Transfer Urban Human Mobility via POI Embedding over Multiple Cities

RENHE JIANG, The University of Tokyo
XUAN SONG and ZIPEI FAN, SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology, The University of Tokyo
TIANQI XIA and ZHAONAN WANG, The University of Tokyo
QUANJUN CHEN, SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology, The University of Tokyo
ZEKUN CAI and RYOSUKE SHIBASAKI, The University of Tokyo

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

This work was supported by Grant-in-Aid for Early-Career Scientists (No. 20280241) of Japan Society for the Promotion of Science (JSPS).
Authors' addresses: R. Jiang, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: jiangrh@csis.u-tokyo.ac.jp; X. Song (corresponding author), SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology, The University of Tokyo, 1088 Xueyuan Avenue, Shenzhen, China; email: songxuan@csis.u-tokyo.ac.jp; Z. Fan, SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: fanzipei@iis.utokyo.ac.jp; T. Xia, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: xiatianqi@csis.u-tokyo.ac.jp; Z. Wang, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: znwang@csis.u-tokyo.ac.jp; Q. Chen, SUSTech-UTokyo Joint Research Center on Super Smart City, Southern University of Science and Technology, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: chen1990@iis.u-tokyo.ac.jp; Z. Cai, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: caizekun@csis.u-tokyo.ac.jp; R. Shibasaki, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Japan; email: shiba@csis.u-tokyo.ac.jp.

2577-3224/2020/12-ART4 $15.00
https://doi.org/10.1145/3416914

CCS Concepts: • **Information systems → Information systems applications**; **Geographic information systems**; • **Computing methodologies → Artificial intelligence**; • **Human-centered computing → Ubiquitous and mobile computing**;

Additional Key Words and Phrases: Big data, human mobility, urban computing, deep learning, transfer learning

ACM Reference format:
Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Zhaonan Wang, Quanjun Chen, Zekun Cai, and Ryosuke Shibasaki. 2020. Transfer Urban Human Mobility via POI Embedding over Multiple Cities. *ACM/IMS Trans. Data Sci.* 2, 1, Article 4 (December 2020), 26 pages.
https://doi.org/10.1145/3416914

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 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, which is approximately 1% of the total population 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 [3, 52, 53, 56]. 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.

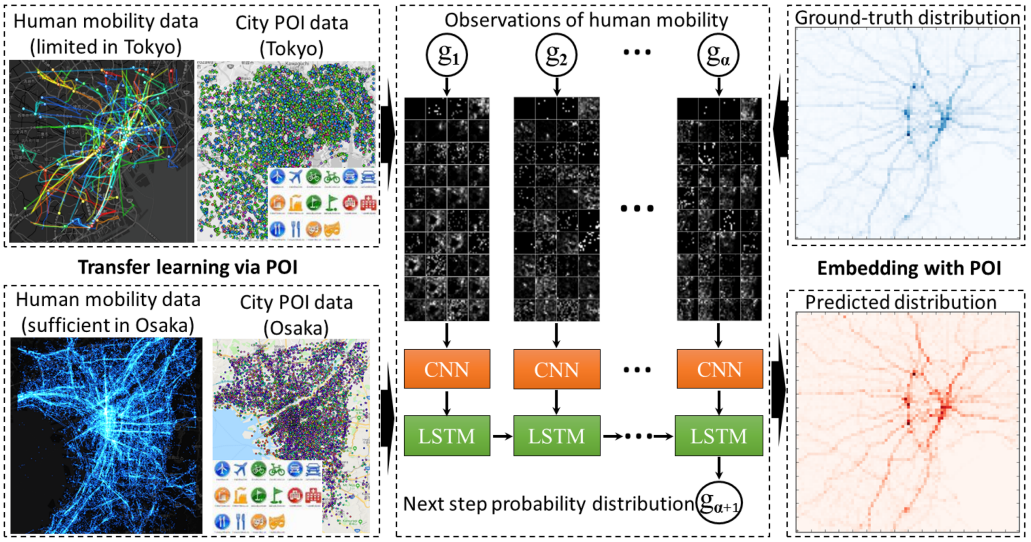


Fig. 1. Can we design an effective approach to build one urban model for predicting human mobility (future distribution of individuals) at a citywide level with limited data? Fusing heterogeneous data (Human mobility data and city POI data) with deep learning technologies may allow us to address this challenge.

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 Figure 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 has 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 article is organized as follows. In Section 2, we introduce our data sources. In Section 3, we illustrate the modeling of citywide human mobility using a deep learning architecture. In Section 4, we explain the details of image-like embedding and transfer learning. We present the results of the experimental evaluation in Section 5, and discuss the results in Section 6. Related works are summarized in Section 7. In Section 8, we give our conclusions and future works.

2 DATA SOURCE

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 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 100 m, 200 m, or 300 m).

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 1. The total numbers of POIs for Tokyo, Osaka, Fukuoka, Sapporo and Tottori were 281,400, 153,377, 47,418, 73,635, and 17,743, respectively, which were used as the five 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.

3 CITYWIDE HUMAN MOBILITY MODELING

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 can indicate a person's

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

Table 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		

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 pre-processing 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. 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 $\{g_1, g_2, \ldots, g_K\}$ and a raw trajectory $\{(t_1, l_1), (t_2, l_2), \ldots, (t_m, l_m)\}$, a grid-mapped human trajectory $traj$ is defined as a sequence of mesh-grids:

$$traj = (t_1, g_1), (t_1, g_2), \ldots, (t_m, g_m), \forall i, l_i \in g_i. \tag{1}$$

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 (Citywide Human Mobility). 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

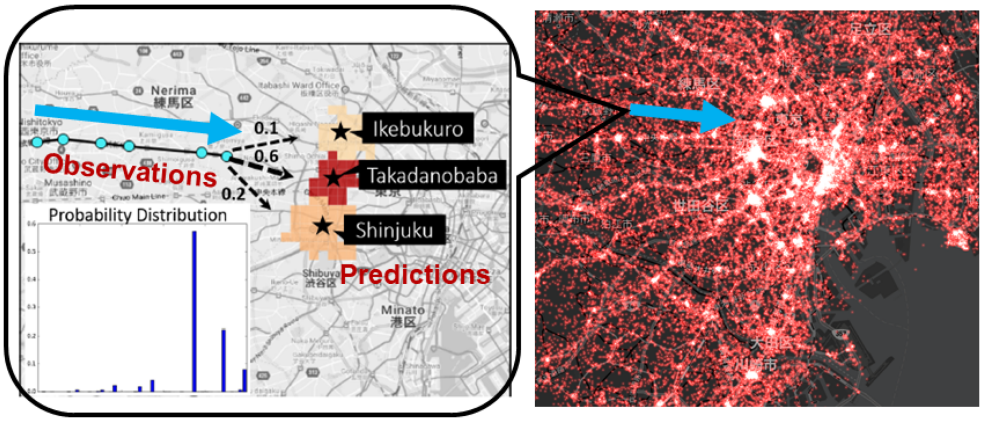


Fig. 2. Citywide Human Mobility Prediction.

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 $x = g_1, g_2, \ldots, g_\alpha$, whereas the corresponding next-step trajectory snippet is represented as $y = g_{\alpha+1}$. Then, the observed human mobility prediction X, and the next step of citywide human mobility Y can be represented as follows:

$$X = \{x^1, x^2, \ldots, x^N\}, Y = \{y^1, y^2, \ldots, y^N\}, \tag{2}$$

where N denotes for the number of samples, i.e., the number of people whose mobility data are available for model training.

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

$$\theta = \underset{\theta}{\operatorname{argmin}} H(P(\widehat{Y} \mid X), Q(Y \mid X)), \tag{3}$$

where Y denotes the ground-truth for the next-step mobility distribution, $\widehat{Y}$ 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. Furthermore, one trajectory snippet x is essentially representing one pattern of urban human mobility within α observation steps, since the same mobiliy $g_1, g_2, \ldots, g_\alpha$ can be observed from a group of different people at different time peoriods. For instance, as shown in Figure 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 six-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 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 [6, 20] 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 [8]. 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 [45] and sentiment analysis [46]. 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 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 6,400 500 m× 500 m mesh-grids in our study, whereas there are just 2,500 to 3,000 most common words in

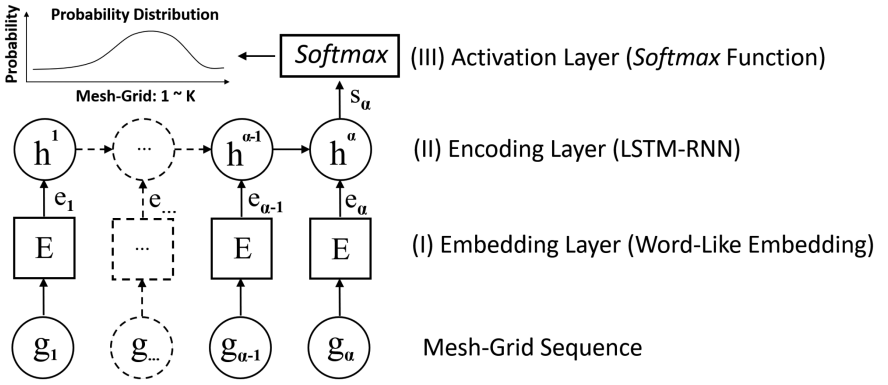


Fig. 3. Deep Sequential Modeling Architecture (Word-like Embedding).

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 Figure 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 $(e_1, e_2, \ldots, e_\alpha)$ 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 a probability distribution over K different mesh-grids $\{g_1, g_2, \ldots, g_K\}$. 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 $(e_1, e_2, \ldots, e_\alpha)$ as follows:

$$i_\alpha = \sigma(W_i e_\alpha + U_i s_{\alpha-1} + b_i), \tag{4}$$

$$f_\alpha = \sigma(W_f e_\alpha + U_f s_{\alpha-1} + b_f), \tag{5}$$

$$o_\alpha = \sigma(W_o e_\alpha + U_o s_{\alpha-1} + b_o), \tag{6}$$

$$\widetilde{C_\alpha} = tanh(W_c e_\alpha + U_c s_{\alpha-1} + b_c), \tag{7}$$

$$C_\alpha = i_\alpha \odot \widetilde{C_\alpha} + f_\alpha \odot C_{\alpha-1}, \tag{8}$$

$$s_\alpha = o_\alpha \odot tanh(C_\alpha), \tag{9}$$

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 (3).

4 EMBEDDING AND TRANSFERRING

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.

4.1 Image-like Embedding 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:

$$\begin{aligned} g.POI &= (f_1, f_2, \ldots, f_\sigma), \forall i \in [1, \sigma], \\ f_i &= |\{poi \mid poi.coordinate \in g \wedge poi.category = i\}|, \end{aligned} \tag{10}$$

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:

$$r = \left\{ g' \mid |g'.ix - g.ix| \leq \frac{\eta - 1}{2} \wedge |g'.iy - g.iy| \leq \frac{\eta - 1}{2} \right\}, \tag{11}$$

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 g can be extended to obtain an artificial POI image ρ represented by an $\eta \times \eta \times \sigma$ tensor. To this end, the model input has been extended from mesh-grid sequence $\{g_1, g_2, \ldots, g_\alpha\}$ to poi-image sequence $\{\rho_1, \rho_2, \ldots, \rho_\alpha\}$.

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 [27], 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, convolutional layers and pooling layers are employed in our deep sequential learning architecture as an advanced embedding component. The convolutional feature $f_{i,j}^{(conv)}$ at pixel (i, j) is calculated as

$$f_{i,j}^{(conv)} = ReLU(w \cdot px_{i,j} + b), \tag{12}$$

where w and b are the weight and bias matrix, and $px_{i,j}$ 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 $f_{i,j}^{(pool)}$ is calculated using the max-pooling operation:

$$f_{i,j}^{(pool)} = MaxPooling\left(f_{m,n}^{(conv)}\right), \forall (m,n) \in \mathbb{R}_{ij}, \tag{13}$$

where $\mathbb{R}_{ij}$ is the local neighborhood around pixel (i, j). By stacking several convolutional and pooling layers (2 conv layers → 1 pool layer → 2 conv layers → 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 and the features extracted by the CNNs step by step are visualized in the first row of Figure 4. Similarly, the processing paths for the POI images of Shinjuku Station and Shinagawa Station are displayed in the second/third row of Figure 4, respectively. To better illustrate how image-like embedding is performed, the flowchart has been drawn as Figure 5 by taking a grid-mapped human trajectory (Tokyo Station → Shinjuku Station → Shinagawa Station) as an example.

Originally, grid-mapped human trajectory is represented as a $\alpha \times 1$ vector, i.e., $\{g_1, g_2, \ldots, g_\alpha\}$. 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, i.e., $\{\rho_1, \rho_2, \ldots, \rho_\alpha\}$. By replacing naive word-like embedding with CNN-based image-like embedding, our deep sequential learning architecture essentially becomes a deep video model as shown in Figure 6, which takes a four-dimensional shape tensor as the input. The hierarchical geographical features inside a region are extracted by stacked CNNs 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 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 Reference [18]. LSTM-on-CNNs architecture has been proposed in the filed of computer vision for visual recognition and description [11]. 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.

4.2 Transfer Learning via POI Embedding

Transfer learning is a powerful tool that helps deep learning models to achieve better performance [40]. 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

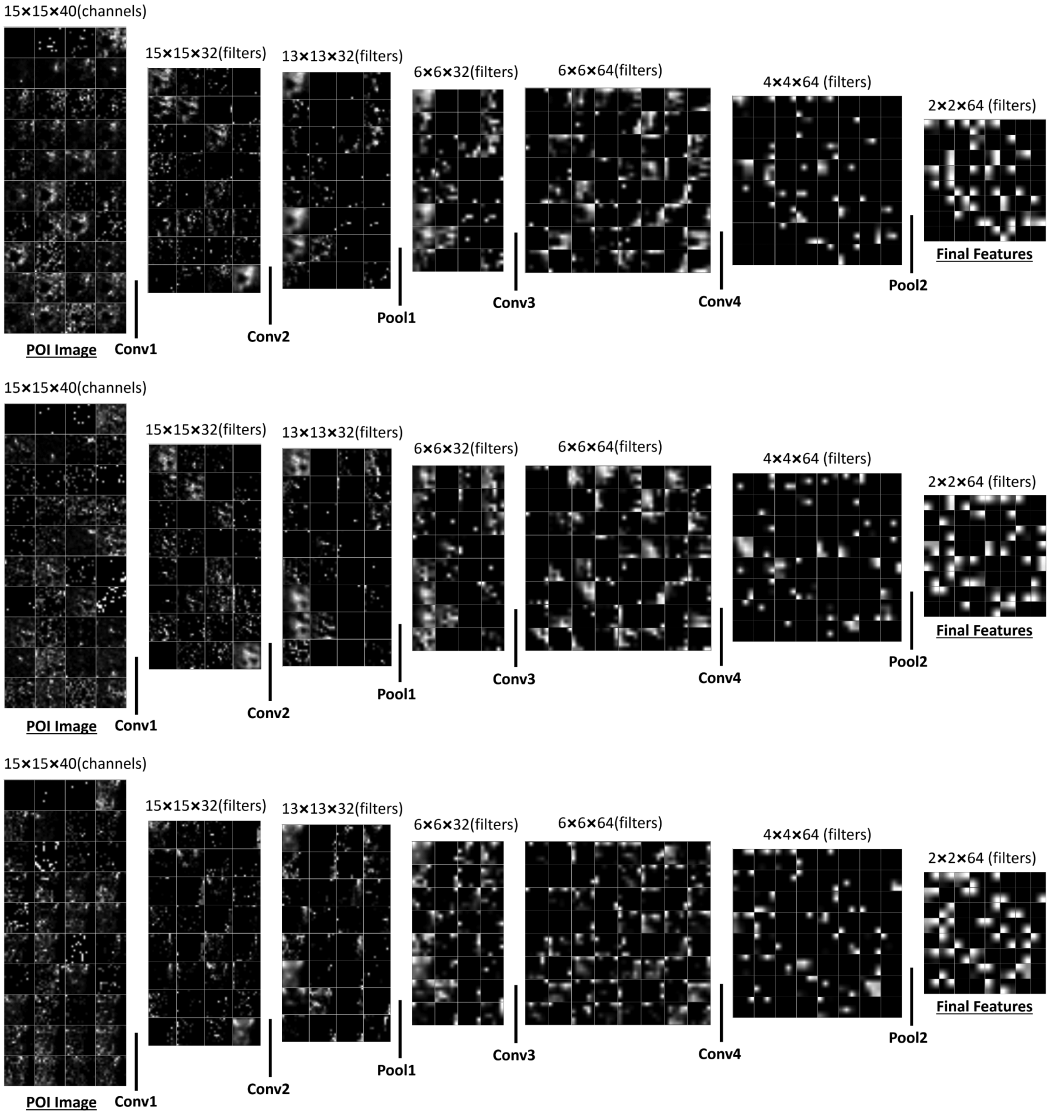


Fig. 4. Visualization of the extracted features by embedding CNNs for a 15 × 15 POI image with the grid containing [Tokyo Station (1st row), Shinjuku Station (2nd row), Shinagawa Station (3rd row)] as its centroid. The feature maps for each layer along the processing path are displayed in a block, where 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).

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

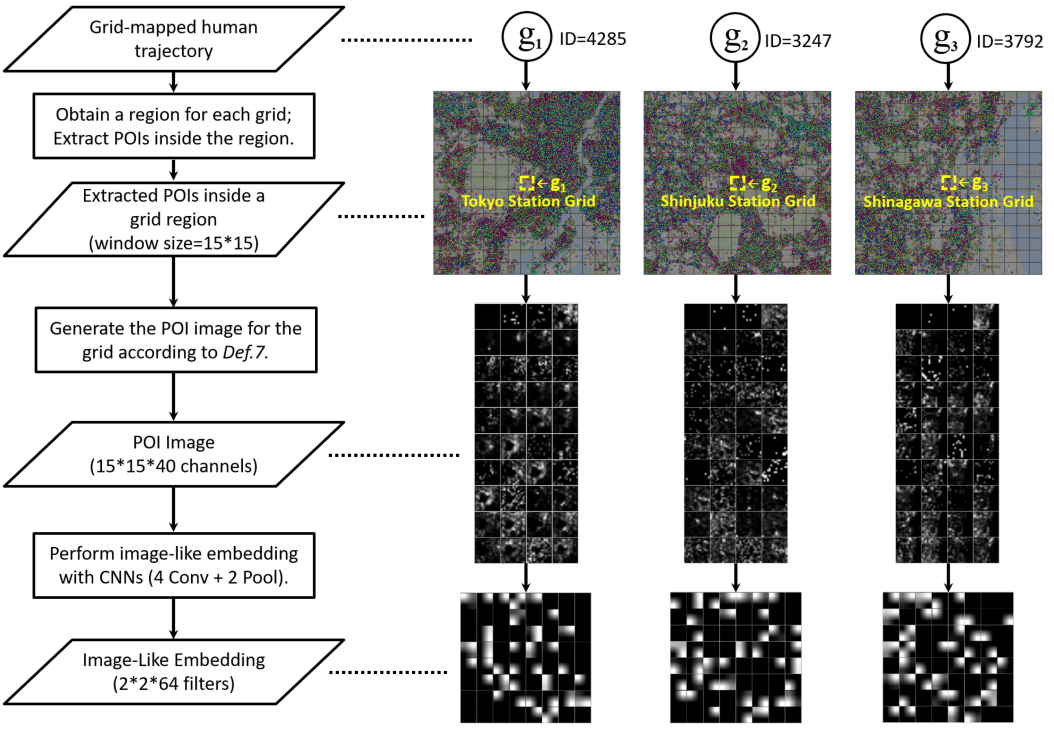


Fig. 5. Flowchart of image-like embedding using a grid-mapped human trajectory g_1 (Tokyo Station) $\rightarrow$ g_2 (Shinjuku Station) $\rightarrow$ g_3 (Shinagawa Station) as an example. For each mesh-grid in the given trajectory, first a region with the mesh-grid as its centroid will be obtained according to Definition 6, then the POIs inside the region will be extracted to generate the POI image according to Definition 7. Through a series of CNNs, the extracted final features from the POI image can be seen as the embedding result.

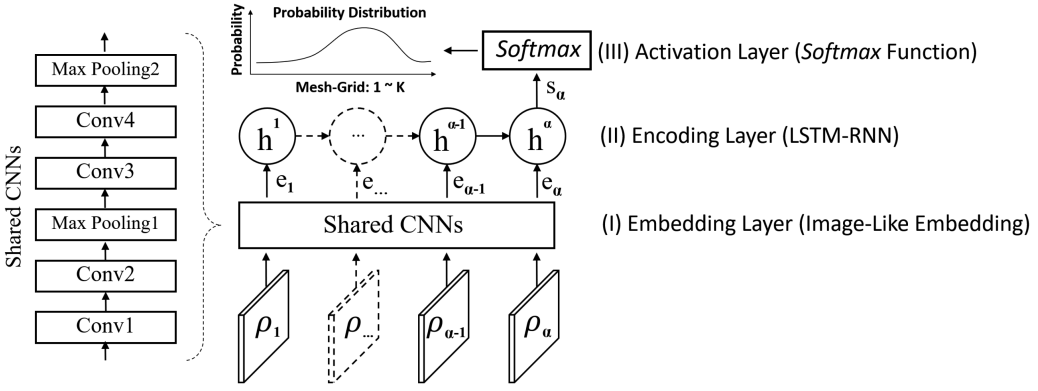


Fig. 6. Deep Sequential Modeling Architecture (Image-like Embedding).

trained or embedded due to the lack of corresponding training data in A.[2] Meanwhile, the CNN architectures for image-like embedding can remain the same for different cities. These advantages

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

of image-like embedding over word-like embedding will be further validated and discussed in Section 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 with the data from 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 the data from target city. Experiments showed that the latter obtained better performance, when limiting the training data of target city to 1%, 5%, and 10% of the total. Note that the transfer learning technique employed here is essentially "few-shot learning" [42], namely, to build the prediction model with few training samples. The few-shot setting originates from the motivation mentioned in Section 1, that is the data limitation issue for training human mobility prediction model in a real-world application scenario. For different cities, we may collect the GPS trajectories from different data sources, such as taxi GPS data collected from car-hailing platform (DiDi Chuxing or Uber), bike GPS data collected from bicycle-sharing system (Mobike, Ofo, Jump), mobile phone GPS data collected from telecom operators (NTT DoCoMo, AU, Softbank in Japan or China Telecom and China Unicom), and geo-tagged GPS data collected from location-based social networks (Twitter, Facebook, Foursquare). Even for one data source, due to the user privacy issue, companies may only provide sample data for research purposes. Thus, each data source could only cover a small portion of the total population of a city. For example, in our study, the data we collected through NTT DoCoMo could cover just approximately 1% of the total population. Last, considering that our goal is to build a prediction model based on the observed human mobility, we have to input the partial trajectories anyway for a new city, so "zero-shot learning" may not be an appropriate setting for a mobility prediction task like ours.

5 EXPERIMENT

Experimental Setup: We randomly selected two consecutive weeks as our experimental period and conducted evaluations in five 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.[3] 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. By utilizing these four 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.[4] More geographical details about these five areas are summarized as Table 2. Python and some Python libraries such as Keras [7] and TensorFlow [1] 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, and 7,268 trajectories were generated for Tokyo, Osaka, Fukuoka, Sapporo, and Tottori, respectively, where the average trajectory lengths (number of points) were approximately 85.0, 76.5, 69.3, 67.7, 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 3. We randomly selected 60% of

[3] https://en.wikipedia.org/wiki/Japan.
[4] https://en.wikipedia.org/wiki/Tottori_Prefecture.

Table 2. Geographic Details of Experimental Cities

City	Geographic Interval	No. 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
Tottori	$Long. \in [134.12, 134.32]$, $Lat. \in [35.44, 35.56]$	1,200

Table 3. Data Information of Experimental Cities

City	No. of Trajectory	Average Trajectory Length	No. of Samples (x-y pair)
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
Tottori	7,268	58.2	386,751

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 450 m $\times$ 450 m) 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 and 1,200 mesh-grids were generated for the Sapporo 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 1,024 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) KNN. A KNN-based learning model [9] 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 [41] is built to predict the target value by learning simple decision rules.
(4) RandomForest. A random forest [32] 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) Grid-POI Vector. This model only considered the POIs inside each grid. According to *Definition 5*, we could count the frequencies of each POI category and get a 40-dimension POI vector for each grid. Each trajectory snippet was then represented by a sequence of 40-dimensional POI vectors. The vectors were scaled into [0,1]. To handle these POI-vector sequences as inputs, an LSTM-RNN layer with 256 hidden units was utilized as the embedding layer. An additional LSTM-RNN layer with 256 hidden units followed as the encoding layer.

Table 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 2)
Learning Rate	RMSprop Optimizer	0.001
Batch Size	Training Process	1024

(6) Grid-POI Vector+Transfer. This model applied transfer learning to "Grid-POI Vector" model mentioned above. The embedding layer (1st LSTM-RNN layer) was pre-trained with sufficient data from other urban area. Then the model was continuously trained with the limited data from the target area. The other settings on transfer learning were kept the same as "Word-like Embedding+Transfer" (8).

(7) Word-like Embedding. This is the deep learning model shown in Figure 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 Figure 3 also contained 256 hidden units.

(8) Word-like Embedding+Transfer. This model applied transfer learning to "Word-like Embedding" model 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 model (7) 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 model (7).

(9) 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.

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 (9). The embedding layer and the encoding layer were pre-trained with the same transfer learning settings mentioned in (8). All the parameter settings of the experiments are summarized as Table 4.

Table 5. Performance Evaluation of Citywide Human Mobility Prediction for Tokyo

Model (Tokyo)	1% data	5% data	10% data	50% data
N-Gram	7.06	4.43	3.42	1.91
KNN	7.07	3.54	2.81	1.95
DecisionTree	7.28	4.11	3.02	1.74
RandomForest	5.07	2.68	2.37	1.77
Grid-POI Vector	5.86	3.13	2.57	1.88
Grid-POI Vector + Transfer^{O}	4.43	2.72	2.33	1.82
Word-like Embedding	4.52	2.21	1.73	**1.23**
Image-like Embedding	3.10	1.73	1.53	1.25
Word-like Embedding + Transfer^{O}	4.27	2.18	1.75	1.28
Image-like Embedding + Transfer^{O}	**2.75**	**1.65**	**1.51**	1.28

Table 6. Performance Evaluation of Citywide Human Mobility Prediction for Osaka

Model (Osaka)	1% data	5% data	10% data	50% data
N-Gram	7.07	4.51	3.52	1.97
KNN	7.19	3.61	2.87	2.03
DecisionTree	7.19	4.06	2.96	1.73
RandomForest	4.90	2.60	2.30	1.75
Grid-POI Vector	5.76	2.90	2.31	1.69
Grid-POI Vector + Transfer^{T}	4.22	2.48	2.12	1.64
Word-like Embedding	4.49	2.08	1.63	**1.19**
Word-like Embedding + Transfer^{T}	3.96	2.03	1.65	1.23
Image-like Embedding	2.85	1.63	1.45	1.21
Image-like Embedding + Transfer^{T}	**2.62**	**1.56**	**1.40**	1.21

Evaluation metric: We evaluated the performance of the proposed models using Cross-entropy, which 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 is used as the primary metric in the evaluation, which is defined as follows:

$$CrossEntropy = \frac{1}{n}\sum_{i}^{n}\sum_{k}^{K} -y_i^{(k)} \log\left(\hat{y}_i^{(k)}\right) \tag{14}$$

where n is the number of samples, K is the mesh-grid number for each urban area, $y^{(k)}$ and $\hat{y}^{(k)}$ are the true probability and predicted probability based on each mesh-grid, respectively.

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 5~9, which shows that based on all five cities: (1) our model performed better than the others when the amount of training data was small (1%, 5%, and 10%); (2) embedding the trajectory into a POI-vector sequence simply by considering the POIs inside each grid was not sufficiently effective; (3) when training data was small (1%, 5%, and 10%), deep learning models using work-like embedding could not be effectively trained either; (4) when 50% of the data were used as training data, deep-learning models using word-like embedding performed better than the other models; (6) even without transfer learning, image-like embedding still performed better than word-like embedding or Grid-POI vector with small training datasets, where the advantage increased as the

Table 7. Performance Evaluation of Citywide Human Mobility Prediction for Fukuoka

Model (Fukuoka)	1% data	5% data	10% data	50% data
N-Gram	6.47	3.78	2.91	1.77
KNN	5.28	3.30	2.66	2.10
DecisionTree	5.63	3.44	2.52	1.66
RandomForest	3.49	2.23	2.00	1.70
Grid-POI Vector	3.96	2.19	1.83	1.43
Grid-POI Vector + Transfer^{T}	3.31	2.06	1.78	1.42
Word-like Embedding	3.04	1.66	1.44	**1.21**
Word-like Embedding + Transfer^{T}	2.93	1.73	1.49	1.23
Image-like Embedding	2.20	1.55	1.43	1.25
Image-like Embedding + Transfer^{T}	**2.10**	**1.50**	**1.39**	1.24

Table 8. Performance Evaluation of Citywide Human Mobility Prediction for Sapporo

Model (Sapporo)	1% data	5% data	10% data	50% data
N-Gram	6.37	3.16	2.40	1.58
KNN	4.42	2.53	2.20	1.70
DecisionTree	5.39	2.53	2.08	1.62
RandomForest	3.88	2.00	1.75	1.52
Grid-POI Vector	4.01	2.04	1.67	1.25
Grid-POI Vector + Transfer^{T}	3.11	1.85	1.56	1.23
Word-like Embedding	2.78	1.50	1.30	**1.09**
Word-like Embedding+Transfer^{T}	2.56	1.53	1.32	1.11
Image-like Embedding	2.09	1.47	1.33	1.14
Image-like Embedding+Transfer^{T}	**1.95**	**1.39**	**1.28**	1.14

Table 9. Performance Evaluation of Citywide Human Mobility Prediction for Tottori

Model (Tottori)	1% data	5% data	10% data	50% data
N-Gram	5.66	2.75	2.12	1.41
KNN	4.30	2.54	2.09	1.52
DecisionTree	4.59	2.54	2.09	1.49
RandomForest	3.85	2.03	1.72	1.43
Grid-POI Vector	4.33	2.06	1.67	1.24
Grid-POI Vector + Transfer^{T}	3.08	1.87	1.58	1.22
Word-like Embedding	2.86	1.45	1.22	**0.99**
Word-like Embedding+Transfer^{T}	2.73	1.50	1.26	1.01
Image-like Embedding	2.17	1.36	1.22	1.02
Image-like Embedding+Transfer^{T}	**2.07**	**1.31**	**1.17**	1.01

amount of training data became smaller. In general, we achieved consistent experiment results on all five cities. Our proposed methodology had a clearer advantage for larger urban areas (Tokyo and Osaka) when using less training dataset (1%∼10%).

Case Study: Cross Entropy is a widely used measurement for comparing distributions, and we also aim to evaluate our model in a more intuitive way through some case studies. The

Table 10. Case Study on Shinjuku Station Area (Tokyo)

Model (Tokyo)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	−2,003	12.67%	−386	2.44%
KNN	1,160	7.34%	985	6.23%
DecisionTree	−3,887	24.60%	−295	1.87%
RandomForest	430	2.72%	288	1.82%
Grid-POI Vector	−1,276	8.07%	105	0.66%
Grid-POI Vector+Transfer^{O}	915	5.79%	87	0.55%
Word-like Embedding	448	2.83%	45	0.28%
Word-like Embedding+Transfer^{O}	1,838	11.63%	384	2.43%
Image-like Embedding	403	2.55%	49	0.31%
Image-like Embedding+Transfer^{O}	**122**	**0.77%**	**8**	**0.05%**

ground-truth probability can be represented more intuitively through the density of the population. Therefore, two additional metrics regarding density were employed in the case studies to check the ground-truth density and the predicted density for selected areas in the city. 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:

$$Err = \hat{d}_{area} - d_{area}, \tag{15}$$

$$RE = \frac{|\hat{d}_{area} - d_{area}|}{d_{area}}, \tag{16}$$

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*.

Using the metrics *Err* and *RE*, we conducted three case studies to compare the ground-truth density and predicted density. The first case study is on Shinjuku Station area (Tokyo), which can be seen as a typical central business area. This area consists of 6 × 6 neighboring mesh-grids, with Shinjuku Station locating at the central mesh-grid. The results conducted with 1% and 5% training data are listed as Table 10, from which we can also see that our proposed model **Image-like Embedding+Transfer** could achieve the best performance. The second case study listed as Table 11 is on The University of Tokyo area (Tokyo), which is taken as a typical educational area (containing 5 × 5 neighboring mesh-grids). The third case study as shown in Table 12 is on the Odori Park area (Sapporo), which is a famous sightseeing spot located in the center of Sapporo City (containing 5 × 5 neighboring mesh-grids). Similar results were achieved on these two case studies as the first one. Moreover, when using 1% training data, we plot the locations of the mesh-grids on our proposed model **Image-like Embedding+Transfer** could achieve the best prediction performance based on the metrics *Err* and *RE*. Three cities Tokyo (red marker), Osaka (green marker), and Sapporo (yellow marker) are taken as three examples illustrated as Figure 7. Through it, we could observe that **Image-like Embedding+Transfer** was the most effective model on most of the urban area when training data was limited to 1%.

Table 11. Case Study on Tokyo University Area (Tokyo)

Model (Tokyo)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	−636	13.00%	−183	3.74%
KNN	69	1.41%	−129	2.64%
DecisionTree	−2,833	57.91%	−469	9.59%
RandomForest	232	4.74%	207	4.23%
Grid-POI Vector	305	6.23%	88	1.80%
Grid-POI Vector+Transfer^{O}	−70	1.43%	294	6.01%
Word-like Embedding	942	19.26%	−95	1.94%
Word-like Embedding+Transfer^{O}	761	15.56%	171	3.50%
Image-like Embedding	495	10.12%	−137	2.80%
Image-like Embedding+Transfer^{O}	**44**	**0.90%**	**64**	**1.31%**

Table 12. Case Study on Odori Park Area (Sapporo)

Model (Sapporo)	1% data		5% data	
	Err	RE	Err	RE
N-Gram	−1,925	8.89%	−769	3.55%
KNN	179	0.83%	1,331	6.14%
DecisionTree	−1,677	7.74%	606	2.80%
RandomForest	1,044	4.82%	839	3.87%
Grid-POI Vector	986	4.55%	297	1.37%
Grid-POI Vector+Transfer^{T}	927	4.28%	802	3.70%
Word-like Embedding	−495	2.29%	94	0.43%
Word-like Embedding+Transfer^{T}	570	2.63%	530	2.45%
Image-like Embedding	726	3.35%	−93	0.43%
Image-like Embedding+Transfer^{T}	**−128**	**0.59%**	**59**	**1.31%**

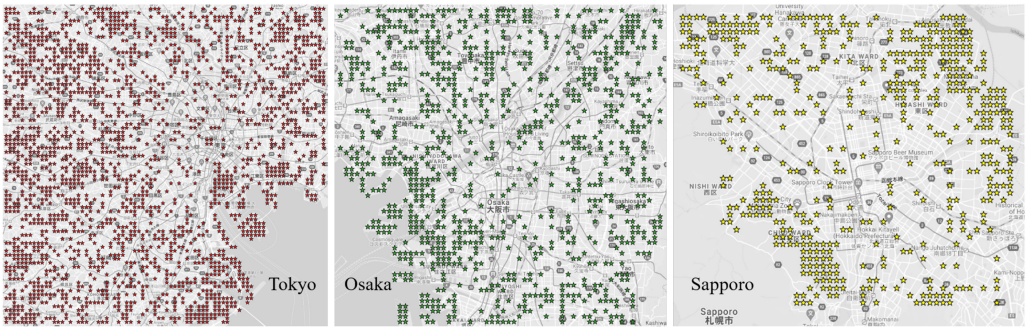
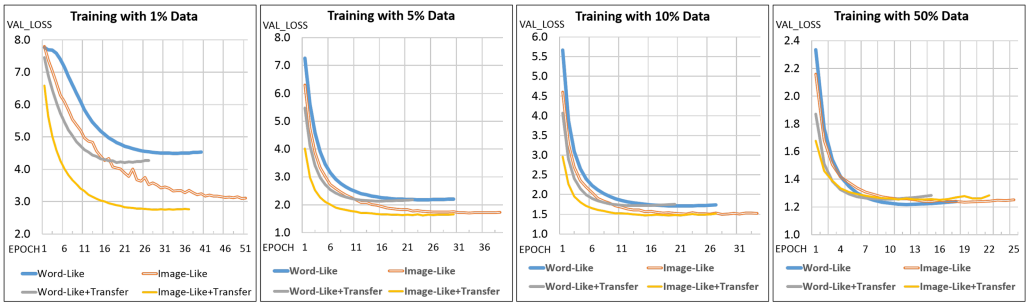
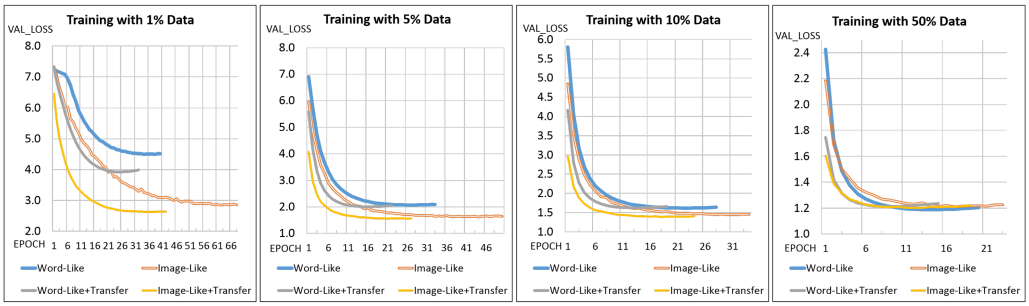


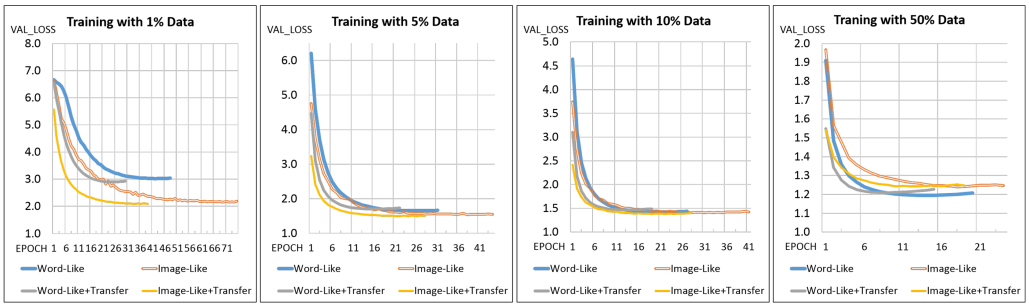
Fig. 7. Visualization of the locations on which **Image-like Embedding+Transfer** could give the best prediction performance based on *Err* and *RE* when using 1% training data. The locations are represented with colored marker. From left to right, Tokyo, Osaka, and Sapporo are given as three examples, represented with red, green, and yellow marker, respectively.



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



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



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

Fig. 8. Learning curves of transfer learning on three cities.

6 DISCUSSION

First, we continue to verify 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 Figure 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.

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

Table 13. 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**

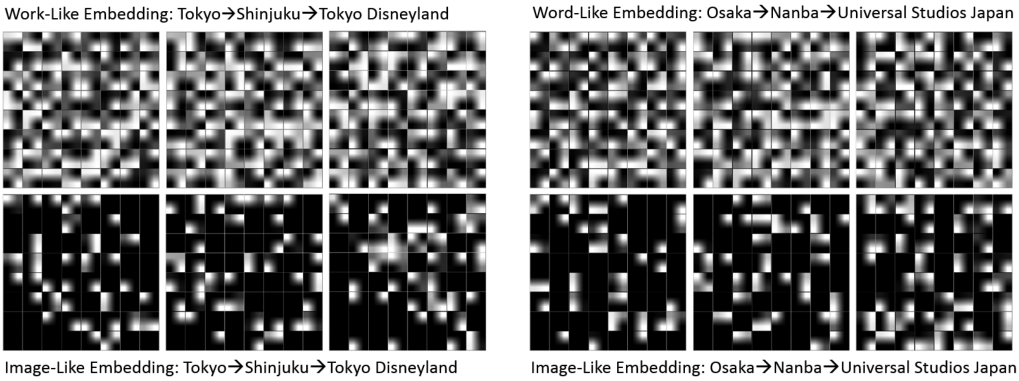


Fig. 9. Visualization of word-like embedding and image-like embedding for "Tokyo Station → Shinjuku Station → Tokyo Disneyland Station" and "Osaka Station → Nanba Station → Universal Studios Japan." Word-like embedding results are listed on the top, and the image-like embedding results are listed at the bottom. The 256-dimension vector of word-like embedding is reshaped to a $2 \times 2 \times 64$ tensor so that it can be visualized in a similar way with image-like embedding.

(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 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 5. The cosine similarity between those corresponding places of two cities was calculated and summarized as Table 13, from 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 Figure 9. And some basic statistic values are listed as Table 14 to further elaborate the difference between word-like embedding and image-like embedding. Through Table 14, we could see the standard deviation of image-like embedding is larger than word-like embedding.
(2) As mentioned in Section 4, word-like embedding will not work well with transfer learning due to the different numbers of mesh-grids for different cities (e.g., 6,400 mesh-grids for

Table 14. 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

Table 15. Verification of ID Problem of Word-like Embedding

Osaka Model	Before Training		After Training		Difference	
	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**

Tokyo and 3,600 mesh-grids for Osaka). When using word-like embedding with transfer learning, the Osaka model had to be built with a 6,400 × 256 embedding matrix instead of a 3,600 × 256 embedding matrix so that transfer learning from Osaka to Tokyo could work. If not so, then mesh-grid IDs of Tokyo in [3,600, 6,400) cannot be taken as input by the model. However, the IDs in [3,600, 6,400) will not be well trained or embedded due to the lack of corresponding training data in Osaka. Four test cases were listed as Table 15. 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 15, we could see those two IDs in [3,600, 6,400) 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.

7 RELATED WORK

The existing researches on the human mobility prediction have been conducted from multifold perspectives. On the techniques, there are traditional pattern-based methods as well as emerging deep learning leveraged models. On the research scale, both individual-level behaviors and city-level mobility have been extensively studied throughout recent years. Furthermore, given different definitions of problems, the citywide mobility forecasting task derives various applications (e.g., emergency management, traffic conditions).

Trajectory-pattern-based methodologies have been proposed to predict future movement of individual person [23, 28, 38]. 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 [43]. Zheng [55] proposed an unsupervised learning algorithm for location prediction. Social-LSTM [2] is an advanced multi-agent model, which builds a separate LSTM network for each person. ST-RNN by Reference [34] utilizes time/distance-specific transition matrices to model spatio-temporal contexts. However, the model does not explicitly take semantic meaning into consideration. SERM [50] is recurrent model designed for semantic trajectory. A RNN architecture similar with our word-like embedding model was proposed in Reference [12] for destination prediction task. Reference [16] focused on human mobility prediction from the sparse and lengthy trajectories. 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 [33], but they utilized big check-in data from location-based network services [29–31, 39, 48]. Similarly, graph-liked embedding has been applied in References [10, 49] to capture latent data representations for tasks in location-based recommendations and social networks.

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 [13, 15, 25]. Furthermore, modeling human mobility for very large populations [22, 24] and simulating human emergency mobility following disasters [47] are other topics that are close to ours. However, all of these approaches had different problem definitions and modeling methods. For example, the approaches [47] required disaster information such as intensity of earthquake and damage level as additional input data; the deep learning model [24] needed to be built based on dynamic city Region-of-Interest (ROI) discovered from massive human mobility data. Forecasting the citywide crowd density [19, 54] 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 [26, 44] 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 taxi demand [4, 5, 21, 35–37, 51].

CityCoupling [14] first utilized massive human mobility data to discover the corresponding locations between the source city and the target city, then generated the mapped human trajectories in the target city under event situations like a big earthquake. Reference [17] also generated the human mobility in a new city by generating the Origin-Destination (OD) pairs and paths from a learned mobility intention model. CityCoupling [14] generates trajectories for a special situation of the target city, and Reference [17] generates trajectories for a newly built city. These two studies fall into the category of trajectory simulation and generation, while ours is still a problem of trajectory prediction, with specific constraints, namely, limited training data. In summary, our work distinguishes from others in three aspects. First, it innovatively fuses big heterogeneous POI data with citywide human trajectories to accurately predict a probability distribution of human mobility at city level. Besides, a generic image-liked embedding mechanism is proposed to capture latent semantic representations to enrich vanilla human mobility data. Moreover, transfer learning gets involved to share the knowledge learned from city to city with similar POI components, to improve the predicting power with limited mobility data.

8 CONCLUSION

In this article, 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 or extended 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) Current problem setting is few-shot learning on trajectory prediction, and we would like to challenge the zero-shot learning on trajectory generation or simulation as the next research direction.

REFERENCES

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

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

[3] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in location-based social networks: A survey. *GeoInformatica* 19, 3 (2015), 525–565.

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

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

[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. Retrieved from https://arXiv:1409.1259.

[7] Francois Chollet. 2015. keras. Retrieved from https://github.com/fchollet/keras.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. Retrieved from https://arXiv:1412.3555.

[9] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Trans. Info. Theory* 13, 1 (1967), 21–27.

[10] Daizong Ding, Mi Zhang, Xudong Pan, Duocai Wu, and Pearl Pu. 2018. Geographical feature extraction for entities in location-based social networks. In *Proceedings of the World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 833–842.

[11] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2625–2634.

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

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

[14] Zipei Fan, Xuan Song, Ryosuke Shibasaki, Tao Li, and Hodaka Kaneda. 2016. CityCoupling: Bridging intercity human mobility. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 718–728.

[15] Zipei Fan, Xuan Song, Tianqi Xia, Renhe Jiang, Ryosuke Shibasaki, and Ritsu Sakuramachi. 2018. Online deep ensemble learning for predicting citywide human mobility. *Proceedings of the ACM Conference on Interactive, Mobile, Wearable, and Ubiquitous Technologies* 2, 3 (2018), 1–21.

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

[17] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Li Song, Hui He, and Yu Zheng. 2020. What is the human mobility in a new city: Transfer mobility knowledge across cities. In *Proceedings of the Web Conference*. 1355–1365.

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

[19] Minh X. Hoang, Yu Zheng, and Ambuj K. Singh. 2016. Forecasting citywide crowd flows based on big data. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'16)*.

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

[21] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. 2014. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Trans. Intell. Transport. Syst.* 15, 5 (2014), 2191–2201.

[22] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. 2012. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. ACM, 239–252.

[23] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. 2008. A hybrid prediction model for moving objects. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08)*. Ieee, 70–79.

[24] Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Quanjun Chen, Qi Chen, and Ryosuke Shibasaki. 2018. Deep ROI-based modeling for urban human mobility prediction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–29.

[25] Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Quanjun Chen, Satoshi Miyazawa, and Ryosuke Shibasaki. 2018. Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

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

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

[28] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. 2010. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1099–1108.

[29] Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. 2018. Scalable content-aware collaborative filtering for location recommendation. *IEEE Trans. Knowl. Data Eng.* 30, 6 (2018), 1122–1135.

[30] Defu Lian, Xing Xie, Vincent W. Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. 2015. CEPR: A collaborative exploration and periodically returning model for location prediction. *ACM Trans. Intell. Syst. Technol.* 6, 1 (2015), 8.

[31] Defu Lian, Kai Zheng, Yong Ge, Longbing Cao, Enhong Chen, and Xing Xie. 2018. GeoMF++ scalable location recommendation via joint geographical modeling and matrix factorization. *ACM Trans. Info. Syst.* 36, 3 (2018), 1–29.

[32] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R News* 2, 3 (2002), 18–22.

[33] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1043–1051.

[34] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

[35] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 865–873.

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

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

[38] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. Wherenext: A location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 637–646.

[39] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. 2012. Mining user mobility features for next place prediction in location-based services. In *Proceedings of the IEEE 12th International Conference on Data Mining (ICDM'12)*. IEEE, 1038–1043.

[40] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.
[41] J. Ross Quinlan. 1987. Simplifying decision trees. *Int. J. Man-Mach. Studies* 27, 3 (1987), 221–234.
[42] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*
[43] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T. Campbell. 2011. Nextplace: A spatio-temporal prediction framework for pervasive systems. In *Proceedings of the International Conference on Pervasive Computing*. Springer, 152–169.
[44] Masamichi Shimosaka, Keisuke Maeda, Takeshi Tsukiji, and Kota Tsubouchi. 2015. Forecasting urban dynamics with mobility logs by bilinear Poisson regression. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 535–546.
[45] Richard Socher, John Bauer, Christopher D. Manning, et al. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 455–465.
[46] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1631–1642.
[47] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Ryosuke Shibasaki, Nicholas Jing Yuan, and Xing Xie. 2015. A simulator of human emergency mobility following disasters: Knowledge transfer from big disaster data. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.
[48] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. 2015. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1275–1284.
[49] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 15–24.
[50] Di Yao, Chao Zhang, Jianhui Huang, and Jingping Bi. 2017. SERM: A recurrent model for next location prediction in semantic trajectories. In *Proceedings of the ACM Conference on Information and Knowledge Management*. ACM, 2411–2414.
[51] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[52] Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 186–194.
[53] Wei Zeng, Chi-Wing Fu, Stefan Müller Arisona, Simon Schubiger, Remo Burkhard, and Kwan-Liu Ma. 2017. Visualizing the relationship between human mobility and points of interest. *IEEE Trans. Intell. Transport. Syst.* 18, 8 (2017), 2271–2284.
[54] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.
[55] Jiangchuan Zheng and Lionel M. Ni. 2012. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the ACM Conference on Ubiquitous Computing*. ACM, 153–162.
[56] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: Concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol.* 5, 3 (2014), 1–55.

Received June 2019; revised June 2020; accepted August 2020