing Urban Human Mobility with Deep Learning

（深層学習を用いた都市スケールの群衆移動
のモデリング及び解析に関する研究）

姜 仁河

Abstract

Rapidly developing location acquisition technologies have provided us with big GPS trajectory data, which offers a new means of understanding people's daily behaviors as well as urban dynamics. In this study, a raw GPS log dataset was collected from approximately 1.6 million mobile phone users in Japan over a three-year period (August 1, 2010 to July 31, 2013). With such data, predicting human mobility at the city level will be of great significance for transportation scheduling, urban regulation, and emergency management. However, in order to do this, we still confront with a few challenges. (1) The urban area is too large, taking the Greater Tokyo area as an example, it is the most populous metropolitan area in the world, has an urban area that can reach 3.925 km^2, and its metropolitan area can be 14.034 km^2. How to deal with such huge spatial domain is the first challenge; (2) Collected data used for model training are often limited to a small portion of the total population, and we can't collect every citizen's long-term historical trajectory data. How to train an effective model with limited training data (e.g., 1% of the total population) is the second challenge; (3) Under some real-world application scenario such as crowd management and crowd monitoring, it is important to predict crowd mobility as well as crowd density, especially for the latter one. Because high crowd density naturally means high risk for accidents. How to simultaneously predict crowd density and crowd flow is the third challenge; (4) Sometimes we may have no time or historical data for training a prediction model. Moreover, when some big events happen such as an earthquake, typhoon, and national festival, people change their behaviors from their routine activities. Thus, a model trained with historical data can't work very well for such kind of circumstances. Recently, the success of deep learning in the fields of computer vision and natural language processing motivates us to consider deep learning techniques as highly potential solutions to our problems, because it has the following advantages: (1) It can handle real big data; (2) It can model highly complex spatiotemporal system; (3) It can deal with multimodal distribution; (4) It can fuse multiple heterogeneous data. Then we propose four deep-learning-based solutions to address those four challenges and demonstrate the superior performances to the baseline methodologies. Finally, we summarize all of the works including some complementary works, discuss the limitations of current proposed solutions, and point out the future works.

Acknowledgements

First, the very special thanks to my parents, Zaiyun Jiang and Ping Qiao, also to other family members. My parents and other family members have given me tremendous help during my life. They are the motivation for me to do everything. I was born in a small city in Liaoning Province where the educational resources were limited, however, my parents and grandparents still made me receive the best education from elementary school to middle school and high school. I remember the slogan written above the gate of my high school: "from here to the world". When I first saw this as a 15-year-old teenager, I wondered if this could come true for me one day in the future. Somehow these encouraging words were well imprinted on my brain. After many years, I saw another encouraging sentence at the first page of the guidebook for Doctoral Program of School of Engineering of the University of Tokyo: "Ph.D. is the passport to the world". What a coincidence. What a sequence. Now I am so close to the completion of the Doctoral course, but I hope I can hang in there in every challenge and keep going on the long long road. From here to the world with my passport, bravity, and courage.

Second, a lot of thanks to my four supervisors in my Doctoral course in the University of Tokyo, Prof. Ryosuke Shibasaki, Prof. Yoshihide Sekimoto, Prof. Xuan Song, and Dr. Zipei Fan. Many thanks to my committee members for my Ph.D. dissertation, Prof. Takashi Oguchi, Prof. Takashi Fuse, and Prof. Masamichi Shimosaka. Also, many thanks to other research colleagues in my Doctoral course, Dr. Quanjun Chen, Tianqi Xia, Satoshi Miyazawa, Dr. Qi Chen, Zhiling Guo, Dou Huang, Shuzhe Huang, Ruochen Si, Min Lu, Guangming Wu, Tengyao Yu, Xiaoya Song, Dr. Haoran Zhang, Xiaodan Shi, Yuxuan Wang, Wenxiao Jiang, Dr. Yi Sui, Zekun Cai, Zhaonan Wang.

Third, many thanks to my supervisors and colleagues in my Master course in Nagoya University, Prof. Yoshiharu Ishikawa and Prof. Chuan Xiao, Dr. Xi Guo, Dr. Tingting Dong, Jing Zhao, Pei Wang, Dr. Jie Zhang, Dr. Yuya Sasaki, Kento Sugiura, Teppei Inaba, Takeshi Sugiyama, Arata Hayashi, Lei Ma. Moreover, I feel so grateful to my undergraduate university, Dalian University of Technology, where I learned computer science and second foreign lauguage Japanese. Thanks to the friends in 0807 and Room 401–407, Shuaiyao Wang, Tong Shen, Feng Ding, Lingjun Shan, Chao Xue, Lei Yan, Guifan Zhang, Zhengyao Li, Lijun Kuang, Yue Li, Guoliang Xu, Sihan Zhang. Last, thanks to the people who ever helped me from 1990, especially my friends, Lu Yao, Shanwei Wang, Renze Zhang, Heng Yu, Yan Wang, Ao Liu, Cheng Zhang, Xin Wang.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Big Human Mobility Data

A raw human trajectory collected from an individual person is essentially a sequence of timestamped locations: (*timestamp*, *location*), which can indicate a person's location according to a captured timestamp. A typical GPS trajectory data is shown in Fig.1.1.

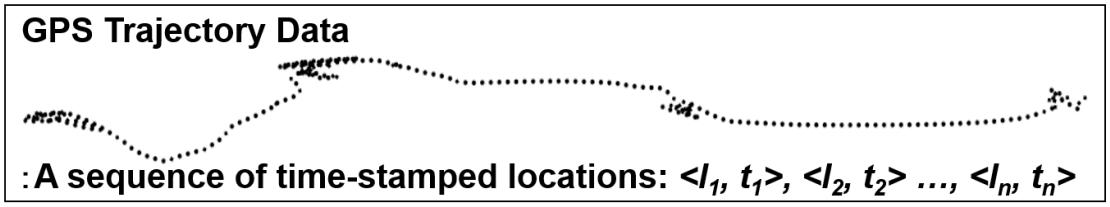


FIGURE 1.1: A typical GPS trajectory data.

Due to the continuing development of location acquisition technologies, massive GPS trajectory data are generated by various sources such as car navigation systems, mobile phones, location-based social networks, and Wi-Fi log as shown in Fig.1.2. Comparing to other data sources, mobile phone has its huge advantage. For example, data from car navigation system are biased, data from location-based social networks have low sampling rate, and wifi log data may have small data volume.

In this study, "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

FIGURE 1.2: GPS trajectory data are generated from various data sources.

is GPS data (latitude, longitude) sent at a minimum period of about 5 minutes, 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). It contains approximately 30 billion GPS records, and the total size of the data is more than 1.5 terabytes. Each record contains user ID, latitude, longitude, altitude, timestamp and positioning accuracy level, there are three levels due to different satellite's signal strength, correspondingly the positioning error would be within 100m, 200m or 300m. Anonymization and aggregation are carefully conducted to protect user privacy. Here is a snapshot of a normal weekday morning in Tokyo and Osaka as shown in Fig.1.3.

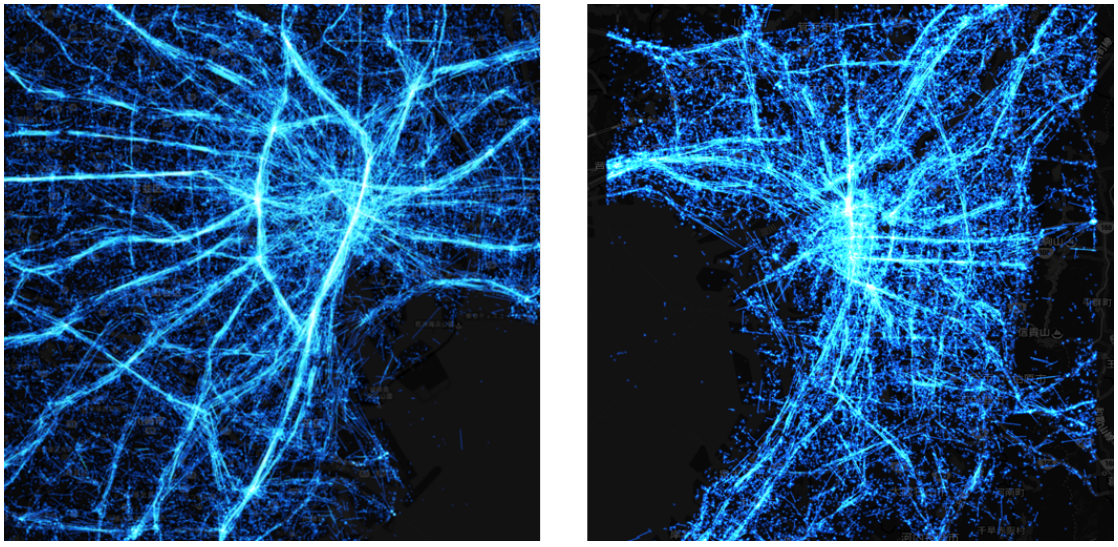


FIGURE 1.3: One snapshot on a normal weekday 8.am. in Tokyo and Osaka.

Obviously, human activities in city are closely linked with point-of-interest (POI) information, which can reflect the semantic meaning of human mobility. By combining human mobility data and city POI data, a more effective representation of human mobility can be expected. Moreover, although cities can have different types, scales, and

developmental levels, the POI distrubtions similar with each other. For example, a business area often has more POIs (e.g. offices, shopping malls, and restaurants) and locates at central part of city, while a residential area comes in an opposite way. Human mobility in different cities generally follow the similar patterns. Taking commuting pattern for example, people move from residential area to central business area to work and then return to residential district. This provides us the possibilities to transfer human mobility knowledge between cities via POI information. Thus, in this study, in addition to GPS trajectory data, we also collected big POI data for every major city in Japan as geographical data by utilizing "Telepoint Pack DB February 2014" provided by ZENRIN DataCom Co., Ltd [1]. In the original database, each record is a registered land-line telephone number with coordinates (latitude, longitude) and industry category information included. We treated each "telepoint" as one specific POI.

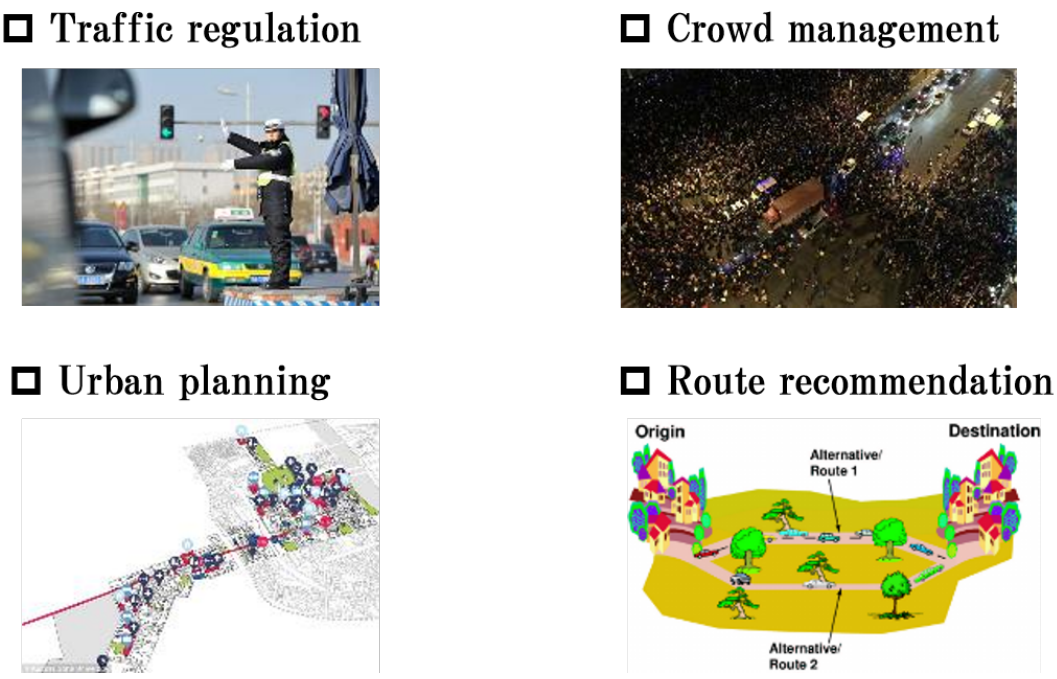


FIGURE 1.4: Urban computing scenarios.

1.2 Urban Human Mobility

Simulating/predicting human mobility (individuals' behaviors) with big trajectory data for a large urban area is a significant research topic, which is related to a variety of urban computing scenarios as shown in Fig.1.4. For example, it will be of great significance for transportation scheduling, urban regulation, and emergency management.

[1] https://joras.csis.u-tokyo.ac.jp/dataset/show/id/14000201400

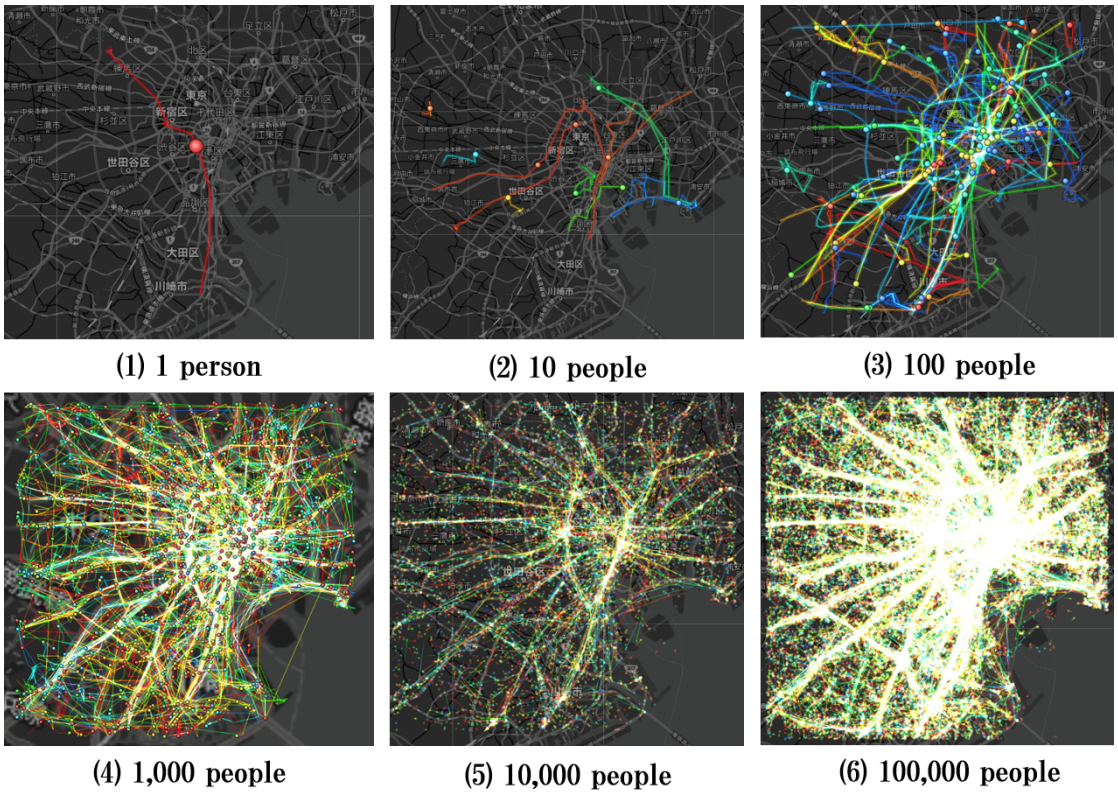


FIGURE 1.5: Urban human mobility (from 1 person's mobility to 0.1 million peoples' mobility).

Here is an illustration from 1 person's mobility to 0.1 million peoples' mobility as shown in Fig.1.5. Individual human trajectory prediction has been widely studied in recent years in the field of urban computing, but it is very difficult to expand such kind of individual modeling methodology to a citywide level. Since there are millions of people in a big city such as Tokyo, Shanghai, and Hong Kong, it is just infeasible to build a prediction model for each person by using his/her long historical data, which can also be an infringement on individual privacy. Moreover, crowd management under emergency situations is considered as a direct application scenario of human mobility prediction model. For this scenario, comparing with precisely mastering each individual's location, knowing and controlling the crowd density for any urban region is the real demand of governments (e.g. police) or public service operators (e.g. subway/bus companies, mobile service providers). Thus, in this study, our goal is to build one general model to effectively predict human mobility at a citywide level. However, in order to model and analyze urban human mobility, we still confront with a few challenges.

1.3 Challenges

Modeling and analyzing urban human mobility has never been an easy task for the following reasons: (1) The urban area is too large, taking the Greater Tokyo area as an example, it is the most populous metropolitan area in the world, has an urban area that can reach 3.925 km^2, and its metropolitan area can be 14.034 km^2. How to deal with such huge spatial domain is the first challenge; (2) Collected data used for model training are often limited to a small portion of the total population, and we can't collect every citizen's long-term historical trajectory data. How to train an effective model with limited training data (e.g., 1% of the total population) is the second challenge; (3) Under some real-world application scenario such as crowd management and crowd monitoring, it is important to predict crowd mobility as well as crowd density, especially for the latter one. Because high crowd density naturally means high risk for accidents. How to simultaneously predict crowd density and crowd flow is the third challenge; (4) Sometimes we may have no time or historical data for training a prediction model. Moreover, when some big events happen such as an earthquake, typhoon, and national festival, people change their behaviors from their routine activities. Thus, a model trained with historical data can't work very well for such kind of circumstances. The four challenges have been proposed and summarized as Fig.1.6

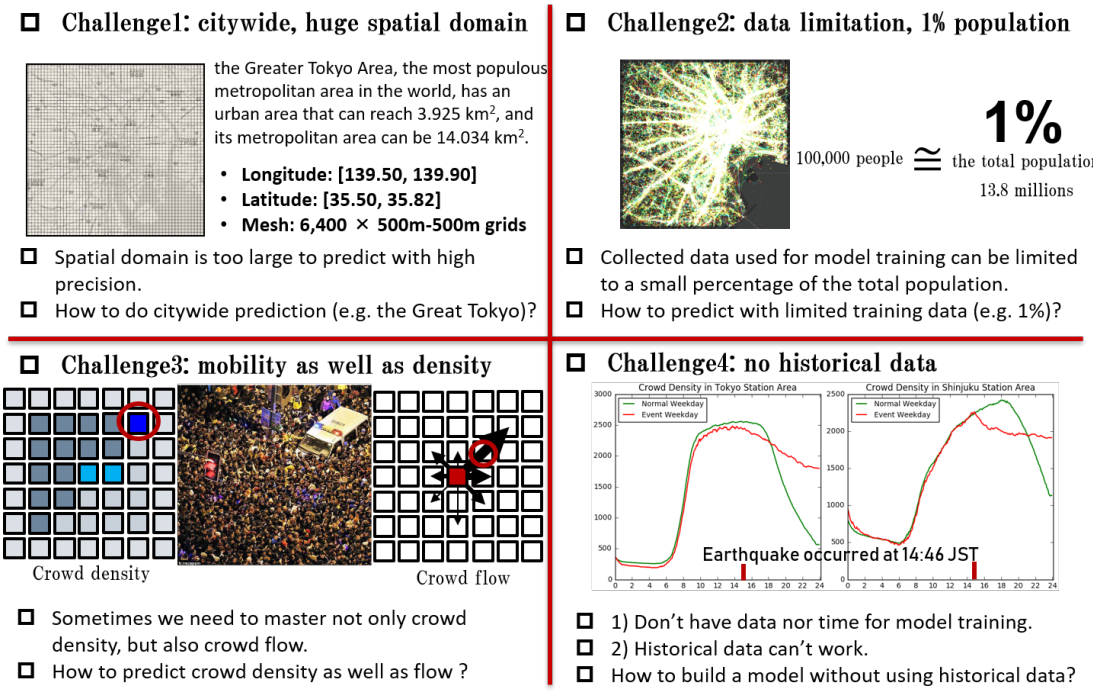


FIGURE 1.6: Challenges on modeling and analyzing urban human mobility.

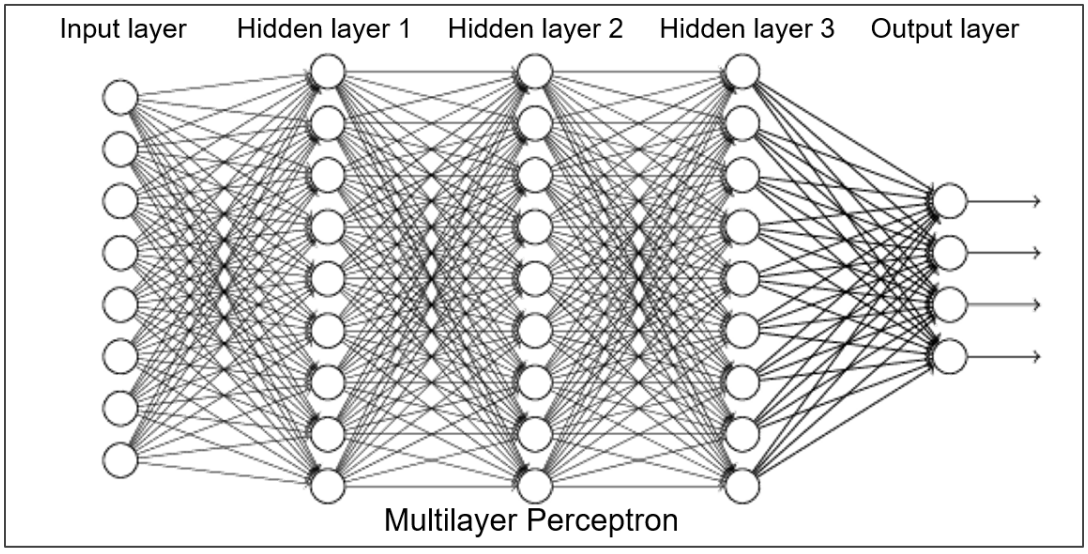


FIGURE 1.7: Multiple layers of neural networks.

1.4 Deep Learning

Emerging deep learning technologies have taken over the world since 2016 (The year of AI). It belongs to Neural Networks family, a branch of Machine Learning. As shown in Fig.1.7, it attempts to learn multiple levels of representation of increasing complexity/abstraction. "Deep learning is part of state-of-the-art systems in various disciplines, particularly computer vision and automatic speech recognition (ASR). Results on commonly used evaluation sets such as TIMIT (ASR) and MNIST (image classification), as well as a range of large-vocabulary speech recognition tasks have steadily improved. Convolutional neural networks (CNNs) were superseded for ASR by CTC for LSTM. The impact of deep learning in industry began in the early 2000s, when CNNs already processed an estimated 10% to 20% of all the checks written in the US, according to Yann LeCun. Industrial applications of deep learning to large-scale speech recognition started around 2010. The 2009 NIPS Workshop on Deep Learning for Speech Recognition[68] was motivated by the limitations of deep generative models of speech, and the possibility that given more capable hardware and large-scale data sets that deep neural nets (DNN) might become practical. It was believed that pre-training DNNs using generative models of deep belief nets (DBN) would overcome the main difficulties of neural nets.[69] However, it was discovered that replacing pre-training with large amounts of training data for straightforward backpropagation when using DNNs with large, context-dependent output layers produced error rates dramatically lower than then-state-of-the-art Gaussian mixture model (GMM)/Hidden Markov Model (HMM)

and also than more-advanced generative model-based systems. The nature of the recognition errors produced by the two types of systems was characteristically different, offering technical insights into how to integrate deep learning into the existing highly efficient, run-time speech decoding system deployed by all major speech recognition systems. Analysis around 2009-2010, contrasted the GMM (and other generative speech models) vs. DNN models, stimulated early industrial investment in deep learning for speech recognition, eventually leading to pervasive and dominant use in that industry. That analysis was done with comparable performance (less than 1.5% in error rate) between discriminative DNNs and generative models. In 2010, researchers extended deep learning from TIMIT to large vocabulary speech recognition, by adopting large output layers of the DNN based on context-dependent HMM states constructed by decision trees. Advances in hardware enabled the renewed interest. In 2009, Nvidia was involved in what was called the big bang of deep learning, as deep-learning neural networks were trained with Nvidia graphics processing units (GPUs). That year, Google Brain used Nvidia GPUs to create capable DNNs. While there, Ng determined that GPUs could increase the speed of deep-learning systems by about 100 times. In particular, GPUs are well-suited for the matrix/vector math involved in machine learning. GPUs speed up training algorithms by orders of magnitude, reducing running times from weeks to days. Specialized hardware and algorithm optimizations can be used for efficient processing."[1]

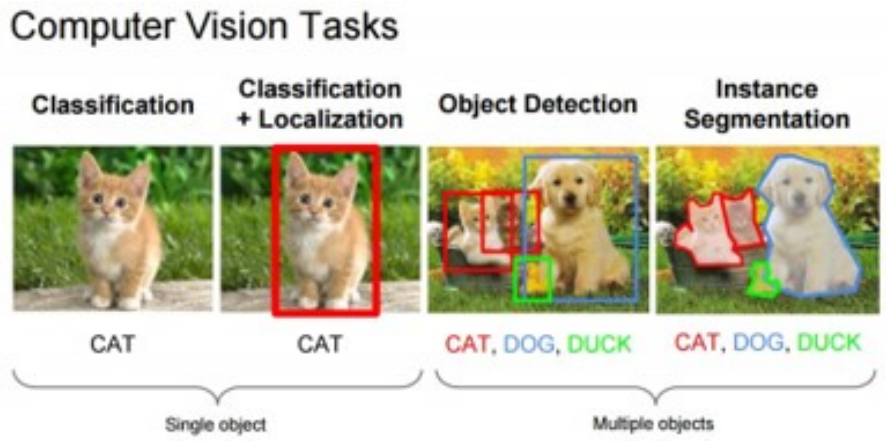


FIGURE 1.8: Deep learning on computer vision.

The success of deep learning in the fields of computer vision (shown as Fig.1.8) and natural language processing (shown as Fig.1.8) motivates us to consider deep learning techniques as highly potential solutions to our problems, because it has the following

[1] https://en.wikipedia.org/wiki/Deep_learning

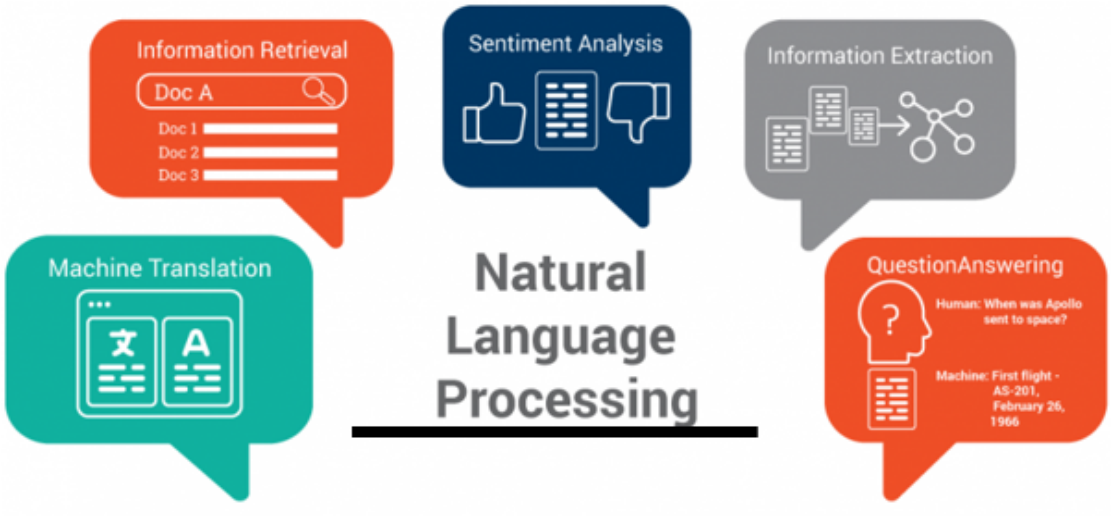


FIGURE 1.9: Deep learning on natural language processing.

advantages: (1) It can handle real big data; (2) It can model highly complex spatiotemporal system; (3) It can deal with multimodal distribution; (4) It can fuse multiple heterogeneous data. Four deep-learning-based solutions have been proposed to address those four corresponding challenges as shown in Fig.1.10.

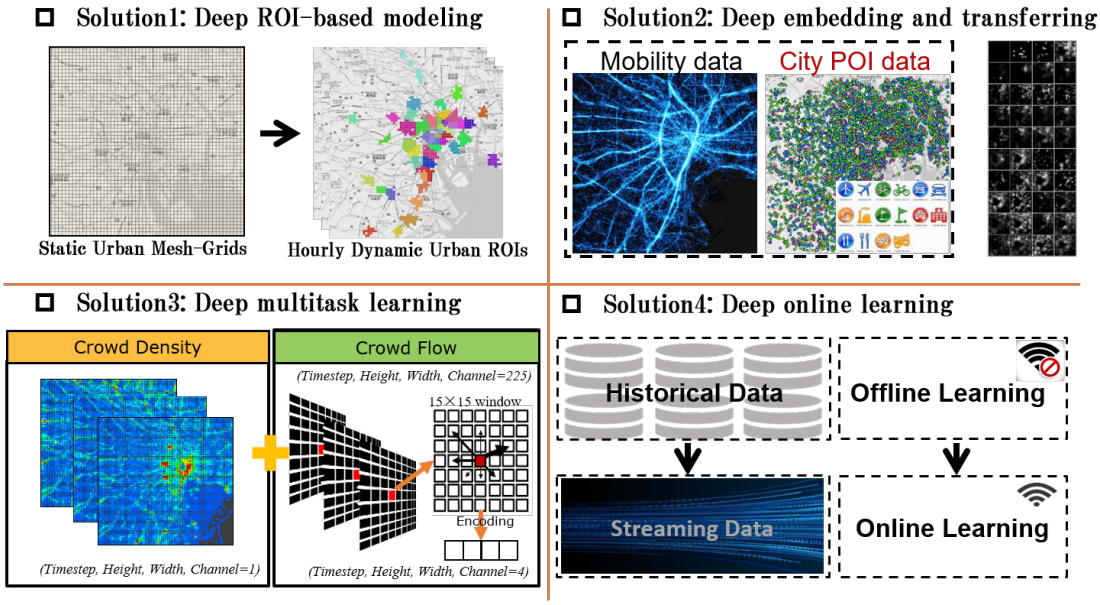


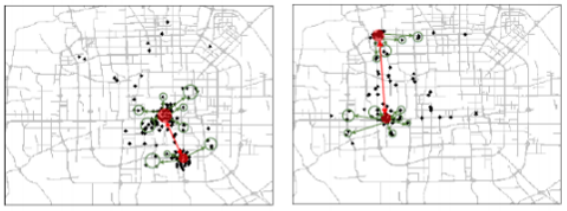
FIGURE 1.10: Deep learning based solutions.

Chapter 2

Related Works

2.1 Classical Methods on Human Mobility

Recently, various studies were conducted on human mobility data (e.g. mobile phone GPS log data, taxi GPS data, and location-based services data). These are summarized as urban computing problems in [5].



(a) check-ins of user A (b) check-ins of user B

Figure 2: Sparsity of major and minor hubs

Table 1: Comparison of location prediction methods
CI: check-in, SMP: spatial mobility pattern, TC: text content
IT: individual temporal patterns, SR: social relationship
CF: collaborative filtering, HT: heterogeneous mobility datasets

methods	target			feature					
	CI	GPS	Wifi	SMP	TC	IT	SR	CF	HT
PSMM [6]	√			√		√	√		
W^4 [40]	√			√	√	√			
M5Tree[25]	√					√	√		
CEPR [19]	√			√		√		√	
SHM [12]	√					√	√		
gSCorr [11]	√					√	√		
DBN [26]	√					√	√		
NextPlace [27]		√	√			√			
WhereNext [23]		√				√			
Markov [1]		√							
RCH(Our Model)	√			√		√		√	√

FIGURE 2.1: Comparison of location prediction methods [1].

Trajectory-pattern based methodologies have been proposed to predict future movement of individual person [6–8]. An approach based on nonlinear time series analysis of the arrival and residence times of users has been proposed, which focused on predicting most important places of each user [9]. J. Zheng [10] proposed an unsupervised learning algorithm for location prediction. Some collaborative approaches have been proposed to take social relationships of users into account for location prediction and

recommendation, but they utilized big check-in data from location-based network services [1, 11, 12]. Comparison of location prediction methods have been summarized in Fig.2.1 by [1].

CityMomentum [2] is a state-of-the-art prediction approach that can predict short-term human mobility at a citywide level. The performance of CityMomentum is not good enough since it is constructed based on a simple first-order Markov model.

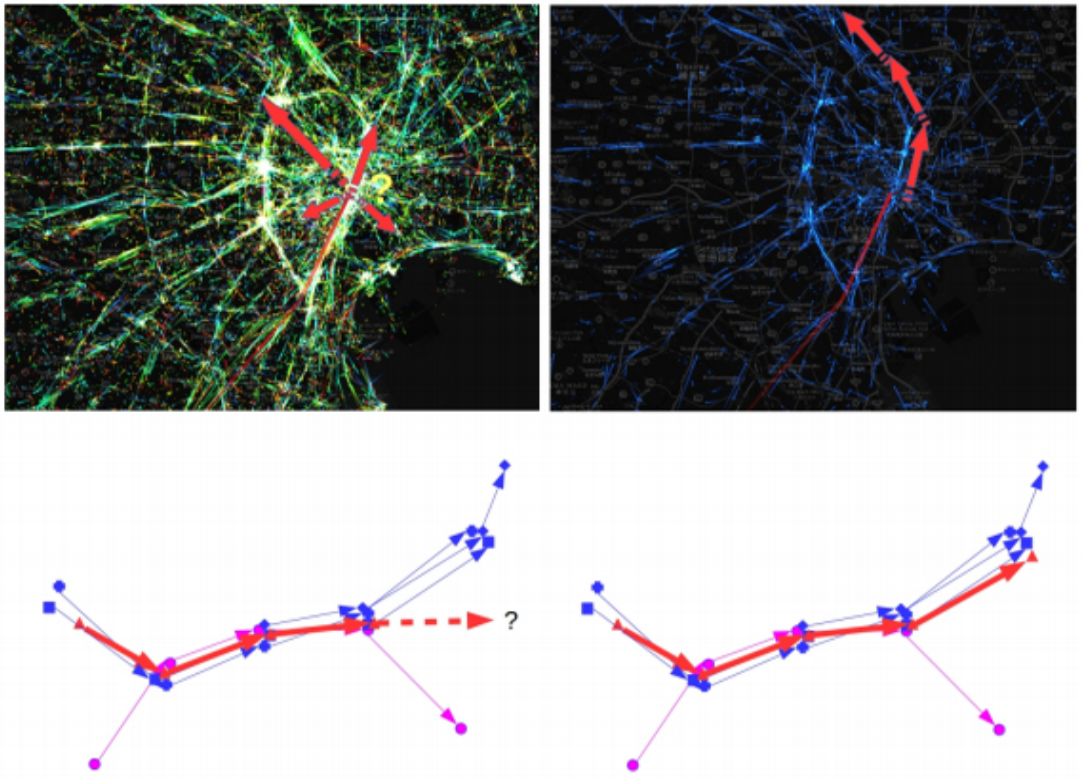


FIGURE 2.2: The main idea about CityMomentum[2].

CityProphet[13, 14] and [15] utilize query data of Smartphone APP to forecast only crowd density other than crowd flow. [16, 17] conduct transition estimation from aggregated population data, and [18] estimates the transition populations using inflow and outflow defined by [19]. Some researchers tried to detect the urban anomalies from mobility data based on statistical methodologies [20, 21]. Modeling human mobility for very large populations [2, 22] and simulating human emergency mobility following disasters [23] are similar problems to ours, however, their models are built based on millions of individuals' mobility.

Understanding the basic life patterns of the flow of people [24] and recommending location-based services [25, 26] are studies that utilize the tensor factorization approach

to decomposing urban human mobility. [12] conducted next place prediction in location-based services based on user features. Using population-scale data, [27] detected popular temporal modes and [28] modeled urban population of multiple cellphone networks. Traffic flow, which can be seen as a special human mobility constraint on road networks, has also been studied in the form of traffic flow prediction [29] and traffic congestion monitoring [30], but they also start with some kind of individual model for each road segment. Moreover, C. Song [31] explored the upper bound of the predictability of human mobility. J. Zheng [10] proposed an unsupervised learning algorithm for location prediction. A more advanced trajectory calibrating algorithm was proposed in [32]. Many factors (e.g. sampling rate, data type) are demonstrated to affect the modeling performance of human mobility [22]. A transfer learning framework was designed to transfer knowledge of the hourly air quality between cities [33]. [34] proposed a unified approach to predicting original taxi demands based on large-scale online platforms. Construction of urban movement knowledge graph from GPS trajectory has been proposed for understanding people lifestyles[35].

Furthermore, modeling human mobility for very large populations [22, 36] and simulating human emergency mobility following disasters [37, 38] are other topics that are close to ours. However, all of these approaches had different problem definitions and modeling methods. For example, the approaches required disaster information such as intensity of earthquake and damage level as additional input data in [37, 38]. In addition, they did not use the power of deep-learning technologies.

2.2 Deep Learning on Human Mobility

Social-LSTM [39] is an advanced multi-agent model, which builds a separate LSTM network for each person. ST-RNN [40] utilized RNN to model spatio-temporal transitions. SERM [41] is recurrent model designed for semantic trajectory. A RNN architecture similar with our word-like embedding model was proposed in [42] for destination prediction task. These models focused on individual mobility and were validated with small-scale trajectory dataset, which are difficult to be applied to our urban mobility modeling task. [43] proposed online deep ensemble learning for predicting citywide human mobility.

Forecasting the citywide crowd flow [3, 19, 44] are related works, which build a time-series prediction model based on inflow and outflow, which can only indicate how many

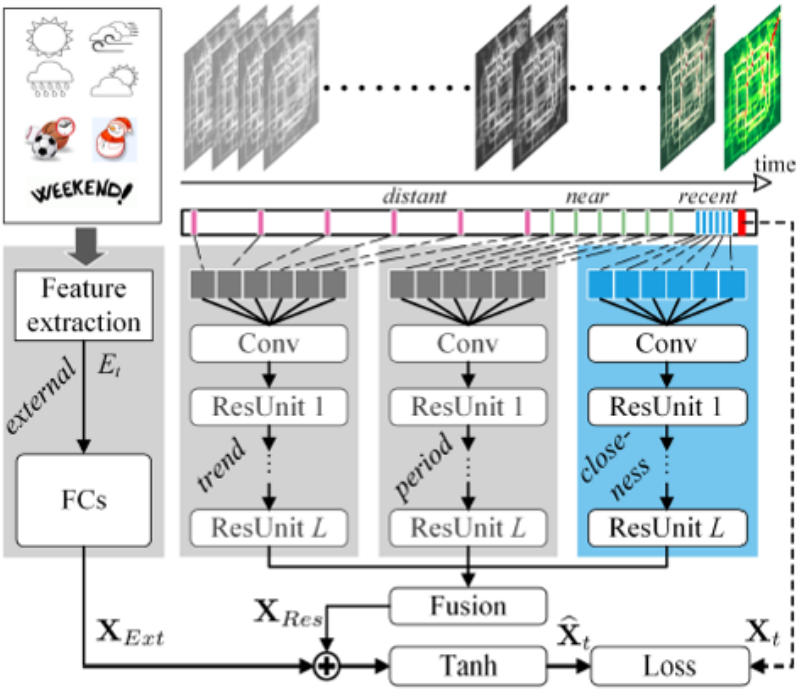


Figure 3: ST-ResNet Architecture. Conv: Convolution; ResUnit: Residual Unit; FC: Fully-connected.

FIGURE 2.3: The main idea about ST-ResNet[3].

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. The main idea about ST-ResNet is shown in Fig.2.3.

Moreover, some studies also applied deep learning to predict the traffic flow, traffic speed, congestion, transportation mode as well as human mobility, and taxi demand[29, 30, 45–49]. [4] proposed Deep Multi-View Spatial-Temporal Network for taxi demand prediction. Bidirectional RNN is reported to be more effective for taxi destination prediction task [50]. DeepMove[51] is a framework for predicting human mobility with attentional recurrent networks, which focused on the problem of mobility prediction from the sparse and lengthy trajectories. [52] developed deep generative models of urban mobility. [53] is a deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data.

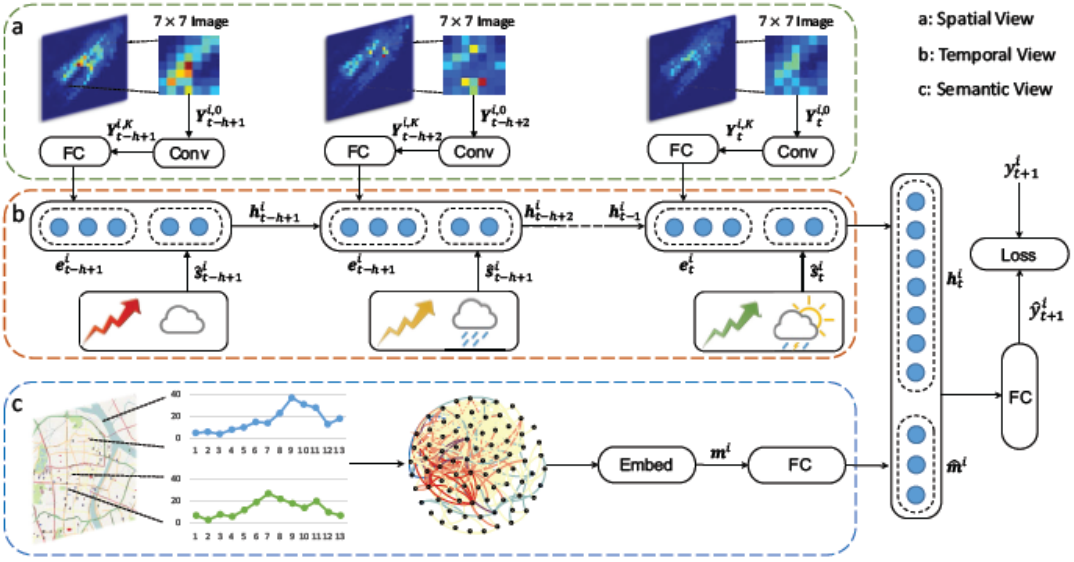


FIGURE 2.4: The main idea about DMVST-Net[4].

2.3 Other Methods on Human Mobility

Traditional multi-agent based human mobility simulation (e.g. Matsim and Aimsun as shown in Fig.2.5) needs: 1) A large amount work of parameterizing the city; 2) A large number of assumptions on human behavior; 3) A rich expertise in citywide human mobility to make it work properly. [54] did agent-based modeling of taxi behavior simulation with probe vehicle data.

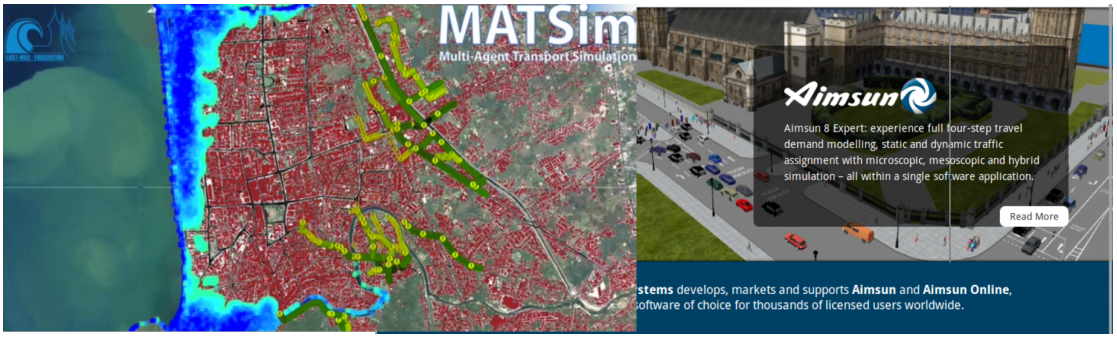


FIGURE 2.5: MATSim (the Multi-Agent Transport Simulation Toolkit) is an open source software development project developing agent-based software modules intended for use with transportation planning models. Aimsun Live is a simulation-based traffic forecasting solution, developed and marketed by Aimsun.

Research on processing classic trajectory data has drawn a lot of attention, in which the following three topics are most representative.

(1)Clustering: TRACLUS (Trajectory Clustering)[55] is a method for clustering trajectories that first splits inputed trajectories into small segments, and then conducts clustering on those segments.

Object	Semantic Trajectory
o_1	$< p_3(g), p_1(o), p_7(g), p_9(r) >$
o_2	$< p_5(o), p_7(g), p_2(o), p_7(g), p_{10}(r) >$
o_3	$< p_3(g), p_6(r) >$
o_4	$< p_2(o), p_1(o), p_6(r), p_8(g), p_{11}(r) >$
o_5	$< p_{12}(g), p_8(g), p_{11}(r), p_4(o) >$

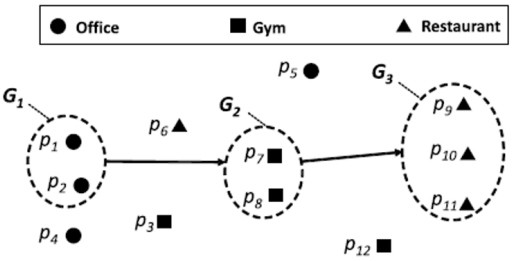


FIGURE 2.6: Semantic trajectory database and fine-grained sequential pattern [61].

(2)Pattern Discovery: T-Pattern (Trajectory Pattern)[56] is a method for discovering trajectory patterns, which are represented by a sequence like $RoI_0 \rightarrow t_1\ RoI_1 \rightarrow t_2$ ... $\rightarrow t_n RoI_n$, where RoI_i(Region of Interest) represents the discovered region passed through by a plenty of trajectories and t_i stands for the transition time between two RoIs. In addition to T-Pattern, flock[57], swarm[58], convoy[59], gathering[60] and splitter[61] are also proposed as specific trajectory patterns.

(3)Optimal Route Discovery: MPR(Most Popular Route)[62] takes start point and end point as inputs and returns the most popular route between the two points from trajectories. Apart from the start point and end point, TPMFP(Time Period-based Most Frequent Path)[63] also takes a time period as input and returns the most frequent path between the two points during that time period.

Chapter 3

Problem and Overview

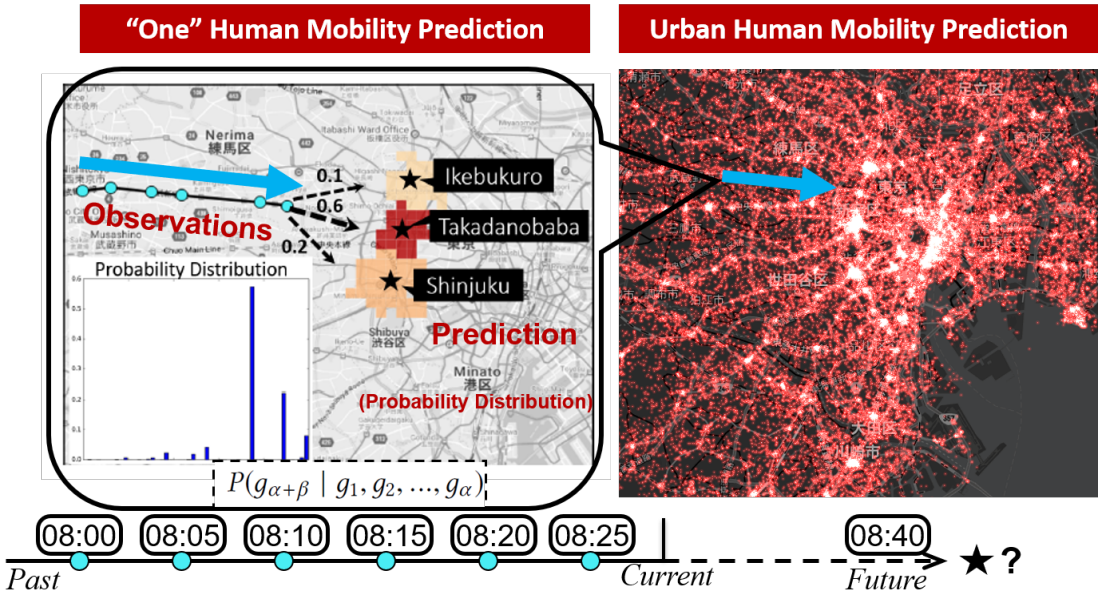


FIGURE 3.1: General objective of this study.

3.1 General Objective

The general objective of this study is to predict human mobility, namely future distribution of millions' individuals for a large urban area with high precision. Like the weather forecast, if it is possible to predict population distribution in real time, the result can be shared with various people and agencies for various kinds of disaster/event response. Based on the observed mobility data from different people, we can give a probability distribution as the prediction result for next step. For example, for a group

of commuting people lets say 1000 people on the Seibu-Shinjuku line observed from 08:00 O'clock to 08:30, we could precisely predict around 60% of people will go to Takadanobaba station, 20% will go to Shinjuku station, 10% will go to Ikebukuro station. In this way, given all peoples' observed mobility, we can predict human mobility for the entire urban area. It should be noted that for one person's mobility prediction, we are concerned only about whether the model can precisely predict the next location with the highest probability. However, a large crowd of people can share the same observed trajectories (e.g. commuters taking the same train) but they may go to different places after some time. Thus, for citywide human mobility prediction, our model should precisely predict the overall probability distribution of the next possible destinations. With such model being deployed as an online service, we can precisely predict and simulate how many person will enter a certain region in real time, which can play an important role in controlling the crowd density for a city especially when some irregular events happen. The trained model can generate or predict multiple steps of human mobility in an autoregressive manner. Multiple steps of mobility can be generated one step by one step according to the probability distribution in a similar way to a text generator. For example, given the first word "how", the second word can be generated as "are", then the third can be "you". If the second was generated as "old", then the next two words could be "are you" with higher probability. Moreover, if one step corresponds to 5 minutes time interval, generating next six steps of human mobility means that we can get a next-30-minutes mobility prediction. For instance, our model can take all of the 6-step observations from 07:35~08:00 as inputs and report the prediction result for 08:05~08:30 at 08:00. This can help us understand how the crowd dynamics are evolving step by step under a crowd management application scenario.

3.2 How to Predict Citywide Domain?

Rapidly developing location acquisition technologies have provided us with big GPS trajectory data, which offers a new means of understanding people's daily behaviors as well as urban dynamics. With such data, predicting human mobility at the city level will be of great significance for transportation scheduling, urban regulation, and emergency management. In particular, most urban human behaviors are related to a small number of important regions, referred to as Regions-of-Interest (ROIs). Therefore, in this study, a deep ROI-based modeling approach is proposed for effectively predicting urban human mobility. Urban ROIs are first discovered from historical trajectory data, and urban

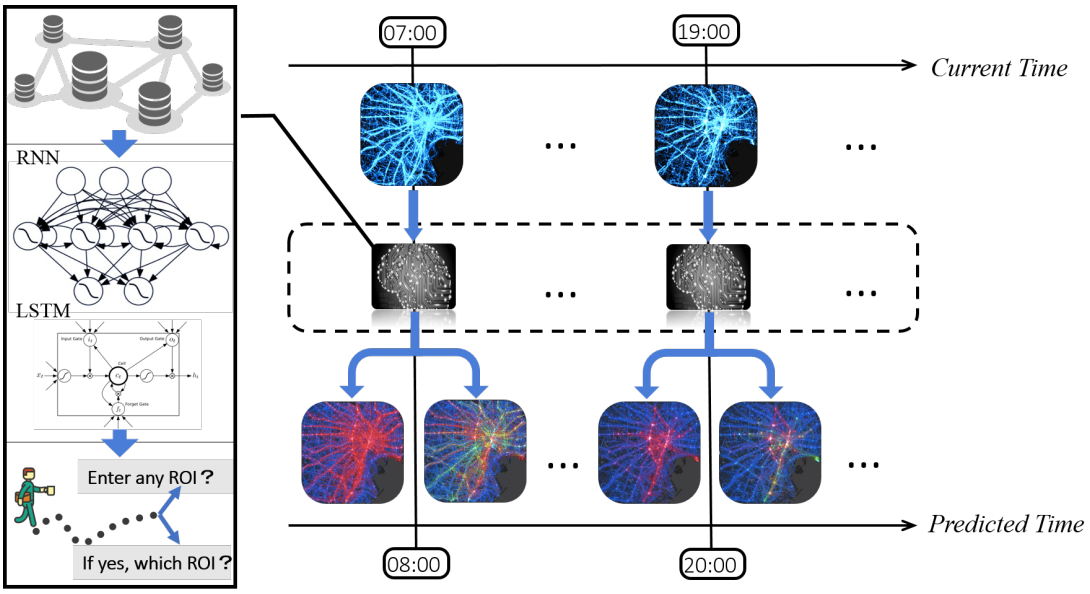


FIGURE 3.2: Can we design an ROI-based approach for predicting short-term human mobility at the citywide level with high precision? Big human mobility data and deep-learning technologies provide us with the opportunity to implement this.

human mobility is designated using two types of ROI labels (ISROI and WHICHROI). Then, urban mobility prediction is modeled as a sequence classification problem for each type of label. Finally, a deep-learning architecture built with recurrent neural networks is designed as an effective sequence classifier. Experimental results demonstrate that the superior performance of our proposed approach to the baseline models and several real-world practices show the applicability of our approach to real-world urban computing problems.

The complete framework is demonstrated in Fig. 3.2 and Fig.3.3.

The main contributions of this solution are as follows:

- To the best of our knowledge, this solution is the first attempt to utilize urban ROI to model human mobility at a citywide level.
- An effective mining algorithm is proposed to discover urban ROIs given historical raw trajectory data.
- The proposed approach constructs a deep-sequence learning model with RNN to effectively predict urban human mobility.
- Our approach can be easily applied to real-world simulations and has been verified as a highly applicable approach.

Input	Observed α steps of grid-mapped human trajectory: $x=(g_1, g_2, g_3, \ldots, g_\alpha)$
Intermediate	(1) (Grid-based modeling) Probability distribution over mesh-grids at next β-th timestep: $y=g_{\alpha+\beta}$ (2) (ROI-based modeling) Probability distribution over ROIs at next β-th timestep: $y=r_{\alpha+\beta}$
Output	Predicted position (centroid of mesh-grid/ROI) at next β-th timestep: $o=(\text{longitude}_{\alpha+\beta}, \text{latitude}_{\alpha+\beta})$
Training Mode	Offline training (Using historical trajectory data to train and validate the model)

FIGURE 3.3: The input and output for solution 1.

3.3 How to Predict with Limited Data?

Rapidly developing location acquisition technologies provide a powerful tool for understanding and predicting human mobility in cities, which is very significant for urban planning, traffic regulation, and emergency management. However, with the existing methodologies, it is still difficult to accurately predict millions of peoples' mobility in a large urban area such as Tokyo, Shanghai, and Hong Kong, especially when collected data used for model training are often limited to a small portion of the total population. Obviously, human activities in city are closely linked with point-of-interest (POI) information, which can reflect the semantic meaning of human mobility. This motivates us to fuse human mobility data and city POI data to improve the prediction performance with limited training data, but current fusion technologies can hardly handle these two heterogeneous data. Therefore, we propose a unique POI-embedding mechanism, that aggregates the regional POIs by categories to generate an artificial POI-image for each urban grid and enriches each trajectory snippet to a four-dimensional tensor in an analogous manner to a short video. Then we design a deep learning architecture combining CNN with LSTM to simultaneously capture both the spatiotemporal and geographical information from the enriched trajectories. Furthermore, transfer learning is employed to transfer mobility knowledge from one city to another, so that we can fully utilize other cities' data to train a stronger model for the target city with only limited data available. Finally, we achieve satisfactory performance of human mobility prediction

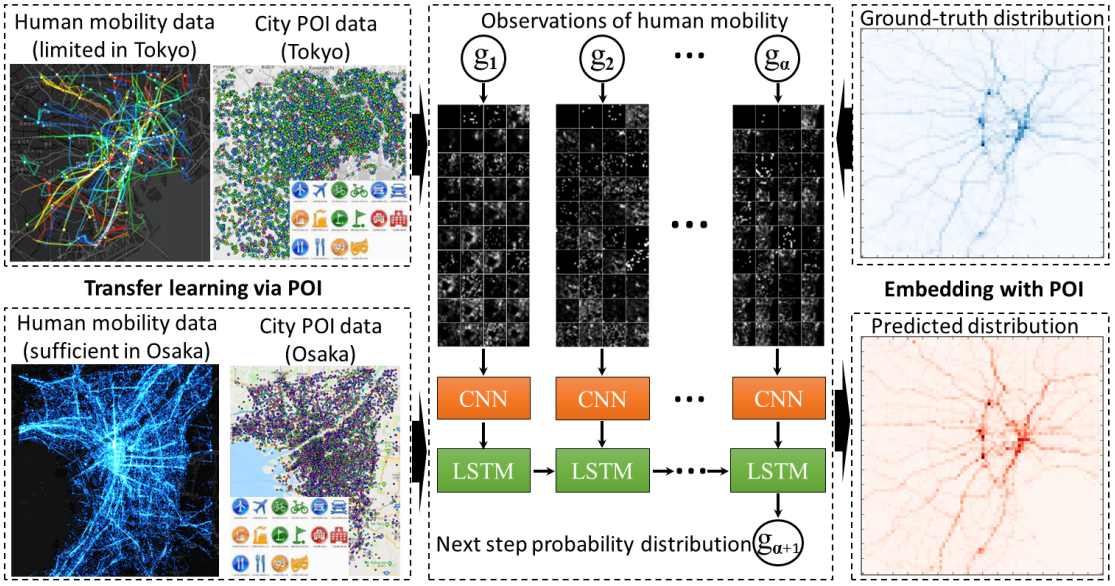


FIGURE 3.4: Can we design an effective approach to build one urban model for predicting human mobility (future distribution of individuals) at a citywide level with limited data? Fusing heterogeneous data (Human mobility data and city POI data) with deep learning technologies may allow us to address this challenge.

Input	Observed α steps of grid-mapped human trajectory: $x=(g_1, g_2, g_3, \ldots, g_\alpha)$
Output	Predicted probability distribution over mesh-grids at next timestep: $y=g_{\alpha+1}$
Training Mode	Offline training (Using historical trajectory data to train and validate the model)

FIGURE 3.5: The input and output for solution 2.

at the citywide level using a limited amount of trajectories as training data, which has been validated over six urban areas of different types and scales.

The complete framework is demonstrated in Fig. 3.4 and Fig.3.5.

The main contributions of this solution can be summarized as follows:

- We constructed a standard deep sequence learning model for accurately predicting a probability distribution of human mobility at the citywide level.
- We proposed a novel sequential embedding method called image-like embedding that uses city POI data to enrich the original human mobility data with geographical features, where we applied CNNs to the standard model to obtain more effective representations.

- Transfer learning was employed to work together with image-like embedding mechanism. Through this, we can transfer mobility knowledge from source city to target city via POI information, if the source city have relatively sufficient mobility data and the target city only have limited data.
- We evaluated our approach based on multiple urban areas using different amounts of training data and demonstrated the advantages of our method compared with other baseline approaches.

3.4 How to Predict Mobility as Well as Density?

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 DeepUrbanVideo 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.

The complete framework is demonstrated in Fig. 3.6 and Fig.3.7.

This solution 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.

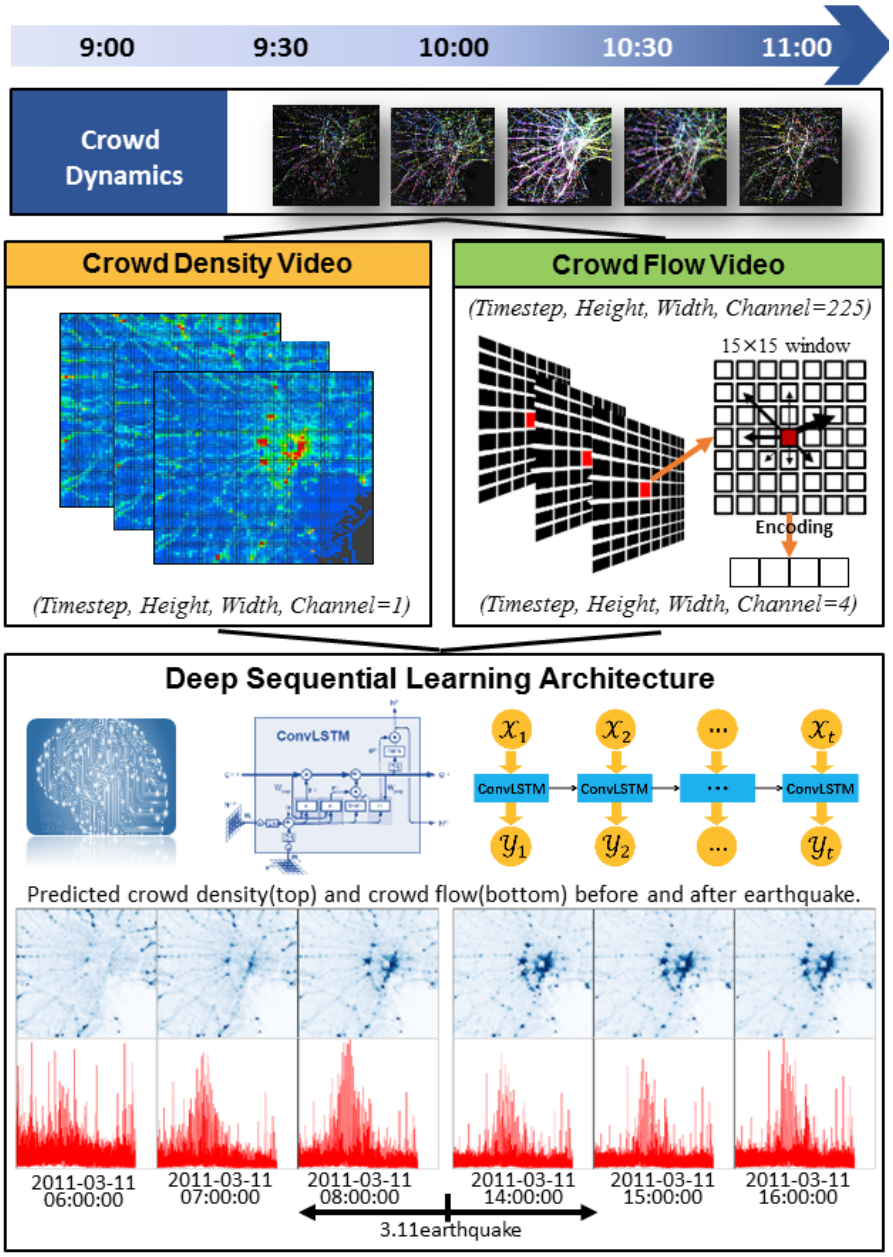


FIGURE 3.6: Can we design an effective real-world system for predicting citywide crowd dynamics at big events? Real-time human mobility data as well as deep learning technologies allow us address this high-social-impact problem.

- 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.

- Using the predicted crowd density and flow video, we further build a series of dynamic crowd mobility graph to help conduct probabilistic reasoning of crowd movements during big events.

Input	Observed α steps of crowd density and crowd flow: $x_d=(d_1, d_2, d_3, \ldots, d_\alpha)$, $x_f=(f_1, f_2, f_3, \ldots, f_\alpha)$
Output	Predicted β steps of crowd density and crowd flow: $y_d=(d_{\alpha+1}, d_{\alpha+2}, \ldots, d_{\alpha+\beta})$, $y_f=(f_{\alpha+1}, f_{\alpha+2}, \ldots, f_{\alpha+\beta})$
Training Mode	Offline training (Using historical trajectory data to train and validate the model)

FIGURE 3.7: The input and output for solution 3.

- 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.

3.5 How to Predict without Historical Data?

Big human mobility data are being continuously generated through a variety of sources, some of which can be treated and used as streaming data for understanding and predicting urban dynamics. With such streaming mobility data, the online prediction of short-term human mobility at the city level can be of great significance for transportation scheduling, urban regulation, and emergency management. In particular, when big rare events or disasters happen, such as large earthquakes or severe traffic accidents, people change their behaviors from their routine activities. This means people's movements will almost be uncorrelated with their past movements. Therefore, in this study, we build an online system called DeepUrbanMomentum to conduct the next short-term mobility predictions by using (the limited steps of) currently observed human mobility data. A deep-learning architecture built with recurrent neural networks is designed to effectively model these highly complex sequential data for a huge urban area. Experimental results demonstrate the superior performance of our proposed model as compared to the existing approaches. Lastly, we apply our system to a real emergency scenario and demonstrate that our system is applicable in the real world. The complete framework is demonstrated in Fig. 3.8 and Fig.3.9.

It has the following key characteristics:

- It is built and tested based on a big human mobility data source, which stores the GPS records of 1.6 million users over three years.

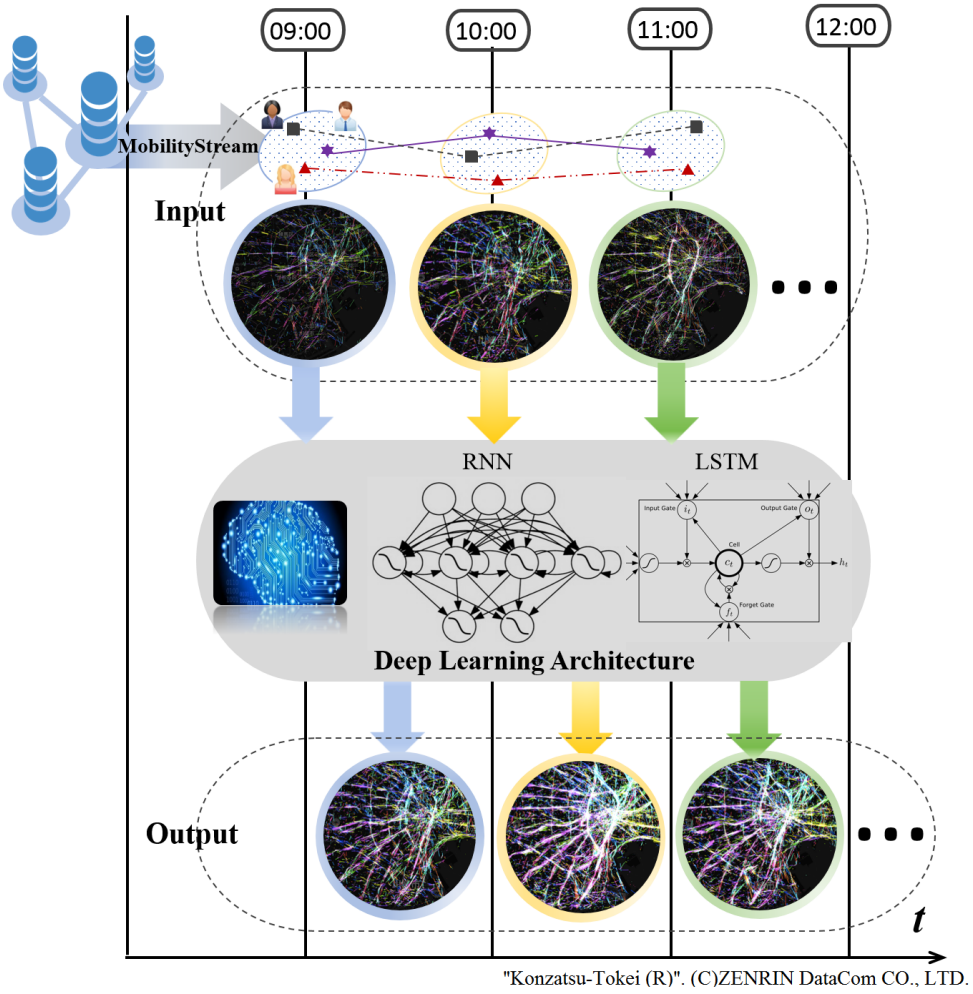


FIGURE 3.8: Can we develop an online intelligent system for short-term human mobility prediction with high precision by using recent momentary mobility at a citywide level? Big human mobility data and deep-learning technologies provide us with the opportunity to implement this system.

Input	Observed first-α steps of human trajectory: x=(lon_1, lat_1), (lon_2, lat_2), (lon_3, lat_3), …, (lon_α, lat_α)
Output	Predicted next-β steps of human trajectory: y=($lon_{\alpha+1}$, $lat_{\alpha+1}$), ($lon_{\alpha+2}$, $lat_{\alpha+2}$), …, ($lon_{\alpha+\beta}$, $lat_{\alpha+\beta}$)
Training Mode	Online training (Using one-hour streaming trajectory data to train and validate the model)

FIGURE 3.9: The input and output for solution 4.

- It is built as an online prediction system driven by mobility stream and deep-learning technologies.

- It constructs a deep-sequence learning model with RNN for effective multi-step predictions.
- It is applied to real-world scenarios and verified as a highly deployable prototype system.

Chapter 4

Citywide Domain: Deep ROI-Based Modeling

4.1 Introduction

A large urban area such as Beijing, Shanghai, or Tokyo usually has multiple Central Business Districts (CBDs), and people are likely to gather at those CBDs for working, shopping, and entertainment. Apart from CBDs for daily human behaviors, some stadiums and tourist spots can also be hot urban regions for non-daily human behaviors. For example, Tokyo Disneyland, one of the most famous theme parks, far away from the Tokyo Station, can attract over 100,000 visitors a day. In addition, people visit Beijing Workers' Stadium to see a sports game or a concert. Essentially, most urban human behaviors (daily and non-daily) and urban phenomena can be related to these hot regions, officially mentioned as Regions-of-Interest (ROIs).

Hot regions mean high crowd density, which naturally makes these areas at high risk for various accidents. Recall the tragedy on New Year's Eve in Shanghai, when around 300,000 people gathered to celebrate the arrival of 2015 near Chen Yi Square on the Bund, which is the most representative tourist spot in Shanghai. However, the large crowd was not well controlled, and a stampede occurred where 36 people died and 47 were injured in the tragedy. For governments and public infrastructure operators such as metro companies, an ROI always should always have a higher priority over other regions, and they should always allocate more resources to some ROIs to prevent emergency situations from occurring. If not, a tragedy like the Shanghai stampede could

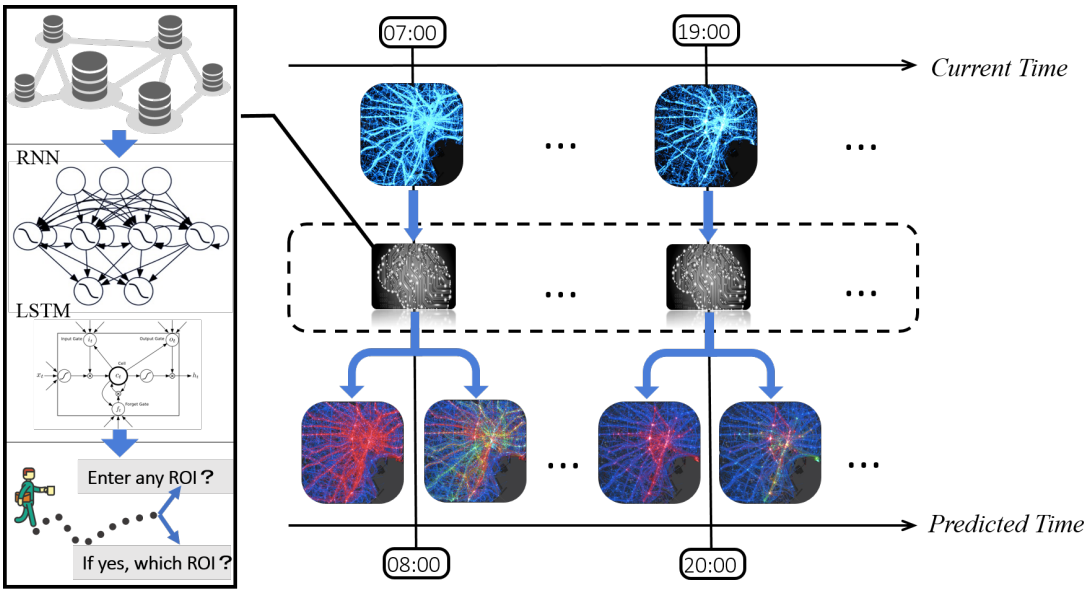


FIGURE 4.1: Can we design an ROI-based approach for predicting short-term human mobility at the citywide level with high precision? Big human mobility data and deep-learning technologies provide us with the opportunity to implement this.

happen again. Therefore, paying close attention to urban ROIs can be of great significance for transportation scheduling, urban regulation, and emergency management.

On the other hand, rapidly developing location acquisition technologies have provided us with big data on human mobility. Obviously, this offers a new means of understanding people's daily behaviors as well as urban dynamics, which has been seen as one of the most promising research topics in both academia and industry. However, modeling big historical trajectory data has never been an easy task for the following reasons: (1) The spatial domain is very large for a big urban area, and is difficult to predict with high precision. For example, the Greater Tokyo Area, the most populous metropolitan area in the world, has an urban area that can reach 3.925 km^2, and its metropolitan area can be 14.034 km^2. [1] (2) In order to build a human mobility prediction model or mine some mobility knowledge, we usually have to collect trajectory data over a long period of time. The large temporal scale can be another big challenge for us. (3) Collected raw trajectory data can be noisy and sparse owing to the limitation of location acquisition technology itself or user privacy problems. Handling the raw trajectory data is a significant issue that we have to face.

Motivated by all the above, in this article, we propose an effective ROI-based approach for understanding and predicting human mobility with big raw and historical trajectory data. Specifically, given a few steps of observed mobility from one person, predicting

[1] https://en.wikipedia.org/wiki/Greater_Tokyo_Area

where this person will go next is the core problem we have been working on in the field of urban computing. The key idea is to convert this problem into the following two problems based on ROI: (1) Next, will this person enter any ROI? (2) If yes, which ROI will this person enter next? Intuitively, our new modeling approach will first determine if the input trajectory is so important that it can have a key influence on the urban area. Then, it will predict which urban area the trajectory will influence. It can be very effective, straightforward, and of high usability for human mobility prediction at a citywide level due to the significance of ROI as mentioned above. In order to deliver this idea, the complete framework demonstrated in Fig. 4.1 is designed as follows:

Framework Overview. Given multiple days of trajectory data (e.g., one month), our goal is to build a 24-h mobility prediction system on these historical data, which iteratively runs in an online updating mode that takes observed human mobility within the current hour as input and predicts how the mobility will be in the next hour. In order to do this, first, dynamic ROIs for each hour will be discovered through divide-and-merge mining from raw trajectory database. This divides the entire trajectory dataset by day, finds the ROIs corresponding to a specified hour (e.g., 07:00~07:59) from each day, and finally merges the discovered ROIs of each day by the hour. Through this, 24 sets of ROIs for 24 h are prepared. Second, all subtrajectories in a continuous 2 h are retrieved. The data for the first hour are utilized as input features, and the second set of hourly data are used to generate the two types of ROI-related target labels mentioned above: "Enter-ROI-Or-Not" and "Which-ROI." Last, 24 mobility prediction models are built using deep Recurrent Neural Networks (RNNs), each of which consists of a typical binary classification model and a multiclass classification model. All of these processing steps form a complete and effective mining-learning-predicting framework for modeling urban human mobility on raw and historical trajectory data.

The main contributions of this paper are as follows:

- To the best of our knowledge, this paper is the first attempt to utilize urban ROI to model human mobility at a citywide level.
- An effective mining algorithm is proposed to discover urban ROIs given historical raw trajectory data.
- The proposed approach constructs a deep-sequence learning model with RNN to effectively predict urban human mobility.

- Our approach can be easily applied to real-world simulations and has been verified as a highly applicable approach.

The remainder of this paper is organized as follows: Section 4.2 gives an overview of our data source. Section 4.3 gives the problem definition for the entire framework. Section 4.4 explains the details of mining algorithms for dynamic urban ROIs. Section 4.5 explains the modeling details and deep-learning architectures. Section 4.6 shows the experimental details and a performance evaluation. Section 4.7 describes the simulation results in a real-world scenario. Section 4.8 introduces studies related to our research. Section 4.9 contains summaries, the limitations of our approach, and our future work.

4.2 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). It contains approximately 30 billion GPS records, and the total size of the data is more than 1.5 terabytes. Each record contains user ID, latitude, longitude, altitude, timestamp and positioning accuracy level (there are three levels due to different satellite's signal strength, correspondingly the positioning error would be within 100m, 200m or 300m). However, the record interval exceeds 5 minutes occasionally due to loss of signal or battery power. Besides, the positioning function would be suspended when no motion is detected, in this case no records will be uploaded. After filtering the records outside of Tokyo and Osaka, the two prefectures have about 1.2 billion and 650 million records for 680365 and 394980 unique users, respectively. In this research, we randomly take one month of data of Tokyo as the simulation dataset, and get around 120000 distinct user IDs for each day (from 00:00 to 23:59), which is approximately equal to 1% of the total population of the Greater Tokyo Area.

To better simulate a real-time situation for our online system, this dataset is stored on a Hadoop cluster containing 32 cores, 32 GB of memory, and 16 TB of storage, which can run 28 tasks simultaneously. Furthermore, we use Hive on top of Hadoop to make the entire system support SQL-like spatial queries. Therefore, GPS trajectories of a specified city and day can be retrieved in a short response time, and our database can be regarded as a nearly real-time data source.

4.3 Problem Definition

Definition 4.1 (Raw Human Trajectory)**.** A raw human trajectory collected from an individual person is essentially a sequence of 3-tuple: $(timestamp, latitude, longitude)$, which can indicate a person's location according to a captured timestamp. In our research, it will be denoted as a sequence of (d, t, l)-tuple by decomposing $timestamp$ to d, t and simplifying $(latitude, longitude)$ as l, where d is the date and t is the time for a 24-h clock. Furthermore, a historical raw trajectory database is denoted as $HTDB$.

Urban ROI discovery will be conducted using the raw trajectories with long-period records(i.e. one-month in our research), and more details will be described in Section 4.4. Note that the raw trajectory has a lot of temporal uncertainties because of different time intervals between two consecutive timestamps. In order to effectively model and predict citywide human movements, it motivates us to reduce temporal uncertainty by calibrating the raw trajectory to have equal time intervals Δt, which is defined as follows:

Definition 4.2 (Calibrated Human Trajectory)**.** A calibrated human trajectory from time t_1 to t_m is a sequence of timestamp-location pairs denoted as: (t_1, l_1), (t_2, l_2), ..., (t_m, l_m) that satisfies:

$$\forall i \in [1, m), |t_{i+1} - t_i| = \Delta t$$

The calibration operation is essentially to apply a linear interpolation to the raw trajectories to get the unified timestamps and the corresponding locations. A calibrated human trajectory database is further denoted as TDB. Then, urban human mobility prediction can be defined as follows:

Definition 4.3 (Observed Urban Human Mobility)**.** Given a TDB, an observation time t and an integer α, current urban mobility X_t is defined as follows:

$$X_t = \{traj \mid traj \in TDB \wedge \forall i, t - \alpha\Delta t < traj.t_i \leq t\}$$

which intuitively means α steps of urban human mobility are observed at time t.

Definition 4.4 (Next Urban Human Mobility)**.** Given a TDB, an observation time t, and an integer β, the next urban mobility $X_{t+\beta}$ is defined as follows:

$$X_{t+\beta} = \{traj \mid traj \in TDB \wedge traj.t = t + \beta\Delta t\}$$

which means the human's locations of the entire urban area at the moment of $t + \beta\Delta t$.

Definition 4.5 (Urban Human Mobility Prediction)**.** Given the observed urban human mobility X_t, urban human mobility prediction is to construct a model $P(\widehat{X}_{t+\beta} \mid X_t)$, in which $\widehat{X}_{t+\beta}$ is the predicted next urban human mobility.

Our goal is to build an online updating system that can report the prediction results every hour. In order to do this, Δt is set to 10 minutes, α is set to 6 and β is set to {1,3,6} which means our system will take the observed urban human mobility within one hour as inputs to report three prediction results corresponding to the moment of "Next-10-Minutes", "Next-30-Minutes" and "Next-60-Minutes" respectively.

Now, based on a historical trajectory dataset, the problem of urban human mobility prediction has been successful defined. In Section 4.4, we will show the details about discovering urban ROIs from historical raw trajectory data. Then based on the discovered urban ROIs, we will explain our deep ROI-based modeling approach for urban human mobility prediction in Section 4.5.

4.4 Urban ROI Discovering

An urban ROI is a polygon region frequently traversed by the trajectories in $HTDB$, which makes the region with maximal density within a specified time span. Urban ROI discovery is a nontrivial task for urban computing. For professionals working in government or a public service operator, they may be able to roughly point out the hot regions according to their daily experience. However, by exploiting big human mobility data,

dynamic urban ROIs with higher precision can be expected. Instead of utilizing only one day's trajectory data, multiple days' data are required for highly convincing results. Furthermore, daily human behaviors can be quite different from a long-time-period perspective, such as a weekday or weekend, this month and next month, or summer and winter. To satisfy the need for this kind of human behavior analysis, exploiting multiple days' trajectory data of a large urban area becomes indispensable. Moreover, owing to the limitation of location acquisition technology itself (e.g., loss of signal or low battery power) or user privacy problems, massive raw trajectory data can be very noisy and sparse. Handling the raw trajectory data at the citywide level is another significant issue we have to dealt with.

Here, in our approach, a divide-and-merge mining algorithm is designed to discover urban ROIs with multiple days' data. The entire mining process is demonstrated in Fig. 4.2, which can be summarized as follows:

- First, multiple days' raw trajectories are divided by the day, and pattern-based filtering is proposed an advanced preprocessing technique for ROI discovery.
- Second, after trajectory filtering, for each hour, MeanShift [64] is applied to find the point clusters with maximal density as preliminary ROIs.
- Third, to merge multiple days' preliminary ROIs, all centroids from the same time span of each day are once again clustered by MeanShift, and an urban ROI is approximately represented by several neighboring fine-grained grids.

In summary, our urban ROI discovering approach has the following characteristics that make it unique : (1) Be capable of handling large raw trajectory data. (2) Has high applicability for long-period historical trajectory data. (3) Has high compatibility with the prediction task for urban human mobility.

4.4.1 One-Day Preliminary ROI Discovering

As defined above, an ROI is a region with maximal population density in a time span, which is an hour in our research. Given raw trajectories, we can get all the points within each hour, and apply a clustering algorithm to these points to generate a preliminary ROI, which is essentially a point cluster identified by a centroid with maximal density. Two additional definitions are listed below:

FIGURE 4.2: Mining process for urban ROI discovery from multiple days' trajectories.

Algorithm 1: One-Day Preliminary ROI Discovering

Input: One-day raw trajectories DRT, a minimum support δ, a mesh M, a kernel bandwidth h.

Output: Hourly Preliminary ROI sets. `// Each hour has a set of ROIs and each ROI is a cluster of points.`

$PROI \leftarrow \emptyset$;
for *each* $HH \in [0, 1, ..., 23]$ **do**
  $\mathcal{T} \leftarrow$ TrajectoryFiltering(DRT, δ, M) ;
  `/* 4.2 Pattern-Based Trajectory Filtering. */`
  $\mathcal{P} \leftarrow$ GetHourPoints($\mathcal{T}$, HH);
  `/* Iterate over all trajectories in` $\mathcal{T}$ `to pick out the points within` $HH:00:00 \sim HH:59:59$. `*/`
  $PROI[HH] \leftarrow$ MeanShift($\mathcal{P}$, h);
return $PROI$

Definition 4.6 (Point)**.** Given a trajectory, a sequence of (d, t, l)-tuple, each (d, t, l)-tuple is called a *point*. When clustering is applied, a point will be treated as a normal 2D point with a $(longitude, latitude)$ coordinate. Points with the same l and different (d, t) will be treated as different points.

Definition 4.7 (Preliminary ROI)**.** Given the points within a specified hour (e.g., 08:00:00~08:59:59), an ROI is a cluster of points, and a centroid is a point located at the maxima of a density function.

Through our definition of an urban ROI, one can realize that MeanShift is a good solution to locate the maximal density as well as to get point clusters. MeanShift can automatically set the number of clusters instead of relying on only one parameter called the kernel bandwidth, which holds a physical meaning: the size of the region to search

through. In our approach, a flat kernel is used for kernel density estimation in MeanShift, which is defined as follows:

$$K(x) = \begin{cases} 1 & \text{if } \|x\|^2 \leq h \\ 0 & \text{if } \|x\|^2 > h \end{cases}$$

where $\|x\|^2$ indicates the two-dimensional Euclidean distance between points.

Note that in addition to the proposed definition of an ROI, when comparing it with other clustering algorithms such as K-Means, GMM, and DBSCAN, the most widely used ones for spatial clustering, we still find that MeanShift is the most appropriate method. K-Means and GMM require the user to set the number of clusters. For a large urban area like that in our case, the number will be very difficult to determine. If the number is set too small, clusters with too large an area will be generated, which will make the urban mobility prediction—which ROI this person will enter—become meaningless. DBSCAN, which requires a number threshold $minPts$ and a distance threshold ϵ, is also inclined to suffer from the same problems: (1) difficulty in determining the parameters and (2) ease of generating too-large clusters, especially for the second one, because an expanding strategy is used to form clusters in DBSCAN.

Since our historical trajectory data is raw data, there may exist many noise points as well as too many normal points. This problem can be easily noticed in the fist subfigure of Fig. 4.2, which is a snapshot of all raw trajectories of one day at "08:00:00." Spatial clustering on citywide points from raw trajectories cannot be applied without filtering the trajectory data first, especially for a noise-sensitive algorithm such as MeanShift. In addition, MeanShift is not highly scalable, as it requires many nearest-neighbor searches during execution.

Therefore, pattern-based trajectory filtering is proposed as an advanced preprocessing technique for our approach. More details about the trajectory filtering will be introduced in the next subsection. A one-day preliminary ROI discovering is summarized as Algorithm 1.

4.4.2 Pattern-Based Trajectory Filtering

Before conducting any spatial analysis, such as spatial clustering, a raw trajectory dataset should be well preprocessed to filter out unnecessary trajectories including noisy

Algorithm 2: Trajectory Filtering

Input: One-day raw trajectories DRT, a minimum support δ, a mesh M.
Output: Filtered one-day trajectories.

$\mathcal{T}, \mathcal{L}, \mathcal{S} \leftarrow \emptyset, \emptyset, \emptyset$; /* $\mathcal{T}$ is final result, $\mathcal{L}$ is intermediate trajectory set, $\mathcal{S}$ is cube sequence set. */
$CUBES \leftarrow$ GenerateCubes(M, $HH \in$ [0,1,...,23]);
for *each traj in DRT* **do**
  $ID \leftarrow \emptyset$;
  Initilize a cube trajectory ctj : $ctj.id \leftarrow traj.id$, $ctj.sequence \leftarrow \emptyset$;
  Initilize a trajectory tj : $tj.id \leftarrow traj.id$;
  for *each point in traj* **do**
    $CID \leftarrow$ InWhichCube(*point*, $CUBES$);
    if $CID \neq ID$ **then**
      Append CID to $ctj.sequence$, append *point* to tj;
    $ID \leftarrow CID$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup ctj$, $\mathcal{L} \leftarrow \mathcal{L} \cup tj$;
$PA \leftarrow$ PrefixSpan($\mathcal{S}$, *min-length*=2, *max-length*=2);
PT, $PC \leftarrow$ GetPatternTIDs(PA), GetPatternCubes(PA);
/* PT is pattern-matched trajectory-ID list, PC is pattern-matched cube-ID list. */
for *each traj in* $\mathcal{L}$ **do**
  if *traj.id in PT* **then**
    Initilize a trajectory tj : $tj.id \leftarrow traj.id$;
    for *each point in traj* **do**
      $CID \leftarrow$ InWhichCube(*point*, $CUBES$);
      if *CID in PC* **then**
        Append *point* to tj;
    $\mathcal{T} \leftarrow \mathcal{T} \cup tj$;
return $\mathcal{T}$

ones. For our ROI discovery, some unnecessary points should also be pruned out. To satisfy these needs, we propose a sequence-pattern-based filtering strategy in our approach. Our strategy follows the same idea as most outlier detection methodologies: if a data sample does not follow a number of companions, this sample can be considered as an outlier. Similarly, a definition of a trajectory pattern is proposed in our approach. If a trajectory does not match any trajectory pattern, it will be filtered out from the dataset used for urban ROI discovery.

Definition 4.8 (Spatiotemporal Cube)**.** Let a time span be equally divided by n. We can get n small continuous time spans, each denoted as τ. Given a mesh of an urban area, it consists of m fine-grained grids of the same size, each denoted as σ. By multiplying the

time spans and mesh grids, we can get $n \cdot m$ small spatiotemporal cubes c, where each c is a 3D region corresponding to a (τ, σ)-pair.

Definition 4.9 (Cube Trajectory)**.** Given a point, a (d, t, l)-tuple of one trajectory sequence, it can be mapped into a cube c if the following is satisfied:

$$t \in c.\tau \ \wedge \ l \in c.\sigma.$$

Through this kind of mapping, a trajectory can be converted into a sequence of a cube, called cube trajectory ct. Note that if several continuous (d, t, l)-tuples match the same cube, the cube will be only appended once into the cube sequence.

Definition 4.10 (Trajectory Pattern)**.** Consider a cube trajectory database $CTDB = \{ct_1, ct_2, ..., ct_n\}$, where ct_i is a sequence of cubes $(c_1, c_2, ..., c_n)$. Given a minimum support threshold δ, a trajectory pattern $tp = (p_1, p_2, ..., p_n)$ is defined as a frequent sequence pattern with minimum length, as follows:

$$support_{tp} = |\{ct \mid ct \in tp\}| \geq \delta \ \wedge \ tp.length \geq 2$$

where $ct \in tp$ is satisfied if there exists a subsequence of ct, $ct' = (c'_1, c'_2, ..., c'_n)$ such that $\forall i, \ c'_i = p_i$. In the rest of the article, we say a trajectory $traj$ can match a trajectory pattern tp if $ct \in tp$ is satisfied, where ct is the corresponding cube trajectory of $traj$.

Note that a "good" trajectory should have at least two "good" points that can match a certain length-2 trajectory pattern. Otherwise, if a trajectory holds only one "good" point, it cannot be treated as a real trajectory. This is why a minimum length equal to 2 is introduced in the definition.

Furthermore, the definition comes with a minimum length, which means we have to discover all of the patterns longer than 2. However, according to the Apriori Property [65]:

- Any subsequence of a frequent pattern is also frequent.

we can say that:

- If a trajectory $traj$ can match a trajectory pattern tp with length ¿2, $traj$ must match a length-2 pattern tp', where tp' is a subsequence of tp.

Based on this, we can only discover the length-2 patterns, which can notably improve the performance.

Definition 4.11 (Trajectory Filtering)**.** Given a raw trajectory $traj$ and a set of trajectory patterns, trajectory filtering is conducted from the following aspects:

- Redundant points of $traj$ will be filtered out. If a trajectory holds several continuous points located in the same cube, only the first point will be preserved as the representative point. Thus, this person stays stationary in a region within a time span.
- The noisy trajectory $traj$ will be filtered out if it cannot match any trajectory pattern.
- Noisy points of $traj$ will be filtered out. Let a trajectory point be mapped to a cube c. If c does not occur in any trajectory pattern, the point will be taken as a noisy point.

Our proposed pattern-based trajectory filtering is summarized as Algorithm 2. Lines 3-12 of the algorithm generate cube trajectories for trajectory pattern mining and filter redundant points of inputted raw trajectories. At line 13, PrefixSpan [66], one of the most widely used sequential mining algorithms, is applied. To meet our proposed definition, apart from the minimum support, we make PrefixSpan capable of taking a minimum length and a maximum length as two extra parameters, where both are set to 2. At line 14, from the discovered trajectory patterns, we extract the cube IDs that can have at least one trajectory pattern, as well as the trajectory IDs that can match at least one pattern. Lines 15-22 filter out the noisy trajectories that cannot match any pattern, and also the noisy points whose cube cannot occur at any pattern. For lines 8 and 19, which determine the cube to which a given point belongs, instead of searching over all cubes, we can directly calculate the cube ID using the timestamp and location coordinates of the point.

Through our proposed trajectory filtering, we can make sure that each cube traversed by at least δ "good" (length ≥ 2) trajectories will be preserved, while cubes without a sufficient number of "good" (length ≥ 2) trajectories passing by will be filtered out in the ROI discovering phase. From Fig. 4.2, we can see the effectiveness by comparing the trajectories before and after filtering. This can be as strong enough as a preprocessing method for ROI discovering. Some other filtering methodologies may also be

Algorithm 3: Multiple-Day Preliminary ROI Merging

Input: Multiple days of hourly preliminary ROI sets $DPROI$, a kernel bandwidth h, a mesh M, an integer K.

Output: Hourly urban ROI sets.

$ROI \leftarrow \emptyset$;
$GRIDS \leftarrow$ GenerateGRIDS(M);
for *each* $HH \in [0, 1, ..., 23]$ **do**
  $HHROI \leftarrow$ Get all days' preliminary ROI sets for HH from $DPROI$;
  $\mathcal{R}, \mathcal{P}, \mathcal{U} \leftarrow \emptyset, \emptyset, \emptyset$; /* $\mathcal{R}$ is urban ROI for HH, $\mathcal{P}$ is cluster centroids, $\mathcal{U}$ is merged clusters. */
  for *each roi in PROI* **do**
    $\mathcal{P} \leftarrow \mathcal{P} \cup roi.centroid$;
  $C \leftarrow$ MeanShift($\mathcal{P}, h$); /* C is centroid-clusters (clusters of centroids from mutiple days). */
  for *each* c *in* C **do**
    Initialize a cluster c' : $c'.centroid \leftarrow c.centroid$, $c'.points \leftarrow \emptyset$;
    for *each centroid in* $c.points$ **do**
      $c'.points \leftarrow c'.points \cup HHROI[centroid].points$;
    $\mathcal{U} \leftarrow \mathcal{U} \cup c'$;
  $Q \leftarrow$ Select top-K largest clusters in $\mathcal{U}$;
  for *each* q *in* Q **do**
    Initialize a region r : $r.center \leftarrow q.centroid$, $r.region \leftarrow \emptyset$;
    $\mathcal{R} \leftarrow \mathcal{R} \cup r$;
    for *each point in* $q.points$ **do**
      $g \leftarrow$ InWhichGRID($point, GRIDS$);
      $r \leftarrow GetNearestRegion(g, \mathcal{R})$;
      $r.region \leftarrow r.region \cup g$;
  $ROI[HH] \leftarrow \mathcal{R}$;
return ROI

considered, such as Trajectory Clustering[67]. However, these can be highly complex and difficult to apply to citywide-level trajectory datasets. Since our filtering algorithm takes only two parameters, mesh size and minimum support, it can be easily tuned and set.

4.4.3 Multiple-Day Preliminary ROI Merging

Given a set of preliminary ROIs discovered on multiple days, a merging process is proposed for our approach, which is summarized as Algorithm 3. Remember that each preliminary ROI is a cluster of trajectory points identified by a centroid point. Our method

of merging is to cluster all these centroids collected from multiple days by applying MeanShift once again to get so-called *centroid-clusters* (Lines 4-8). All preliminary ROIs are merged into large clusters. The centroid of each *centroid cluster* is utilized as the centroid of the merged cluster (Lines 9-14). Then, the top-K largest merged clusters are selected as the final urban ROIs (Line 15). Here, a threshold K is introduced because for different applications and users, different ROI numbers should be set flexibly when using our approach. Last, each candidate grid is extracted by iterating over all points of the top-K largest merged clusters. Each grid is assigned to the nearest cluster centroid to form the final urban ROIs (Lines 16-22). Thus, an urban ROI is approximately represented by fine-grained grids. Since each grid belongs only to a certain urban ROI, any two ROIs will be disjointed from each other.

4.5 Urban Human Mobility Modeling

Our approach is designed for urban human mobility modeling on a big historical trajectory database. Given a time t, a model can be built through a supervised learning process by taking current observed urban mobility X_t as the inputs and next urban human mobility $X_{t+\beta}$ as the outputs. By referring to this model, given a real-time observed mobility of the same time t, the next urban mobility can be predicted. For example, a mobility model can be built with one month of historical trajectory data (e.g., October). The current-time urban mobility from 07:00 to 07:59 is taken as inputs, and the next-time urban mobility from 08:00 to 08:59 is used as outputs. Assume that now is 08:00 on November 1. If we can get a person's mobility observations from 07:00 to 07:59, we can predict where this person will go from 08:00 to 08:59. Further, an online real-time prediction is available like this by building 24 models, one for each hour.

It should be noted that for one person's mobility prediction, we are concerned only about whether the model can precisely predict the next location with the highest probability. However, a large crowd of people can always share the same observed trajectories but they will head to different places after some time. This is one of the main characteristics of urban human mobility. Thus, our model should predict the overall probability distribution with high accuracy.

4.5.1 Modeling Urban Mobility on Grid

An observed α steps of one person can be simplified as: $x_t = l_1, l_2, ..., l_\alpha$, and a next β-th step mobility of this person $x_{t+\beta} = l_{\alpha+\beta}$ can be modeled and predicted based on mesh grids as follows:

$$P(g_{\alpha+\beta} \mid g_1, g_2, ..., g_\alpha), \ \forall i, l_i \in g_i \tag{4.1}$$

where g represents the mesh-grid.

Then, given all the current observed mobility data X_t from a TDB, urban human mobility prediction basically involves obtaining a predicted probability distribution $P(\widehat{X}_{t+\beta} \mid X_t)$, which should be as close as possible to the true probability distribution $Q(X_{t+\beta} \mid X_t)$. Therefore, we can obtain a grid-based model with the parameters θ that satisfies:

$$\theta = \underset{\theta}{\operatorname{argmin}} H(P(\widehat{X}_{t+\beta} \mid X_t), Q(X_{t+\beta} \mid X_t)) \tag{4.2}$$

where $X_{t+\beta}$ denotes the true next step of human mobility, $\widehat{X}_{t+\beta}$ denotes the predicted results for the whole urban area, and $H(\cdot)$ is the cross-entropy, which is widely used to measure the similarity between two probability distributions.

Modeling mobility on grid is similar to an n-gram language model, which is a typical probabilistic sequential model for predicting the next item in such a sequence in the form of a (n-1)-order Markov model. Like the language model, in which each word from a sequence of text or speech is a discrete value, each location is converted to a spatial grid for mobility modeling. However, it is difficult to directly apply this for modeling human mobility on a huge urban area:

- It will contain too many grids. In our case, around 6000 spatial grids will be generated for our experimental area with a 500-meter meshing. Even for a Bi-Gram model (first-order Markov), it still will lead to an extremely sparse transition matrix.
- The current mobility sequence can be long. Maintaining a high-order Markov model such as 5-gram model can be very inefficient. Typically for a language model, only the Bi-Gram model and Tri-Gram model are considered.

Hence, a new modeling approach should be designed to improve this naive grid-based n-gram model. As mentioned in Section 4.1, people mobility is related to urban ROIs to

a large degree. Because of this, urban ROIs are always a first concern for public service operators (e.g., governments, event organizers, and infrastructure companies). Instead of predicting mobility on massive fine-grained grids with poor performance, mobility prediction on urban ROIs with high precision is much more valuable and applicable. On the other hand, deep-learning technologies such as RNNs [68] and their special variants of long short-term memory (LSTM) networks [69] have provided an impressive performance in modeling sequential data[70] such as speech and text[71]. In particular, for our approach, they can help us model human mobility as a sequence classification problem and bring superior performance to classical methodologies. Motivated by the above, modeling urban human mobility on ROIs with deep neural networks is proposed in the following.

4.5.2 Modeling Urban Mobility on ROI

Definition 4.12 (ISROI Label)**.** Given next urban human mobility $X_{t+\beta}$ and a set of urban ROIs w.r.t $[t, t+\beta\Delta t]$, for a trajectory $x_{t+\beta} \in X_{t+\beta}$, the $ISROI$ label for $x_{t+\beta}$ is defined as follows:

$$z_{t+\beta} = x_{t+\beta}.ISROI = \begin{cases} 1, & \text{if } \exists\ j,\ x_{t+\beta}.l \in ROI_j \\ 0, & \text{otherwise} \end{cases}$$

The definition is essentially a formulation to the answer (1) (Enter any ROI or not ?) mentioned above. And the $ISROI$ label sets for $X_{t+\beta}$ is denoted with $Z_{t+\beta}$.

Definition 4.13 (WHICHROI Label)**.** Similarly, given next urban human mobility $X_{t+\beta}$ and a set of urban ROIs w.r.t $[t, t+\beta\Delta t]$, for a trajectory $x_{t+\beta} \in X_{t+\beta}$, the $WHICHROI$ label for $x_{t+\beta}$ is defined as follows:

$$y_{t+\beta} = x_{t+\beta}.WHICHROI = \begin{cases} j, & \text{if } \exists\ j,\ x_{t+\beta}.l \in ROI_j \\ \emptyset, & \text{otherwise} \end{cases}$$

The definition is essentially a formulation to the answer (2) (Enter which ROI ?) mentioned above. And the $WHICHROI$ label sets for $X_{t+\beta}$ is denoted with $Y_{t+\beta}$.

Given a historical raw trajectory database $HTDB$, through our proposed urban ROI discovering algorithm, hourly urban ROIs can be retrieved. Then next urban human mobility $X_{t+\beta}$ can be attached with two kinds of ROI-labels to get $Z_{t+\beta}$ and $Y_{t+\beta}$: (1)

ISROI label and (2) *WHICHROI* label. This is done by iterating over each point in $X_{t+\beta}$. Similarly, given an observed mobility $x_t = l_1, l_2, ..., l_\alpha$ of a person, the next ISROI-label and WHICHIROI-label prediction can be modeled as follows:

$$P(z_{t+\beta} \mid g_1, g_2, ..., g_\alpha), \ \forall i, l_i \in g_i \tag{4.3}$$

$$P(y_{t+\beta} \mid g_1, g_2, ..., g_\alpha), \ \forall i, l_i \in g_i \tag{4.4}$$

Specifically, z is labeled by $z = 0, 1$, which means this person will enter any ROI or not. y is labeled by $y = 1, 2, ..., K$, and K is the number of urban ROI, that means which ROI this person will enter.

Given the observed urban human mobility X_t, corresponding next *ISROI* labels $Z_{t+\beta}$ and *WHICHROI* labels $Y_{t+\beta}$, urban human mobility related to urban ROI areas can still follow multimodal distribution, which means a crowd of people who share the same observed mobility may eventually arrive at different urban ROIs. Therefore, ROI-based urban mobility prediction is to construct two probability distribution : $P(\widehat{Z}_{t+\beta} \mid X_t)$ and $P(\widehat{Y}_{t+\beta} \mid X_t)$, which should be as close as possible to the true probability distribution $Q(Z_{t+\beta} \mid X_t)$ and $Q(Y_{t+\beta} \mid X_t)$ respectively. $\widehat{Z}_{t+\beta}$ represents the predicted *ISROI* labels and $\widehat{Y}_{t+\beta}$ represents the predicted *WHICHROI* labels. Therefore, an ROI-based model consists of (1) ISROI model with the parameters η (2) WHICHROI model with the parameters ϵ that satisfy the following objective functions respectively:

$$\eta = \underset{\eta}{\operatorname{argmin}} H(P(\widehat{Z}_{t+\beta} \mid X_t), Q(Z_{t+\beta} \mid X_t)) \tag{4.5}$$

$$\epsilon = \underset{\epsilon}{\operatorname{argmin}} H(P(\widehat{Y}_{t+\beta} \mid X_t), Q(Y_{t+\beta} \mid X_t)) \tag{4.6}$$

where $H(\cdot)$ is the cross-entropy as mentioned above.

Working Mode: our urban human mobility prediction system can work on two modes when the models proposed above are well trained by big human mobility data. (1) ROI mode. First the system utilize ISROI model to discriminate which trajectories will enter urban ROIS, then for those ROI trajectories, it utilizes WHICHROI model to predict the probability distribution over each urban ROI. This mode can be very effective for some emergency situations or for the users who only care about some key urban areas such as railway operators. (2) Hybrid mode. To predict the entire urban human mobility, a hybrid mode is performed to combine grid-based modeling and ROI-based modeling together. Given an input trajectory, ISROI model is first utilized to judge the input trajectory will enter any ROI or not. If yes, WHICHROI model is utilized

as same as ROI mode, otherwise, a grid-based model will be utilized to generate the next possible grid with a probability distribution. Through this hybrid mode, all urban human mobility can be covered and predicted.

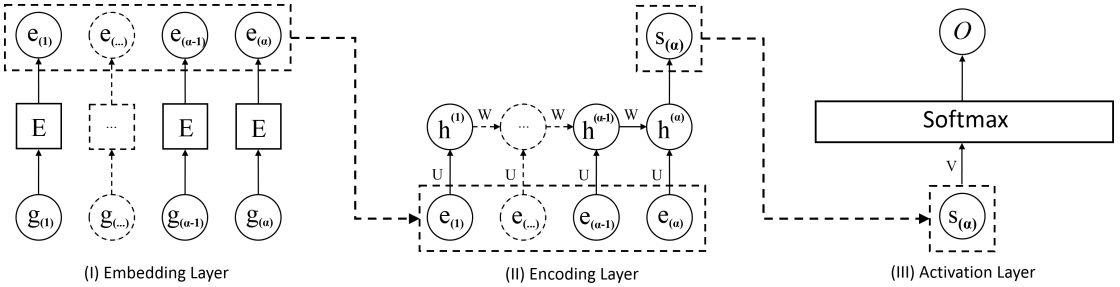


FIGURE 4.3: Deep Sequential Modeling Architecture.

4.5.3 Deep Sequential Modeling Architecture

Compared with traditional neural networks, RNNs are specially designed for sequential data modeling. In traditional neural networks, neurons in one layer and its neighboring layers are fully connected, whereas neurons in the same layer do not have any connections. Such structures cannot effectively deal with the situation when data are not independent, such as words in a sentence. The typical structure of a simple RNN is shown in Fig.4.1. We can see that the neighboring neurons in the same hidden layer are connected to one another so that the network can memorize former information and have an impact on the output of the current timestep τ. Therefore, the total input not only contains the input at the timestep τ, but also the output at the timestep τ-1. To train an RNN, the standard method is "backpropagation through time" (BPTT). It begins by unfolding RNN through time and generalizes the backpropagation for feed-forward networks to minimize the total error. In other words, it can also use gradient descent in proportion to the procedure to change the weight matrix like traditional neural networks.

To implement the sequence classification models defined as Equation 4.3 and 4.4, an RNN-based deep-learning architecture is constructed as shown in Fig.4.3. It works in the following steps: (1) The first layer is an embedding layer to turn an integer of grid id into a vector of continuous values, which is a more efficient low-dimension vector representation than a one-hot vector. (2) The second layer is an encoding layer constructed by the RNN, where a *tanh* function is used to map the α steps of the embedded mobility $(e_1, e_2, ..., e_\alpha)$ into a single latent vector s_α, which can be taken as auto-extracted features for the entire sequence. (3) The third layer is an activation layer, where a

$Softmax$ function is used to convert the latent vector s_α to probability values for each class.

The formulas that govern the entire computation in our architecture are as follows:

$$e_\alpha = Eg_\alpha \tag{4.7}$$

$$s_\alpha = \tanh(Ue_\alpha + Ws_{\alpha-1}) \tag{4.8}$$

$$o = \text{softmax}(Vs_\alpha) \tag{4.9}$$

where g represents the input mobility sequence $(g_1, g_2, ..., g_\alpha)$, s is the value of each hidden state in the encoding layer, o represents the output vector $(o_1, o_2, ..., o_n)$ and each o_i is the probability for each class, E is the embedding matrix in the embedding layer, W and U are weight matrices in the encoding layer, and V is the weight matrix in the activation layer. All of these weight parameters will be determined by applying the BPTT algorithm to minimize the cross entropy as defined in Equation 4.5 and 4.6. The algorithm details are omitted in this paper. This deep sequential modeling architecture has high applicability that can be easily applied to grid-based (GRID) modeling, ISROI modeling, and WHICHROI modeling by modifying only the number of hidden units in the final activation layer. Specifically, for GRID, ISROI, WHICHROI modeling, the number should be set to the number of mesh-grids, 2, and the number of urban ROIs (i.e. K) respectively.

Table 4.1: Parameter Description Table for Urban ROI Discovering

Parameter	Relevant Component	Tuned Value
Mesh M	Trajectory Filtering, ROI Merging	ΔLon=0.005, ΔLat=0.004
–	–	$\approx$ 450 m, 450 m
Minimum Support δ	Trajectory Filtering	10
Bandwidth h	ROI Discovering, ROI Merging	500 m
Top-K K	ROI Merging	50

Table 4.2: Parameter Description Table for Deep Sequential Learning

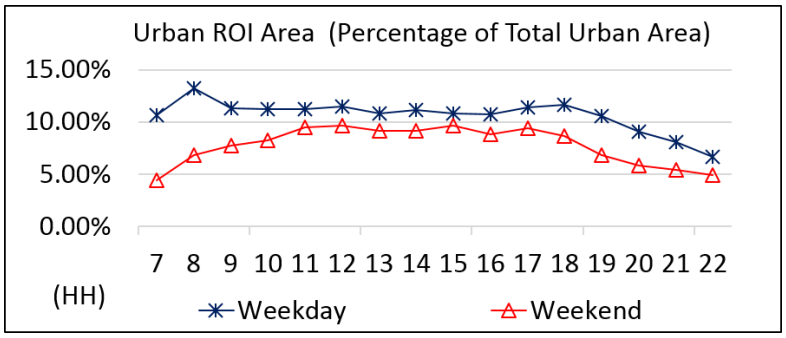
Parameter	Relevant Component	Tuned Value
Embedding Vector e	Embedding Matrix (Embedding Layer)	64
Encoding Vector s	RNN (Encoding Layer)	64
Output Vector o	Softmax (Activation Layer)	ISROI:2, WHICHROI:K
–	–	GRID: grid number
Learning Rate	RMSprop (Optimizer)	0.001
Batch Size	Training process	512

4.6 Experiment

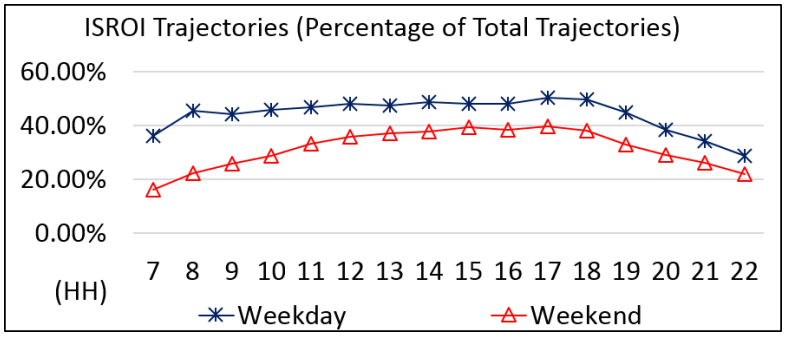
4.6.1 Experimental Setup and Parameter Settings

Experimental setup: In this research, from our three-year historical trajectory dataset, we randomly chose one month (October 2011) for the Greater Tokyo Area ($longitude \in [139.5, 139.9]$, $latitude \in [35.5, 35.8]$) as the experimental urban area. Urban human behaviors on weekdays and weekends are distinct from each other, so we separated this one month of data into two parts: a weekday dataset and a weekend dataset. Furthermore, trajectories were naturally segmented according to date, which means we had an individual small trajectory dataset on each day of October 2011. The weekday dataset contained around 1,700,000 trajectories, while the weekend dataset contained around 660,000 trajectories. For each dataset, we selected 60% of the trajectories as a training dataset, 15% of the data as a validation dataset, and the remaining 25% for testing. Based on this setup, given one-hour observed urban human mobility X_t ($X_t.\alpha = 6$) three types of urban human mobility predictions $\widehat{X}_{t+1}$, $\widehat{X}_{t+3}$ and $\widehat{X}_{t+6}$ ("Next-10-Minutes," "Next-30-Minutes," and "Next-60-Minutes") were tested. Correspondingly, three types of ISROI predictions $\widehat{Z}_{t+1}$, $\widehat{Z}_{t+3}$ and $\widehat{Z}_{t+6}$, and three types of WHICHROI predictions $\widehat{Y}_{t+1}$, $\widehat{Y}_{t+3}$ and $\widehat{Y}_{t+6}$ were also taken as the evaluation targets.

Parameter settings: For the mining process, all parameter settings are summarized in Table 4.1. The parameters were tuned and selected by us through a series of background indexes such as (1) the urban ROI area (percentage of total urban area) and (2) the percentage of ISROI trajectories, which is the total number of trajectories that will enter one of the urban ROIs in the next hour. Some additional investigations were conducted to determine the appropriate total ROI number for the Tokyo area and the mesh size. For the learning part, the key parameter settings are summarized in Table 4.2. A 64-dimension embedding vector was used in the embedding layer, and a 64-dimension latent representation was used in the encoding layer. The RMSprop algorithm was adopted in our system to govern the entire training process. All parameter settings were kept the same for the weekday dataset and weekend dataset. Java was used to implement the mining algorithm of our approach, while Python and some Python libraries including Keras [72] and TensorFlow [73] were used to implement the learning part.



(a) Urban ROI Area



(b) ISROI Trajectories

FIGURE 4.4: Effectiveness Evaluation for Urban ROI Discovering

4.6.2 Evaluation for Urban ROI Discovering

With these parameters, discovered urban ROIs keep the properties as shown in Fig. 4.4. This demonstrates a 50/10 rule for urban human mobility in which an average 10% urban area can cover nearly 50% urban human mobility, which can be an analogy to the Pareto principle (also known as the 80/20 rule). In addition, for ISROI, a binary-class classification, 50% is an ideal percentage for positive samples and negative samples. We have also verified that how many hot stations in Tokyo area can be covered by our discovered top-50 urban ROIs. In Fig.4.5, each of the 50 urban ROIs is represented by a random color, and each hot station is represented with a small red marker. We can see that among the top 100 hot stations in Greater Tokyo Area, around 80 stations are covered by the discovered 50 urban ROIs.

In the following subsections, we evaluated our deep ROI-based modeling approach from the following three aspects: (1) comparing deep sequential classification models with non-deep-learning classification models for ISROI and WHICHROI modeling, (2) comparing ROI-based modeling with classical grid-based modeling for the prediction, and (3) studying how the key parameters impact the overall performance.

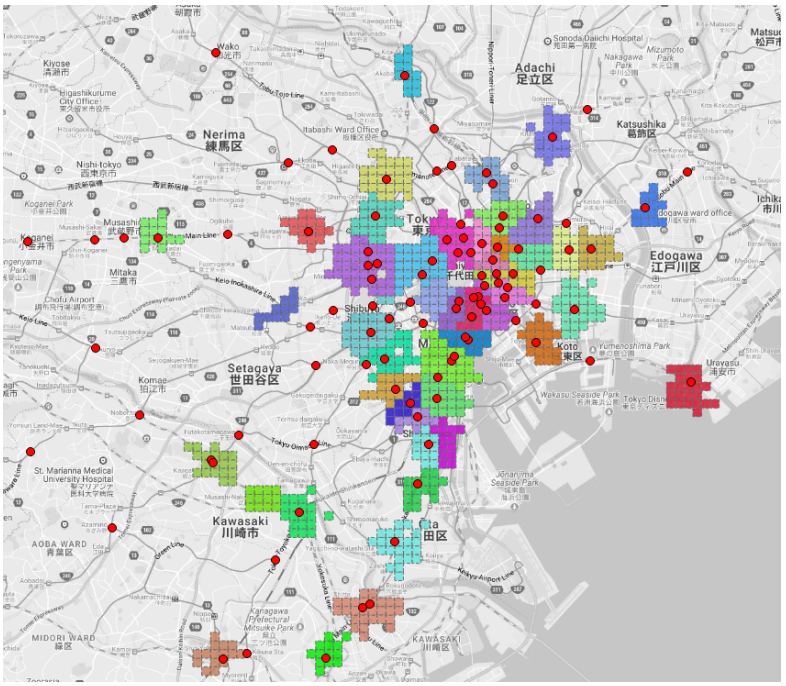


FIGURE 4.5: Top-50 Urban ROIs and Top-100 Hot Stations

4.6.3 ISROI and WHICHROI Modeling Evaluation

In this subsection, we compare our proposed deep sequential learning architecture with classical models for ISROI and WHICHROI modeling.

Metrics: We evaluated the performance of the proposed models using two metrics. Cross-entropy (denoted as LOSS) was used as the loss functions for both ISROI and WHICHROI modeling. It describes the loss between two probability distributions, and it was also used as the primary metric in the evaluation. Furthermore, to be intuitive, accuracy (ACC) was used as the secondary metric to help us get a straightforward understanding of the results. Note that urban human mobility essentially follows a multi-modal distribution because people share the same short-term trajectories but they will go to different places after some time. Thus, LOSS is a better metric than ACC for our urban human mobility prediction task. Specifically, these metrics are defined as follows:

$$LOSS_{ISROI} = \frac{1}{n}\sum_{i}^{n} -[z_i^{(0)} \log(\hat{z}_i^{(0)}) + z_i^{(1)} \log(\hat{z}_i^{(1)})] \tag{4.10}$$

$$ACC_{ISROI} = \frac{1}{n}\sum_{i}^{n} 1(z_i = \hat{z}_i) \tag{4.11}$$

$$LOSS_{WHICHROI} = \frac{1}{n}\sum_{i}^{n}\sum_{k}^{K} -y_i^{(k)} \log(\hat{y}_i^{(k)}) \tag{4.12}$$

$$ACC_{WHICHROI} = \frac{1}{n}\sum_{i}^{n} 1(y_i = \hat{y}_i) \tag{4.13}$$

where n is the number of samples, $z^{(0)}$, $z^{(1)}$ denote the one-hot true probability of ISROI (i.e., if ISROI = 1, $z^{(0)} = 0$, $z^{(1)} = 1$. Otherwise, $z^{(0)} = 1$, $z^{(1)} = 0$), $\hat{z}^{(1)}$, and $\hat{z}^{(0)}$ are the predicted probability about entering the ROI or not, z and $\hat{z}$ are the true ISROI label and the predicted ISROI label (with the highest probability), K is the ROI number, $y^{(k)}$ and $\hat{y}^{(k)}$ are the one-hot true probability (i.e., if WHICHROI = k, $y^{(k)} = 1$ and $y^{(\neq k)} = 0$) and the predicted probability on each ROI, respectively, and y and $\hat{y}$ are the true WHICHROI label and the predicted WHICHROI label (with the highest probability).

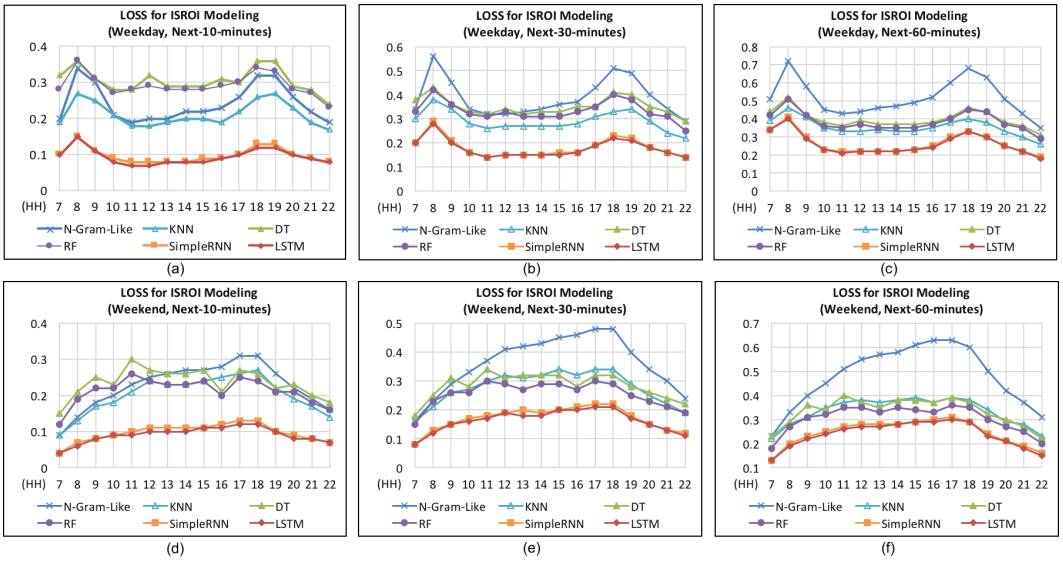


FIGURE 4.6: ISROI Modeling Evaluation by LOSS

Comparison models: (1) N-Gram-Like. This is a widely used algorithm for modeling sequential data, especially in the field of natural language processing. In our study, we applied this model based on a gridded space to predict the ISROI and WHICHROI labels. Here, we calculated the probability distribution for ISROI and WHICHROI based on the last three steps. This is similar to a Four-Gram model (3-order Markov model). (2) KNN. A KNN-based learning model [74] is a type of instance-based learning where classification is computed from a simple majority vote of the nearest neighbors of each point. (3) DT. A decision tree [75] is built to predict the target value by learning simple decision rules. (4) RF. A random forest [76] is constructed with a multitude of decision trees, which usually can bring better performance than a decision tree. For (2)~(4),

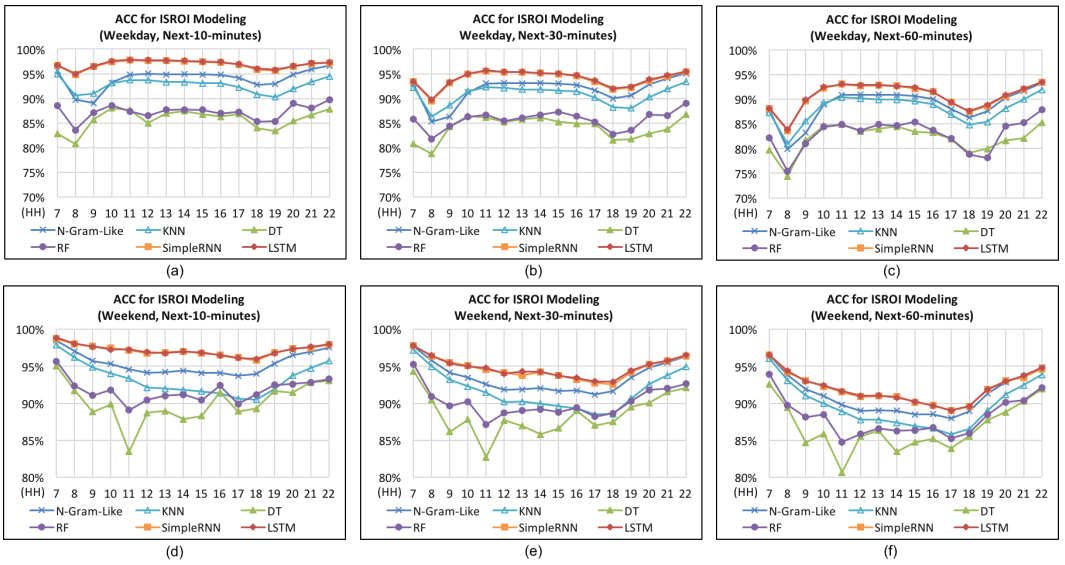


FIGURE 4.7: ISROI Modeling Evaluation by ACC

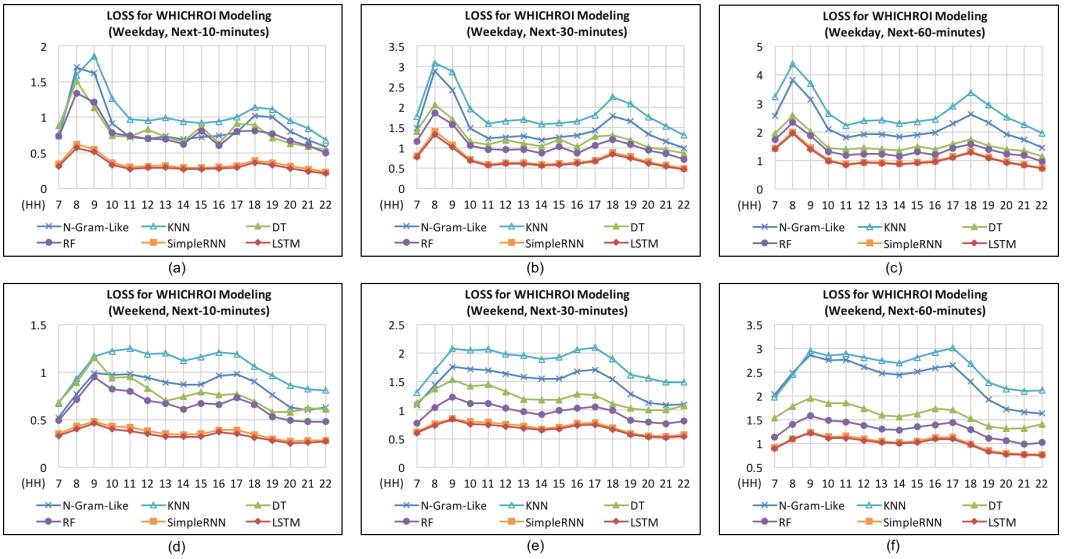


FIGURE 4.8: WHICHROI Modeling Evaluation by LOSS

these classical methodologies are extended to output the probability distribution for ISROI and WHICHROI rather than only the predicted label. As the input features, the grid-mapped trajectories with α steps are represented as α-dimension vectors, and each dimension stores the grid ID for the corresponding step. These four models are used as our baselines, which can very effective for both binary classification (ISROI) and multiclass classification (WHICHROI). (5) SimpleRNN. This is a deep sequential learning architecture constructed with traditional RNNs. (6) LSTM. We also implement a deep sequential learning architecture with LSTM[69], which shares the same architecture with RNN except that ordinary neurons in traditional RNNs are replaced with special computation blocks, namely LSTM. This has shown superior performance to traditional

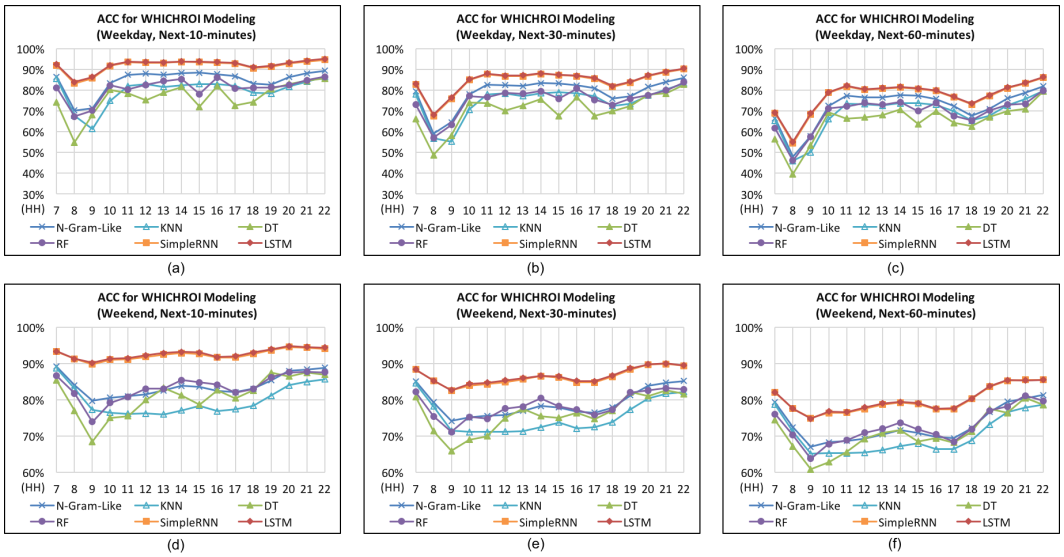


FIGURE 4.9: WHICHROI Modeling Evaluation by ACC

RNNs for long time-series modeling; therefore, we want to test if it can further improve the performance of our short-term prediction.

Evaluation results: Fig. 4.6~4.9 (LOSS of ISROI, ACC of ISROI, LOSS of WHICHROI, and ACC of WHICHROI) are the evaluation results of six comparison models for ISROI and WHICHROI modeling using LOSS and ACC as the metrics, where only the results for the core hours 07:00 ~ 22:59 are given. In each figure, three subfigures (a)~(c) on the top show the "Next-10-Minutes," "Next-30-Minutes," and "Next-60-Minutes" on weekdays, and the subfigures (d)~(f) at the bottom show the results on weekends. Through Fig. 4.6~4.9, we can see that

(1) Compared with N-Gram-Like model, KNN, DecisionTree, and RandomForest, traditional RNN and LSTM-RNN achieve the best performance for both ISROI and WHICHROI modeling over different prediction time intervals (10 min, 30 min, and 60 min). Note that the performances of SimpleRNN and LSTM are almost the same, which indicates that a traditional RNN is sufficient for encoding only 1-h short urban mobility.

(2) As the prediction time interval increases, the dependency on current mobility becomes weak; therefore, the prediction performances decrease owing to this reason. However, our deep-learning methods can still achieve satisfactory performances even for the "Next-60-Minutes," where the ACC of ISROI remains around 90% on both weekdays and weekends (Fig. 4.7-(c) and Fig. 4.7-(f)), and the ACC of WHICHROI remains around 80% on both weekdays and weekends (Fig. 4.9-(c)

and Fig. 4.9-(f)). These high accuracies demonstrate high applicability in real-world prediction systems.

(3) The urban mobility prediction results show different oscillating patterns on weekdays and weekends. On weekdays, all models perform relatively badly around 08:00 am, the typical morning commuting hours in Tokyo. All models also perform badly around 6:00 pm, the evening commuting hours in Tokyo, but not that bad compared to 08:00 am. This makes sense because the commuting behaviors are much more obvious in the morning than in the evening. Some salarymen tend to stay late in the office under Japan's overtime culture, so the commuting hours can be from 6:00 pm to 10:00 pm or even later. On weekends, the models give similar and stable performances during the daytime (9:00 am-6:00 pm), and better performances appear during other hours (early morning or at night) since human behaviors become less active. Note that urban human mobility running on a highly complicated transportation system will change drastically during rush hours on weekdays, and urban human mobility will follow a much more complex multimodal distribution during these hours. It is difficult to effectively measure WHICHROI modeling by ACC since people with the same current trajectories may head to different ROIs after some time. We will show an example of this in the next section. From Fig. 4.8 and Fig. 4.9, we can also observe that the evaluation results of ACC oscillate more drastically than the LOSS results on weekdays.

In summary, deep sequential learning architecture built with RNNs demonstrates superior performance for ISROI and WHICHROI modeling over classical methodologies.

4.6.4 Evaluation between Grid-based and ROI-based Modeling

In this subsection, we compare ROI-based modeling with classical grid-based modeling for urban human mobility prediction. How grid-based modeling and ROI-based modeling work for urban mobility prediction will be described as follows:

(1) A grid-based modeling method defined by Equation 4.1 can also be implemented with a deep sequential learning architecture by modifying only the Softmax layer with the mesh-grid number. For an input trajectory, a probability distribution over all mesh grids can be outputted by our deep sequential model, and one grid will be

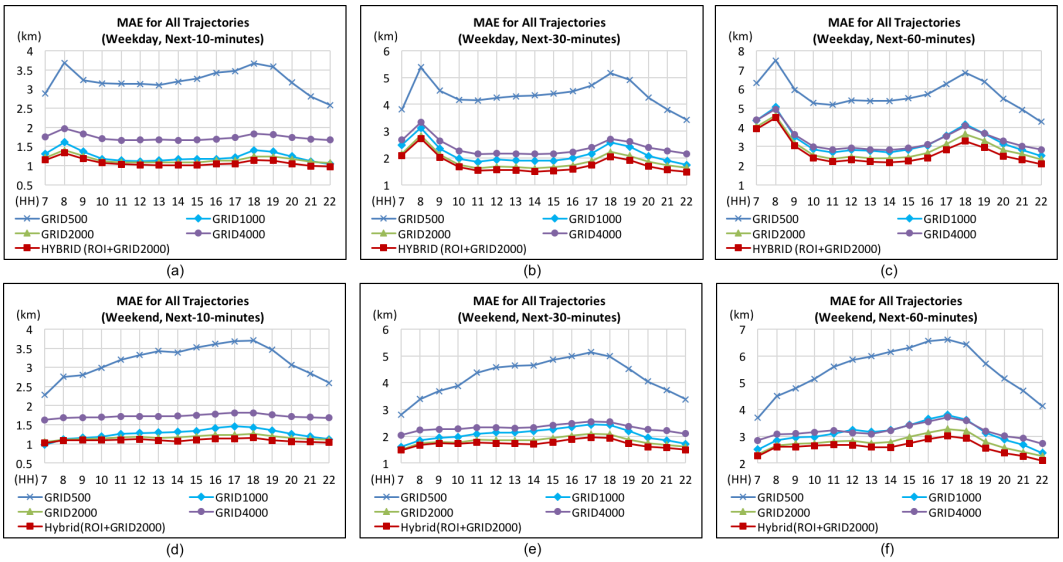


FIGURE 4.10: MAE for All Trajectories

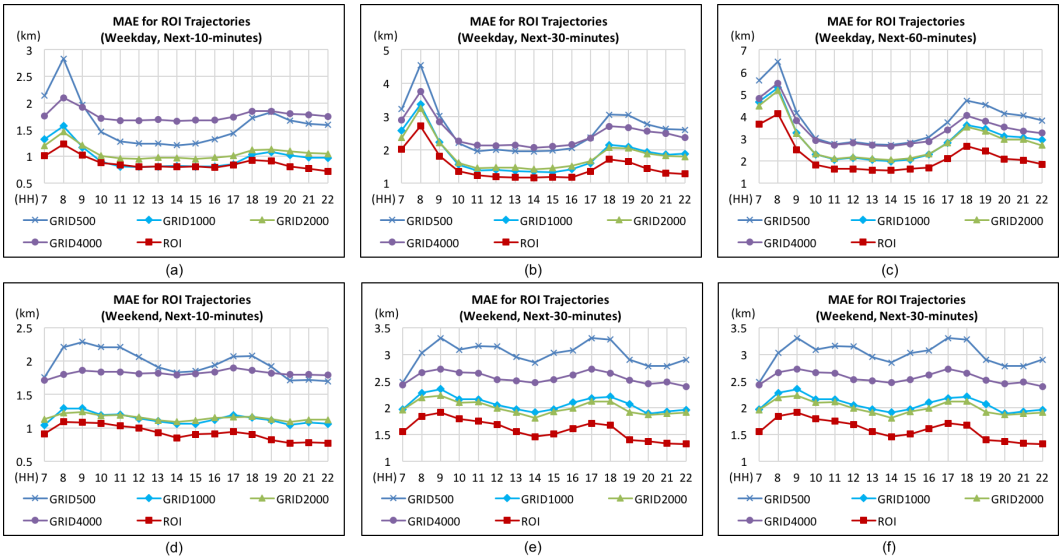


FIGURE 4.11: MAE for ROI Trajectories

generated as the prediction result by probability sampling with that probability distribution. Eventually, the centroid of the chosen grid will be utilized as the predicted point.

(2) An ROI-based modeling method will perform in a hybrid mode to predict the entire urban human mobility. Given an input trajectory, an ISROI model is first utilized to judge whether the input trajectory will enter any ROI. If yes, the WHICHROI model is further utilized to predict the probability distribution over all urban ROIs. Then, one urban ROI will be generated as the prediction result by probability sampling, and the centroid of the predicted ROI will be utilized as the predicted point.

Otherwise, instead of a WHICHROI model, a grid-based model will be utilized to generate the centroid of one sampled grid as the final predicted point. This follows the same procedure as (1).

Through such mechanisms, grid-based and ROI-based modeling can be fairly compared with each other. Furthermore, to better check the differences between these two modeling methods, we run two separate evaluations as follows: (1) Utilize all testing trajectories to run on a grid-based model and ROI-based hybrid model to test the overall performance. (2) Pick up the ROI trajectories (the ones that enter urban ROIs) from the entire testing dataset to run on a grid-based model and a WHICHROI model.

Metrics: We redefine the Mean Absolute Error (MAE) as follows:

$$\text{MAE} = \frac{1}{n}\left[\sum_{i=1}^{n} \|l_i - \widehat{l_i}\|\right]$$

where n is the total number of trajectories, and $\|l - \widehat{l}\|$ is the Euclidean distance between the real location and the predicted grid centroid or the predicted ROI centroid, which will be measured in kilometers (km).

Comparison models: Here, four types of meshes are utilized to implement the grid-based modeling, and the parameters of these four meshes are listed in Table 4.3. The four grid-based models are denoted as GRID500, GRID1000, GRID2000, and GRID4000. They all implement a deep sequential learning architecture with traditional RNNs following the same parameter settings listed in Table 4.2. The only difference is that the Softmax layer will be modified according to the corresponding mesh-grid number.

Table 4.3: Mesh Parameter

Mesh Name	Parameter
GRID500	(ΔLon = 0.005, ΔLat=0.004 $\approx$ 450 m, 450 m) $\times$ 6000
GRID1000	(ΔLon = 0.01, ΔLat=0.008 $\approx$ 900 m, 900 m) $\times$ 1480
GRID2000	(ΔLon = 0.02, ΔLat=0.016 $\approx$ 1800 m, 1800 m) $\times$ 360
GRID4000	(ΔLon = 0.04, ΔLat=0.032 $\approx$ 3600 m, 3600 m) $\times$ 90

Evaluation results: The overall evaluation results measured by MAE for all testing trajectories are summarized in Fig. 4.10. The evaluation results focusing only on ROI trajectories are summarized in Fig. 4.11. Similarly, subfigures (a)~(c) show the MAE results of "Next-10-Minutes," "Next-30-Minutes," and "Next-60-Minutes" on weekdays. Subfigures (d)~(f) show the MAE results on weekends. The ROI-based model

works in a hybrid mode along with the GRID2000 model. This is denoted as Hybrid(ROI+GRID2000) in Fig. 4.10. The WHICHROI model is denoted as ROI for simplicity in Fig. 4.11. Through Fig. 4.10~4.11, we can observe that (1) similar oscillating patterns appear with ISROI and WHICHROI modeling on weekdays and weekends, and (2) as the prediction time interval increases, the MAEs also become large. Additional key points are listed as follows:

(1) In Fig. 4.10, GRID2000 achieves the best performance among these four grid-based models. An ROI-based hybrid model can further improve the overall performance by working with the GRID2000 model.

(2) In Fig. 4.11, for ROI trajectories, ROI-based modeling (i.e., the WHICHROI model) holds a clear advantage over grid-based modeling. Especially on weekdays, the advantage becomes more obvious when the prediction time interval increases to 60 min, which can be crucial for conducting a multistep prediction.

In summary, ROI-based modeling shows better performance for urban human mobility prediction than classical grid-based modeling.

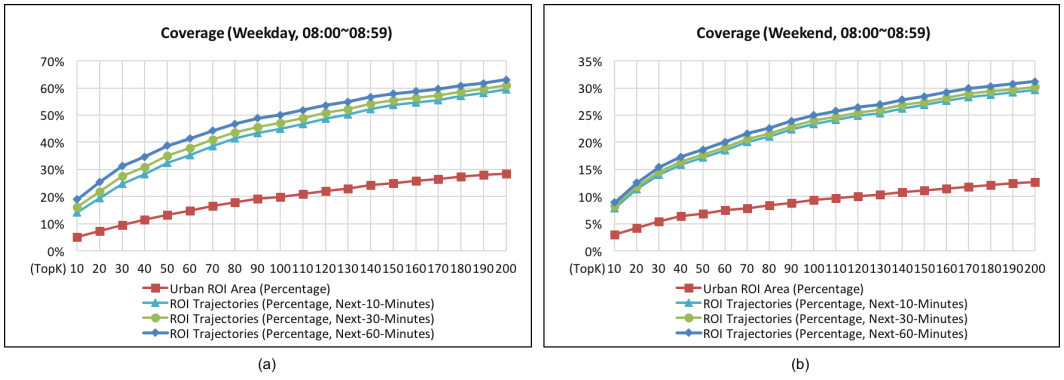


FIGURE 4.12: Coverage w.r.t TopK

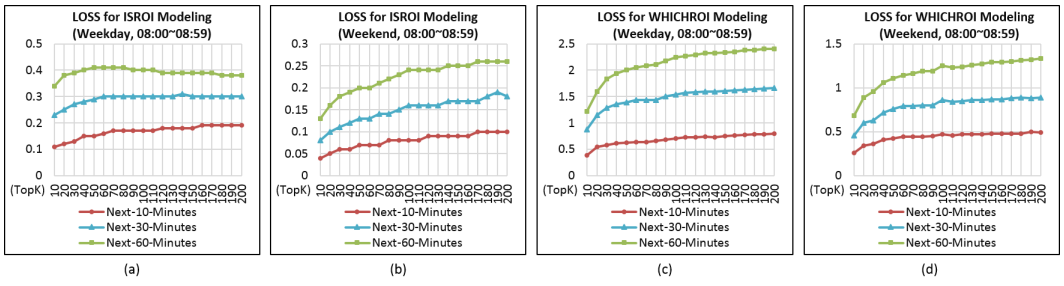


FIGURE 4.13: LOSS of ISROI and WHICHROI w.r.t TopK

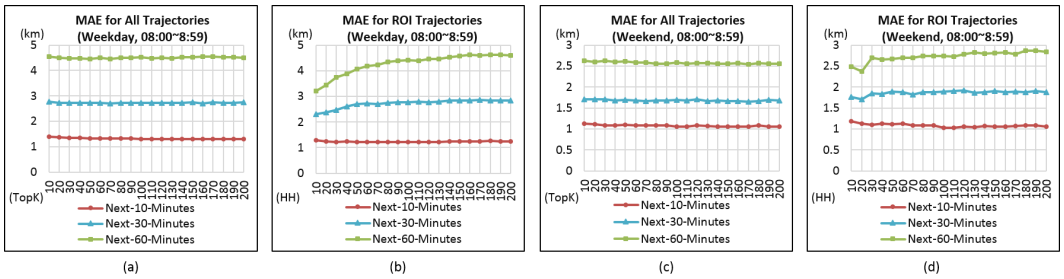


FIGURE 4.14: MAE w.r.t TopK

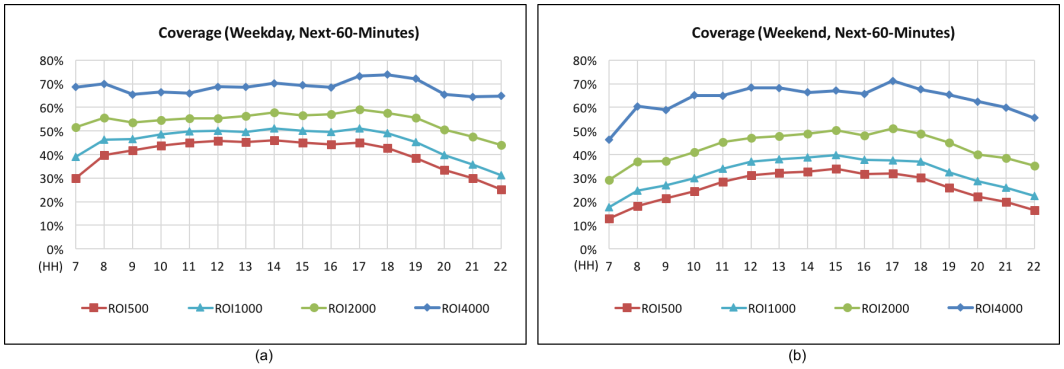


FIGURE 4.15: Coverage w.r.t ROI Size

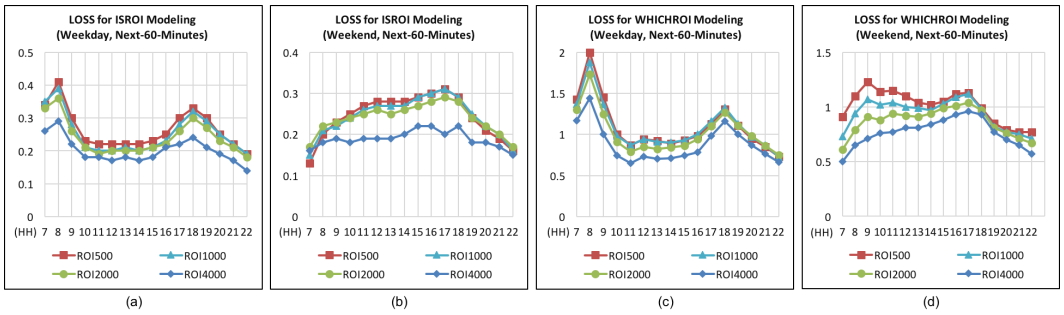


FIGURE 4.16: LOSS of ISROI and WHICHROI

4.6.5 Parameter Study

We mainly study how two key parameters impact on the overall performance: (1) the ROI number denoted as TopK; (2) the ROI size. By varying the two parameters, we evaluate how the coverage (Percentage of ROI trajectories), LOSS of ISROI and WHICHROI model and MAE of mobility prediction will be effected. Specifically, the results of Coverage, LOSS, MAE with respect to TopK are summarized as Fig.4.12~4.14, and the results with respect to ROI size are summarized as Fig.4.15~4.17.

TopK: We vary TopK from 10 to 200 to get different numbers of urban ROI. To make the results look more straightforward, we only list the evaluation results of 08:00~08:59

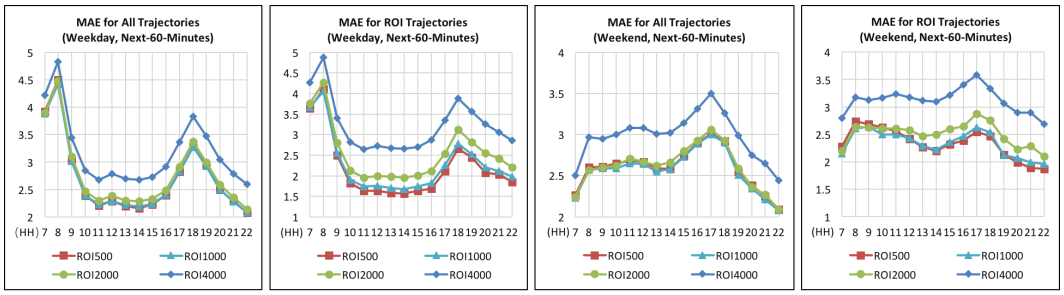


FIGURE 4.17: MAE w.r.t ROI Size

rather than 07:00~22:59 in Fig.4.12~4.14. Evaluation results w.r.t TopK are listed as follows:

1. Through Fig.4.12, we can see that as TopK increases, coverage of urban ROI area and ROI trajectories both become larger, and the slope of coverage of ROI trajectories is higher than coverage of urban ROI area. Furthermore, we can also see that from 08:00 to 08:59 more trajectories will enter ROI area as time goes by, and this phenomenon is more obvious on weekday than weekend.

2. In Fig.4.13, the ISROI LOSS of weekday keeps stable after TopK is bigger than 50; the ISROI LOSS of weekend has a clear growth as the TopK goes bigger; the WHICHROI LOSS of both weekday and weekend have a slight growth after TopK is larger than 50.

3. Through Fig.4.14(a)(c), we can see that the MAE for all testing trajectories does not change too much along with TopK for both weekday and weekend. "Next-60-Minutes" MAE for the ROI trajectories shown in Fig.4.14(b)(d) becomes larger as TopK increases, and the MAE results keep relatively stable for "Next-10-Minutes" and "Next-30-Minutes"prediction.

In general, ISROI and WHICHROI modeling and urban mobility predictions are stably performed over TopK.

Size of ROI: We utilize four different meshes with the parameters listed in Table.4.3 to vary the ROI size, and denote the four kinds of ROI as ROI500, ROI1000, ROI2000 and ROI4000 respectively. Similarly, to make the results look more straightforward, we only list the evaluation results of "Next-60-Minutes" prediction in Fig.4.15~4.17. Evaluation results w.r.t ROI size are listed as follows:

1. Through Fig.4.15, we see that as ROI size increases, the trajectory coverage becomes higher, but the difference between RO500 and ROI1000 is just around 5%, which is not so big.

2. The LOSS results of ISROI and WHICHROI on ROI500, ROI1000 and ROI2000 are similar with each other as shown in Fig.4.16. The ROI-based modeling can perform relatively stably on ROIs with different size.

3. To predict the entire urban mobility, we still utilize GRID2000 model mentioned above to work along with the four ROI models of different ROI size. From Fig.4.17(a)(c), we can see that ROI500, ROI1000 and ROI2000 achieve almost the same performances. As shown in Fig.4.17(b)(d), for the prediction only on ROI trajectories, ROI500 and ROI1000 obtain better performances than the other two.

Based on these results, the size of the fine-grained grids used for approximate ROI representation can be up to around 1000 meters. Furthermore, these two parameters TopK and ROI size will both have impact on the urban mobility prediction, and these two parameters should fine-tuned together for a real-world application.

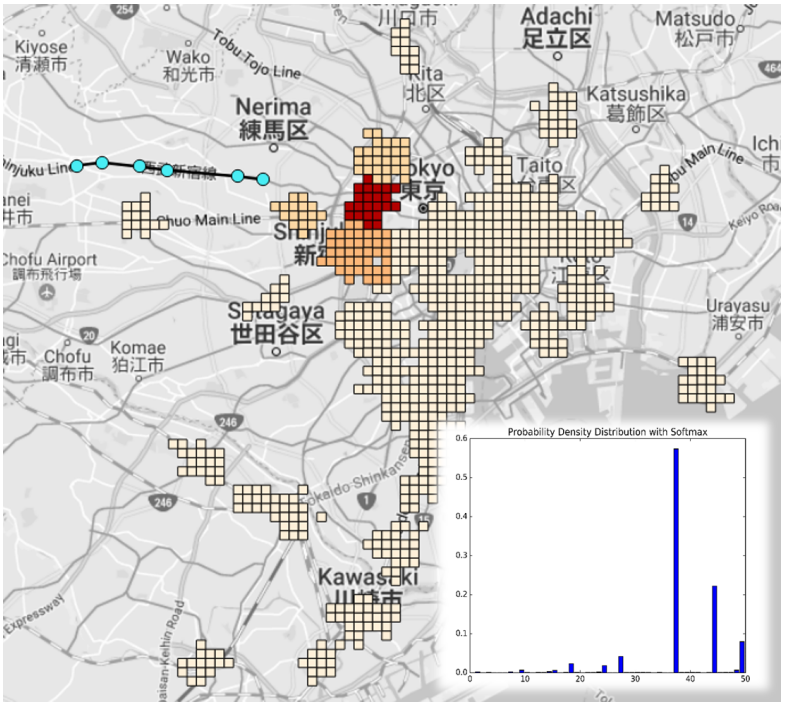
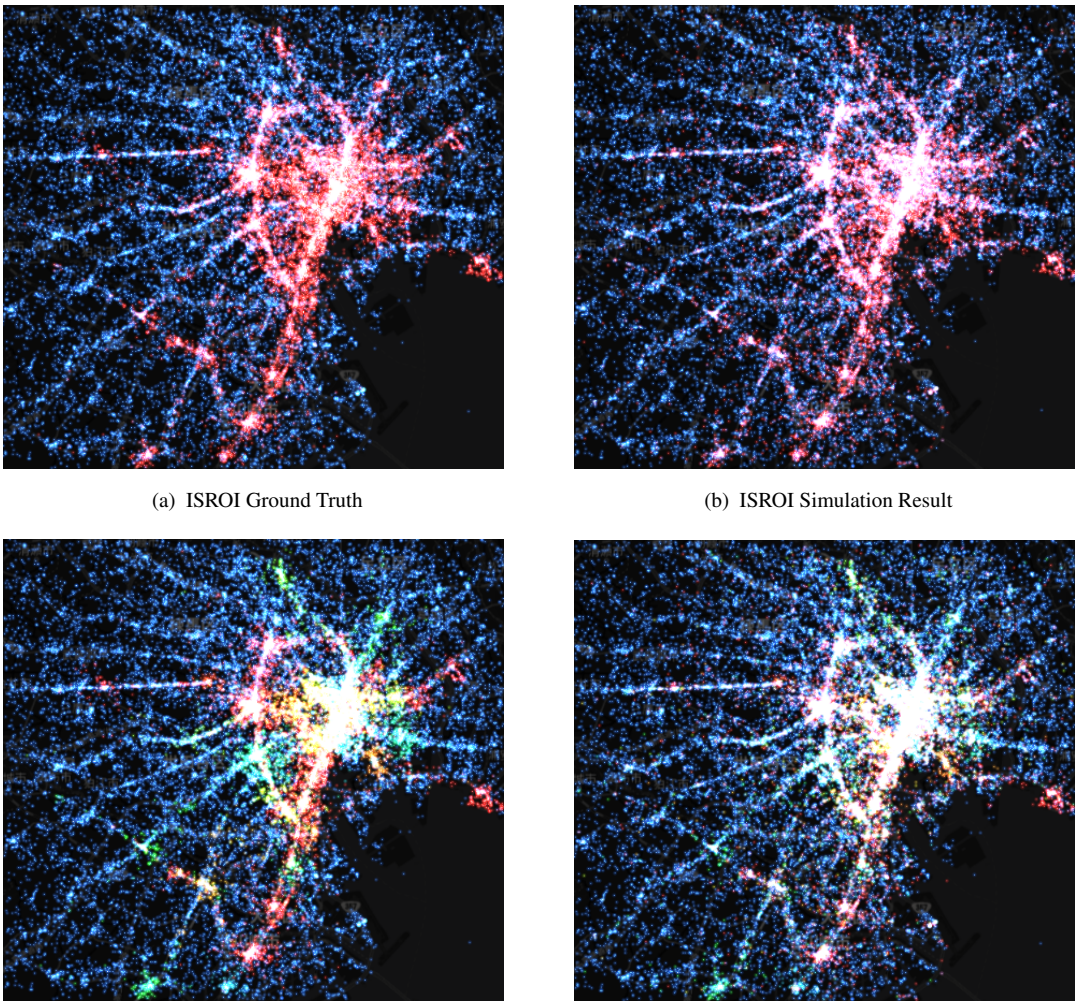


FIGURE 4.18: Simulation for One Person.



(a) ISROI Ground Truth

(b) ISROI Simulation Result

(c) WHICHROI Ground Truth

(d) WHICHROI Simulation Result

FIGURE 4.19: Simulation for Urban Mobility

4.7 Real-world Simulation

In the experimental stage, our proposed models are built based on the dataset of October 2011. For a real-world simulation, we assume that now is 08:00:00 on November, 1, 2011. A mobility prediction model of 08:00 ~ 08:59 has been successfully trained with last month's historical trajectory data. Then, simulations for one specific person and for the entire urban human mobility can be conducted with the trained model.

Simulation for one person: Here, we assume that now Tom is on his way from home to work by the "Seibu Shinjuku Line" [2], which connects Shinjuku Station of Tokyo with Hon-Kawagoe Station of Saitama Prefecture. His observed trajectories from 07:00 to 07:59 are represented by a couple of blue marks at the top left of Fig. 4.18. Our

[2]https://en.wikipedia.org/wiki/Seibu_Shinjuku_Line

proposed model can output a probability distribution on each ROI with Softmax, as shown at the bottom right of Fig. 4.18. Then, the next possible destination of Tom can be predicted as a heatmap, where the dark color represents high probability. For this person, his top-three possible destinations during the next hour are Takadanobaba Station, Shinjuku Station, and Ikebukuro Station, which are quite reasonable results from an empirical perspective. In particular, the prediction in a probability distribution format can be very useful for predicting a large crowd of people who share the same observed trajectories. For example, if 1,000 people are on the same train with Tom, we can predict that around 600 people will go to Takadanobaba Station, around 200 people will go to Shinjuku Station, and around 100 people will go to Ikebukuro Station.

Simulation for urban mobility: Given one-hour observed urban human mobility of 07:00~07:59, a simulation for "Next-30-Minutes" urban human mobility prediction was conducted using the data of 2011-11-01. The visualization results for this simulation are listed in Fig. 4.19. The ground truth and simulation results for ISROI are shown in Fig. 4.19(a) and Fig. 4.19(b), respectively. Those trajectories that will enter a certain urban ROI during the next hour are marked in red; otherwise, the trajectories are marked in blue. We can see that the urban human trajectories are clearly divided into two parts. The ground truth and simulation results for WHICHROI are shown in Fig. 4.19(c) and Fig. 4.19(d), respectively. The random color indicates which urban ROI it will enter. The simulation result has high similarity with the ground truth for both ISROI and WHICHROI prediction. Through this simulation, we can further confirm that our ROI-based modeling on urban human mobility is effective even for this morning rush hour.

4.8 Related Work

Recently, various studies were conducted on human mobility data (mobile phone GPS log data, taxi GPS data, and location-based services data). These are summarized as urban computing problems in [5].

Trajectory-pattern based methodologies have been proposed to predict future movement of individual person [6, 7]. An approach based on nonlinear time series analysis of the arrival and residence times of users has been proposed, which focused on predicting most important places of each user [9]. These models focused on individual mobility,

which are difficult to be applied to our urban mobility modeling task. Some collaborative approaches have been proposed to take social relationships of users into account for location prediction and recommendation, but they utilized big check-in data from location-based network services [1, 11, 12].

CityMomentum [2] is the most closely related work to ours; however, it builds a prediction model with real-time current observed data, while ours predicts the mobility using historical data of the same hour. Most important, our work is the first attempt to model urban mobility based on urban ROIs, which has shown its own advantages over grid-based modeling approach like [2]. Furthermore, modeling human mobility for very large populations [13, 22, 36] and simulating human emergency mobility following disasters [37, 38] are other topics that are close to ours. However, all of these approaches had different problem definitions and modeling methods. For example, the approaches required disaster information such as intensity of earthquake and damage level as additional input data in [37, 38]. In addition, they did not use the power of deep-learning technologies. Forecasting citywide crowd density [3, 19] is a related endeavor based on deep learning that builds a long time-series model for predicting the density value for each grid, whereas our approach aims at predicting citywide human mobility through sequence classification.

Understanding the basic life patterns of the flow of people [24] and recommending location-based services [25, 26] are studies that utilize the tensor factorization approach to decomposing urban human mobility, whereas ours utilizes urban ROIs to represent urban mobility and can provide more details of human mobility in comparison with an O-D matrix. Traffic flow, which can be seen as a special human mobility constraint on road networks, has also been studied in the form of traffic flow prediction [29] and traffic congestion monitoring [30], but they also start with some kind of individual model for each road segment. Some researchers have applied deep learning to traffic problems such as traffic speeds and transportation modes along with human mobility [45–49]. Moreover, C. Song [31] explored the upper bound of the predictability of human mobility. J. Zheng [10] proposed an unsupervised learning algorithm for location prediction. A more advanced trajectory calibrating algorithm was proposed in [32].

4.9 Conclusions

In this article, an effective ROI-based approach was proposed for predicting the next urban human mobility with historical trajectory data. A divide-and-merge mining algorithm was designed to discover urban ROIs from raw trajectories of multiple days. Based on the discovered urban ROIs, urban human mobility was modeled with two questions: (1) Will this person enter any ROI? and (2) Which ROI he will enter? To implement this, a deep sequential architecture was specially designed with an RNN as an effective sequence-classification model. The experimental results demonstrated the superior performance of our proposed approach compared to the baseline models. Furthermore, we applied our approach to a real-world simulation and verified its applicability.

Our modeling approach is a complete and highly applicable framework that can handle large and raw trajectory data with long-period historical records. It demonstrates the advantage of combing data-mining methodologies and deep-learning technologies to obtain the deep knowledge about urban human behaviors from big trajectory data. It can be easily utilized to implement real-world mobility prediction systems for various urban areas basing on different trajectory data sources. For instance, if we can get similar data source of Shanghai (big historical data for mining and training, real-time streaming data for prediction), we can deploy our system to run on top of Shanghai ROIs for next 10, 30, 60 minutes mobility prediction. As one of the most important ROIs of Shanghai, the population density of the Bund can be accurately predicted in advance by our system, thus, we believe that a stampede tragedy like that New Year's Eve in Shanghai can be prevented from happening again.

Our study has some room for improvement in the following aspects: (1) Our approach is still struggling to deal with situations in which urban mobility is full of sudden changes such as morning rush hour. (2) Currently, only one month's data was used for our experimental evaluation. To validate the scalability and improve the overall performance of our approach, we need to try more trajectory data over one month. Furthermore, our approach was only validated on the Greater Tokyo Area, the applicability and limitations for other urban areas should also be checked in the future work.

Chapter 5

Limited Data: Deep Embedding and Transffering

5.1 Introduction

Understanding and predicting citywide human mobility are considered as important problems for urban planning, traffic regulation and emergency management. Due to the continuing development of location acquisition technologies, massive GPS trajectory data are generated by sources such as mobile phones, car navigation systems, WLAN networks, and location-based social networks, and they provide the opportunity to solve the problem of human mobility prediction. Individual human trajectory prediction has been widely studied in recent years in the field of urban computing, but it is very difficult to expand such kind of individual modeling methodology to a citywide level. Since there are millions of people in a big city such as Tokyo, Shanghai, and Hong Kong, it is just infeasible to build a prediction model for each person by using his/her long historical data, which can also be an infringement on individual privacy. Moreover, crowd management under emergency situations is considered as a direct application scenario of human mobility prediction model. For this scenario, comparing with precisely mastering each individual's location, knowing and controlling the crowd density for any urban region is the real demand of governments (e.g. police) or public service operators (e.g. subway/bus companies, mobile service providers). Thus, in this study, our goal is to build one general model to effectively predict human mobility at a citywide level. Our problem is then defined as predicting the probability distribution for locations of a large

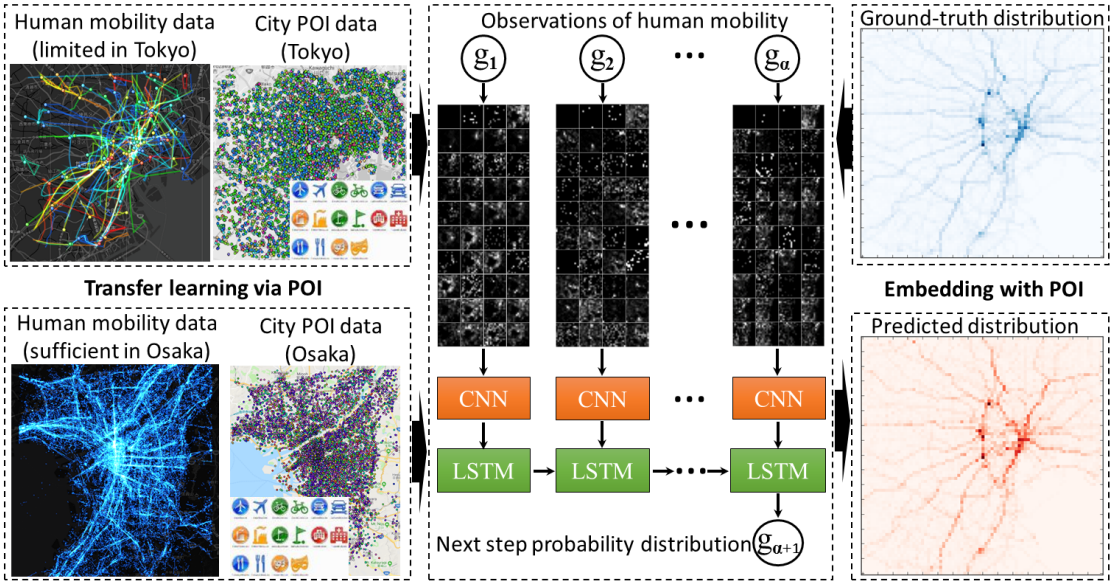


FIGURE 5.1: Can we design an effective approach to build one urban model for predicting human mobility (future distribution of individuals) at a citywide level with limited data? Fusing heterogeneous data (Human mobility data and city POI data) with deep learning technologies may allow us to address this challenge.

group of people at the next time step, which can meet the demands on crowd management. However, to implement such kind of prediction model, the following challenges need to be addressed. (1) Citywide human mobility is a highly nonlinear and complex phenomenon with multimodal distribution, and we can hardly achieve satisfactory prediction models for a large urban area with the classical methods. (2) The human mobility data used for model training can be limited to a small percentage (e.g. 1%~10%) of the total population, because it is impossible to collect every citizen's trajectory data for a large city. In our case, we have tried our best to collect up to 100,000 peoples' mobility data of Tokyo area, but it is difficult for us to collect any more data nor the total 9 million peoples' mobility data of Tokyo. To address these, we aim to design an approach to obtain a more effective representation of human mobility using heterogeneous data and advanced AI technologies especially the emerging deep learning technologies.

Obviously, human activities in city are closely linked with point-of-interest (POI) information, which can reflect the semantic meaning of human mobility. By combining human mobility data and city POI data, a more effective representation of human mobility can be expected. Moreover, although cities can have different types, scales, and developmental levels, the POI distrubtions similar with each other. For example, a business area often has more POIs (e.g. offices, shopping malls, and restaurants) and locates at central part of city, while a residential area comes in an opposite way. Human mobility in different cities generally follow the similar patterns. Taking commuting pattern

for example, people move from residential area to central business area to work and then return to residential district. This provides us the possibilities to transfer human mobility knowledge between cities via POI information. All these motivate us to fuse human mobility data and city POI data to improve the prediction performance especially when the amount of training data is small, but current fusion technologies can hardly handle these two heterogeneous data. Thus, in this study, we propose a deep sequential modeling architecture with a unique POI embedding mechanism for effectively predicting human mobility at the citywide level. In this study, an urban mesh-grid is extended to obtain an artificial POI image by aggregating the regional POIs by categories, where POI information is utilized as geographical features. Then each trajectory snippet is enriched to a four-dimensional tensor in an analogous manner to a short video. An LSTM-on-CNN architecture is designed to simultaneously capture both the spatiotemporal and geographical information from the enriched trajectories, where CNNs are utilized as advanced embedding layers to replace the standard word-like embedding for each mesh-grid to get better representations. The new embedding mechanism can work very well with transfer learning to transfer human mobility knowledge from one city to another, so that other cities' data can be fully utilized to train a stronger model for the target city with only limited data available. Finally, our learning model can achieve satisfactory prediction performance using a limited amount of trajectories as training data. A brief overview of this study has been summarized as Fig.5.1. *To the best of our knowledge, our approach is the first attempt to fuse big heterogeneous data to enhance the performance of citywide human mobility prediction, and our main contributions can be summarized as follows:*

- We constructed a standard deep sequence learning model for accurately predicting a probability distribution of human mobility at the citywide level.
- We proposed a novel sequential embedding method called image-like embedding that uses city POI data to enrich the original human mobility data with geographical features, where we applied CNNs to the standard model to obtain more effective representations.
- Transfer learning was employed to work together with image-like embedding mechanism. Through this, we can transfer mobility knowledge from source city to target city via POI information, if the source city have relatively sufficient mobility data and the target city only have limited data.

- We evaluated our approach based on multiple urban areas using different amounts of training data and demonstrated the advantages of our method compared with other baseline approaches.

The remainder of this paper is organized as follows. In Section 5.2, we introduce our data sources. In Section 5.3, we illustrate the modeling of citywide human mobility using a deep learning architecture. In Section 5.4, we explain the details of image-like embedding and transfer learning. We present the results of the experimental evaluation in Section 7.5, and discuss the results in Section 5.6. Related works are summarized in Section 6.2. In Section 6.8, we give our conclusions as well as explaining the limitations of our method and providing suggestions for future research.

5.2 Data Source

5.2.1 Human Mobility Data

"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 minutes, 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). It contains approximately 30 billion GPS records, and the total size of the data is more than 1.5 terabytes. Each record contains user ID, latitude, longitude, altitude, timestamp and positioning accuracy level (there are three levels due to different satellite's signal strength, correspondingly the positioning error would be within 100m, 200m or 300m).

5.2.2 City POI Data

In this study, we collected big POI data for every major city in Japan as geographical data by utilizing "Telepoint Pack DB February 2014" provided by ZENRIN DataCom Co., Ltd [1]. In the original database, each record is a registered land-line telephone number with coordinates (latitude, longitude) and industry category information included. We treated each "telepoint" as one specific POI. All the POIs were classified into 40 categories as listed in Table 5.1. The total numbers of POIs for Tokyo, Osaka, Fukuoka, Sapporo, Naha and Tottori were 281,400, 153,377, 47,418, 73,635, 35,014 and 17,743 respectively, which were used as the six target cities in our experiments. Furthermore, we used R-tree to index all of the POIs to speed up the range queries. Given an urban mesh, POIs can be retrieved for each mesh-grid by iteratively executing range query.

TABLE 5.1: POI Category Table

Fishery, Agriculture	Mining	Construction
Foods	Textiles, Apparels	Pulp, Paper
Chemicals	Oil, Coal Products	Rubber Products
Ceramics, Glass	Steel	Nonferrous Metals
Metal Products	Machinery	Electric Appliances
Transportation Equipment	Precision Instruments	Other Products
Commerce	Financial Insurance	Real Estate
Transportation(land)	Transportation(sea)	Transportation(air)
Warehousing	Communication	Electric Power, Gas
Technician Related	Sports Facilities	Sports Shop
Entertainment, Restaurant	Resort	Hospital
Large Retail Store	Lifestyle Related Store	Car Related
Education	Public Organization	Other
Dummy	–	–

5.3 Citywide Human Mobility Modeling

5.3.1 Preliminaries

Definition 1 (Human trajectory): The human trajectory collected for an individual person essentially comprises a 3-tuple sequence: (*timestamp*, *latitude*, *longitude*), which

[1] https://joras.csis.u-tokyo.ac.jp/dataset/show/id/14000201400

can indicate a person's location according to a captured timestamp. It can be further denoted as a sequence of (t, l)-pair by simplifying $timestamp$ as t and $(latitude, longitude)$ as l.

Our raw human trajectories were collected with a minimum sampling rate of about 5 minutes, but the record interval exceeds 5 minutes occasionally due to loss of signal or battery power. Besides, the positioning function would be suspended when no motion is detected, in this case no records will be uploaded. Thus, we fully conducted preprocessing to our raw human trajectory dataset in the following step: (1) Conducting data clearning and noise reduction to filter out low-quality trajectories or points. (2) Detecting stay points and conducting trajectory segmentation according to the stay points. After this, redundant points (continuous points located in the same position) will be filtered out. (3) Merge the trajectory segmentations of the same person within 24-hour (00:00~23:59) time interval as one human trajectory. Usually trajectory is mapped onto a mesh-grid or transportation network so that the trajectories can be handled as normal sequential data. In order to cover the entire urban area, we used grid-mapping to simplify the human trajectory as defined in the following.

Definition 2 (Grid-mapped human trajectory): Given a set of mesh-grids for an urban area $\{g_1, g_2, ..., g_K\}$ and a raw trajectory $\{(t_1, l_1), (t_2, l_2), ..., (t_m, l_m)\}$, a grid-mapped human trajectory $traj$ is defined as a sequence of mesh-grids:

$$traj = (t_1, g_1), (t_1, g_2), ..., (t_m, g_m), \forall i, l_i \in g_i. \tag{5.1}$$

A trajectory database TDB refers to a set of grid-mapped trajectories from a certain urban area.

In this study, we would like to focus on exploiting how to effectively predict the distribution of the next step location only based on previous locations from a spatial perspective, therefore only sequential information on the spatial axis were utilized. Then we can treat the raw trajectories as pure sequential data constituted by mesh-grids, and design an effective embedding mechanism for modeling the grid-mapped trajectories, which is the core problem of this study. Trajectory database TDB can be taken as a big corpus like a typical text database in the filed of natural language processing (NLP). Human mobility prediction problem is defined in an analogous manner to word/text modeling in the following.

Definition 3 (Human mobility prediction): Given a trajectory database TDB, we treat it as a big text corpus to generate partial trajectories. Specifically, when observation step α

is given, for each length-m $traj$ ($m > \alpha$), we can obtain (m-α) length-α trajectory snippets and their corresponding next-step trajectory snippets by setting the size of sliding window to 1. The length-α trajectory snippet is denoted as $x = g_1, g_2, ..., g_\alpha$, whereas the corresponding next-step trajectory snippet is represented as $y = g_{\alpha+1}$. One trajectory snippet x is essentially representing one pattern of urban human mobility within α observation steps, since the same mobiliy $g_1, g_2, ..., g_\alpha$ can be observed from a group of different people at different time peoriods. Then, the human mobility prediction for the next step can be modeled as follows:

$$P(y = g_{\alpha+1} \mid x = g_1, g_2, ..., g_\alpha). \tag{5.2}$$

The definition given above can be considered as a direct application of the n-gram language model, which is a typical probabilistic sequential model for predicting the next item in a sequence with the form of an (n-1)-order Markov model. Thus, N-gram is implemented as the first comparison model in our evaluation experiments (Section 7.5).

Definition 4 (Citywide human mobility prediction): Given all the human mobility data X with α steps of observations generated from TDB, citywide human mobility prediction for the next step basically involves obtaining a predicted probability distribution $P(\widehat{Y} \mid X)$, which should be as close as possible to the true probability distribution $Q(Y \mid X)$. Therefore, our goal is to obtain a model with the parameters θ that satisfies:

$$\theta = \underset{\theta}{\operatorname{argmin}}\, H(P(\widehat{Y} \mid X), Q(Y \mid X)), \tag{5.3}$$

where Y denotes the true next-step mobility, $\widehat{Y}$ denotes the predicted results, and $H(\cdot)$ represents the cross-entropy function, which is widely used to measure the divergence between two probability distributions. The lower the cross-entropy is, the two probability distributions have higher similarity. Thus, it is used as the loss function as well as the primary evaluation metric in our supervised learning models.

It should be noted that for one person's mobility prediction, we are concerned only about whether the model can precisely predict the next location with the highest probability. However, a large crowd of people can share the same observed trajectories (e.g. commuters taking the same train) but they may go to different places after some time. Thus, for citywide human mobility prediction, our model should precisely predict the overall probability distribution of the next possible destinations. For instance, as shown

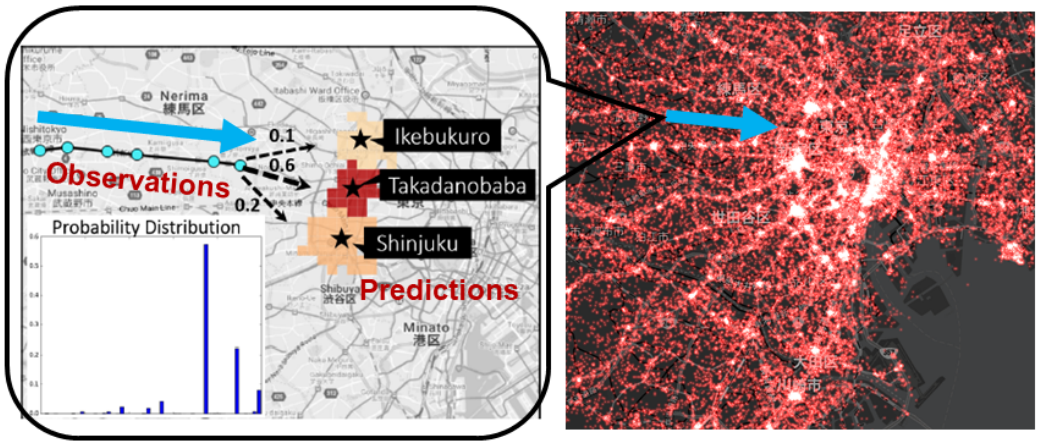


FIGURE 5.2: Citywide Human Mobility Prediction.

in Fig.5.2, we assume that 1,000 people on the same train have the same observed mobility, which is represented by a series of blue marks. We should precisely predict that around 600 people will go to Takadanobaba Station, 200 people to Shinjuku Station, and 100 people to Ikebukuro Station by obtaining the precise probability distribution (0.6, 0.2, and 0.1, respectively). With such model being deployed as an online service, we can precisely predict and simulate how many person will enter a certain region in real time, which can play an important role in controlling the crowd density for a city especially when some irregular events happen.

The trained model can generate or predict multiple steps of human mobility in an autoregressive manner. Multiple steps of mobility can be generated one step by one step according to the probability distribution in a similar way to a text generator. For example, given the first word "how", the second word can be generated as "are", then the third can be "you". If the second was generated as "old", then the next two words could be "are you" with higher probability. Moreover, if one step corresponds to 5 minutes time interval, generating next six steps of human mobility means that we can get a next-30-minutes mobility prediction. For instance, our model can take all of the 6-step observations from 07:35~08:00 as inputs and report the prediction result for 08:05~08:30 at 08:00. This can help us understand how the crowd dynamics are evolving step by step under a crowd management application scenario.

Currently, for simplicity, our prediction model with 1-step ahead is similar to a traditional n-gram language model as shown in Definition 3. We can further build a β-step ahead prediction model in a similar way to skip-gram language model[77], in which the

words need not be consecutive with the next step.

$$P(y = g_{\alpha+\beta} \mid x = g_1, g_2, ..., g_\alpha). \tag{5.4}$$

By modifying Definition 3 to the formula above, given α-step observations of human mobility, this model can directly predict or generate the human mobility at the next β step. If we only want to know the human mobility at a specific timestep in the future, this model would be helpful and effective. But if we want to obtain multi-step mobility predictor or generator, the original definition would be appropriate. Besides, in this study, we would like to focus more on designing a more powerful embedding mechanism for sequential mobility data. Therefore, this skip-gram-like mobility model will not be taken as our main target.

5.3.2 Deep Sequential Modeling Architecture

Citywide human mobility prediction is essentially defined to predict a probability distribution as shown by Definition 4. Since citywide human mobility data comprise highly complex and nonlinear sequential data, the overall probability distribution at next step is essentially a multimodal probability distribution, which is difficult to precisely predict using classical methods. Deep learning techniques such as long short-term memory (LSTM)-recurrent neural networks (RNNs) and gated recurrent unit (GRU)-RNNs [69, 78] are two improved RNNs that are highly successful at modeling highly complex sequential data such as text data and speech data. Specifically, they inherit the basic structure of the RNN but special computation blocks are introduced, i.e., LSTM and GRU, respectively, to replace the ordinary neurons in an RNN. These two architectures obtain similar performance in many deep learning tasks [79]. Hence, in this study, we used LSTM-RNN to implement a deep sequential model to predict the complex probability distribution using a limited amount of training data.

Word-Like Embedding for Grid. Word embedding is a state-of-the-art technique for many NLP tasks, where it is used to convert non-negative integers (i.e. word IDs) to a set of fixed-length dense and continuous-valued vectors. It has shown to boost the performance in NLP tasks such as syntactic parsing [80] and sentiment analysis [81]. One-hot embedding is the most naive embedding technique to map non-negative integers to vectors. However, the dimensionality of the vectors with one-hot embedding is equal to the size of the supported vocabulary, and these vectors are very huge and

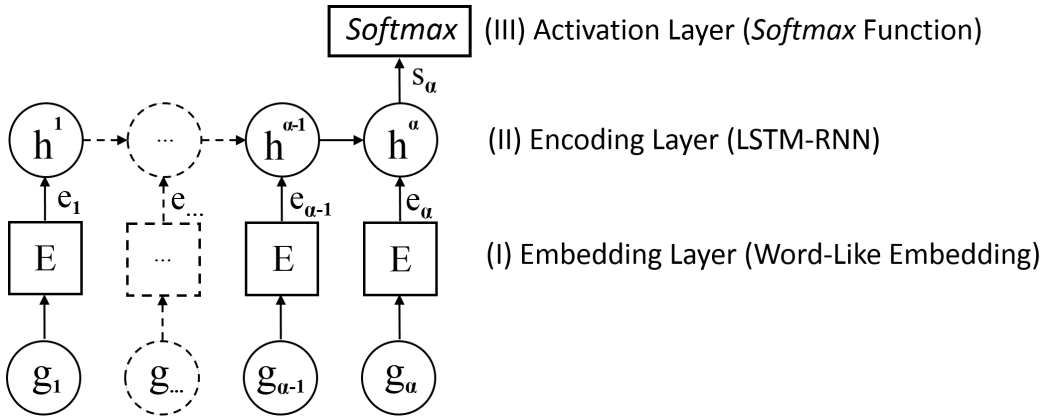


FIGURE 5.3: Deep Sequential Modeling Architecture.

sparse. Therefore, word embedding is employed in most natural language application scenarios to create a more efficient vector representation for each word. It has two huge advantages over one-hot embedding: (1) the representation vector is low-dimensional, far lower than the total size of vocabulary; and (2) the contextual similarity of words can be better captured, which means that if two words have similar semantic meanings, the two embedding vectors have high similarity. In our study, the model runs on grid-mapped trajectory data, it is natural to treat the entire urban mesh as the total vocabulary and each mesh-grid as a word. Each mesh-grid has a unique grid ID in the same way as word ID, which is called word-like embedding for grid. Note that comparing with a typical natural language model, it is more indispensable for our citywide human mobility prediction model to employ the state-of-the-art word-like embedding technique. Because the total number of mesh-grids for a big urban area can be larger than the total number of words in one language. Taking Tokyo area as an example, it is meshed with 6,400 500m×500m mesh-grids in our study, whereas there are just 2,500 to 3,000 most common words in English. Thus, we utilize word-like embedding instead of naive one-hot embedding as the basic technique to construct the mobility prediction model.

The RNN-based deep learning architecture with word-like embedding is constructed as shown in Fig.5.3, which operated according to the following steps: (1) the first layer is an embedding layer that changes an integer of grid id into a vector of continuous values by using an $K \times M$ embedding matrix, where M is the embedding dimension and K is the number of mesh-grid; (2) the second layer is an encoding layer constructed by the LSTM-RNN, where the *tanh* function is used to map the α steps of the embedded mobility $(e_1, e_2, ..., e_\alpha)$ into a single latent vector s_α, which can be taken as the auto-extracted features for the entire sequence; details about the calculation for LSTM are listed below; and (3) the third layer is an activation layer where the $Softmax$ function

is used to convert the latent vector s_α into probability values over K different possible mesh-grids $(g_1, g_2, ..., g_K)$. This architecture can be easily applied to different urban areas by modifying the embedding layer and the activation layer with the new mesh-grid number.

LSTM-RNN. An LSTM has three gates comprising an input gate i, an output gate o, and a forget gate f. Hidden state s_α in an LSTM is calculated iteratively from 1 to α for an input embedded mobility $(e_1, e_2, ..., e_\alpha)$ as follows:

$$i_\alpha = \sigma(W_i e_\alpha + U_i s_{\alpha-1} + b_i) \tag{5.5}$$

$$f_\alpha = \sigma(W_f e_\alpha + U_f s_{\alpha-1} + b_f) \tag{5.6}$$

$$o_\alpha = \sigma(W_o e_\alpha + U_o s_{\alpha-1} + b_o) \tag{5.7}$$

$$\widetilde{C_\alpha} = tanh(W_c e_\alpha + U_c s_{\alpha-1} + b_c) \tag{5.8}$$

$$C_\alpha = i_\alpha \odot \widetilde{C_\alpha} + f_\alpha \odot C_{\alpha-1} \tag{5.9}$$

$$s_\alpha = o_\alpha \odot tanh(C_\alpha), \tag{5.10}$$

where W and U are weight matrices, b is a bias vector, and $\odot$ represents elementwise multiplication. All of the model parameters are determined by applying the standard "backpropagation through time" algorithm, which starts by unfolding the RNN through time and it then generalizes the backpropagation for feed-forward networks to minimize the loss function namely cross-entropy, as defined in Equation 5.3.

5.4 Embedding Trajectory with POI

A standard deep learning model is proposed for modeling the mobility data described in the previous section, which shares most of the same techniques employed by the RNN-based deep natural language model. However, in addition to spatio-temporal information, human mobility in an urban area can also highly rely on geographical information, which can reflect the semantic meaning of human behavior. Thus, we consider to fuse human mobility data and city POI data to obtain a more powerful representation for mobility prediction. A novel embedding mechanism called image-like embedding with POI for grid is proposed to replace the naive word-like embedding for grid in the following.

5.4.1 Image-Like Embedding for Grid with POI

Definition 5 (Grid POI): Given a set of mesh-grids for an urban area and a set of POIs with σ categories, the POIs inside each mesh-grid g can be aggregated by category into a σ-dimension frequency vector as follows:

$$g.POI = (f_1, f_2, ..., f_\sigma), \forall i \in [1, \sigma],$$

$$f_i = |\{poi \mid poi.coordinate \in g \wedge poi.category = i\}| \tag{5.11}$$

where f represents the aggregated frequency based on each POI category. Each f is further scaled into [0,1].

Definition 6 (Grid Region): Given an $\eta \times \eta$ window and a mesh-grid g, we can obtain an $\eta \times \eta$ region r as follows:

$$r = \{g' \mid |g'.ix - g.ix| \leq \frac{\eta - 1}{2} \wedge |g'.iy - g.iy| \leq \frac{\eta - 1}{2}\} \tag{5.12}$$

where ix and iy denote the mesh-grid coordinates in the entire urban mesh. To make g the centroid, η is always set as odd in our method.

Definition 7 (POI Image): According to these definitions, each region can be treated as an image where each mesh-grid inside the region can be seen as a pixel. The category number σ corresponds to the σ channels. Therefore, a mesh-grid can be extended to obtain an artificial POI image represented by an $\eta \times \eta \times \sigma$ tensor.

This embedding mechanism has two advantages for mobility modeling: (1) geographical information is considered because of the POI information, and (2) in addition to the mesh-grid itself, the regional information around the mesh-grid is also taken into consideration by utilizing an $\eta \times \eta$ window. To effectively handle such kind of POI image, we use the-state-of-the-art convolutional neural network (CNN) to extract higher-level feature representation as the embedding vector.

CNN. Compared with traditional neural networks, CNNs were designed specifically for analyzing visual imagery [82], 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. To hierarchically capture the spatial structural information from a POI image, a classical CNN constructed using a convolutional layer and pooling layer is employed in our deep sequential learning architecture as an advanced embedding component. In

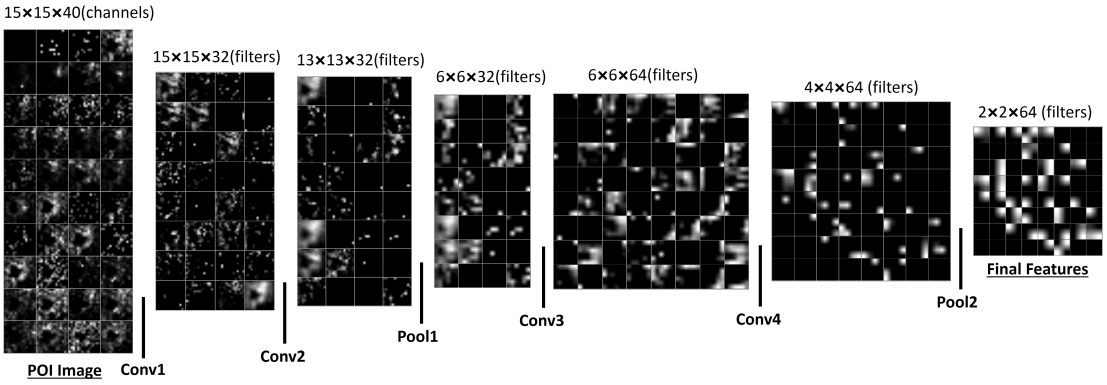


FIGURE 5.4: Visualization of the extracted features by embedding CNNs for a 15×15 POI image with the mesh-grid containing Tokyo Station as its centroid. The feature maps for each layer along the processing path are displayed in a block, where each channel or filter is plotted as a small subfigure (40 channels, 32 filters, and 64 filters are listed with sizes of 10×4, 8×4, and 8×8, respectively).

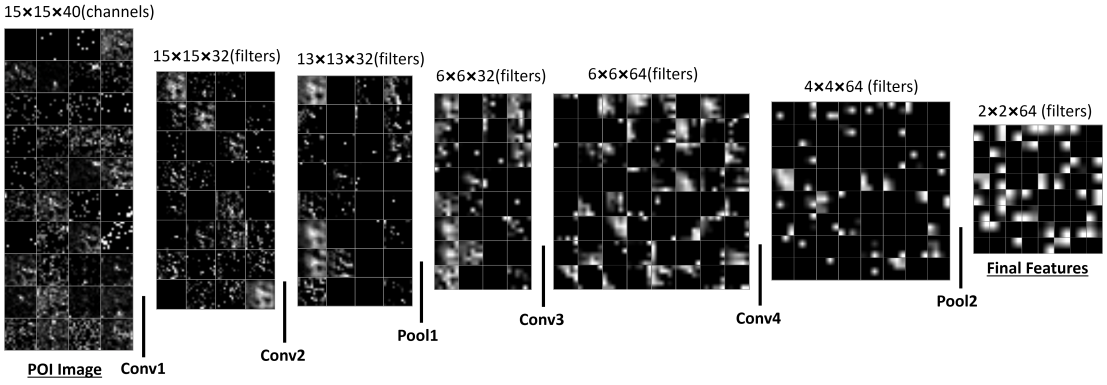


FIGURE 5.5: Visualization of the extracted features by embedding CNNs for a 15×15 POI image with the grid containing Shinjuku Station as its centroid. The feature maps for each layer along the processing path are displayed in a block, where each channel or filter is plotted as a small subfigure (40 channels, 32 filters, and 64 filters are listed with sizes of 10×4, 8×4, and 8×8, respectively).

our method, the convolutional feature value at pixel (i, j) in the k-th feature map, convolutional feature $cf_{i,j,k}$, is calculated as:

$$cf_{i,j,k} = ReLU(w_k pt_{i,j} + b_k) \tag{5.13}$$

where w_k and b_k are the weight and bias matrix of the k-th filter, respectively, and $pt_{i,j}$ is the input image patch centered at pixel (i, j). Kernel size (i.e. the size of the input image patch) needs to be specified for the convolutional operation. In our study, kernel size is set to 3×3, which is widely used in many state-of-the-art computer vision models. *ReLU* is used as the activation function. The pooling feature $pf_{i,j,k}$ is calculated using

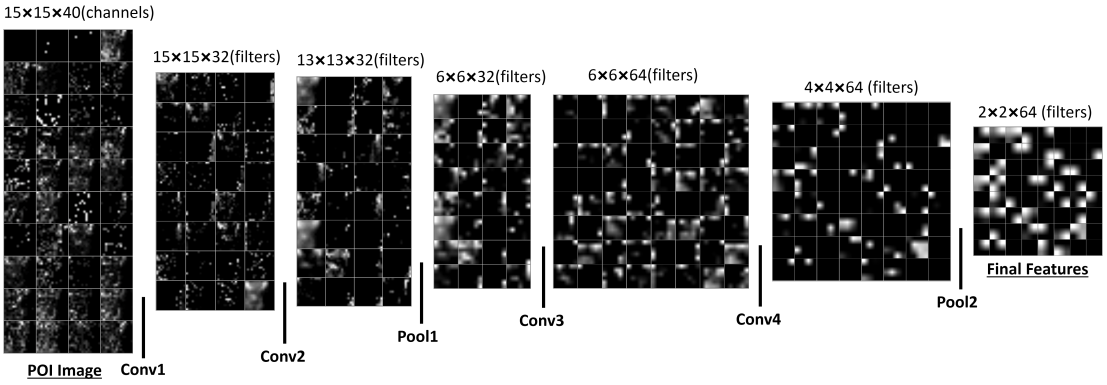


FIGURE 5.6: Visualization of the extracted features by embedding CNNs for a 15×15 POI image with the grid containing Shinagawa Station as its centroid. The feature maps for each layer along the processing path are displayed in a block, where each channel or filter is plotted as a small subfigure (40 channels, 32 filters, and 64 filters are listed with sizes of 10×4, 8×4, and 8×8, respectively).

the max-pooling operation:

$$pf_{i,j,k} = MaxPooling(cf_{m,n,k}), \forall(m,n) \in \mathbb{R}_{ij} \tag{5.14}$$

where $\mathbb{R}_{ij}$ is the local neighborhood around pixel (i, j). By stacking several convolutional and pooling layers (2 conv layers $\rightarrow$ 1 pool layer $\rightarrow$ 2 conv layers $\rightarrow$ 1 pool layer), we can gradually extract higher-level feature representations for a large grid region because one convolution can only capture nearby spatial dependencies. The input POI image around the mesh-grid of Tokyo Station, Shinjuku Station, and Shinagawa Station, as well as the features extracted by the CNNs step by step are visualized in Fig.5.4 ~ 5.6. To better illustrate how image-like embedding is performed, the flowchart has been drawn as Fig.5.7 by taking a grid-mapped human trajectory (Tokyo Station $\rightarrow$ Shinjuku Station $\rightarrow$ Shinagawa Station) as an example.

Originally, grid-mapped human trajectory is represented as a $\alpha \times 1$ vector. Now, with image-like embedding, an input human trajectory can be represented as an $\alpha \times \eta \times \eta \times \sigma$ tensor, which can be considered as an artificial video made from an α-frame $\eta \times \eta \times \sigma$ POI image. By replacing naive word-like embedding with CNN-based image-like embedding, our deep sequential learning architecture essentially becomes a deep video model, which takes a four-dimensional shape tensor as the input. As shown in Fig.**??**, the hierarchical geographical features inside a region are extracted by a CNN and fed into an LSTM layer for sequential prediction. The LSTM is stacked on CNNs (denoted as LSTM-on-CNNs) in combination to exploit both the geographical and sequential information related to citywide human mobility. It should be noted that we set the same

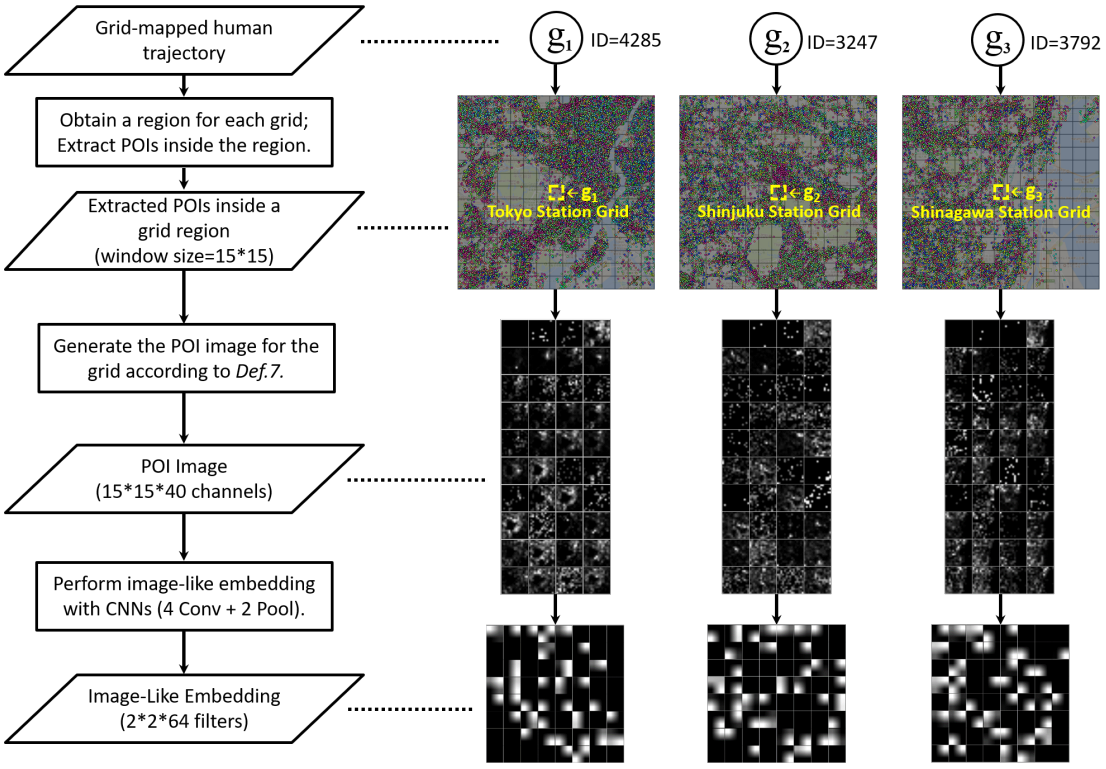


FIGURE 5.7: Flowchart of image-like embedding using a grid-mapped human trajectory g_1 (Tokyo Station) $\rightarrow$ g_2 (Shinjuku Station) $\rightarrow$ g_3 (Shinagawa Station) as an example. For each mesh-grid in the given trajectory, first a region with the mesh-grid as its centroid will be obtained according to Def.6, then the POIs inside the region will be extracted to generate the POI image according to Def.7. Through a series of CNNs, the extracted final features from the POI image can be seen as the embedding result.

CNN layers to be shared across each slice of the total α frames, where this sharing mechanism has several advantages, such as reducing the model complexity and making the network easier to train. The overall networks can still be trained using the standard backpropagation algorithm. Finally, the use of multiple stacked layers of RNNs can also be considered to boost the performance in difficult time-series modeling tasks according to [83]. LSTM-on-CNNs architecture has been proposed in the filed of computer vision for visual recognition and description [84]. However, in our approach, CNNs and LSTM are utilized as embedding component and encoding component separately for the human mobility modeling problem. In particular, a series of CNNs are utilized to replace the standard embedding matrix to generate more powerful embedding vectors for each step of the input human mobility.

5.4.2 Transfer Learning between Cities via POI

Transfer learning is a powerful tool that helps deep learning models to achieve better performance [85]. Citywide human mobility predictions for different cities are highly related tasks, which motivated us to transfer the mobility knowledge learned from one city to improve the learning process for another city. In particular, a mobility prediction model is unlikely to achieve satisfactory performance if sufficient trajectory data are not collected from one urban area. However, if sufficient human mobility data exist for another or more urban areas, we can exploit these large amounts of data from other areas by using transfer learning to boost the performance for the target area.

Image-like embedding with POI is assumed to have good natural compatibility with transfer learning between different cities because: (1) the POI distributions share some common properties between different cities, e.g., a central area often contains more POIs, including shopping malls and offices; and (2) human mobility in different cities generally follow similar patterns and comprise similar semantic meanings. Taking commuting pattern for example, people move from residential area to central business area to work and then return to residential district. All these provide us the possibilities to transfer human mobility knowledge between cities via POI information.

Moreover, the embedding matrix used for word-like embedding must be modified for each city according to the mesh-grid number of that city, which hinders transfer learning. Assuming that we have two cities A, B meshed with K_A and K_B mesh-grids, respectively ($K_A < K_B$), it is difficult to directly transfer the knowledge in A-model to city B because the integers in $[K_A, K_B)$ will not be well trained or embedded due to the lack of corresponding training data in A[1]. Meanwhile, the CNN architectures for image-like embedding can remain the same for different cities. These advantages of image-like embedding over word-like embedding will be further validated and discussed in Section 5.6.

We tested two different transfer methods for our problem: (1) freezing the LSTM-on-CNNs trained from the source city and training a completely new $Softmax$ activation layer for the target city; (2) using the LSTM-on-CNNs as a pre-trained model, connecting it with a new $Softmax$ activation layer, and training the overall networks again with new dataset. Experiments showed that the latter obtained better performance.

[1]The index for mesh-grid starts from 0 in this study.

5.5 Experiment

Table 5.2: Geographic Details of Six Experimental Cities

City	Geographic Interval	Number of Grids
Tokyo	$Long. \in [139.50, 139.90]$, $Lat. \in [35.50, 35.82]$	6,400
Osaka	$Long. \in [135.35, 135.65]$, $Lat. \in [34.58, 34.82]$	3,600
Fukuoka	$Long. \in [130.20, 130.50]$, $Lat. \in [33.46, 33.70]$	3,600
Sapporo	$Long. \in [141.22, 141.47]$, $Lat. \in [43.00, 43.16]$	2,000
Naha	$Long. \in [127.64, 127.74]$, $Lat. \in [26.17, 26.25]$	400
Tottori	$Long. \in [134.12, 134.32]$, $Lat. \in [35.44, 35.56]$	1,200

Table 5.3: Data Information of Six Experimental Cities

City	Number of Trajectory	Average Trajectory Length	Number of Samples (x,y pairs)
Tokyo	33,261	85.0	2,658,077
Osaka	22,182	76.5	1,584,579
Fukuoka	26,425	69.3	1,697,885
Sapporo	24,727	67.7	1,545,297
Naha	15,304	55.1	766,085
Tottori	7,268	58.2	386,751

Table 5.4: Parameter Description Table

Parameter	Relevant Component	Tuned Value
$\Delta Long.$ $\Delta Lat.$	Mesh Size	$\Delta Long.$=0.005, $\Delta Lat.$=0.004
α	Observation Step	5
σ	POI Categories	40
η	Window Size of Grid Region	15
Embedding Dimension	Word-Like & Image-Like Embedding	256
Encoding Dimension	LSTM-RNN Encoding	256
Output Dimension (K)	Softmax Activation	Tokyo:6,400 (etc. in Table 6.2)
Learning Rate	RMSprop Optimizer	0.001
Batch Size	Training Process	1024

Experimental Setup: We randomly selected two consecutive weeks as our experimental period and conducted evaluations in six cities of Japan. As we all know, Japan is a stratovolcanic archipelago consisting of about 6,852 islands. The main islands, from north to south, are Hokkaido, Honshu, Shikoku and Kyushu. The Ryukyu Islands, which include Okinawa, are a chain to the south of Kyushu[1]. Tokyo and Osaka, as the two biggest cities of Japan, were chosen as the representatives of Honshu. Fukuoka and Sapporo were included as the representatives of Kyushu and Hokkaido respectively. Naha was selected to represent Okinawa, which is very far from the main

[1]https://en.wikipedia.org/wiki/Japan

islands. By utilizing these five cities, we would like to verify our proposed framework could be effective among multiple isolated areas. Additionally, Tottori City was selected to verify the effectiveness of our framework in rural areas from Tottori Prefecture of Honshu, which is the least populous prefecture in Japan[2]. More geographical details about these six areas are summarized as Table 6.2. Python and some Python libraries such as Keras[72] and TensorFlow[73] were used in this study. The experiments were performed on a GPU server with a GeForce GTX 1080Ti graphics card installed.

Parameter Settings: We treated the 24-hour (00:00~23:59) GPS log of each individual person as one trajectory, and after pre-processing (e.g., data cleaning, noise reduction, etc.). 33,261, 22,182, 26,425, 24,727, 15,304, and 7,268 trajectories were generated for Tokyo, Osaka, Fukuoka, Sapporo, Naha, and Tottori respectively, where the average trajectory lengths (number of points) were approximately 85.0, 76.5, 69.3, 67.7, 55.1, and 58.2. We set the observation step α as five to obtain length-5 trajectory snippets as inputs and their corresponding next locations as outputs. Total number of generated snippet samples for each city was summarized and listed in Table 5.3. We randomly selected 60% of the data as the training dataset, 20% of the data as the validation dataset, and the remaining 20% as the testing dataset for every city. The mesh size was set to $\Delta Long.$=0.005, $\Delta Lat.$=0.004 (approximately $450m \times 450m$) for each city. Finally, 6,400, 3,600, and 3,600 mesh-grids were generated for the Tokyo area, Osaka area, and Fukuoka area. 2,000, 400, and 1,200 mesh-grids were generated the Sapporo area, Naha area, and Tottori area. So the Softmax activation layer output the probability distribution over the corresponding number of mesh-grids for each city. The RMSprop algorithm was employed to control the overall training process, where the batch size was set to 1024 and the learning rate to 0.001. The training algorithm was stopped early if the loss stopped decreasing based on the validation dataset for five consecutive epochs. All of the learning settings were kept the same for each model and each city.

Baseline models: We considered the following models as baseline models for comparison.

(1) N-Gram. N-Gram is a widely used algorithm for modeling sequential data, especially for text and speech data. Tri-Gram was found to be the most appropriate for our problem.

[2]https://en.wikipedia.org/wiki/Tottori_Prefecture

(2) KNN. A KNN-based learning model [74] is a type of instance-based learning where classification is computed from a simple majority vote of the nearest neighbors of each point.

(3) DecisionTree. A decision tree [75] is built to predict the target value by learning simple decision rules.

(4) RandomForest. A random forest [76] is constructed with a multitude of decision trees to gain better performance. For (2)~(4), these classical methodologies are extended to output the probability distribution over mesh-grids. One-hot encoding was utilized to encode the K mesh-grids for each city, then the grid-mapped trajectories with α steps were converted to $\alpha \times K$-dimension vectors as the final input features.

(5) Word-Like Embedding. This is the deep learning model shown in Fig.5.3 with a typical word-like embedding. An embedding layer, which is essentially an embedding matrix, embedded each grid id into a continuous 256-dimensional vector space. The subsequent LSTM-RNN layer shown in Fig.5.3 also contained 256 hidden units.

(6) Image-Like Embedding. This is our proposed image-like embedding model without transfer learning. We set the window size to 15 and each trajectory snippet was expanded to a video where each frame comprised a 15×15 POI image. A six-layer CNN was utilized in this model, where the first two convolutional layers used 32 filters of 3×3 and the third layer was a 2×2 max-pooling layer. The subsequent two convolutional layers used 64 filters of 3×3 and the sixth layer was a 2×2 max-pooling layer. Using these settings, each mesh-grid could be embedded into a 256-dimensional vector and an LSTM-RNN layer with 256 hidden units followed in the same manner.

(7) Word-Like Embedding+Transfer. This model applied transfer learning to baseline method (5) mentioned above. We assumed that a limited amount of data could be retrieved from one urban area, whereas a large amount of data could be obtained from another urban area. The embedding layer and the encoding layer in method (5) were pre-trained with the sufficient data from other urban area and they were then continuously trained with the limited data from the target area. Specifically, "Transfer^{O}" denotes performing transfer learning based on Osaka model, which is pre-trained with the whole training dataset of the Osaka area (60% of all). And "Transfer^{T}" denotes performing transfer learning based on Tokyo model, which

is pre-trained using the whole training data of the Tokyo area (60% of all). The network settings were kept the same as those in method (5).

Our proposed model is denoted as **Image-Like Embedding+Transfer**. "Transfer^{O}" and "Transfer^{T}" follows the same meaning as mentioned above. The network settings were kept the same as those in method (6). The embedding layer and the encoding layer were pre-trained with the same transfer learning settings mentioned in (7). All the parameter settings of the experiments are summarized as Table 6.3.

TABLE 5.5: Performance Evaluation of Citywide Human Mobility Prediction for Tokyo

Model (Tokyo 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	7.06	42.55%	4.43	57.29%
KNN	7.07	29.84%	3.54	44.17%
DecisionTree	7.28	9.50%	4.11	37.32%
RandomForest	5.07	30.79%	2.68	49.38%
Word-Like Embedding	4.52	27.54%	2.21	51.19%
Image-Like Embedding	3.10	36.21%	1.73	54.96%
Word-Like Embedding+Transfer^{O}	4.27	34.71%	2.18	54.67%
Image-Like Embedding+Transfer^{O}	**2.75**	**45.09%**	**1.65**	**57.98%**

TABLE 5.6: Performance Evaluation of Citywide Human Mobility Prediction for Tokyo

Model (Tokyo 2/2)	10% data		50% data	
	Loss	Acc	Loss	Acc
N-Gram	3.42	60.82%	1.91	64.64%
KNN	2.81	46.43%	1.95	56.39%
DecisionTree	3.02	49.79%	1.74	58.93%
RandomForest	2.37	48.94%	1.77	52.74%
Word-Like Embedding	1.73	59.14%	**1.23**	**67.38%**
Image-Like Embedding	1.53	59.67%	1.25	65.76%
Word-Like Embedding+Transfer^{O}	1.75	60.48%	1.28	67.35%
Image-Like Embedding+Transfer^{O}	**1.51**	**60.89%**	1.28	65.75%

Evaluation metric: We evaluated the performance of the proposed models using two metrics. Cross-entropy (denoted as Loss) was used as the loss function defined in Definition 4, where it describes the predicted loss between the ground-truth and the prediction. Predicting a spatial probability distribution of next step in a large urban area is the goal of our study. Thus, it was used as the primary metric in the evaluation. Furthermore, accuracy (denoted as Acc) was used as the secondary metric, and it is more suitable to evaluate one person's mobility prediction rather than our human mobility prediction task. These two metrics are defined as follows:

Table 5.7: Performance Evaluation of Citywide Human Mobility Prediction for Osaka

Model (Osaka 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	7.07	42.04%	4.51	57.47%
KNN	7.19	22.66%	3.61	44.40%
DecisionTree	7.19	19.78%	4.06	50.31%
RandomForest	4.90	30.44%	2.60	51.21%
Word-Like Embedding	4.49	25.67%	2.08	52.59%
Image-Like Embedding	2.85	37.26%	1.63	56.22%
Word-Like Embedding+Transfer^{T}	3.96	39.02%	2.03	56.79%
Image-Like Embedding+Transfer^{T}	**2.62**	**45.28%**	**1.56**	**58.86%**

Table 5.8: Performance Evaluation of Citywide Human Mobility Prediction for Osaka

Model (Osaka 2/2)	10% data		50% data	
	Loss	Acc	Loss	Acc
N-Gram	3.52	60.95%	1.97	64.75%
KNN	2.87	46.64%	2.03	56.87%
DecisionTree	2.96	51.28%	1.73	60.04%
RandomForest	2.30	50.48%	1.75	54.50%
Word-Like Embedding	1.63	60.25%	**1.19**	**68.02%**
Image-Like Embedding	1.45	60.87%	1.21	66.93%
Word-Like Embedding+Transfer^{T}	1.65	61.98%	1.23	68.25%
Image-Like Embedding+Transfer^{T}	**1.40**	**62.75%**	1.21	67.23%

Table 5.9: Performance Evaluation of Citywide Human Mobility Prediction for Fukuoka

Model (Fukuoka 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	6.47	47.65%	3.78	58.07%
KNN	5.28	33.02%	3.30	47.26%
DecisionTree	5.63	35.20%	3.44	55.56%
RandomForest	3.49	42.12%	2.23	51.69%
Word-Like Embedding	3.04	37.75%	1.66	58.36%
Image-Like Embedding	2.20	45.85%	1.55	58.87%
Word-Like Embedding+Transfer^{T}	2.93	45.21%	1.73	58.72%
Image-Like Embedding+Transfer^{T}	**2.10**	**50.30%**	**1.50**	**60.28%**

$$Loss = \frac{1}{n}\sum_{i}^{n}\sum_{k}^{K} -y_i^{(k)} \log\left(\hat{y}_i^{(k)}\right) \tag{5.15}$$

$$Acc = \frac{1}{n}\sum_{i}^{n} 1(y_i = \hat{y}_i) \tag{5.16}$$

where n is the number of samples, K is the mesh-grid number for each urban area,

TABLE 5.10: Performance Evaluation of Citywide Human Mobility Prediction for Fukuoka

Model (Fukuoka 2/2)	10% data		50% data	
	Loss	Acc	Loss	Acc
N-Gram	2.91	60.29%	1.77	62.65%
KNN	2.66	48.45%	2.10	56.89%
DecisionTree	2.52	56.66%	1.66	61.31%
RandomForest	2.00	51.20%	1.70	52.85%
Word-Like Embedding	1.44	**62.23%**	**1.21**	**66.15%**
Image-Like Embedding	1.43	61.57%	1.25	65.14%
Word-Like Embedding+Transfer^{T}	1.49	**62.23%**	1.23	66.09%
Image-Like Embedding+Transfer^{T}	**1.39**	62.17%	1.24	65.20%

TABLE 5.11: Performance Evaluation of Citywide Human Mobility Prediction for Sapporo

Model (Sapporo 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	6.37	40.45%	3.16	51.01%
KNN	4.42	29.70%	2.53	42.45%
DecisionTree	5.39	20.97%	2.53	49.55%
RandomForest	3.88	31.98%	2.00	47.45%
Word-Like Embedding	2.78	39.36%	1.50	57.25%
Image-Like Embedding	2.09	43.06%	1.47	55.17%
Word-Like Embedding+Transfer^{T}	2.56	45.26%	1.53	**58.53%**
Image-Like Embedding+Transfer^{T}	**1.95**	**48.28%**	**1.39**	58.06%

TABLE 5.12: Performance Evaluation of Citywide Human Mobility Prediction for Sapporo

Model (Sapporo 2/2)	10% data		50% data	
	Loss	Acc	Loss	Acc
N-Gram	2.40	52.89%	1.58	54.77%
KNN	2.20	47.23%	1.70	56.78%
DecisionTree	2.08	55.89%	1.62	65.01%
RandomForest	1.75	49.25%	1.52	51.45%
Word-Like Embedding	1.30	61.45%	**1.09**	65.72%
Image-Like Embedding	1.33	58.30%	1.14	63.94%
Word-Like Embedding+Transfer^{T}	1.32	**61.85%**	1.11	**65.77%**
Image-Like Embedding+Transfer^{T}	**1.28**	60.61%	1.14	64.06%

$y^{(k)}$ and $\hat{y}^{(k)}$ are the true probability and predicted probability based on each mesh-grid, respectively, z is the true grid id, and $\hat{y}$ is the predicted grid id. Here, the grid id with highest probability will generated as the predicted result, which essentially makes the Acc metric equal to $Rank@Top1$. We may further utilize $Rank@TopK$ as the metric by setting different $TopK$ in the future work, which is widely used in recommendation

TABLE 5.13: Performance Evaluation of Citywide Human Mobility Prediction for Naha

Model (Naha 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	4.61	46.92%	2.18	52.99%
KNN	2.91	35.95%	2.12	45.07%
DecisionTree	3.21	38.02%	2.06	54.51%
RandomForest	2.37	44.64%	1.63	51.06%
Word-Like Embedding	1.89	47.42%	1.28	**59.03%**
Image-Like Embedding	1.64	50.47%	1.33	55.91%
Word-Like Embedding+Transfer^{T}	2.11	48.23%	1.35	58.85%
Image-Like Embedding+Transfer^{T}	**1.62**	**51.81%**	**1.27**	58.71%

TABLE 5.14: Performance Evaluation of Citywide Human Mobility Prediction for Naha

Model (Naha 2/2)	10% data		50% data	
	Loss	Acc	Loss	Acc
N-Gram	1.80	54.18%	1.38	55.02%
KNN	1.90	49.40%	1.52	57.53%
DecisionTree	1.86	58.91%	1.50	**64.30%**
RandomForest	1.51	52.41%	1.38	54.98%
Word-Like Embedding	**1.17**	**61.45%**	**1.04**	64.15%
Image-Like Embedding	1.22	59.39%	1.08	63.24%
Word-Like Embedding+Transfer^{T}	1.22	61.15%	1.05	64.00%
Image-Like Embedding+Transfer^{T}	1.20	60.37%	1.08	63.16%

TABLE 5.15: Performance Evaluation of Citywide Human Mobility Prediction for Tottori

Model (Tottori 1/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	5.66	44.07%	2.75	53.45%
KNN	4.30	29.41%	2.54	41.90%
DecisionTree	4.59	25.05%	2.54	48.95%
RandomForest	3.85	32.14%	2.03	48.72%
Word-Like Embedding	2.86	39.59%	1.45	59.57%
Image-Like Embedding	2.17	46.94%	1.36	57.93%
Word-Like Embedding+Transfer^{T}	2.73	44.84%	1.50	59.70%
Image-Like Embedding+Transfer^{T}	**2.07**	**48.86%**	**1.31**	**60.64%**

system.

Entropy is a widely used measurement for comparing distributions. However, we also aim to evaluate our results in a more intuitive way. The ground-truth probability can be represented more intuitively through the density of the population. Therefore, two additional metrics regarding density were employed in the supplementary evaluation to check the ground-truth density and the predicted density for selected areas in the city.

Table 5.16: Performance Evaluation of Citywide Human Mobility Prediction for Tottori

Model (Tottori 2/2)	1% data		5% data	
	Loss	Acc	Loss	Acc
N-Gram	2.12	55.20%	1.41	57.18%
KNN	2.09	48.43%	1.52	58.38%
DecisionTree	2.09	56.85%	1.49	65.90%
RandomForest	1.72	52.89%	1.43	57.78%
Word-Like Embedding	1.22	62.95%	**0.99**	**67.02%**
Image-Like Embedding	1.22	60.64%	1.02	65.81%
Word-Like Embedding+Transfer^{T}	1.26	**63.13%**	1.01	66.88%
Image-Like Embedding+Transfer^{T}	**1.17**	62.73%	1.01	66.29%

Table 5.17: Evaluation of Human Mobility Density for Shinjuku Station Area (Tokyo)

Model (Tokyo 1/2)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	-1215	8.94%	-386	2.84%
KNN	2340	17.22%	1561	11.49%
DecisionTree	-1869	13.75%	-111	0.82%
RandomForest	1688	12.42%	891	6.56%
Word-Like Embedding	1589	11.69%	-208	1.53%
Image-Like Embedding	1343	9.88%	-61	0.45%
Word-Like Embedding+Transfer^{O}	2251	16.56%	244	1.80%
Image-Like Embedding+Transfer^{O}	**1025**	**7.54%**	**-5**	**0.04%**

Table 5.18: Evaluation of Human Mobility Density for Shinjuku Station Area (Tokyo)

Model (Tokyo 2/2)	10% data		50% data	
	Err	RE	Err	RE
N-Gram	-320	2.35%	**-45**	**0.33%**
KNN	1015	7.47%	328	2.41%
DecisionTree	-210	1.55%	51	0.38%
RandomForest	1133	8.34%	614	4.52%
Word-Like Embedding	-110	0.81%	50	0.37%
Image-Like Embedding	-91	0.67%	78	0.57%
Word-Like Embedding+Transfer^{O}	122	0.90%	57	0.42%
Image-Like Embedding+Transfer^{O}	**-57**	**0.42%**	144	1.06%

With these metrics, it will be easier for us to understand if the system is overshooting or undershooting the numbers/densities. They are defined as follows:

$$Err = \hat{d}_{area} - d_{area} \tag{5.17}$$

$$RE = \frac{|\hat{d}_{area} - d_{area}|}{d_{area}}, \tag{5.18}$$

TABLE 5.19: Evaluation of Human Mobility Density for Tokyo University Area (Tokyo)

Model (Tokyo 1/2)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	-636	13.00%	-183	3.74%
KNN	69	1.41%	-129	2.64%
DecisionTree	-2833	57.91%	-469	9.59%
RandomForest	232	4.74%	207	4.23%
Word-Like Embedding	942	19.26%	-95	1.94%
Image-Like Embedding	495	10.12%	-137	2.80%
Word-Like Embedding+Transfer^{O}	761	15.56%	171	3.50%
Image-Like Embedding+Transfer^{O}	**44**	**0.90%**	**64**	**1.31%**

TABLE 5.20: Evaluation of Human Mobility Density for Tokyo University Area (Tokyo)

Model (Tokyo 2/2)	10% data		50% data	
	Err	RE	Err	RE
N-Gram	-191	3.90%	**55**	**1.12%**
KNN	-222	4.54%	-247	5.05%
DecisionTree	-30	0.61%	-173	3.54%
RandomForest	88	1.80%	188	3.84%
Word-Like Embedding	127	2.60%	-79	1.61%
Image-Like Embedding	**9**	**0.18%**	-186	3.80%
Word-Like Embedding+Transfer^{O}	53	1.08%	-123	2.51%
Image-Like Embedding+Transfer^{O}	-288	5.89%	-85	1.74%

TABLE 5.21: Evaluation of Human Mobility Density for Odori Park Area (Sapporo)

Model (Sapporo 1/2)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	-1925	8.89%	-769	3.55%
KNN	179	0.83%	1331	6.14%
DecisionTree	-1677	7.74%	606	2.80%
RandomForest	1044	4.82%	839	3.87%
Word-Like Embedding	-495	2.29%	94	0.43%
Image-Like Embedding	726	3.35%	-93	0.43%
Word-Like Embedding+Transfer^{T}	570	2.63%	530	2.45%
Image-Like Embedding+Transfer^{T}	**-128**	**0.59%**	**59**	**0.27%**

where d_{area} and $\hat{d}_{area}$ are the true density and predicted density on a selected *area*. *Err* represents the prediction error, positive number (+) means overshooting and negative number means undershooting (-). *RE* represents the relative prediction error in percentage. By iteratively checking each sample in $\hat{Y}$ and Y, we can get the ground-truth density d_g and the prediction density $\hat{d}_g$ for each mesh-grid. The density of a selected *area* can be calculated by adding up the densities of the mesh-grids belonging to the *area*.

TABLE 5.22: Evaluation of Human Mobility Density for Odori Park Area (Sapporo)

Model (Sapporo 2/2)	10% data		50% data	
	Err	RE	Err	RE
N-Gram	1582	7.30%	965	4.46%
KNN	864	3.99%	459	2.12%
DecisionTree	**92**	**0.42%**	**330**	**1.52%**
RandomForest	762	3.52%	558	2.58%
Word-Like Embedding	996	4.60%	391	1.81%
Image-Like Embedding	980	4.52%	751	3.47%
Word-Like Embedding+Transfer^{T}	677	3.13%	622	2.87%
Image-Like Embedding+Transfer^{T}	926	4.27%	516	2.38%

Overall performance: We compared the performances of the baselines and our proposed model using different amounts of training data. The overall evaluation results are summarized in Table 5.6~5.16, which shows that based on all six cities: (1) our model performed better than the others on both metrics when the amount of training data was small to 1%; (2) when training data were increased to 5% and 10%, our model outperformed other models in Tokyo and Osaka on both metrics, and also kept an advantage in Fukuoka, Sapporo and Tottori on the primary metric; when using 10% training data, word-like embedding worked best for Naha; (3) when 50% of the data were used as training data, deep-learning models using word-like embedding performed better than the other models. In general, our proposed methodology had a clearer advantage for larger urban areas (Tokyo and Osaka) when using less training dataset (1%~10%). Furthermore, we found that even without transfer learning, image-like embedding still performed better than word-like embedding with small training datasets, where the advantage increased as the amount of training data became smaller. Finally, as shown above, we achieved relatively good performance with only 10% of the data by using **Image-Like Embedding+Transfer**.

Using the metrics *Err* and *RE*, we conducted a supplementary evaluation to compare the ground-truth density and predicted density on three selected areas, namely Shinjuku Station area (Tokyo), Tokyo University area (Tokyo), and Odori Park area (Sapporo). Each area consists of 5×5 neighboring mesh-grids, with Shinjuku Station, Tokyo University, and Odori Park locating at the central mesh-grid respectively. Shinjuku Station area can be seen as a typical central business area, whereas Tokyo University area (Tokyo) as well as Odori Park area (Sapporo) can be seen as areas with special functions and less population density than central area. The evaluation results are listed as Table

5.18, 5.20, and 5.22, from which we can also see that our proposed framework **Image-Like Embedding+Transfer** could achieve the best performance comparing other baselines if only using 1%~ 5% of training data.

Transfer learning performance: We also verified the performance of transfer learning by checking the learning curve obtained by each deep-learning model based on different amounts of training data. The verification results for the Tokyo area, Osaka area and Fukuoka area are presented in Fig.5.8. Through it, we could clearly see transfer learning brought the original models with lower start loss, lower final loss, and higher learning slope, especially with small datasets. Thus, a positive transfer between different urban areas was verified for human mobility predictions. Especially when the training data are limited to 1%, the advantages of **Image-Like Embedding+Transfer** become much more noticeable.

5.6 Discussion

Here we further discuss the advantages of image-like embedding over word-like embedding as follows:

TABLE 5.23: Comparison of Word-Like Embedding and Image-Like Embedding on Capturing The Similarity between Cities

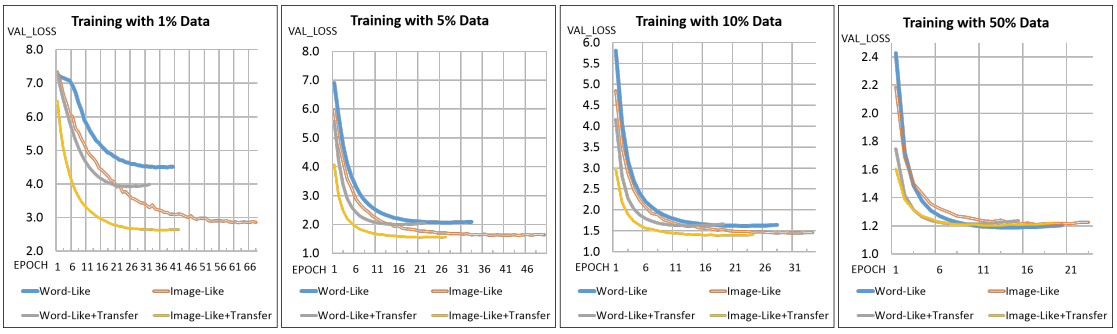
Cosine Similarity	Word-Like Em.	**Image-Like Em.**
Tokyo Station vs Osaka Station	0.004	**0.177**
Shinjuku Station vs Nanba Station	0.118	**0.224**
Tokyo Disneyland vs Universal Studios Japan	0.120	**0.273**

TABLE 5.24: Basic Statistics of Embedding Vectors for "Tokyo Station → Shinjuku Station → Tokyo Disneyland Station"

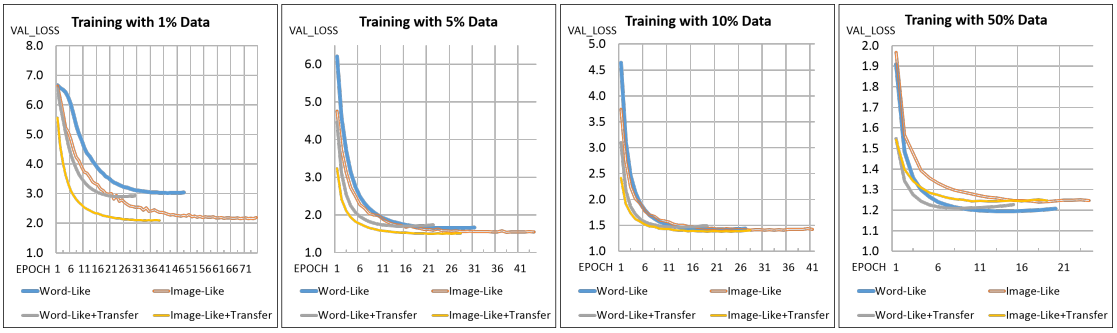
	Word-Like Embedding				Image-Like Embedding			
	Min	Max	Mean	Std	Min	Max	Mean	Std
Tokyo Station	-1.27	1.18	0.05	0.47	0.00	27.00	0.87	2.68
Shinjuku Station	-0.83	0.84	0.06	0.37	0.00	8.19	0.44	1.17
Tokyo Disneyland	-1.00	0.84	0.02	0.28	0.00	3.22	0.32	0.66

(1) Based on public recognition, corresponding relationships between some places from two different cities can be set up. Taking Tokyo and Osaka as example, Tokyo Station can correspond to Osaka Station, because both of them can be seen as the city

(a) Learning Curves of Transfer Learning (Tokyo Model)



(b) Learning Curves of Transfer Learning (Osaka Model)



(c) Learning Curves of Transfer Learning (Fukuoka Model)

FIGURE 5.8: Learning Curves of Transfer Learning on Three Cities.

center. Shinjuku Station of Tokyo can correspond to Nanba Station of Osaka, because they both can be seen as the sub city center. Similar relationship can also be established between Tokyo Disneyland and Universal Studios Japan, which are the two most famous theme parks in Japan located in Tokyo and Osaka respectively. A good embedding method should be capable of capturing the similarity between those corresponding places of two cities. Therefore, we trained two Tokyo models using the whole training dataset (60% of the data) based on the two different embedding methodologies. Two Osaka models were built and trained in the same way. Transfer learning was not applied between the cities and other settings were kept the same as described in Section 7.5. The cosine similarity between those corresponding places of two cities was calculated and summarized as Table 5.23, from

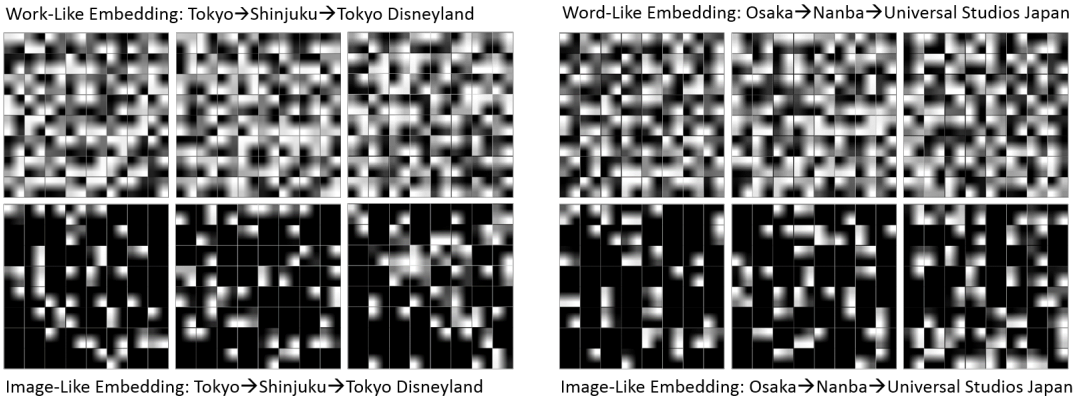


FIGURE 5.9: Visualization of word-like embedding and image-like embedding for "Tokyo Station → Shinjuku Station → Tokyo Disneyland Station" and "Osaka Station → Nanba Station → Universal Studios Japan". Word-like embedding results are listed on the top, and the image-like embedding results are listed at the bottom. The 256-dimension vector of word-like embedding is reshaped to a 2×2×64 tensor so that it can be visualized in a similar way with image-like embedding.

which we can see that image-like embedding could better capture the similarity. The reason behind this is that POI distributions in different cities share some similarity to some degree. The word-like embedding and image-like embedding results of those corresponding places of Tokyo and Osaka were visualized as Fig.5.9. And some basic statistic values are listed as Table 5.24 to further elaborate the difference between word-like embedding and image-like embedding. Through Table 5.24, we could see the standard deviation of image-like embedding is larger than word-like embedding.

TABLE 5.25: Verification of ID Problem of Word-Like Embedding

Osaka	Before Training		After Training		Difference	
Model	Min	Max	Min	Max	L1-norm	L2-norm
ID1=6113 ∈ [3600, 6400)	-0.05	0.05	-0.05	0.05	8.62	0.66
ID2=6399 ∈ [3600, 6400)	-0.05	0.05	-0.05	0.05	8.86	0.68
ID3=1770 ∈ [0, 3600)	-0.05	0.05	**-0.88**	**0.80**	**78.08**	**5.83**
ID4=1821 ∈ [0, 3600)	-0.05	0.05	**-0.85**	**0.77**	**67.14**	**5.14**

(2) As mentioned in Section 5.4, word-like embedding will not work well with transfer learning due to the different numbers of mesh-grids for different cities (e.g. 6400 mesh-grids for Tokyo and 3600 mesh-grids for Osaka). When using word-like embedding with transfer learning, the Osaka model had to be built with a 6400 × 256 embedding matrix instead of a 3600 × 256 embedding matrix so that transfer learning from Osaka to Tokyo could work. If not so, mesh-grid IDs of Tokyo in [3600, 6400) can't be taken as input by the model. However, the IDs in [3600, 6400)

will not be well trained or embedded due to the lack of corresponding training data in Osaka. Four test cases were listed as Table 5.25. We verified how the minimum and the maximum of the embedding vector were updated before and after training. We also measured the difference between the updated value and the initial value with L1-norm and L2-norm. From Table 5.25, we could see those two IDs in $[3600, 6400)$ were not well embedded by the Osaka model with word-like embedding.

(3) Word-like embedding requires more parameters than image-like embedding. Using the settings mentioned above, for the Tokyo area, the word-like embedding layer contained around 1.6 million parameters whereas the image-like embedding layer (CNNs) only required around 0.07 million parameters in total. As the target urban area increased in size, more parameters were required to construct the embedding layer, whereas image-like embedding method could use the same network architecture with the same number of parameters to model different urban areas. This allowed transfer learning to work better as well as enhancing the interpretability and usability for the embeddings between different urban areas.

5.7 Related Work

Recently, various studies were conducted on human mobility data (e.g. mobile phone GPS log data, taxi GPS data, and location-based services data). These are summarized as urban computing problems in [5].

Trajectory-pattern based methodologies have been proposed to predict future movement of individual person [6, 7]. An approach based on nonlinear time series analysis of the arrival and residence times of users has been proposed, which focused on predicting most important places of each user [9]. J. Zheng [10] proposed an unsupervised learning algorithm for location prediction. Social-LSTM [39] is an advanced multi-agent model, which builds a separate LSTM network for each person. ST-RNN [40] utilized RNN to model spatio-temporal transitions. SERM [41] is recurrent model designed for semantic trajectory. A RNN architecture similar with our word-like embedding model was proposed in [42] for destination prediction task. These models focused on individual mobility and were validated with small-scale trajectory dataset, which are difficult to be applied to our urban mobility modeling task. Some collaborative approaches have been proposed to take social relationships of users into account for location prediction

and recommendation, but they utilized big check-in data from location-based network services [1, 11, 12].

Urban human mobility prediction problem was studied in [86], where it involved an auxiliary Region-of-Interest (ROI) mining process from long-period human trajectory data and demonstrated the better effectiveness of ROI than mesh-grid. Predicting citywide human mobility under some rare events such as live concerts or disasters is a related problem, but it built an online prediction model using real-time current observed data [2]. Furthermore, modeling human mobility for very large populations [36] and simulating human emergency mobility following disasters [37, 38] are other topics that are close to ours. However, all of these approaches had different problem definitions and modeling methods. For example, the approaches required disaster information such as intensity of earthquake and damage level as additional input data in [37, 38]. Forecasting the citywide crowd density [3, 19] is another related problem, but a time-series model was built to predict the crowd density for each region of a city, whereas our system predicts the mobility for millions of individuals based on short-term observations. Population prediction model [13, 14] was built for urban dynamics and city-scale irregularity prediction using transit app logs.

Moreover, some studies also applied deep learning to predict the traffic flow, traffic speed, congestion, and transportation mode as well as human mobility[29, 30, 45–49]. Bidirectional RNN is reported to be more effective for taxi destination prediction task [50]. Many factors (e.g. sampling rate, data type) are demonstrated to affect the modeling performance of human mobility [22]. A transfer learning framework was designed to transfer knowledge of the hourly air quality between cities [33]. Previous studies that aimed to understand the basic life patterns in the flows of people [24] and recommend location-based services [25, 26] utilized the tensor factorization approach to decompose citywide human mobility.

5.8 Conclusion

In this paper, we studied the citywide human mobility prediction problem using big GPS trajectory data and POI data. We proposed image-like embedding with POIs to represent a trajectory like an artificial video. We also designed an LSTM-on-CNNs architecture to simultaneously capture both the spatio-temporal and geographical information from citywide human mobility. Transfer learning was employed to work with

image-like embedding to further boost the performance by exploiting the data obtained from different cities. The experimental results obtained based on multiple urban areas demonstrated the superior performance of our proposed model compared with the baseline methods especially when the training data are limited.

However, our method can be improved in the following ways. (1) In addition to POI data, transportation network data and other types of heterogeneous data such as the population density can be utilized as geographical features. (2) Current framework only takes spatial information into account, temporal information could also be used to improve the performance, and conduct the time-series modeling for citywide human mobility. (3) The current dataset only contained approximately 1% of the total population of Japan. Thus, we need to collect more trajectory data from other sources and design reasonable scaling factors in order to simulate and predict citywide human mobility in a more realistic manner.

Chapter 6

Mobility as Well as Density: Deep Multi-task Learning

6.1 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, Shanghai, and HongKong, 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, when around 300,000 people gathered to celebrate the arrival of 2015 near Chen Yi Square on the Bund, which is the most representative tourist spot in Shanghai. However, allocated police forces were not enough, 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.6.1, 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 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.

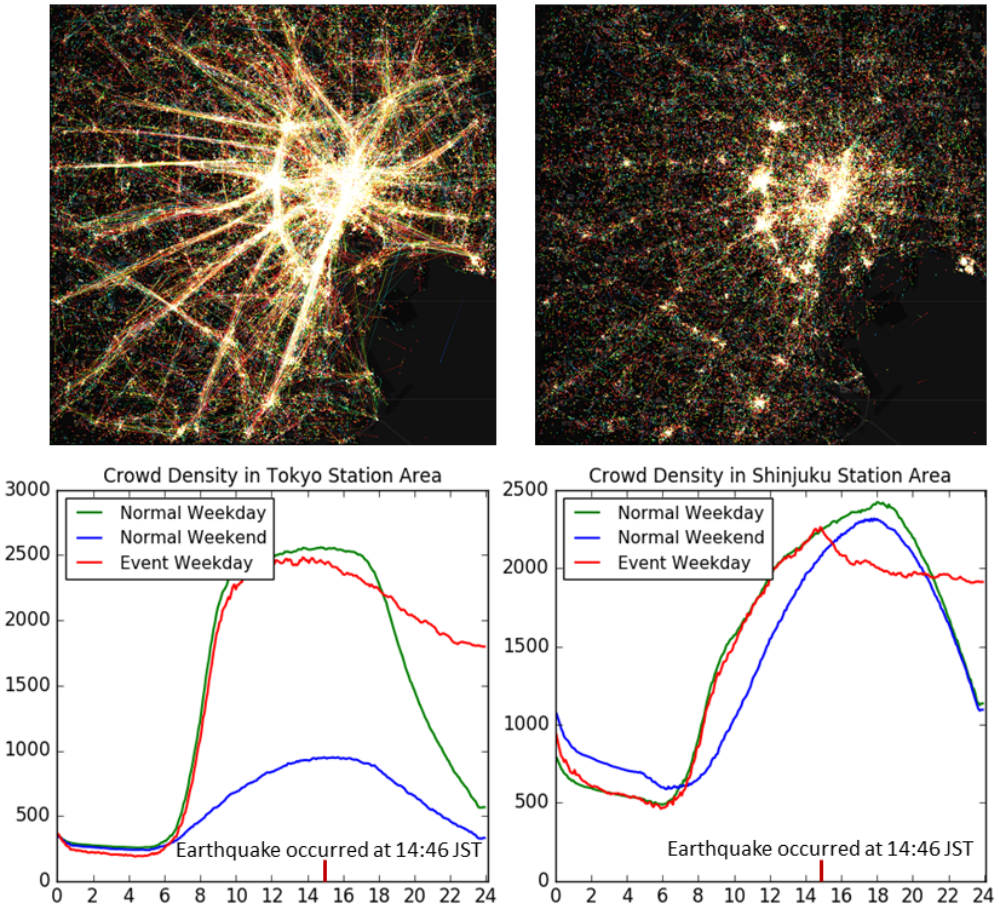


FIGURE 6.1: Citywide human mobility in Tokyo before (upper left) and after (upper right) the Great East Japan Earthquake is listed above. Crowd density in Tokyo station area and Shinjuku station area is listed below.

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 situations at big events. We build an intelligent system called DeepUrbanVideo 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.

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.7.1. 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.
- Using the predicted crowd density and flow video, we further build a series of dynamic crowd mobility graph to help conduct probabilistic reasoning of crowd movements during big events.
- 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.

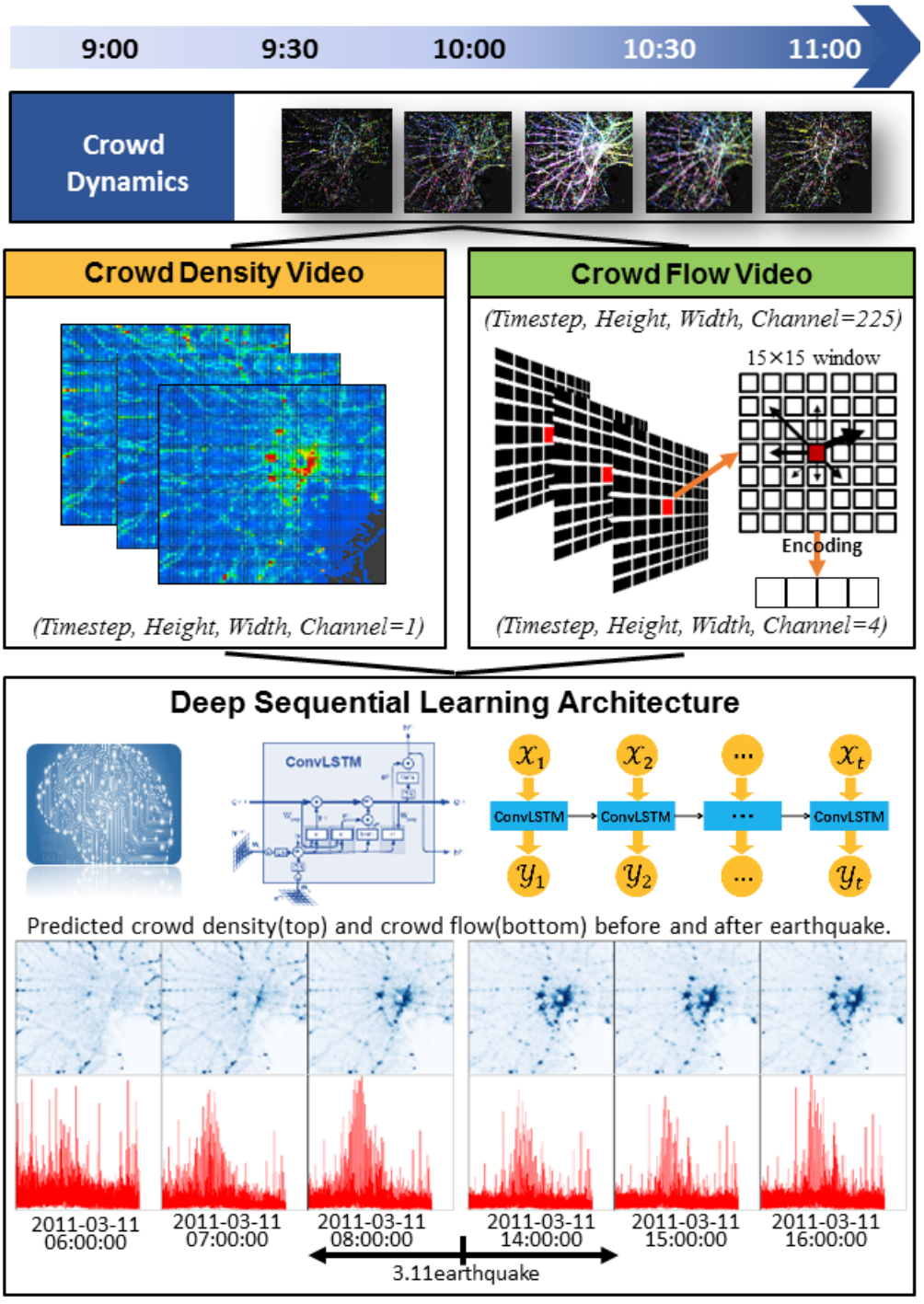


FIGURE 6.2: Can we design an effective real-world system for predicting citywide crowd dynamics at big events? Real-time human mobility data as well as deep learning technologies allow us address this high-social-impact problem.

The remainder of this paper is organized as follows. In Section 6.2, we review some related works. In Section 6.3, we provide a description of our data source. In Section 6.4, we give the problem definition of citywide crowd dynamics prediction. In Section 6.5, we illustrate the proposed deep sequential learning architecture. In Section 6.6, we

explain how to build dynamic crowd mobility graph. We present and discuss the results of the experimental evaluation in Section 7.5. In Section 6.8, we give our conclusions as well as future work.

6.2 Related Work

Forecasting the citywide crowd flow [3, 19, 44] 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[13, 14] and [15] utilize query data of Smartphone APP to forecast only crowd density other than crowd flow. [16, 17] conduct transition estimation from aggregated population data, and [18] estimates the transition populations using inflow and outflow defined by [19]. Some researchers tried to detect the urban anomalies from mobility data based on statistical methodologies [20, 21]. Modeling human mobility for very large populations [2, 22] and simulating human emergency mobility following disasters [23] are similar problems to ours, however, their models are built based on millions of individuals' mobility.

Many recent studies have analyzed human mobility data and they were summarized as urban computing problems by [5]. For example, [24–26] 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. [12] conducted next place prediction in location-based services based on user features. Using population-scale data, [27] detected popular temporal modes and [28] modeled urban population of multiple cellphone networks. Moreover, some studies also applied deep learning to predict the traffic flow, traffic speed, congestion, and transportation mode as well as human mobility[29, 45–49, 87].

6.3 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). It contains approximately 30 billion GPS records, and the total size of the data is more than 1.5 terabytes. Each record contains user ID, latitude, longitude, altitude, timestamp and positioning accuracy level.

6.4 Citywide Crowd Dynamics Modeling

TABLE 6.1: Notation Description

Symbol	Description
M, g	mesh for an urban area, mesh-grid
$Timestep$	the number of observation/prediction steps
$Height$	number of mesh-grids along Latitude-axis
$Width$	number of mesh-grids along Longitude-axis
$Channel$	dimension to store the value for density/flow
Γ_{iu}	each user's(u) trajectory on each day(i)
d_{tm}	crowd density at timeslot t on mesh-grid g_m
f_{tmw}	crowd flow from g_m at t-1 to g_w at t
d_t, f_t	citywide density/flow at t
x_d, x_f	current α steps of density/flow w.r.t t
y_d, y_f	next β steps of density/flow w.r.t t
$\mathbb{X}$	samples from all timeslots for the current
$\mathbb{Y}$	samples from all timeslots for the next
$\wedge$	the predicted results

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 $\{g_1,g_2,...,g_{Height*Width}\}$ 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:

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

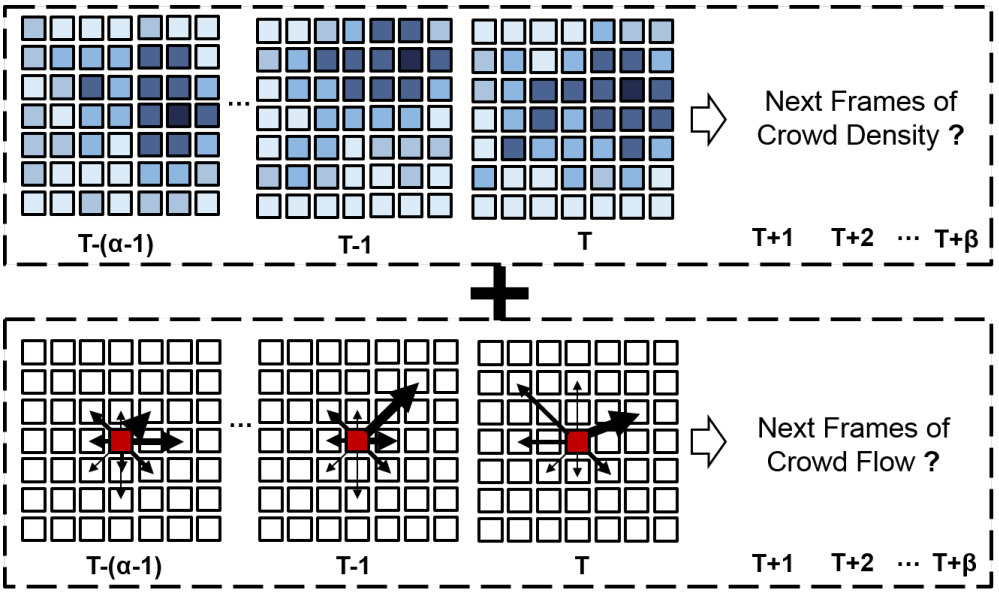


FIGURE 6.3: Citywide Crowd Dynamics Prediction.

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 *Channle* 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.7.1.

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:

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

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

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:

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

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.6.3 demonstrates the overall problem definition mentioned above.

6.5 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[88] 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.7.1, 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:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \odot c_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \odot c_{t-1} + b_f)$$

$$\widetilde{c_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$c_t = i_t \odot \widetilde{c_t} + f_t \odot c_{t-1}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \odot c_t + b_o)$$

$$h_t = o_t \odot tanh(c_t),$$

where W is weight, b bias vector, 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 from this study.

6.5.1 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

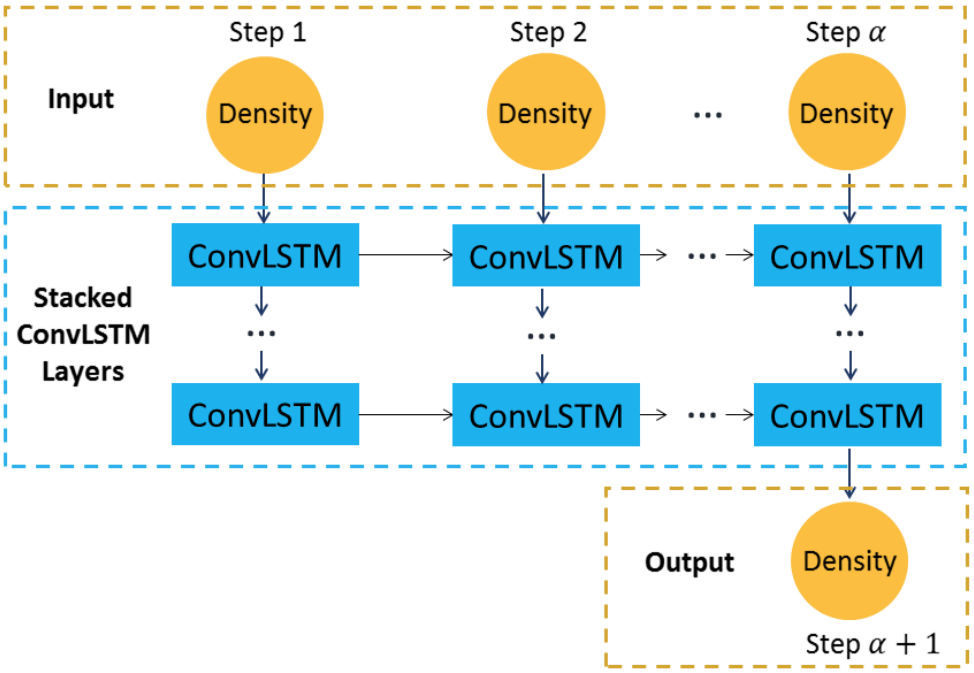


FIGURE 6.4: Stacked ConvLSTM for One-Step Prediction.

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:

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

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 [83]. Thus, a deep architecture constructed with multiple stacked ConvLSTM layers has been shown in Fig.6.4 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.

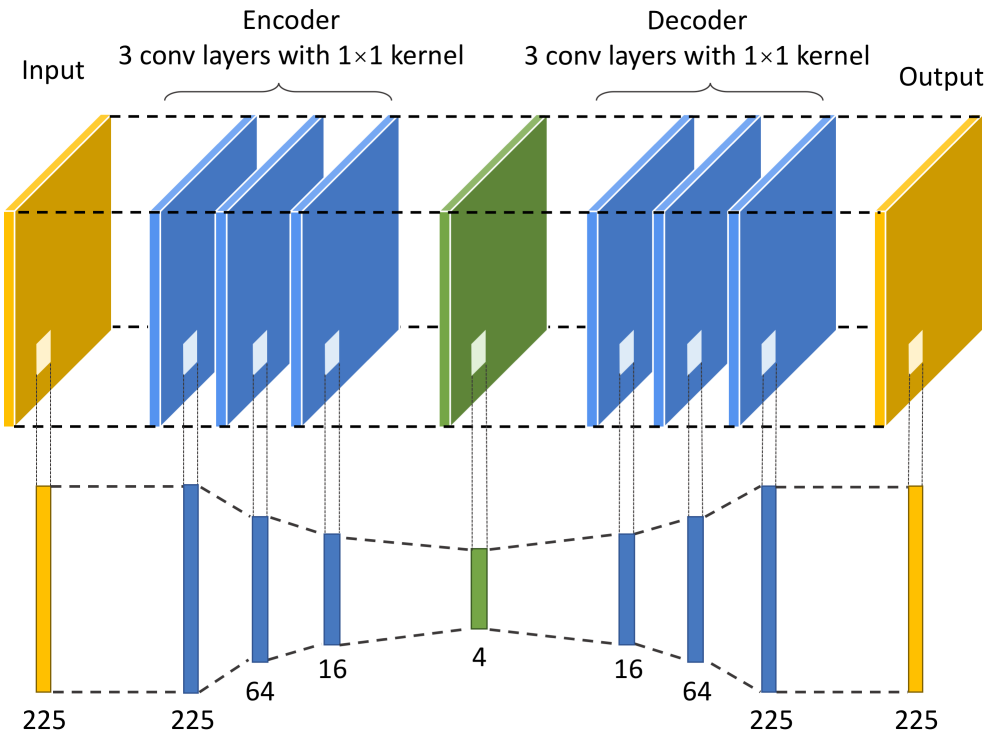


FIGURE 6.5: CNN AutoEncoder for Crowd Flow.

6.5.2 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). 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 [89, 90] to obtain a low-dimension representation of *Channel* for crowd flow.

Compared with traditional neural networks, CNNs were designed specifically for analyzing visual imagery [82], 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 [91, 92]. 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.6.5. 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.

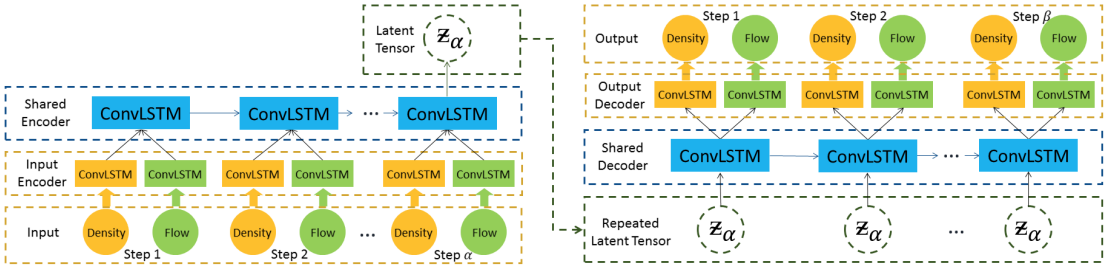


FIGURE 6.6: Multitask ConvLSTM Enocder-Decoder for Simultaneous Multi-Step Prediction of Crowd Density and Crowd Flow.

6.5.3 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.6.4. 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 [88, 93] 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.6.6. The key concept of multi-task learning [94] 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:

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

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.

6.6 Dynamic Crowd Mobility Graph

So far, multiple steps of citywide crowd density and crowd flow can be modeled and predicted simultaneously as crowd dynamics. At this stage, citywide crowd dynamics are still represented with grid-cells, in order to help conduct probabilistic reasoning of crowd movements during big event situations, we build a series of dynamic crowd mobility graph using the predicted crowd dynamics. By using graph instead of mesh, a high-level representation of citywide crowd dynamics can be obtained, more easily and efficiently used for citywide-level event crowd management. Moreover, if severe disasters happen, some parts of the real transportation network might be damaged, therefore we need to build a virtual transportation network to replace or work together the real-world one. Comparing with the static transportation network, our proposed graph is a dynamic one because: (1) One crowd mobility graph is corresponding to one frame of crowd dynamics video; our system can report multiple steps of prediction results, thus we can build a series of graph for each timeslot; (2) Our graph can be updated every 5 minutes by our online updating system.

Specifically, given one frame of predicted crowd density video, we view the centroid of each mesh-grid as a point, the density of the mesh-grid as the weight, then apply weighted KMeans clustering on all the weighted points to get clusters of mesh-grids. Each cluster can be taken as one Region-of-Interest (RoI), as well as one node of the mobility graph. Given one frame of predicted crowd flow video, it is easy to build a transition matrix Ω between each mesh-grid pair. By summing up the total transition number between each node pair, the edges of the mobility graph can be generated. The details of this process are summarized in Algorithm 4. Note that other clustering or RoI construction algorithms (e.g. T-Pattern[56], MeanShift or PopularRoutes[95]) may also be used here, however KMeans is adopted in our system because it can be easily tuned using the only one parameter K.

Algorithm 4: Dynamic Crowd Mobility Graph Building

Input: Citywide crowd density and flow y_d, y_f, a mesh M, node number K.
Output: Dynamic Crowd Mobility Graph

$\Psi \leftarrow \emptyset$; `// dynamic crowd mobility graph.`
for *each* $t \in [t_1, ..., t_\beta]$ **do**
  d_t,$f_t \leftarrow$ retrieve density,flow at t from y_d,y_f;
  $\mathcal{V} \leftarrow$ Node Construction(d_t, M, K) ;
  $\mathcal{E} \leftarrow$ Edge Construction(f_t, M, $\mathcal{V}$) ;
  $\Psi \leftarrow \Psi \cup (\mathcal{V}, \mathcal{E})$;
return Ψ ;
Function *Node Construction (d, M, K)*:
  C, $W \leftarrow \emptyset, \emptyset$;
  for *each* $g_m \in M$ **do**
    $C \leftarrow C \cup g_m$.centroid, $\mathcal{W} \leftarrow \mathcal{W} \cup d_m$;
  $\mathcal{V} \leftarrow$ Weighted Kmeans Clustering(C, W, K);
  return $\mathcal{V}$;
Function *Edge Construction (f, M, $\mathcal{V}$)*:
  $\Omega[|M|][|M|] \leftarrow$ build grid transition matrix using f;
  $\mathcal{E}[|\mathcal{V}|][|\mathcal{V}|] \leftarrow$ initialize node transition matrix with 0;
  for *each pair* $(g_p, g_q) \in (M, M)$ **do**
    $v \leftarrow$ find g_p belongs to which node in $\mathcal{V}$;
    $v' \leftarrow$ find g_q belongs to which node in $\mathcal{V}$;
    $\mathcal{E}[v][v'] \leftarrow \mathcal{E}[v][v'] + \Omega[p][q]$;
  return $\mathcal{E}$;

Given β-step crowd flow video f_1,f_2,...,f_β, using each frame of crowd video, transition matrices Ω_1,Ω_2,...,Ω_β can be calculated. Each Ω_t essentially represents transition probability between each mesh-grid pair in the timeslot t namely Ω_{tij} = P($g_i \rightarrow g_j$) after normalization. Each Ω_t in our final system is just a transition matrix within 5 minutes, which is not sufficient for a relatively long short-term probabilistic reasoning. To address this issue, we view each Ω_t as a first order probability and leverage the first order probability to get higher order transition probabilities. $\Omega_1 \times \Omega_2$ contains all the second order transition probability, thus $\Omega^{1:2}=\Omega_1 + \Omega_1 \times \Omega_2$ after normalization corresponds to the 2-step transition probability, which is the likelihood of transition from g_i to g_j within two steps namely 5*2 minutes. Analogously, we can get the β-step transition probability by calculating the matrix $\Omega^{1:\beta}=\Omega_1 + \Omega_1 \times \Omega_2 + ... + \Omega_1 \times \Omega_2 \times ... \times \Omega_\beta$ with normalization. $\Omega^{1:\beta}$ built with multiple steps of crowd flow can be used to replace the simple first order transition matrix Ω in Algorithm 4 (Edge Construction Function), then crowd mobility graph can contain the transition information within a longer time interval i.e. 5*β minutes.

Table 6.2: Event Information

Event	Date	Training Dates
3.11 Earthquake	2011/03/11	2011/03/01-2011/03/10
Typhoon Roke	2011/09/21	2011/09/11-2011/09/20
New Year's Day	2012/01/01	2011/12/22-2011/12/31
Tokyo Marathon	2011/02/27	2011/02/17-2011/02/26

Table 6.3: Summary of Tuned Parameters

Parameter	Tuned Value
$Height, Width$	80, 80
Time Intervals of One Day	5 minutes×288
Train/Test Timeslots	2880/288
Kernel Window $\eta \times \eta$	15×15
(1)Timestep α/β (Max lead time)	6/6 (30 minutes)
(2)Timestep α/β (Max lead time)	12/12 (60 minutes)
Optimizer	Adam
Epoch	200
Learning Rate	0.0001
Batch Size	4
Scaling Factor for Density/Flow	500/100

Table 6.4: Performance Evaluation of 30 Minutes Ahead Prediction on Four Events

Model 1/2	3.11 Earthquake		Typhoon Roke	
	Density	Flow	Density	Flow
HistoricalAverage	106.032	0.726	75.402	0.519
CopyYesterday	129.436	0.912	85.641	0.592
CityMomentum	27.670	0.653	29.305	0.962
ARIMA	10.430	NA	13.376	NA
VectorAutoRegressive	10.843	NA	13.377	NA
CNN	8.698	0.178	10.245	0.196
CNN Enc.-Dec.	7.115	0.117	8.571	0.187
MultiTask CNN Enc.-Dec.	6.802	0.119	8.226	0.197
ConvLSTM	6.737	0.124	7.959	0.195
ConvLSTM Enc.-Dec.	6.281	0.102	7.508	0.171
MultiTask ConvLSTM Enc.-Dec.	**5.549**	**0.102**	**6.753**	**0.170**

6.7 Experiment

6.7.1 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),

Table 6.5: Performance Evaluation of 30 Minutes Ahead Prediction on Four Events

Model 2/2	New Year's Day		Tokyo Marathon	
	Density	Flow	Density	Flow
HistoricalAverage	0.519	176.013	1.099	33.381
CopyYesterday	110.444	0.660	65.765	0.437
CityMomentum	23.058	0.235	25.774	0.475
ARIMA	8.343	NA	7.808	NA
VectorAutoRegressive	9.511	NA	9.380	NA
CNN	6.178	0.083	6.614	0.100
CNN Enc.-Dec.	5.216	0.079	6.004	0.095
MultiTask CNN Enc.-Dec.	5.158	0.084	5.953	0.097
ConvLSTM	4.679	0.077	5.675	0.094
ConvLSTM Enc.-Dec.	4.500	0.074	5.372	0.089
MultiTask ConvLSTM Enc.-Dec.	**4.117**	**0.074**	**5.012**	**0.086**

Table 6.6: Performance Evaluation of 60 Minutes Ahead Prediction on Four Events

Model 1/2	3.11 Earthquake		Typhoon Roke	
	Density	Flow	Density	Flow
HistoricalAverage	104.604	0.731	75.927	0.529
CopyYesterday	128.133	0.920	86.069	0.601
CityMomentum	32.034	0.570	35.090	0.821
ARIMA	24.296	NA	32.933	NA
VectorAutoRegressive	22.355	NA	29.872	NA
CNN	12.247	0.189	17.670	0.360
CNN Enc.-Dec.	11.372	0.164	13.876	0.245
MultiTask CNN Enc.-Dec.	10.812	0.177	13.800	0.247
ConvLSTM	11.355	0.139	12.285	0.228
ConvLSTM Enc.-Dec.	9.309	0.122	11.186	0.197
MultiTask ConvLSTM Enc.-Dec.	**8.094**	**0.122**	**9.900**	**0.196**

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,

Table 6.7: Performance Evaluation of 60 Minutes Ahead Prediction on Four Events

Model 2/2	New Year's Day		Tokyo Marathon	
	Density	Flow	Density	Flow
HistoricalAverage	175.344	1.102	33.422	0.225
CopyYesterday	106.991	0.645	65.725	0.440
CityMomentum	25.867	0.207	28.825	0.400
ARIMA	15.411	NA	15.259	NA
VectorAutoRegressive	21.072	NA	17.261	NA
CNN	10.469	0.119	12.114	0.223
CNN Enc.-Dec.	8.311	0.097	9.127	0.119
MultiTask CNN Enc.-Dec.	8.153	0.101	9.004	0.124
ConvLSTM	7.615	0.118	9.511	0.140
ConvLSTM Enc.-Dec.	6.885	0.086	7.843	0.103
MultiTask ConvLSTM Enc.-Dec.	**6.496**	**0.085**	**7.483**	**0.101**

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 6.4.

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. All the parameter settings of the experiments are summarized as Table 6.3. The parameter settings were kept the same for each event. Python and some Python libraries such as Keras[72] and TensorFlow[73] were used in this study. The experiments were performed on a GPU server with four GeForce GTX 1080Ti graphics cards.

Baseline models: We considered 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) CityMomentum[2]. 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. (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) 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 [3] 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. (7) 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. (8) Multitask CNN Enc.-Dec. It has 4 Conv layers sharing

a similar multitask architecture as illustrated in Fig.6.6, namely separate input encoding Conv layer, shared encoding and decoding layer, and separate output Conv layer. All the parameters were kept same with (6). (9) ConvLSTM (one-step-by-one-step) and (10) ConvLSTM Enc.-Dec (multi-step-to-multi-step) are the proposed comparison models constructed with four ConvLSTM layers in Section 6.5. 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. 6.5, 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.**.

6.7.2 Performance Evaluation

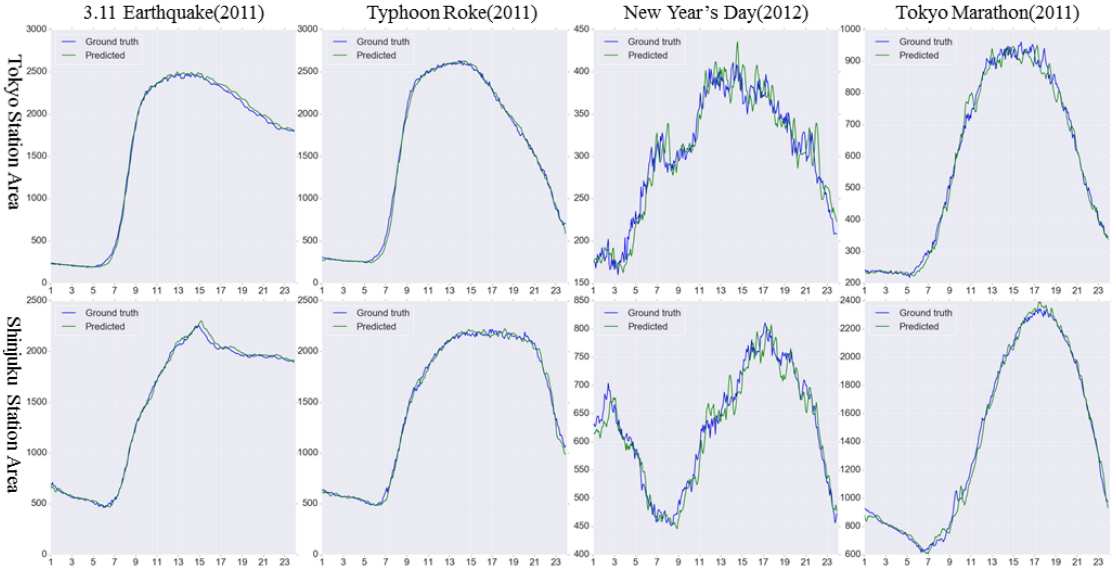
Evaluation metric: We evaluated the performances of the models with Mean Squared Error (MSE) as follows:

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

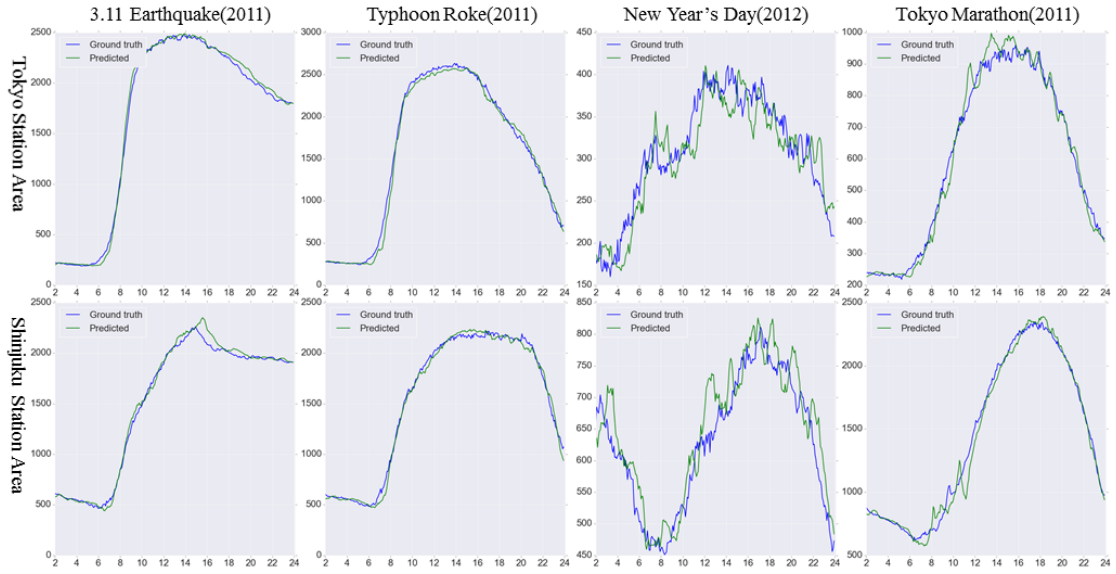
where n is the number of samples, $\mathbb{Y}$ and $\hat{\mathbb{Y}}$ are the ground-truth value and predicted value in 4D tensor format, namely, ($Timestep$, $Height$, $Width$, $Channel$). Density tensor and flow tensor differ at the $Channel$.

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 6.5 for 30 minutes ahead prediction and Table 6.7 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



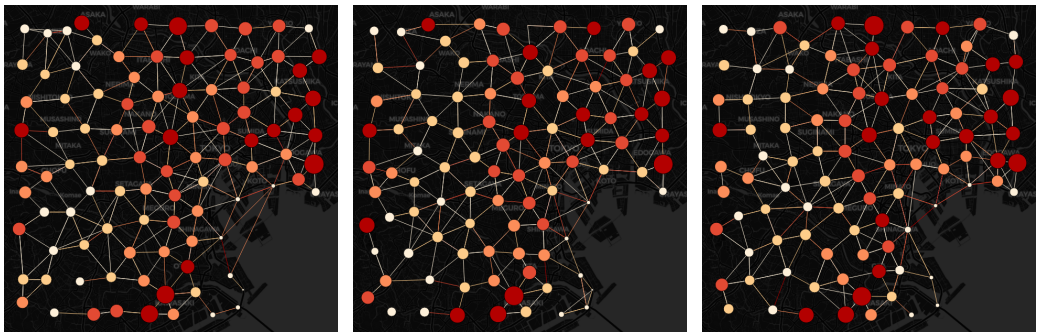
(a) Ground Truth and 30 Minutes Ahead Prediction for Crowd Density



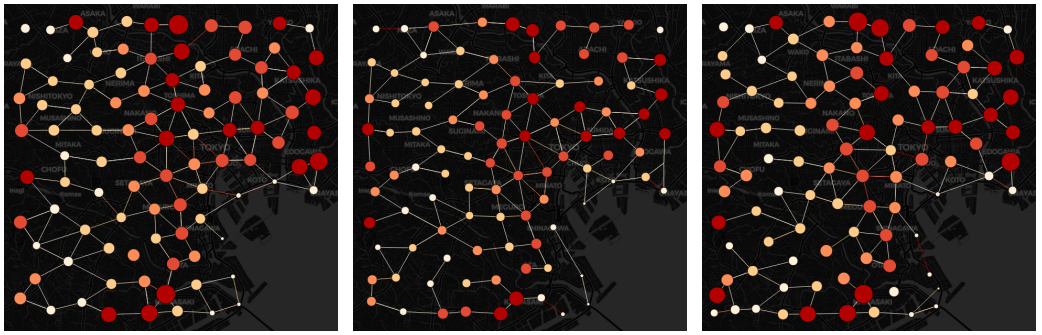
(b) Ground Truth and 60 Minutes Ahead Prediction for Crowd Density

FIGURE 6.7: Visualization of the ground-truth crowd density and the predicted result of our system (MultiTask ConvLSTM Enc.-Dec.) at four events. The prediction lead time is 30 minutes in (a) and 60 minutes in (b) respectively.

Station locating at the central mesh-grid respectively. From Fig.6.7, we can confirm the effectiveness of our model for both 30 minutes and 60 minutes ahead predictions and its high deployability for a real-world online event crowd management system. Referring to the normal weekday and weekend pattern shown in Fig.6.1, we can find that the densities on event weekday (3.11 Earthquake and Typhoon Roke) are very different from normal weekday, and the densities on event weekend (New Year's Day and Tokyo Marathon) also differ a lot from normal weekend. Furthermore, even comparing 3.11



(a) Ground-truth Graph at 2011-03-11 14:00:00 (b) Ground-truth Graph at 2011-03-11 15:00:00 (c) Ground-truth Graph at 2011-03-11 16:00:00



(d) Predicted Graph at 2011-03-11 14:00:00 (e) Predicted Graph at 2011-03-11 15:00:00 (f) Predicted Graph at 2011-03-11 16:00:00

FIGURE 6.8: Visualization of the ground-truth dynamic crowd mobility graph (top) and the predicted results (bottom) at 3.11 Earthquake from 14:00 to 16:00. The larger and darker nodes have higher crowd density and the darker edges represents higher transition probability. The node number K is set to 100, and the edges correspond to 6-step transition matrix $\Omega^{1:6}$.

Earthquake vs. Typhoon Roke or New Year's Day vs. Tokyo Marathon, 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.

Performance on dynamic graph: Using the proposed algorithm in Section 6.6, we build a series of dynamic crowd mobility graph for the 3.11 Earthquake event, and demonstrate three snapshots of the graph at 14:00, 15:00 and 16:00 (the earthquake occurred at 14:46 JST) in Fig.6.8. We generate 100 nodes and build the edge for each node pair based on 6-step transition matrix $\Omega^{1:6}$, which can indicate the crowd flow transition probability in the next 30 minutes. Through Fig.6.8, we could see how the crowd dynamics were gradually evolving during the earthquake. No matter the ground-truth or predictions, it showed quite different details at 14:00, 15:00 and 16:00. Crowd density prediction could achieve very good performances as shown in the previous, and

the nodes of the graph showed a close resemblance between the ground-truth and predictions. However, there still existed some gap between the ground-truth and prediction of the transition probability on edges. Underestimation could be observed through the figure, which left us room for further improvement on the high-dimensional crowd flow video.

6.8 Conclusion

In this study, we built a data-driven intelligent system called DeepUrbanVideo 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. (3) Advanced deep learning technologies such as ResNet[96] and PredNet[97] can also be utilized to further boost the performance, especially for the crowd flow prediction part.

Chapter 7

No Historical Data: Deep Online Learning

7.1 Introduction

The next-generation 5G mobile Internet technologies will mark a new era in the information industry, and they will play an important role in stimulating the growth of the Internet of Things (IoT). Against this background, massive GPS trajectories that are being continuously generated from sources, such as smartphones, GPS devices on cars, WLAN networks, and location-based social networks, become important for use as real-time human mobility data streams. With such valuable streaming data, people's future behaviors and movements can be predicted step-by-step in an online manner, based on an intuitive Markov-like assumption that people's next behaviors mostly rely on their recent ones. Especially, when big rare events or disasters, such as high-magnitude earthquakes happen, people's behaviors and movements will become rather different from their daily routines. Such online short-term predictions using recent momentary mobility will become very necessary and practical. Elevating this to a citywide level, namely predicting *UrbanMomentary* human mobility for a huge urban area, can play a crucial role in effective urban planning, transportation scheduling, and emergency management.

However, even for a short period, human mobility and transportation transitions for a large-scale transportation system are highly complex, which are almost impossible to be effectively modeled using classical methodologies or simple neural network-based models. Emerging deep-learning technologies have demonstrated superior performances on

various datasets (e.g., images, texts, and videos) [68, 89, 90] of existing classical approaches. Hence, in this study, we investigate the various aspects of human mobility during a short period in a large urban area by using a deep-learning-based approach. We also develop an intelligent system for citywide short-term human mobility prediction with high precision compared with the existing approaches.

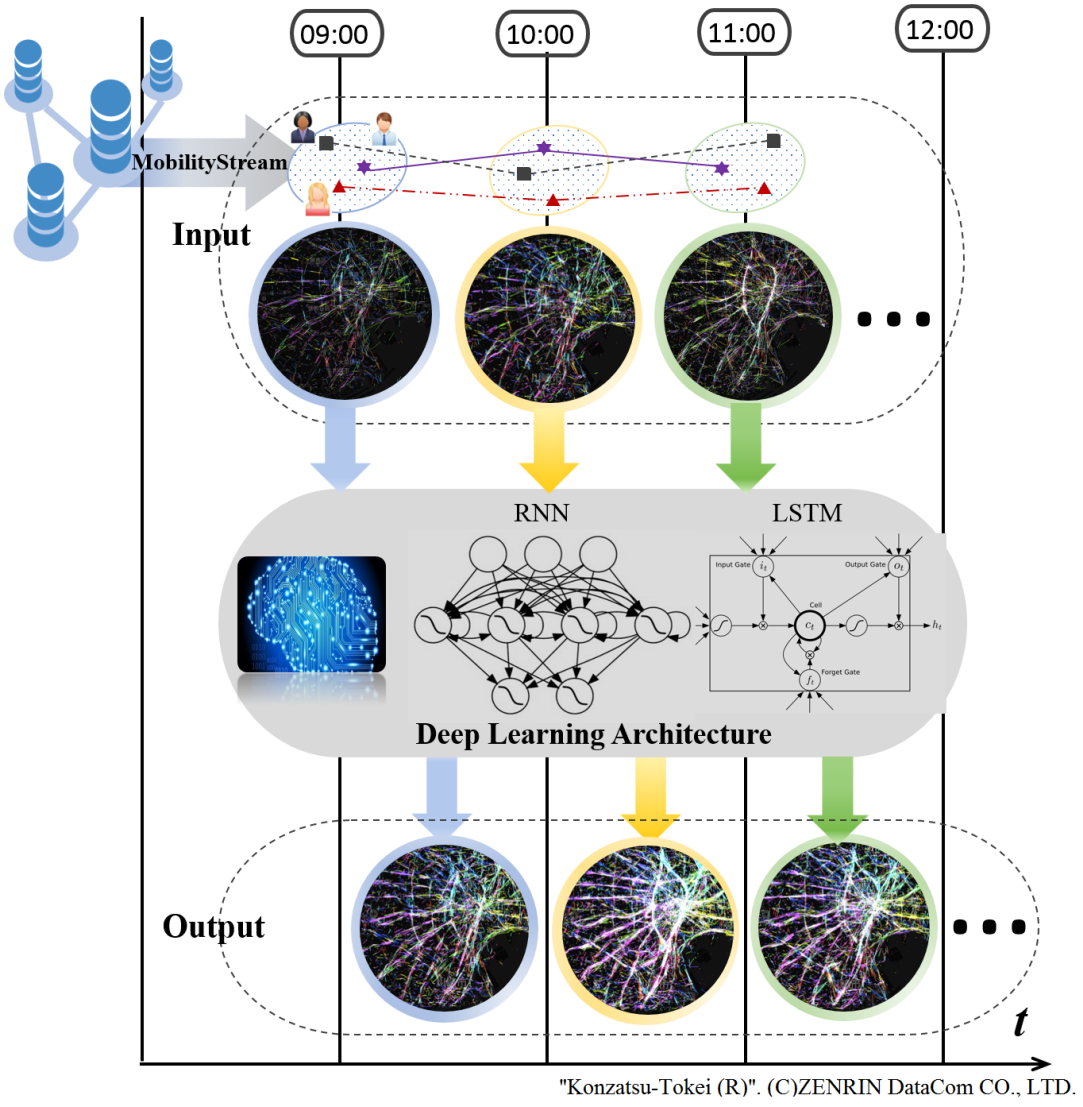


FIGURE 7.1: Can we develop an online intelligent system for short-term human mobility prediction with high precision by using recent momentary mobility at a citywide level? Big human mobility data and deep-learning technologies provide us with the opportunity to implement this system.[1]

[1]"Konzatsu-Tokei (R)" from ZENRIN DataCom CO.LTD is used by us, which refers to people flows data collected by individual location data sent from mobile phone with enabled AUTO-GPS function under users' consent, through the "docomo map navi" service provided by NTT DOCOMO, INC. Those data is processed collectively and statistically in order to conceal the private information. Original location data is GPS data (latitude, longitude) sent in about every a minimum period of 5 minutes and does not include the information to specify individual such as gender or age. In this study, the proposed methodology is applied to raw GPS data by NTT DOCOMO, INC.

In this study, we first collect big human mobility data and process them into calibrated trajectories, and construct an artificial human mobility data stream for a large urban area. Then, we build an online intelligent system called *DeepUrbanMomentum* to continuously take the recent momentary mobility as the input and predict next short-term urban human mobility as the output, as shown in Fig.7.1. The modeling component of our system is based on the deep Recurrent Neural Network (RNN) architecture constructed using two layers: one RNN layer is used to turn the inputted location sequence into a single latent vector containing information about the entire sequence. Then, a functional layer will repeat this latent vector multiple times and pass this vector sequence to another RNN layer that is used to turn this constant sequence into multiple steps of output mobility. This deep model is built essentially as a regression model that can directly take continuous values (location coordinates) as input and output. Finally, given an artificial mobility data stream for a big urban area, DeepUrbanMomentum will automatically conduct an online deep-learning process and report the prediction results of UrbanMomentum by a well-trained model by using the current urban mobility data. To the best of our knowledge, DeepUrbanMomentum is the first system that applies the deep-learning approach to effectively perform online short-term human mobility predictions at a citywide level. It has the following key characteristics:

- It is built and tested based on a big human mobility data source, which stores the GPS records of 1.6 million users over three years.
- It is built as an online prediction system driven by mobility stream and deep-learning technologies.
- It constructs a deep-sequence learning model with RNN for effective multi-step predictions.
- It is applied to real-world scenarios and verified as a highly deployable prototype system.

The remainder of this paper is organized as follows: Section 2 gives an overview of our data source. Section 3 gives the definition of UrbanMomentum and the prediction model. Section 4 explains the modeling details and the deep-learning architectures. Section 5 shows the experimental details, the performance evaluation, and the prediction results in a real-world scenario. Section 6 introduces studies related to our research. Section 7 contains summaries, the limitations of our current system, and our future work.

7.2 Data Source

A 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)[1]. Data collection was conducted by a mobile operator and a private company under an agreement with the mobile phone users. This dataset contains approximately 30 billion GPS records, and the total size of the data is more than 1.5 TB. To better simulate a real-time situation for our online system, this dataset is stored on a Hadoop cluster, containing 32 cores, 32 GB memory, and 16 TB storage, which can run 28 tasks simultaneously. Furthermore, we use Hive on top of Hadoop to make the whole system support SQL-like spatial queries. Therefore, GPS trajectories of a specified city and day can be retrieved in a short response time, and our database can be regarded as a nearly real-time data source that can provide streaming trajectory data to our online system.

7.3 Preliminaries

Definition 1 (Raw human trajectory): The raw trajectory collected from an individual is essentially a sequence of 3-tuple: $(timestamp, latitude, longitude)$, which can indicate a person's location according to a captured timestamp. In the rest of this paper, it is further simplified as a sequence of (t, l)-pairs.

Note that the raw trajectory has a lot of temporal uncertainties because of different time intervals between two consecutive timestamps. Our goal is to predict citywide human movements; therefore, it motivates us to reduce temporal uncertainty by calibrating the raw trajectory to have equal time intervals Δt, which is defined as follows:

Definition 2 (Calibrated human trajectory): A calibrated human trajectory $traj$ from time t_1 to t_m is a sequence of timestamp-location pairs denoted as: (t_1, l_1), (t_2, l_2), ..., (t_m, l_m) that satisfies:

$$\forall i \in [1, m), |t_{i+1} - t_i| = \Delta t$$

In fact, this calibration operation is performed based on the following assumption.

Definition 3 (Temporal certainty assumption): For each individual person, his/her location coordinates can be retrieved every Δt time.

Definition 4 (Urban human mobility stream): Based on the above definitions, urban human mobility can be regarded as a kind of streaming data arriving every Δt time interval, from which we can get n infinite human trajectories corresponding to n individual persons. Furthermore, these n trajectories will all be spatially contained in one urban area denoted as ur. Therefore, an urban human mobility stream is determined by three parameters ur, n, and Δt, which are given as:

$$uhms = F(ur, n, \Delta t)$$

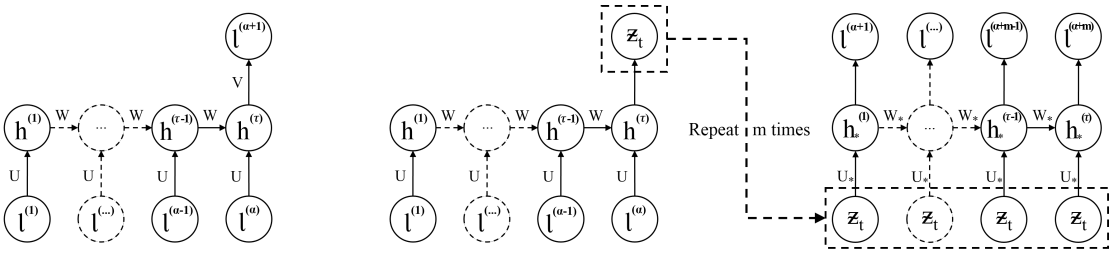


FIGURE 7.2: Deep Sequential Modeling Architecture.

Ideally, n should be the total number of a city's resident population, and Δt should be several seconds. However, this ideal mobility stream is extremely hard to build because of various limitations of location acquisition technologies. Therefore, n is more likely to be the total number of active users of a certain smartphone application, and Δt is set to 10 min or a longer time interval.

Definition 5 (Current urban mobility): given an *uhms*, a current time t and an integer α, current urban mobility X_t is defined as follows:

$$X_t = \{traj \mid traj \in uhms \wedge \forall i, t - \alpha\Delta t \leq traj.t_i \leq t\}$$

which intuitively means current α steps of urban human mobility accumulated from *uhms*.

Definition 6 (Next urban mobility): Similarly, given an *uhms*, a current time t, and an integer β, the next urban mobility X_{t+1} is defined as follows:

$$X_{t+1} =$$
$$\{traj \mid traj \in uhms \wedge \forall i, t + \Delta t \leq traj.t_i \leq t + \beta\Delta t\}$$

which means the next β steps of urban human mobility.

Definition 7 (UrbanMomentum prediction model): Given the current urban mobility X_t, UrbanMomentum prediction will construct a model $P_\theta(\widehat{X}_{t+1} \mid X_t)$, in which θ represents a set of model parameters and $\widehat{X}_{t+1}$ is the predicted next urban mobility. It will be built as a regression model, and its parameters can be obtained by minimizing the prediction error $L(\widehat{X}_{t+1}, X_{t+1})$ as follows:

$$\theta = \underset{\theta}{\operatorname{argmin}} L(\widehat{X}_{t+1}, X_{t+1}) = \underset{\theta}{\operatorname{argmin}} \|\widehat{X}_{t+1} - X_{t+1}\|^2$$

7.4 Short-Term Urban Mobility Modeling

Our system is built based on a given *uhms* to predict the next urban mobility $\widehat{X}_{t+1}$ using X_t in an online manner. It is a relatively easy task when $\widehat{X}_{t+1}.\beta = 1$, which means the online system accumulates the current α steps of the urban human mobility X_t and uses them to predict only one-step of the next urban mobility. However, it is always not sufficient to give out only the one-step prediction, especially during times of emergencies, such as rare events or some natural disasters. Therefore, a more meaningful prediction $\widehat{X}_{t+1}$ with a large β called Short-Term Urban Mobility Prediction becomes the main task of our online prediction system.

7.4.1 SimpleRNN Modeling Architecture

Given a X_t and based on our temporal certainty assumption, a current mobility of one person can be simplified as: $x_t = l_1, l_2, ..., l_\alpha$, and similarly a next short-term prediction can be represented as: $\widehat{x_{t+1}} = l_{\alpha+1}, l_{\alpha+2}, ..., l_{\alpha+\beta}$. It can be further modeled as:

$$P(l_{\alpha+1}, l_{\alpha+2}, ..., l_{\alpha+\beta}) = \prod_{i=1}^{\beta} P(l_{i+\alpha} \mid l_i, l_{i+1}, ..., l_{i+\alpha-1}) \tag{7.1}$$

Spatial Continuity. This model is similar to the n-gram model, which is a typical probabilistic sequential model for predicting the next item in such a sequence in the form of a (n-1)-order Markov model. However, the longitude and latitude of each location l is a continuous value in our problem definition because of the spatial continuity for which it is not simple to utilize the Markov model. Some may suggest a classical methodology that partitions the whole area into massive grids to convert the continuous space into discrete values. It is still difficult because our online system has to predict short-term

human mobility for large urban areas, such as the Great Tokyo Area. Even with 1000-meter meshing, it still generates about 4,000 grids for the whole urban area (3,925 km^2), which will lead to an extremely sparse transition matrix if we apply the Markov model based on this huge mesh. In conclusion, urban human mobility on a continuous large-scale area is a highly complex phenomenon, which cannot be modeled without using classical methodologies. The above information motivates us to employ deep-learning technologies, such as RNNs [68], and their special variants of the long short-term memory (LSTM) networks [69], to our system for mobility modeling. These have provided an impressive performance in modeling sequential data, such as speech and text. In particular, for our system, they can help us model human mobility on a continuous large urban space in a regression manner.

Recurrent Neural Networks. Compared with traditional neural networks, RNNs are specially designed for sequential data modeling. In traditional neural networks, neurons in one layer and its neighboring layers are fully connected, whereas neurons in the same layer do not have any connections. Such structures cannot effectively deal with the situation when data are not independent, such as words in a sentence. The typical structure of a simple RNN is shown in Fig.7.1. We can see that the neighboring neurons in the same hidden layer are connected with one another so that the network can memorize former information and have an impact on the output of the current timestep τ. Therefore, the total input not only contains the input at the timestep τ, but also the output at the timestep τ-1. To train an RNN, the standard method is "backpropagation through time" (BPTT).

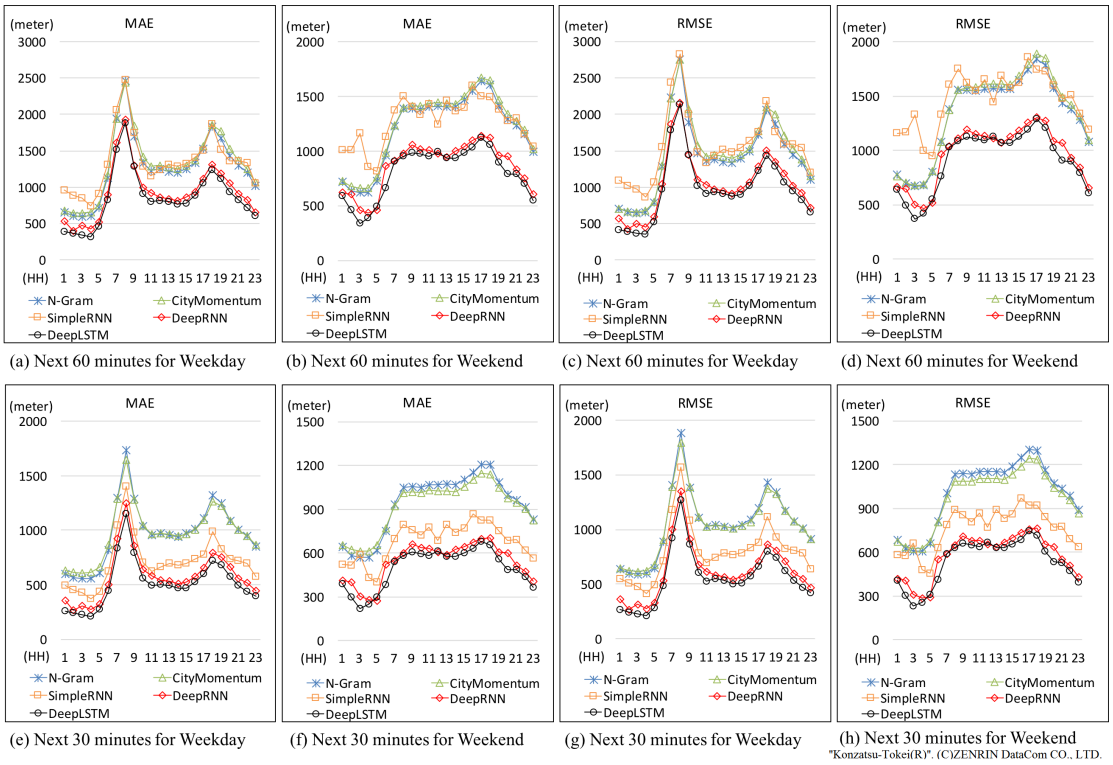
Our goal is to build an Urban Mobility Model for short-term prediction described by Equation (7.1) using an RNN. A simple recurrent network structure is depicted in Fig.7.2-(a), which typically contains an input layer, a hidden layer and an output layer, where *tanh* is used in the hidden layer for mapping inputs into a s single latent vector also called the latent representation. *ReLU* is used as the final activation function, and mean-squared-error (mse) is the objective function defined in Definition 8.

The formulas that govern the whole computation in our architecture are as follows:

$$s_\tau = \tanh(Ui_\tau + Ws_{\tau-1}) \tag{7.2}$$

$$o_\tau = \mathrm{ReLU}(Vs_\tau) \tag{7.3}$$

where i_τ represents the input $(l_i, l_{i+1}, ..., l_{i+\alpha-1})$, o_τ represents the output $l_{\alpha+i}$ described in Equation (7.1), W and U are weight matrices in the hidden layer and V is weight



(a) Next 60 minutes for Weekday (b) Next 60 minutes for Weekend (c) Next 60 minutes for Weekday (d) Next 60 minutes for Weekend

(e) Next 30 minutes for Weekday (f) Next 30 minutes for Weekend (g) Next 30 minutes for Weekday (h) Next 30 minutes for Weekend

FIGURE 7.3: Performance Evaluation of Weekday and Weekend.

matrix in the output layer. All these weight parameters will be determined by applying the BPTT algorithm as mentioned above; the algorithm details will be omitted in this paper.

With this construction, an n-gram-like mobility regression model called SimpleRNN is built; the model can take continuous value of location coordinates as input and output.

7.4.2 DeepRNN Modeling Architecture

Short-Term mobility can be modeled and computed as defined in (7.1) in an iterative one-by-one manner. One major limitation of this model is to predict a relatively long short-term mobility. 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 multi-step-to-one-step modeling in (7.1) with multi-step-to-multi-step modeling aimed at achieving better performance on "long" short-term predictions. This is defined as follows:

$$
\begin{aligned}
&P(l_{\alpha+1}, ..., l_{\alpha+\beta}) \\
&= \prod_{i=0}^{\lceil \frac{\beta}{m} \rceil - 1} P(l_{\alpha+i\cdot m+1}, ..., l_{\alpha+i\cdot m+m} \mid l_{1+i\cdot m}, ..., l_{\alpha+i\cdot m})
\end{aligned} \tag{7.4}
$$

where m is multiple output steps at one time.

To deliver this idea, a deep-learning architecture called DeepRNN is constructed as shown in Fig.7.2-(b). It works in the following steps: (1) the first hidden layer of RNN maps the α steps of the inputted mobility into a single latent vector h, which contains information about the entire sequence; (2) this vector is repeated m times; and (3) another hidden layer of RNN is used to turn this constant sequence into the m steps of the output mobility. Similarly, SimpleRNN, *tanh*, and *ReLU* are used as activation functions in these two RNN layers. Our deep architecture is similar to a sequence-to-sequence model [93], and the two RNN layers act as an encoder and a decoder.

7.5 Experiment

7.5.1 Experimental Setup and Parameter Setting

From our big human mobility database, we select one month of data (October 2011) and divide them into two parts, weekday dataset and weekend dataset, since urban human behaviors on weekdays and weekends are distinct from each other. Based on these, we construct two independent urban human mobility streams, denoted as $uhms_d$ (weekday) and $uhms_e$ (weekend), respectively, where $uhms_d.n \approx 112,360$ and $uhms_e.n \approx 94,812$ averaged by each day, $uhms_d.\Delta t$ and $uhms_e.\Delta t$ are both set to be 10 min, $uhms_d.ur$ and $uhms_e.ur$ are both set to the Greater Tokyo Area by default (Long. $\in$ [139.5, 139.9], Lat. $\in$ [35.5, 35.8]).

Then, the two types of UrbanMomentum predictions are tested on these two streams. One is called "Next 60 minutes" with $\widehat{X}_{t+1}.\beta$ equal to 6, and another is called "Next 30 minutes" with $\widehat{X}_{t+1}.\beta$ equal to 3. This means our system will predict the next-one-hour or next-half-hour urban human mobility in each report. Based on the empirical tuning result, we found the current urban mobility $X_t.\alpha = 3$ and $m = 3$ in DeepRNN would be appropriate.

Lastly, all settings about modeling training and testing are kept the same in these two cases. For both SimpleRNN and DeepRNN, a 64-dimension vector is used as the latent representation Z_t of the entire X_t. The RMSprop algorithm is adopted in our system to govern the whole training process. We randomly select 80% of the data for model training and use the remaining 20% for validation, which is used to early-stop our training algorithm if the validation error is converged. This early-stopping strategy is very crucial for an online learning system like ours. Python and some Python libraries including Keras[72] and TensorFlow[73] are used to implement our system.

7.5.2 Performance Evaluation

Comparison models: (1) N-Gram. It is a widely used algorithm for modeling sequential data, especially in the filed of natural language processing. In our study, we applied this model basing on a gridded space to predict next possible grid. Then the location coordinates were generated randomly inside the predicted grid from a uniform distribution. In order to avoid sparsity problem on a large urban area, we utilized Four-Gram model with $\Delta Long.$=0.01 $\times$ $\Delta Lat.$=0.008 (approximately $900m \times 900m$) as the mesh-size. (2) CityMomentum [2]. It was firstly proposed for this kind of momentary mobility prediction at the citywide level. It is a predicting-by-clustering framework using a mixture of multiple random Markov chains. Each of them is an improved first-order markov model that considers not only the next-step probability from one subject's movements, but also the probability based on the cluster's movements, where the cluster is a bunch of subjects sharing similar movements with the subject. The parameter settings used in our experiment were kept same with the original paper. (3)~(4) SimpleRNN and DeepRNN. These are the two models proposed by us. (5) DeepLSTM. We also implement another comparison model with LSTM[69] called DeepLSTM, which shares the same architecture with DeepRNN except that ordinary neurons in traditional RNNs are replaced with special computation blocks namely LSTM. It has shown superior performance to traditional RNNs for long time-series modeling; therefore, we want to test if it can further improve the performance for our short-term prediction system.

Evaluation metrics: For n trajectories in a given *uhms*, the next β steps of locations will be predicted by every report of our online system. Therefore, to evaluate the overall accuracy of simulation results in a simpler way, we redefine two different metrics, the

mean absolute error (MAE) and the root-mean-square error (RMSE), as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{\beta}\sum_{j=1}^{\beta}\|l_{ij} - \widehat{l}_{ij}\|\right]$$

$$\text{RMSE} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{\beta}\sum_{j=1}^{\beta}\|l_{ij} - \widehat{l}_{ij}\|^2\right]^{\frac{1}{2}}$$

where $\|l - \widehat{l}\|$ means the Euclidean distance between the real location and the predicted one for each trajectory at each step, which will be measured in meters.

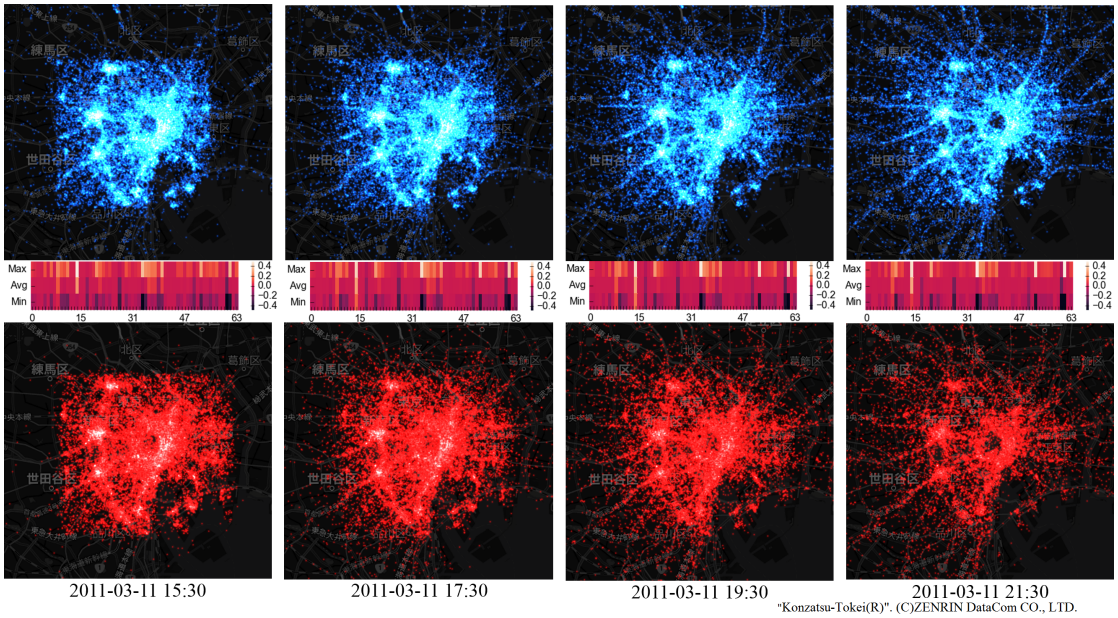


FIGURE 7.4: Visualization for human mobility in the core area of Tokyo in first six hours after the Great East Japan Earthquake. The prediction results are listed on the top in blue, and the corresponding ground truths are at the bottom in red. The 64-dimensional latent representations of UrbanMomentum learned by RNN at each timestamp are listed in the middle. The maximum, average and minimum are calculated across each dimension (0~63) separately as a concise summary of the entire representations. [1]

Performance comparison: Using the two metrics above, we compared the performances of the baseline models and our proposed deep-learning-based models for each hour by averaging each day's result. The evaluation results are summarized in Fig.7.3, and the results of the "Next 60 minutes" and "Next 30 minutes" are listed as Fig.7.3-(a)~(d) and Fig.7.3-(e)~(h), respectively. We can see that the DeepRNN model outperformed N-Gram, CityMomentum and SimpleRNN in each subfigure, and the advantage

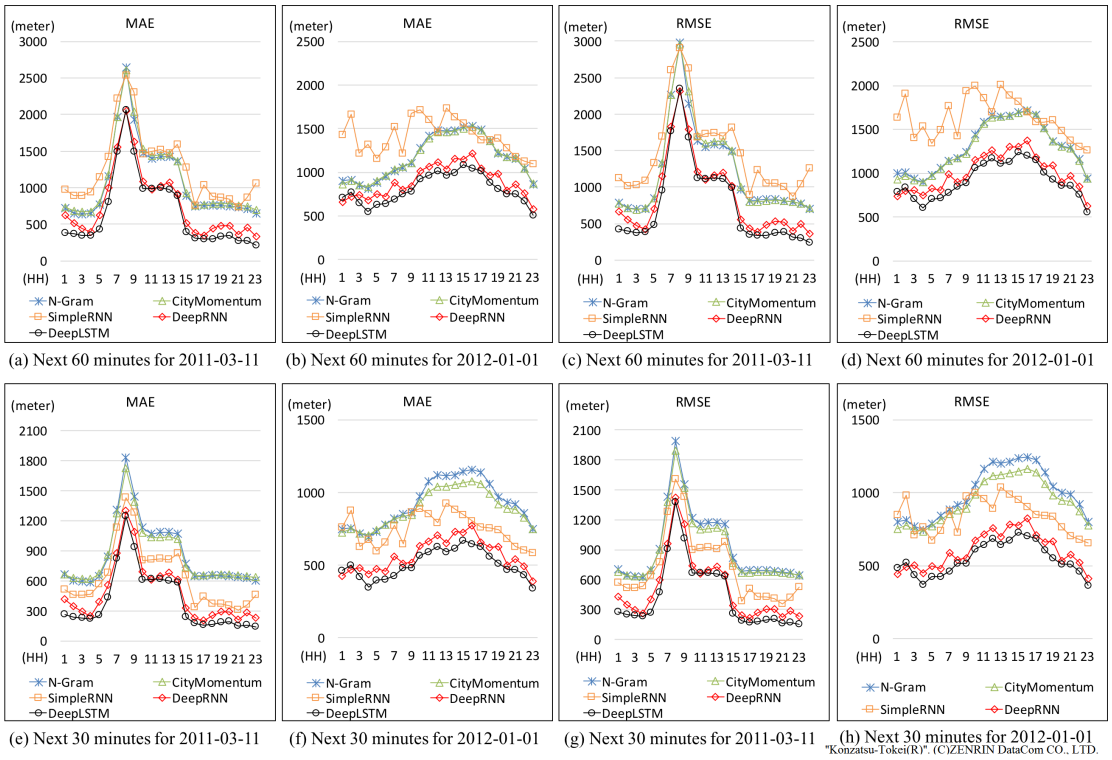


FIGURE 7.5: Performance Evaluation of 3.11 Japan Earthquake and New Year' Day.

over SimpleRNN is more obvious for a relatively long-term "Next 60 minutes" prediction than a short "Next 30 minutes" one. Furthermore, DeepLSTM achieved better performances than DeepRNN, which demonstrates that the traditional RNN is not sufficient for modeling short momentary human movements. All the models performed relatively badly around 8:00 am and 6:00 pm of weekday-the typical morning and evening rush hours in Tokyo. Urban human mobility running on a highly complicated transportation system will change drastically during these hours, and UrbanMomentum becomes hard to predict by just using a few recent observations, which are a major limitation of our system for the normal weekday scenario.

7.5.3 Application to Real-World Scenario

We apply our prototype system to two real-world scenarios: **(1) 3.11 Japan Earthquake (2011-03-11)**. On March 11, 2011, at approximately 2:46 pm local time, the 9.0 magnitude Great East Japan Earthquake occurred off the east coast of Japan; this is considered one of the most powerful earthquakes worldwide. The earthquake caused a great impact on people's behaviors in the Great Tokyo Area. We apply our system to this major emergency scenario to validate its applicability by using a constructed

mobility stream of 2011-03-11 00:00~23:59. Our system gave out reports of "next 60 minutes" and "next 30 minutes" short-term urban mobility prediction every hour. Taking about 30,000 people who were in the core area of Tokyo at 3:00 pm as observation targets, we selected 15:30, 17:30, 19:30, and 21:30 (approximate 6 hours after the earthquake happened) as four evaluation timestamps, and compared their predicted locations with the ground truth. The visualization results are shown in Fig.7.4. This figure shows that our system can work with a relatively high accuracy level to predict urban human mobility after such a huge disaster and a slow evacuation process. Fig.7.4 also demonstrates the encoding RNN has effectively learned the latent representations of UrbanMomentum for the different timestamps after the earthquake. Furthermore, we used the same quantitative measures and summarized the evaluation results as Fig.7.5-(a)(c)(e)(g). Through the figures, we can see the different performance result around 3:00 pm comparing with normal weekdays because of the huge influence of the earthquake on urban transportation system. **(2) New Year's Day (2012-01-01)**. New Year can also be treated as a kind of rare event although it is not that rare as 3.11 earthquake. There are a number of New Year celebrations in Tokyo area, such as Disney Land New Year party and Shibuya square countdown. Especially, for "Hatsumode" (the first visit in Buddhist temple or shrine), a large crowd of people gather at Meiji Shrine, Sensoji Temple and Zojoji Temple, and most of the railway lines operate overnight on the New Year's Eve for this. All these make urban human behaviors very different from normal days. We constructed a mobility stream of 2012-01-01 00:00~23:59 to test the performance of our system under this scenario and summarized the evaluation results as Fig.7.5-(b)(d)(f)(h). Different performance result during midnight (01:00~05:00) can also be observed through the figures. DeepLSTM still achieved the best performances under both of the scenarios.

7.6 Related Work

CityMomentum [2] is the most closely related work with ours; however, our deep-learning-based approach can outperform it as shown in our experiments. Simulating human emergency mobility following disasters was addressed in [37, 38], but it required disaster information such as intensity of earthquake and damage level as additional input data. Modeling human mobility for very large populations [13, 22, 36] are research topics close to ours, but still different from our problem definition. Moreover, traffic flow has also been studied in [29, 30]. However, all of these approaches did not use

the power of deep learning technologies. Forecasting citywide crowd density [3, 19] is a related endeavor based on deep learning, which builds a long time-series model for each region of a city, whereas our system predicts citywide short-term mobility based on the recent observations rather than a long-period's. Some researchers also have applied deep learning to predict traffic flow, traffic speed, congestion, and transportation mode along with human mobility [45–49]. Moreover, various studies conducted on human mobility data, are summarized as urban computing in [5]. C. Song [31] explored the upper bound of predictability of human mobility. J. Zheng [10] proposed an unsupervised learning algorithm for location prediction.

7.7 Conclusions

In this study, we collected big human mobility data to construct artificial mobility data streams for large urban area. Based on this, we build an intelligent system call DeepUrbanMomentum for online short-term mobility prediction at a citywide level based on momentary human movements that we achieved. A deep RNN was specially designed as an effective multistep-to-multistep prediction model. Experimental results demonstrated the superior performance of our proposed model compared to the existing approaches and other shallow models. Furthermore, we applied our system to a real-world scenario and verified its applicability.

Our system has some room for improvement in the following areas: (1) Our system is still struggling to deal with the situation when urban mobility is full of sudden changes such as morning rush hours. (2) Other heterogeneous data, such as transportation network and Point-of-Interest data, can also be used as auxiliary features for deep-learning models. (3) More sophisticated preprocessing will be included to improve the overall performance of our system. Particularly, we will apply map matching algorithm and trajectory calibration algorithm [32] to improve the quality of the raw trajectories.

Chapter 8

Conclusion

Rapidly developing location acquisition technologies have provided us with big GPS trajectory data, which offers a new means of understanding people's daily behaviors as well as urban dynamics. In this study, a raw GPS log dataset was collected from approximately 1.6 million mobile phone users in Japan over a three-year period (August 1, 2010 to July 31, 2013). With such data, predicting human mobility at the city level will be of great significance for transportation scheduling, urban regulation, and emergency management. However, in order to do this, we still confront with a few challenges. (1) The urban area is too large, taking the Greater Tokyo area as an example, it is the most populous metropolitan area in the world, has an urban area that can reach 3.925 km^2, and its metropolitan area can be 14.034 km^2. How to deal with such huge spatial domain is the first challenge; (2) Collected data used for model training are often limited to a small portion of the total population, and we can't collect every citizen's long-term historical trajectory data. How to train an effective model with limited training data (e.g., 1% of the total population) is the second challenge; (3) Under some real-world application scenario such as crowd management and crowd monitoring, it is important to predict crowd mobility as well as crowd density, especially for the latter one. Because high crowd density naturally means high risk for accidents. How to simultaneously predict crowd density and crowd flow is the third challenge; (4) Sometimes we may have no time or historical data for training a prediction model. Moreover, when some big events happen such as an earthquake, typhoon, and national festival, people change their behaviors from their routine activities. Thus, a model trained with historical data can't work very well for such kind of circumstances. Recently, the success of deep learning in the fields of computer vision and natural language processing motivates us

to consider deep learning techniques as highly potential solutions to our problems, because it has the following advantages: (1) It can handle real big data; (2) It can model highly complex spatiotemporal system; (3) It can deal with multimodal distribution; (4) It can fuse multiple heterogeneous data. Then we propose four deep-learning-based solutions to address those four challenges and demonstrate the superior performances to the baseline methodologies. Finally, we summarize all of the works including some complementary works, discuss the limitations of current proposed solutions, and point out the future works.

Each task in the citywide human mobility is an under-explored problem, there is a large room for improvement in the future: 1) improving accuracy 2)new application scenarios. Also we need to explore more challenges/problems on citywide human mobility prediction, and propose corresponding solutions. We also try to apply deep learning technologies to traffic flow/density prediction on transportation network. Our study has some room for improvement in the following aspects: (1) Our approach is still struggling to deal with situations in which urban mobility is full of sudden changes such as morning rush hour. (2) Currently, only one month's data was used for our experimental evaluation. To validate the scalability and improve the overall performance of our approach, we need to try more trajectory data over one month. (3) In addition to POI data, transportation network data and other types of heterogeneous data such as the population density can be utilized as geographical features. (4) Current framework only takes spatial information into account, temporal information could also be used to improve the performance, and conduct the time-series modeling for citywide human mobility. (5) The current dataset only contained approximately 1% of the total population of Japan. Thus, we need to collect more trajectory data from other sources and design reasonable scaling factors in order to simulate and predict citywide human mobility in a more realistic manner. (6) Advanced deep learning technologies such as ResNet[96] and PredNet[97] can also be utilized to further boost the performance, especially for the crowd flow prediction part. (7) Exploit the maximum prediction lead time of our system. (8) More sophisticated preprocessing will be included to improve the overall performance of our system. Particularly, we will apply map matching algorithm and trajectory calibration algorithm [32] to improve the quality of the raw trajectories.

Bibliography

[1] Defu Lian, Xing Xie, Vincent W Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. Cepr: A collaborative exploration and periodically returning model for location prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):8, 2015.

[2] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 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*, pages 559–569. ACM, 2015.

[3] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.

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

[5] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[6] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM, 2009.

[7] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 70–79. Ieee, 2008.

[8] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1099–1108. ACM, 2010.

[9] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. In *International Conference on Pervasive Computing*, pages 152–169. Springer, 2011.

[10] Jiangchuan Zheng and Lionel M Ni. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 153–162. ACM, 2012.

[11] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015.

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

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

[14] Masamichi Shimosaka, Keisuke Maeda, Takeshi Tsukiji, and Kota Tsubouchi. Forecasting urban dynamics with mobility logs by bilinear poisson regression. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pages 535–546. ACM, 2015.

[15] Jingbo Zhou, Hongbin Pei, and Haishan Wu. 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*, pages 19–41. Springer, 2018.

[16] Yasunori Akagi, Takuya Nishimura, Takeshi Kurashima, and Hiroyuki Toda. A fast and accurate method for estimating people flow from spatiotemporal population data. In *IJCAI*, pages 3293–3300, 2018.

[17] Akihito Sudo, Takehiro Kashiyama, Takahiro Yabe, Hiroshi Kanasugi, Xuan Song, Tomoyuki Higuchi, Shinya Nakano, Masaya Saito, and Yoshihide Sekimoto. 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*, page 5. ACM, 2016.

[18] Yusuke Tanaka, Tomoharu Iwata, Takeshi Kurashima, Hiroyuki Toda, and Naonori Ueda. Estimating latent people flow without tracking individuals. In *IJCAI*, pages 3556–3563, 2018.

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

[20] Huichu Zhang, Yu Zheng, and Yong Yu. Detecting urban anomalies using multiple spatio-temporal data sources. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):54, 2018.

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

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

[23] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Teerayut Horanont, Satoshi Ueyama, and Ryosuke Shibasaki. 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*, pages 1231–1239. ACM, 2013.

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

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

[26] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 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*, pages 831–840. ACM, 2014.

[27] Fengli Xu, Tong Xia, Hancheng Cao, Yong Li, Funing Sun, and Fanchao Meng. Detecting popular temporal modes in population-scale unlabelled trajectory data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):46, 2018.

[28] Zhihan Fang, Fan Zhang, Ling Yin, and Desheng Zhang. Multicell: Urban population modeling based on multiple cellphone networks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):106, 2018.

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

[30] Po-Ta Chen, Feng Chen, and Zhen Qian. Road traffic congestion monitoring in social media with hinge-loss markov random fields. In *2014 IEEE International Conference on Data Mining*, pages 80–89. IEEE, 2014.

[31] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[32] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 833–844. ACM, 2013.

[33] Ying Wei, Yu Zheng, and Qiang Yang. Transfer knowledge between cities. In *KDD*, pages 1905–1914, 2016.

[34] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1653–1662. ACM, 2017.

[35] Chenyi Zhuang, Nicholas Jing Yuan, Ruihua Song, Xing Xie, and Qiang Ma. Understanding people lifestyles: construction of urban movement knowledge graph

from gps trajectory. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[36] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 239–252. Acm, 2012.

[37] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. Intelligent system for urban emergency management during large-scale disaster. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8208.

[38] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Ryosuke Shibasaki, Nicholas Jing Yuan, and Xing Xie. A simulator of human emergency mobility following disasters: Knowledge transfer from big disaster data. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[39] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[40] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, pages 194–200, 2016.

[41] Di Yao, Chao Zhang, Jianhui Huang, and Jingping Bi. Serm: A recurrent model for next location prediction in semantic trajectories. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2411–2414. ACM, 2017.

[42] Yuki Endo, Kyosuke Nishida, Hiroyuki Toda, and Hiroshi Sawada. Predicting destinations from partial trajectories using recurrent neural network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2017.

[43] Zipei Fan, Xuan Song, Tianqi Xia, Renhe Jiang, Ryosuke Shibasaki, and Ritsu Sakuramachi. Online deep ensemble learning for predicting citywide human mobility. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):105, 2018.

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

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

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

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

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

[49] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *IJCAI*, pages 2618–2624, 2016.

[50] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021*, 2015.

[51] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1459–1468. International World Wide Web Conferences Steering Committee, 2018.

[52] Ziheng Lin, Mogeng Yin, Sidney Feygin, Madeleine Sheehan, Jean-Francois Paiement, and Alexei Pozdnoukhov. Deep generative models of urban mobility. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[53] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 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*, pages 984–992. ACM, 2018.

[54] Saurav Ranjit, Apichon Witayangkurn, Masahiko Nagai, and Ryosuke Shibasaki. Agent-based modeling of taxi behavior simulation with probe vehicle data. *ISPRS International Journal of Geo-Information*, 7(5):177, 2018.

[55] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[56] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339. ACM, 2007.

[57] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups of-trackable moving point objects. In *International Conference on Geographic Information Science*, pages 132–144. Springer, 2002.

[58] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2): 723–734, 2010.

[59] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[60] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. On discovery of gathering patterns from trajectories. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 242–253. IEEE, 2013.

[61] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas La Porta. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *Proceedings of the VLDB Endowment*, 7(9):769–780, 2014.

[62] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. 2011.

[63] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M Ni. Finding time period-based most frequent path in big trajectory data. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, pages 713–724. ACM, 2013.

[64] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[65] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[66] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and MC Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224, 2001.

[67] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[68] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[70] Alex Graves. *Supervised Sequence Labeling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. ISBN 978-3-642-24796-5. URL http://dx.doi.org/10.1007/978-3-642-24797-2.

[71] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[72] Francois Chollet. keras. https://github.com/fchollet/keras, 2015.

[73] 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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[74] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[75] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[76] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[77] Xuedong Huang, Fileno Alleva, Hsiao-Wuen Hon, Mei-Yuh Hwang, Kai-Fu Lee, and Ronald Rosenfeld. The sphinx-ii speech recognition system: an overview. *Computer Speech & Language*, 7(2):137–148, 1993.

[78] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[79] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[80] Richard Socher, John Bauer, Christopher D Manning, et al. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465, 2013.

[81] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

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

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

[84] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[85] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[86] Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Quanjun Chen, Qi Chen, and Ryosuke Shibasaki. Deep roi-based modeling for urban human mobility prediction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):14, 2018.

[87] Yu Yang, Fan Zhang, and Desheng Zhang. 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):48, 2018.

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

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

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

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

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

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

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

[95] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2012.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[97] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.