StepDeep: A Novel Spatial-temporal Mobility Event Prediction Framework based on Deep Neural Network

Bilong Shen*
Tsinghua University
Beijing, China
shenbilong@gmail.com

Xiaodan Liang
Carnegie Mellon University
Pittsburgh, USA
xiaodan1@cs.cmu.edu

Yufeng Ouyang
Arizona State University
Tempe, USA
youyan16@asu.edu

Miaofeng Liu
University of Science and Technology of China
He Fei, China
lmf916@mail.ustc.edu.cn

Weimin Zheng
Tsinghua University
Beijing, China
zwm-dcs@tsinghua.edu.cn

Kathleen M. Carley
Carnegie Mellon University
Pittsburgh, USA
kathleen.carley@cs.cmu.edu

ABSTRACT

A mobility event occurs when a passenger moves out or takes off from a particular location. Mobility event prediction is of utmost importance in the field of intelligent transportation systems. It has a huge potential in solving important problems such as minimizing passenger waiting time and maximizing the utilization of the transportation resources by planning vehicle routes and dispatching transportation resources. Recently, numerous mobility pattern mining methods have been proposed to predict the transportation supply and demand in different locations. Those methods first reveal the event patterns of each Place of Interests (POI) independently and then employ a separate region function as a post-processing step. This separate process, that disregards the intrinsic spatial and temporal pattern correlations between POI, is sub-optimal and complex, resulting in a poor generalization in different scenarios. In this work, we propose a **S**patial-**T**emporal mobility **E**vent **P**rediction framework based on **Deep** neural network (`StepDeep`) for simultaneously taking into account all correlated spatial and temporal mobility patterns. `StepDeep` not only simplifies the prediction process but also enhances the prediction accuracy. Our `StepDeep` proposes a novel problem formulation towards an end-to-end mobility prediction framework, that is, switching mobility events over time in an area into an event video and then posing the mobility prediction problem as a video prediction task. Such a novel formulation can naturally encode spatial and temporal dependencies for each POI. `StepDeep` thus predicts the spatial-temporal events by incorporating the new time sensitive convolution filters, spatial sensitive convolution filters, and spatial-temporal sensitive convolution filters into a single network. We conduct experimental evaluations on a real-world 547-day New York City taxi trajectory dataset, which show that `StepDeep` provides higher prediction accuracy than five existing baselines. Moreover, `StepDeep` is generalizable and can be applied to numerous spatial-temporal event prediction scenarios.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Spatial and physical reasoning**; • **Information systems** → **Spatial-temporal systems**;

KEYWORDS

Spatial temporal data mining; Intelligent transportation systems; Deep learning; Time series prediction; Convolutional neural network

ACM Reference Format:
Bilong Shen, Xiaodan Liang, Yufeng Ouyang, Miaofeng Liu, Weimin Zheng, and Kathleen M. Carley. 2018. StepDeep: A Novel Spatial-temporal Mobility Event Prediction Framework based on Deep Neural Network. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3219819.3219931

1 INTRODUCTION

Intelligent transportation systems such as Uber, Lyft, and Didi have been widely used in recent years. Passengers can easily acquire cars by sending a request via mobile applications. The service companies will arrange a vehicle to pick them up. Intelligent transportation system provides three benefits. For passengers, such apps make it convenient to find a car. For drivers, such platforms supply an easy way to earn money. For cities, they bring economic growth and create a safer environment.

However, there are some fundamental challenges in the intelligent transportation system that hinders its intelligence level hence limits its applicability. Firstly, how to balance transportation resources is a critical issue. For example, some passengers may not easily be served at some locations due to a limited amount of nearby drivers, while at other places the drivers may not be able to find a passenger because of limited requests. Secondly, creating a good policy to minimize average waiting time is also challenging. A good policy should not only shorten waiting time of passengers but also

*This work was done while the author was a visiting scholar in Carnegie Mellon University.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219931

maximize the opportunities for drivers to pick up more passengers. The desired way to resolve this is to guide the drivers to the locations where the orders may occur. Thirdly, maximizing seat utilization can be improved. As the current average occupancy rate of the vehicle is very low [5], ridesharing[13, 14] is a pattern of transportation in which people with similar itineraries and time schedules utilize spare seats in the same vehicle and share the travel cost. Route planning based solely on solo orders[15] made at the time of request may not be an optimal solution for ridesharing that requires information on future orders.

More specifically, predicting future orders in intelligent transportation systems can be regarded as a mobility event[1] prediction task that estimates the requests at each specific location in the future. A precise transportation event prediction can thus balance vehicle allocations, plan routes efficiently, and maximize seat utilization.

Event prediction is quite challenging mainly due to the complex spatial and temporal dependencies embedded in the event patterns. First, all event time series demonstrate strong temporal dynamic properties. That is, different locations have different event patterns affected by a variety of factors. For instance, places with incidents are affected by factors such as rush hours or emergency events. To resolve these difficulties, previous methods used many specific models for predictions of events at different days, time, and locations. They often need lots of separate training and tuning steps that prohibit the model deployment to diverse situations and scenarios[10, 26]. Second, the spatial event patterns in the neighborhood Place of Interests (POI) may affect each other. For instance, the residential area close to a commercial area may have numerous requests. Prior approaches have not properly taken such spatial event dependencies into account. On the other hand, multi-variate features such as the locations of each POI and area, time series features (e.g., holidays or working days), event conditions (e.g., temperature, humidity, rain, snow) may significantly affect the spatial-temporal events. Previous studies did not utilize these features well enough to predict the spatial-temporal events. Instead, they used hand-crafted feature design, which is difficult to be formulated. Consequently, utilizing POI as a feature to predict the orders in some methods is imprecise since POI can only describe the category or name but cannot reveal the importance of the POI. Previous research thus need to carefully adjust the models according to complex domain knowledge. For instance, if we only consider the category and name of POI, Grand Central Terminal[2] connects to hundreds of regions would be no important than the small terminal that only connect to few regions.

To incorporate spatial and temporal event dependencies simultaneously for each event prediction of an area, we propose an end-to-end **S**patial-**T**emporal mobility **E**vent **P**rediction framework based on **Deep** neural network (`StepDeep`) for achieving this goal. Instead of treating the past patterns of each POI independently as in previous works, we introduce a new problem definition for better encoding spatial-temporal context: 1) We represent the mobility events of an area at a specific time as one spatial slice, in which the value at each location indicates the number of people leaving or arriving; 2) The slices over time and past days can be combined together into an event video; 3) Auxiliary information, such as weather conditions and holidays, presented as a hidden layer to simplify the processing of environment features; 4) The event predictions at all locations can be treated as a pixel-wise video prediction task by considering the spatial and temporal information. Such a simple yet powerful formulation allows our model to be easily extended for more complex tasks with diverse factors that needs to be considered at once. Inspired by the advances accomplished by deep learning methods, we developed a spatial-temporal event prediction network that is constructed by several time sensitive convolution filters, spatial sensitive convolution filters, and spatial-temporal sensitive convolution filters. The `StepDeep` framework has shown to achieve superior accuracy for event prediction without any complex tunning and feature selection process. It is also general enough for many other spatial-temporal event detection areas. To summarize, our contributions in this paper are:

(1) To better address the event prediction (i.e. predicting future orders), we propose a novel problem definition for naturally encoding the spatial-temporal dependencies of event patterns. We regard the mobility event of the area as one slice. We combines slices over time into a video representation.
(2) A novel `StepDeep` network structure is proposed for exploiting the spatial and temporal correlations in event slices, which includes time sensitive convolution filters, spatial sensitive convolution filters, and spatial-temporal sensitive convolution filters.
(3) We propose a novel event slice encoding method to describe the events occurring in distinct locations. This method encodes the events and complex environment, into be two visual channels and one hidden mask, which achieves good generalization in different scenarios.
(4) We evaluated our method using a real-world taxi trajectory dataset of New York City in 547 days. In the experiment, we show that our simple yet effective framework provides excellent prediction ability on spatial-temporal datasets.

2 RELATED WORK

Urban Events Pattern Mining. Urban events mining uses human activity to reveal the pattern of the cities, it is also one kind of urban computing[28]. There are many novel works on human mobility prediction. T-Drive [25] uses historical trajectories combined with weather data to build the moving pattern graph. In this model, the Variance-Entropy-Based Clustering approach is adopted to estimate the distribution of travel time between two landmarks in different time slots. The model also adopts a two-step computing method to select the practically fastest and customized route for end users. Zhang et al.[27] use GPS historical database to analyze the service strategies from passenger-searching strategies, passenger-delivery strategies, and service-region preferences. The relationship between the avenue and the service strategies are informed by feature matrix. Xu et al. [22] focus on the latent social interaction between taxi drivers and driving behaviors correspond to the social propagation scheme. They present a two-stage framework to reveal the latent pattern propagation within cab drivers, which is based on classic social influence theory. These works use different methods to model

[1] We represent "mobility event" by "event" in the following paragraphs of this article.
[2] Grand Central Terminal is a central hub for transit in New York City.

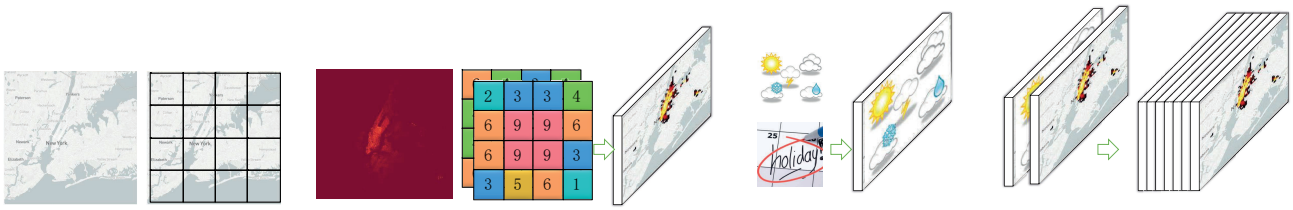


Figure 1: Event slice Encoding step by step. (a) shows spatial decomposition, (b) shows embedding the historic events to spatial pixels, (c) shows encoding the environment features to hidden channels, (d) shows generating the history slices by historical data.

the human mobility. Our work focuses on the prediction of the mobility volumes that people arrive in or depart from particular areas which is different from them.

Spatial Temporal Model in Deep Learning. There are also some deep learning methods focus on spatial-temporal model such as rDNN [7],eRCNN [21], DeepSense [24], BSSPredict[23], and DeepSD [20]. rDNN [7] proposes a regularized DNN that jointly exploits the feature relationships and the class relationships for improving categorization performance. eRCNN [21] uses a deep model to predict the traffic speed on second and third ring roads of Beijing city. They predict the speed of the road on part of the road network, while we want predict events in the whole area. The difference between their works and ours is that we utilize new point of view to solve the prediction problem on the spatial-temporal events. Liu et al. [23] proposes a spatiotemporal bicycle mobility model and a traffic prediction mechanism. The model application is different from the area flow prediction. DeepSense [24] combines RNN and CNN to address the aforementioned noise and feature customization challenges in a unified manner. Huang et al. [6] utilizes CNN to predict geolocation of tweets. DeepSD [20] is the most similar work to our work. It uses a full connected deep model to predict the gap of orders for different areas. The differences between `StepDeep` and DeepSD are as follows. First, DeepSD considers the location separately. This approach misses the relationship between locations at different timestamps. On the other hand, `StepDeep` considers the whole area instead of a solo area. This approach can utilize high-level information to predict events. Second, DeepSD utilizes embedding method to describe the similarity between different areas, while `StepDeep` utilizes temporal locality and spatial locality with specific convolutional filters.

3 PRELIMINARY

3.1 Event Slice Encoding

The target of mobility event slice encoding is to generate the historical slices, which include spatial-temporal information, event information and environmental information, and will be used for predicting events of next time slice. Event slice encoding consists of spatial decomposition, time series decomposition, event decomposition, and environment decomposition.

Spatial Decomposition. Spatial Decomposition is the first step of slice encoding. It generates one grid for each small spatial area. A widely used grid splitting method as in [2, 11] is applied to describe the region of cities. Given an area $\mathcal{A}$, the goal of spatial decomposition is to represent the whole area $\mathcal{A}$ to many small grids. As shown in Figure 1(a), by grid spatial partition method, the whole area is partitioned to small equal-sized grids. Each grid represents a small area of the city. As the definition in [1]. The width of each grid is g_w, the height is g_h. The decomposition matrix is consists of $H \times W$ grids, $H = \frac{D_h}{g_h}$, $W = \frac{D_w}{g_w}$, where D_w is the distance from west to east of the $\mathcal{A}$ region, and D_h is the distance from south to north of $\mathcal{A}$ region. Each grid can be viewed as a pixel of a snapshot. In other words, the spatial decomposition turns the spatial data of the city into a snapshot with $H \times W$ pixels, and each pixel in the snapshot represents a small region in the city.

Time Series Decomposition. The target of time slice decomposition is to decompose the historical data to a series of snapshots, which altogether form a video. In details, we turn the historical events with spatial information into the snapshots of a video, each snapshot represents the events of the whole city in a specific time window. The time window is represented by $[t, t + \delta]$. For example, $[t, t + 30min]$, $t = 9:00AM$ means the time slice represents the events of a particular area from $9:00$ to $9:30$. The fineness of time series is controlled by δ. The time series decomposition generates slices to describe the events by time and generate different time slices $\mathcal{S}$. Given an area $\mathcal{A}$, the slice of time t is $\mathcal{S}_t$.

Event Decomposition. As illustrated in Figure 1(b), the target of event decomposition is used to encode the event slice $\mathcal{E}$ in different time slice $\mathcal{S}$ for area $\mathcal{A}$. $\mathcal{E}_{t,c}^{x,y}$ denote the event of location (x, y) of c channel in time slice of $\mathcal{S}_t$. The urban event can be detected by trajectories generated by moving vehicles. The mobility event slice can be the summary of the number of trips, the number of people arrived in or departed from an area, or the traffic condition. For example, the mobility event of taxi passengers recorded by taxi trips contains origin and destination information. For a grid, the mobility event pattern can be represented by the number of the request to leave the grid and the number of trips arrive in the grid. The mobility event decomposition can generate 2 channels, one is arriving channel, the other is leaving channel. In other words, events decomposition generates the different channel of the one-time snapshot to describe the mobility event. As shown in Figure 1(b), after event decomposition, $\mathcal{E}_{12,0}^{2,1} = 6$ means in 12th slice $\mathcal{S}_{12}$, there were 6 people called Uber in the location of pixel $(2, 1)$.

Environment Decomposition. As depicted in Figure 1(c), the environment decomposition generates attention mask $\mathcal{M}_t$ of $\mathcal{S}_t$,

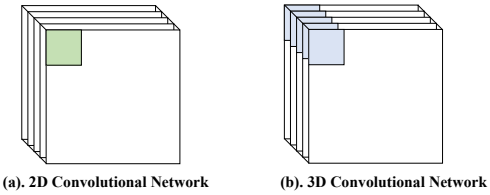


Figure 2: Slice region of 2D convolutional Networks and 3D convolutional networks. 2D convolution network misses temporal information, while 3D convolution networks concern spatial and temporal information.

which can drive or affect the events. Many changes in environments can influence the events, such as weather condition, air condition, density of population, and traffic condition. To enhance the prediction accuracy and generalization ability of our model, a novel attention mechanism should make good use of other data influencing the event maps. First, we collect the weather, holiday and other environment data in the same region. Then we extract them and pack the values of features for one day into a vector:

$$(v_0, v_1, v_2, ..., v_i).$$

Different values of v_k can be the different states of weather such as fog, rain, snow, binary number whether of holiday or not, non-negative float number as temperature and humidity. Before combining these vectors with the former training pipeline, we extend a vector with a dimension of k to an attention tensor with a dimension of $k \times H \times W$ by duplication, where H and W denote the height and width of event slice S_t.

Following the steps shown in Figure 1(d), we obtain the event slices with a mask layer of environmental information. These time-sequenced slices can be regarded as a video clip which can be used to predict the next slice.

3.2 Problem Definition

Given the past event decomposition data and environment decomposition data, our goal is to predict the events of next time slot. We formally define the problem as below. Suppose the current slot time is $\mathcal{S}_t$, the spatial area is $\mathcal{A}$. We have a slice $\mathcal{S}$ of area $\mathcal{A}$ at a given time t. We have observed that the historical events in the past: each time stamp from $\mathcal{E}_t, \mathcal{E}_{t-1}, \mathcal{E}_{t-2}, \mathcal{E}_{t-3}, ..., \mathcal{E}_1$, and the environment attention mask $\mathcal{M}_t, \mathcal{M}_{t-1}, \mathcal{M}_{t-2}, \mathcal{M}_{t-3}, ..., \mathcal{M}_1$. Now the task is, to predict $\hat{\mathcal{E}}_{t+1}$ in time slice $\mathcal{S}_{t+1}$ for time $t+1$. The objective is to make the predicted events $\hat{\mathcal{E}}_{t+1}$ in slice $\mathcal{S}_{t+1}$ as close as possible to $\mathcal{E}_{t+1}$, which is the real mobility events in time slice $\mathcal{S}_{t+1}$.

4 STEPDEEP CONVOLUTIONAL NETWORKS

In this section, we first introduce the 3D convolutional networks. Then we define the 3D Spatial-temporal convolutional networks for spatial temporal event prediction. The networks are constructed by 3 kinds of tensors. 1). 1D time sensitive convolutions to encode time series information. 2). 2D convolutions to encode spatial dimension. 3). 3D spatial-temporal convolutions to combine the spatial and temporal information.

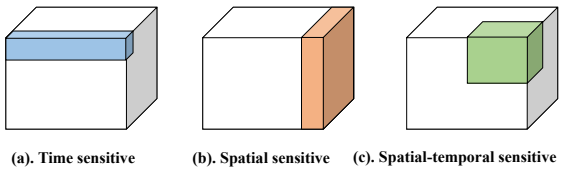


Figure 3: Illustration of perception region of different convolution filters.

4.1 3D Convolutional Networks

3D convolutional network is suitable to predict events in spatial and temporal datasets. Compare with 2D convolutional network, 3D convolutional network can combine spatial and temporal information well. As shown in Figure 2, a 2D convolution can only process 2D information, which means that only the information in the same time slice will be processed while the temporal information will be lost. As 3D convolution can process multiple layers, the spatial information and temporal information can be processed at the same time. In previous works, $3 \times 3 \times 3$ tensors[19] were selected to construct the network to deal with the video data. However, this kind of convolution cannot be directly used for the event prediction. We need to propose a new model based on both 2D and 3D convolutional networks to predict the spatial-temporal events in particular areas.

4.2 Overall Network Structure

The overall architecture of `StepDeep` is introduced in Figure 4. The net contains 7 layers with weights; all of the 7 layers are convolutional layers. The output of the last convolutional layer is fed to a $2 \times 64 \times 64$ output and generate the event slice $\mathcal{E}_{t+1}$ of time $t+1$.

Table 1: Network parameters of each layer

	Filter Num	Filter Size	Sensitive
Layer 1	128	$3 \times 1 \times 1$	Time
Layer 2	128	$1 \times 3 \times 3$	Spatial
Layer 3	256	$3 \times 3 \times 3$	Spatial-temporal
Layer 4	128	$5 \times 1 \times 1$	Time
Layer 5	128	$1 \times 3 \times 3$	Spatial
Layer 6	64	$3 \times 3 \times 3$	Spatial-temporal
Layer 7	2	$13 \times 1 \times 1$	Time

As shown in Figure 4, time sensitive convolution filters, spatial sensitive convolution filters, and spatial-temporal sensitive convolution filters are adopted by the spatial-temporal progress cells. The filter number and filter size is shown in Table 1. The first spatial-temporal progress cell consists of the first three convolutional layers. Then another spatial-temporal progress cell follows, consists of layer 4 to layer 6. The first cell's temporal sensitive filter is $3 \times 1 \times 1$, while the second cell's is $5 \times 1 \times 1$. Both cells are using padding technique to avoid changing the spatial-temporal slice shape. The last layer is an output layer without padding technique and used for generating the prediction value $\hat{\mathcal{E}}_{t+1}$.

Insight of Padding and Pooling. One of the significant differences between `StepDeep` and the other convolutional neural

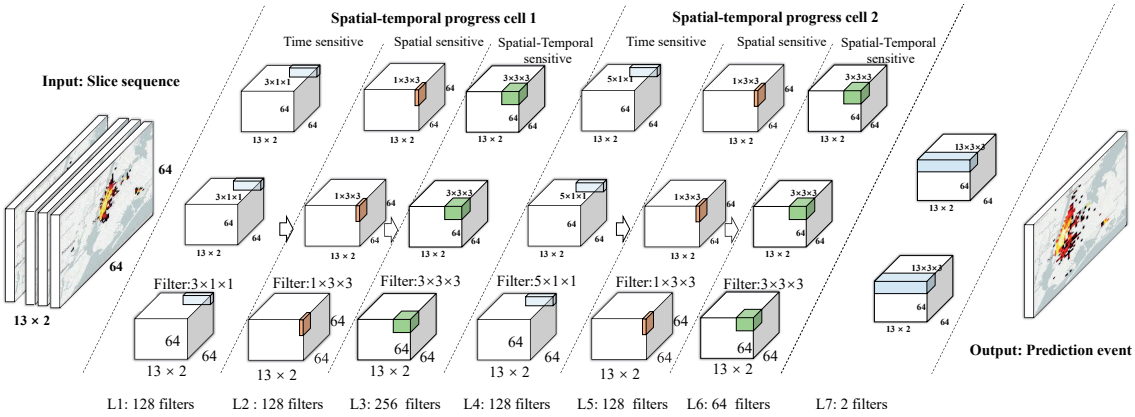


Figure 4: Network architecture of StepDeep model. Time sensitive, spatial sensitive, and spatial-temporal sensitive filters are colored with blue, orange, and green, respectively.

networks is that we utilize padding instead of pooling. As in the real video processing, the information spread by the pixel can be pooled because the information is presented by the shape of the pixels. However, in our spatial-temporal $\mathcal{S}$, each pixel saves the hidden information as the attraction of POI in this location, the region pattern exists in this pixel. As a result, the pooling method would disturb the presentation of each location. Therefore, we do not adopt the pooling layer. Another technique is using padding method to keep the structure of the video slices stable. The most important thing to utilize padding is also to keep the hidden feature in the pixels. The padding technique can deal with the boundary problem if the shape of the $\mathcal{S}$ is not divisible by the size of filter. Our model treats the spatial-temporal dataset as a video, but each pixel contains different information compared to real video. This is why StepDeep uses padding technique rather than pooling layers.

4.3 Model Description

The Convolution Layer. The convolution layer is the most important part of our CNN model. 3D convolutions in the convolution stages of CNNs compute features from both spatial and temporal dimensions. 3D convolution stacks multiple contiguous slices together. By this filter, the spatial and temporal information is connected to multiple contiguous slices in the previous layer, thereby capturing spatial locality and time locality information. Formally, the (x, y, z) element of the convolution neuron matrix, on jth channel generated by the ith filter is calculated by

$$c_{ij}^{xyz} = ReLU(\sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{D_i-1} w_{ijm}^{pqd} c_{(i-1)m}^{(x+p)(y+q)(z+d)} + b_{ij}). \quad (1)$$

w_{ijm}^{pqd} is the (p, q, d) element of the filter connect the mth channel, D_i is the size of 3D filter in temporal dimension, $c_{(i-1)m}^{(x+p)(y+q)(z+d)}$ is the $(x + p, y + q, z + d)$ element of the m feature channel in $i - 1$ layer, b_{ij} is a bias for the filter k. Detail of CNN's convolution layer implement could be found in [9].

We combine three kinds of filters into the Models: time sensitive convolution filters, spatial sensitive convolution filters, and spatial-temporal sensitive convolution filters. As shown in Figure 3. Firstly, time sensitive convolution filter is $D \times 1 \times 1$ shape, the function of this filter is to infer the temporal dimension features. To pass all the time series on the boundary, we use padding method and set padding number $n_p^D = \frac{D-1}{2}$, D is the length of time sensitive filter. Secondly, spatial sensitive convolution filter is a $1 \times S \times S$ filter. The function of this filter is to detect the feature on spatial dimension. We also use the padding method to make sure all the pixels of the $\mathcal{S}$ is scanned, $n_p^S = \frac{S-1}{2}$. The third type of filters, named spatial-temporal sensitive convolution filter, detects both spatial and temporal features with shape $K \times K \times K$. Padding method is used for enabling the filter to scan all the pixels in three dimensions, $n_p^K = \frac{K-1}{2}$.

Active Function. For all Fully-Connected layers, we select Rectified Linear Unit (ReLU) as the activation function. The ReLU activation function is defined by Equation 2:

$$ReLU(x) = \max(0, x). \quad (2)$$

The Output Layer. The output layer generates a final prediction value as

$$c_{ij}^{xyz} = \eta(\sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{D_i-1} w_{ijm}^{pqd} c_{(i-1)m}^{(x+p)(y+q)(z+d)} + b_{ij}). \quad (3)$$

Here i is the number of last layer. The output layer needs to generate integer values to represent the spatial-temporal events. In our case, the output is the number of orders will depart from the $\mathcal{S}_{i,j}$ and number of people will arrive in the $\mathcal{S}_{i,j}$. In the output layer, we adopt a modified ReLU function as the activation function, which is defined as

$$\eta(x) = \lfloor x + 0.5 \rfloor. \quad (4)$$

Lost Function. For the output layer, we select Root Mean Squared Error (RMSE) as the loss function. Formally, we use $\mathcal{E}_t$ to denote the real mobility at time t, and use $\hat{\mathcal{E}}_t$ to denote the value of predicted mobility. Then, the rooted mean squared error can be calculated by Equation 5, N is the number of the data samples.

$$loss(\mathcal{E}_t, \hat{\mathcal{E}_t}) = \sqrt{\frac{\sum |\mathcal{E}_t - \hat{\mathcal{E}_t}|^2}{N}}. \quad (5)$$

4.4 Network Training

The proposed neural networks are implemented with PyTorch[3]. No pre-trained networks are compatible with this deep architecture, so we need to train the network from scratch. The number of filters in layer 1, 2, 3, 4, 5, 6, 7 are 128, 128, 256, 128, 128, 64, and 2, respectively.

Optimization Method. We select Adaptive Moment Estimation (Adam)[8] method to adapt learning rates. Adam essentially combines RMSProp[18] with momentum. Learning rate is set to 0.001 and droped to its 9/10 every 500 iterations. The drop out method is utilized to prevent over-fitting. The 50% probability is selected.

Initial Parameters. We initialize the parameters by Xavier normal initializer which generates sample from a truncated normal distribution centered on 0 with

$$stddev = \sqrt{\frac{2}{N_i + N_o}}. \quad (6)$$

N_i and N_o represent the number of inputs and outputs units in the weight tensor, respectively.

Attention Mechanism. In the prediction of spatial-temporal event distribution, there exists a bunch of noise event points in the margin area and some lost points in the center area, which shows the shortage in capturing the accurate locations of the correlated data points and temporal gradual deformation pattern. Therefore, the model needs more attention to the useful feature points in view of the sparsity. We design a novel attention mechanism based on attention mask $\mathcal{M}$ as in Figure 5. Following values of features for one day into a vector:

$$(temperature, humidity, fog, rain, snow, holiday).$$

In the vector, *temperature* and *huminity* are non-negative float numbers, fog, *rain*,*snow*, and *holiday* are binary values. A dimension of 6 to an attention tensor with a dimension of $6 \times H \times W$ by duplication, where H and W denote the height and width of our original event maps. Considering the redundant values and dimensions in our tensors, we utilize an attentive feature extraction convolutional layer to map the original attention values to new meaningful values according to their contribution to the data distribution respectively.

5 EXPERIMENT AND EVALUATION

In this section, we evaluate our novel model on a real mobility dataset of New York City. In Section 5.1, we first introduce the dataset and the method to generate real mobility event from the raw dataset. In Section 5.2, we introduce several state-of-art algorithms that can be applied in mobility prediction. In Section 5.3 we show the preliminary experiment results. In Section 5.4, we compare `StepDeep` with baseline approaches. Then we compare the effective of different models in Section 5.5.

[3]http://pytorch.org/

5.1 Dataset

Map Region. We select New York City as the analysis place as New York is one the most famous cities in the world and the trajectories of this city is published by the government. The geographic region of New York City is selected by the bounding box of this area. The bounding box is described by two points, left-bottom and right-top. The details of the city region are shown in Table 2.

Table 2: Number / time period of trips we select to analyze, the left-bottom and right-top points of analysis region.

Description	Value
Number of trips	210,591,270
Time Period of trips	01/01/2015 to 06/30/2016
Left-bottom of map box	$(-74.259090^o,\ 40.477399^o)$
Right-top of map box	$(-73.700181^o,\ 40.916179^o)$

Trajectories Dataset. We utilize the taxi trajectory dataset of New York City to generate the event slices. The taxi trajectory dataset contains the trajectories of yellow taxis in New York City. The taxi data set is from New York City Taxi and Limousine Commission (TLC) [16]. The time period is from January 1st 2015 to June 30th 2016, 547 days in total. It contains 210,591,270 trips of taxi. Each trip includes features such as pick-up and drop-off dates/times, pick-up and drop-off locations, and trip distances. The details of trips are also shown in table 2.

Spatial decomposition. We generate the spatial-temporal events by the spatial region data and mobility data of taxi trips. We partition the map zone to 64×64 pixels, each side of the pixel represents the real spatial length of around 0.5 mile.

Spatial Temporal Event Generation. We generate the slice of different time by $\delta = 30mins$. Each slice represents the spatial-temporal mobility event in this region during a specific time period. By the result of spatial decomposition, we calculate the number of trips leave the grid and the number of trips arrive in the grid. Then each slice of 64×64 pixels picture represents the spatial temporal mobility $\mathcal{E}_i$ in slice $\mathcal{S}_i$. The mobility picture contains two channels, one channel is the mobility of people arrive in the area $\mathcal{A}$, and the other channel represents the event of people leave this $\mathcal{A}$.

Environment. We conduct experiments on a 64-bit Ubuntu 17.10 computer with 6 Intel 3.10GHz and 128GB memory, 4 NVIDIA TITAN X GPUs (12GB DDR5).

5.2 Baselines

To evaluate the effectiveness of our framework, we compare `StepDeep` with the following baselines, of which the parameters are fine-tuned.

Empirical Average. The empirical average is calculated by the average number in a particular area. We use the empirical average as the predict value. The prediction value of the pixel area $a \in \mathcal{A}$ is calculated by Equation 7

$$\frac{\sum_{t \in (0, |T|), a \in \mathcal{A}} \mathcal{E}_a^t}{|T|}. \quad (7)$$

The time interval is same with `StepDeep`, 30 minutes. T is the set of time slices.

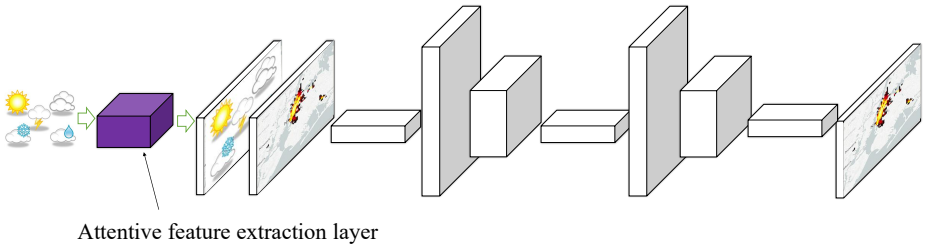


Figure 5: Architecture of attentive feature extraction layer.

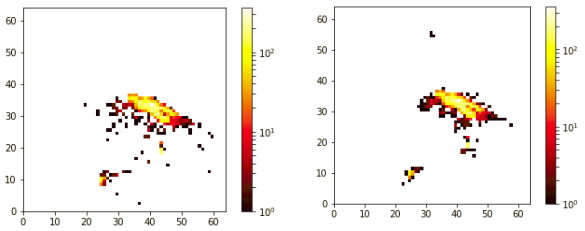
LASSO. Least Absolute Shrinkage and Selection Operator (LASSO) is a linear regression analysis method that performs both variable selection and regularization. We implement the LASSO prediction model with the Scikit-learn library[12] .

GBDT. Gradient Boost Decision Tree (GBDT) is widely used in data mining applications these years. Extreme Gradient Boosting (XGBoost) [3] is a state-of-art GBDT implementation used for supervised learning problems and has a good performance in many areas such as competitions of kaggle platform[4]. We select XGBoost[5] as a baseline to predict the events.

RF. Random Forest (RF) is a powerful ensemble learning method for classification, regression and other tasks. It fits a number of classifying decision trees on various sub-samples of the dataset and improve the predictive accuracy and control over-fitting by averaging. We implement the Random forest prediction model with the Scikit-learn library[12].

DeepSD. DeepSD is a deep model to predict the gap of supply and demand. We implemented this method and modified the gap to the mobility event corresponding to our `StepDeep`. It is implemented with PyTorch.

5.3 Preliminary Experiment Results for Prediction



(a) Real orders (b) Predict orders

Figure 6: Visualization of real and prediction orders. The color denotes the number of orders for real and prediction.

First, we demonstrate the effectiveness of `StepDeep` on the generated spatial-temporal event dataset illustrated in Section 5.1 by a fine-tuned 7-layer spatial-temporal multi-scale convolutional network with region distribution attention. Each layer captures the short and long range correlations among the event maps on different scales of height, width and depth of input tensors. In each epoch

[4]https://www.kaggle.com/
[5]https://github.com/dmlc/xgboost

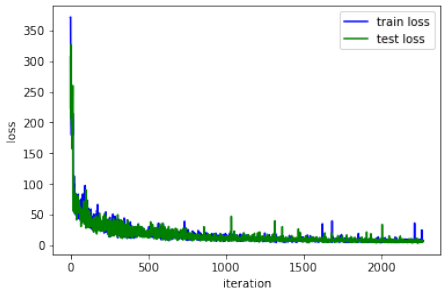


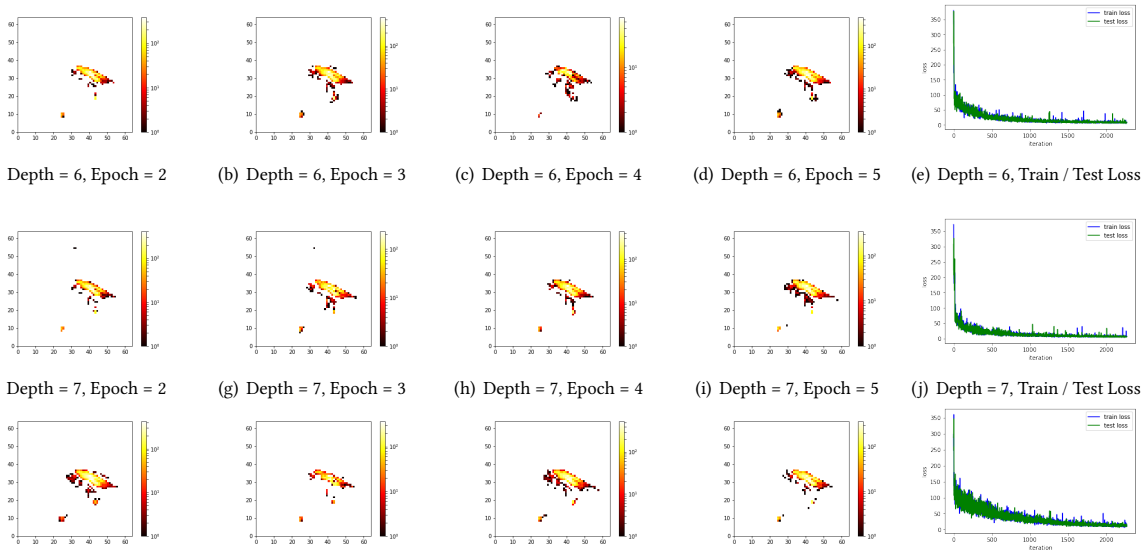
Figure 7: Training and testing loss for fine-tuned 7-layered StepDeep Model.

of training, validation and testing, event map dataset is divided into three corresponding parts, frames in them are shuffled each time as to iterate on all partitions with subscripts. For each element in a training batch, we first select a prediction target identifier i one by one in the training set, then we trace back to its former event map frames with the strong dependency. Specifically, the 1st, 2nd, 47th, 48th, 335th, 336th, 337th, 671st, 672nd, 673rd, 1439th, 1440th, 1441st steps (each step stands for 30 minutes in the real data) of precedent frames are selected to form a bunch of frame for training the prediction capability for our target frame i. The weather and holiday data of different time are utilized to enhance the attention of the network for the data dense region. We set a maximum epoch of 100 and iteration of 454 times in each epoch (in fact, all tuning model parameters converge in 10 epochs), choose 5 most related weather condition (temperature, humidity, fog, rain, and snow) values to extend as an attention tensor concatenated to the original event map channels to form the final input tensor. Our model converges within 5 epochs of training in this optimal configuration and achieve an excellent average testing RMSE loss of 2.7286 on the testing set. Figure 6 shows an output map with predicting orders and a real order in a heat map way[6], after training we can observe that `StepDeep` learned a good command of the spatial-temporal data distribution in the sequenced input event maps. Figure 7 depicts the training and testing loss in this process.

5.4 Comparison with baselines

By conducting the training and testing experiments on the event dataset with the baselines in Section 5.2 and our 7-layer fine-tuned

[6]The central part has data, while the remaining area is empty because the other parts are sea and the area that does not belong to New York City.



(a) Depth = 6, Epoch = 2 (b) Depth = 6, Epoch = 3 (c) Depth = 6, Epoch = 4 (d) Depth = 6, Epoch = 5 (e) Depth = 6, Train / Test Loss

(f) Depth = 7, Epoch = 2 (g) Depth = 7, Epoch = 3 (h) Depth = 7, Epoch = 4 (i) Depth = 7, Epoch = 5 (j) Depth = 7, Train / Test Loss

(k) Depth = 8, Epoch = 2 (l) Depth = 8, Epoch = 3 (m) Depth = 8, Epoch = 4 (n) Depth = 8, Epoch = 5 (o) Depth = 8, Train / Test Loss

Figure 8: Selected results of varying depth of layers and epochs.

model, we demonstrate the competitive performance of our model in Table 3. We evaluate the performance of each model from a regression indicator: Rooted Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\vec{preds} - \vec{labels})^2}. \quad (8)$$

It is observed that our model provides higher prediction accuracy than the other 5 methods on RMSE.

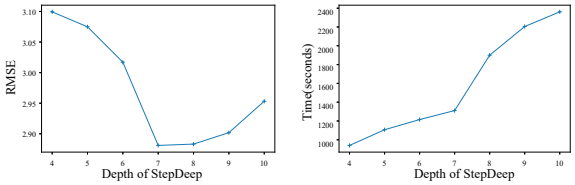
Table 3: Performance of `StepDeep` compared to baselines

Model/Method	RMSE
Average of pixels	17.6266
LASSO	3.3196
GBDT	3.3211
Random Forest	3.3367
DeepSD	3.2101
`StepDeep` without attention	2.8810
`StepDeep` with attention	2.7286

5.5 Model Ablation and Further Analysis

To have a clear understanding of each component in `StepDeep`, we conduct a model ablation to observe the effectiveness of each novel local structure and crucial parameter for performance in our paradigm. By choosing the depth and use of attention, we can achieve an excellent balance performance flexibly. The comparative results are shown mainly in the following subsections.

5.5.1 Effectiveness of depth in `StepDeep`. In previous deep learning methods, deep neural networks benefit from depth in numerous occasions, as is analyzed in [17]. However, the spatial-temporal data we are working on is to some extent different to the usual vision data processed by common CNN, which has more information on data distribution and detailed pixel values. In a visualized way, the shape of the visible data region on the heat map evolutes gradually, for which we utilize external weather and holiday constriction as attention to the valuable distribution regions. To capture more pixel-level feature details, multi-scale kernels designed for multiple receptive fields are proposed in our model inspired by [4]. Due to the special function and designed structure in `StepDeep`, we conducted an ablation parameter study with respect to the depth in building our model. We take the 4-layer convolutional model as the initial configuration, at every step, we add one more layer to the existed model while keeping other existed parameters intact and conduct a training, validation and testing experiment, in which the maximum epoch is set to 10, the numerical ratio of training, validation and testing samples is 6:2:2. Figure 8 shows the selected visualized training outputs and results with depth equal to 6,7,8, where the reasonable output maps and corresponding training and testing curves are laid out.



(a) Influence of depth on RMSE (b) Influence of depth on convergence time

Figure 9: Influence of depth on performance.

The utmost important factors of prediction problems are accuracy and efficiency. We evaluate the accuracy by the average testing RMSE for each ablation model, the performance curves with depth from 4 to 10 are shown in the Figure 9. We can observe that in general the accuracy grows with more layers utilized, the increasing

speed slows down when depth gets large, however, the performance gets worse when depth exceeds a threshold. The efficiency is measured by how long it takes for a model to converges on its parameters. The time-depth curve is shown in Figure 9, where the performance decrease when depth is increased. With the different monotonicity of two results, we can see that 7 layers is the best choice for the application of this model.

5.5.2 Effectiveness of extra attention. From the former results we can see that in the prediction of spatial-temporal event distribution, noisy points and missing values in prediction have a negative effect on the RMSE visually, which is also why we conduct an environment-attentive training on StepDeep. To this end, we add the attention mechanism described in Section 4.4 to ablation models of different depths in Section 5.5.1. With the training and testing hyper parameters keeping intact, the results of models with 5-9 layers are shown in Figure 10 and comparison between 7-layer model with and without attention is shown in Figure 11. We can find that the best depth combined with environment attention mechanism is 7. Fewer layers are unable to adequately show the effect of attention mechanism for not making good use of receptive field to pass attention information through enough deep multi-scale convolutional layers, while more layers makes a heavy burden of training due to extended input tensors and a large amount of parameters. As a result, we select the 7-layer with environment attention as the best application model for StepDeep up to now.

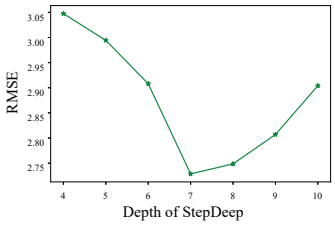
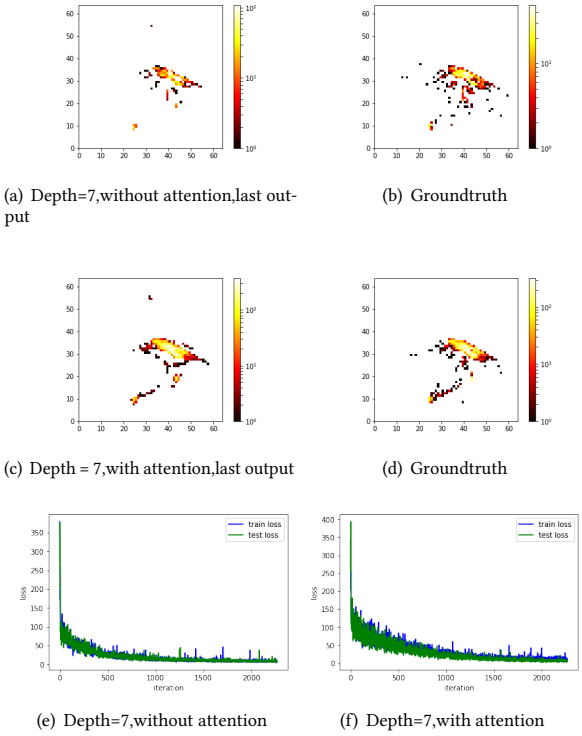


Figure 10: Influence of the layers combined with attention .

5.5.3 Discussion on selection of the temporal correlation stride. Spatial-temporal event evolution not only have correlations with the nearby time periods but also with the distant time periods. For example, to predict the events of [9:00 AM.,9:30 AM], we should not only consider the events of [8:30 AM,9:00 AM] but also the events of the corresponding time periods of the past days. We compare the different time strides' contribution to enhance the model's prediction capability. By experiments, we found that for a prediction target slice, selecting slices located in the relative temporal locations in Section 5.3 benefits the most, where the temporal correlation strides are examined through past hours (the 1st and 2nd slices), correspondence time in the past day (the 47th and 48th slices), correspondence time in past weeks (the 335th, 336th, 337th, 671st, 672nd, and 673rd slices), and past month (the 1439th, 1440th, and 1441st slices). These slices reflects historical information of different time spans, and the historical video generated through this information can effectively predict the events of the next slice through our framework.



(a) Depth=7,without attention,last output (b) Groundtruth

(c) Depth = 7,with attention,last output (d) Groundtruth

(e) Depth=7,without attention (f) Depth=7,with attention

Figure 11: Comparison on 7-depth model, without and with environment attention.

6 CONCLUSION

This paper proposes a framework (named StepDeep) to predict the spatial-temporal events in particular areas. The framework is an end-to-end deep neural network which overcomes the shortage of previous methods that need to understand the POI of the location or that has a low prediction accuracy. StepDeep can automatically predict the events by utilizing historical data. We conduct the experiments on a real-world dataset of New York City, which contains the real taxi trajectories in 547 days. The results show that StepDeep outperforms 5 baselines. Even more, the proposed StepDeep framework is highly flexible and can be easily generalized. Last but not the least, the model opens a new view, that turns the mobility event prediction problems into the video prediction problems, to process the spatial-temporal event data. We will continue to improve our framework for check-in events prediction, twitter-location prediction, and traffic condition prediction.

ACKNOWLEDGMENTS

The authors would like to sincerely thank the reviewers for many valuable comments and insightful suggestions. We appreciate the help from Felicia Natali, Binxuan Huang, Dong Wang, and Hanqing Yin. This work is supported by the Office of Naval Research (USA) (N000140811186), the National Grand Fundamental Research 973 Program of China (2014CB340402). Bilong Shen's work is also supported by the China Scholarship Council.

REFERENCES

[1] Pablo Samuel Castro, Daqing Zhang, Chao Chen, Shijian Li, and Gang Pan. 2013. From taxi GPS traces to social and community dynamics: A survey. *ACM Computing Surveys (CSUR)* 46, 2 (2013), 17.

[2] Chao Chen, Daqing Zhang, Nan Li, and Zhi-Hua Zhou. 2014. B-Planner: Planning bidirectional night bus routes using large-scale taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems* 15, 4 (2014), 1451–1465.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 785–794. https://doi.org/10.1145/2939672.2939785

[4] Adam Coates and Andrew Y Ng. 2011. Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*. 2528–2536.

[5] Keivan Ghoseiri, Ali Ebadollahzadeh Haghani, Masoud Hamedi, and MAUT Center. 2011. *Real-time rideshare matching problem*. Mid-Atlantic Universities Transportation Center Berkeley.

[6] Binxuan Huang and Kathleen M Carley. 2017. On Predicting Geolocation of Tweets using Convolutional Neural Networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 281–291.

[7] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. 2017. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[8] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[10] Moshe Lichman and Padhraic Smyth. 2014. Modeling human location data with mixtures of kernel densities. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 35–44. https://doi.org/10.1145/2623330.2623681

[11] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 410–421.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[13] Bilong Shen, Yan Huang, and Ying Zhao. 2016. Dynamic ridesharing. *SIGSPATIAL Special* 7, 3 (2016), 3–10.

[14] Bilong Shen, Ying Zhao, Yan Huang, and Weimin Zheng. 2017. Survey on Dynamic Ride Sharing in Big Data Era. *Journal of Computer Research and Development* 54, 1, Article 34 (2017), 15 pages. https://doi.org/10.7544/issn1000-1239.2017.20150729

[15] Bilong Shen, Ying Zhao, Guoliang Li, Weimin Zheng, Yue Qin, Bo Yuan, and Yongming Rao. 2017. V-Tree: Efficient kNN Search on Moving Objects with Road-Network Constraints. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. 609–620. https://doi.org/10.1109/ICDE.2017.115

[16] New York City Taxi and Limousine Commission (TLC). 2017. The yellow taxi trip records. (2017). http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

[17] Matus Telgarsky. 2016. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485* (2016).

[18] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[19] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 4489–4497.

[20] Dong Wang, Wei Cao, Jian Li, and Jieping Ye. 2017. DeepSD: Supply-Demand Prediction for Online Car-Hailing Services Using Deep Neural Networks. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. 243–254. https://doi.org/10.1109/ICDE.2017.83

[21] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 499–508.

[22] Tong Xu, Hengshu Zhu, Xiangyu Zhao, Qi Liu, Hao Zhong, Enhong Chen, and Hui Xiong. 2016. Taxi Driving Behavior Analysis in Latent Vehicle-to-Vehicle Networks: A Social Influence Perspective. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 1285–1294. https://doi.org/10.1145/2939672.2939799

[23] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. 2016. Mobility Modeling and Prediction in Bike-Sharing Systems. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16)*. ACM, New York, NY, USA, 165–178. https://doi.org/10.1145/2906388.2906408

[24] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 351–360. https://doi.org/10.1145/3038912.3052577

[25] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*. 99–108. https://doi.org/10.1145/1869790.1869807

[26] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2013. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE Trans. Knowl. Data Eng.* 25, 10 (2013), 2390–2403. https://doi.org/10.1109/TKDE.2012.153

[27] Daqing Zhang, Lin Sun, Bin Li, Chao Chen, Gang Pan, Shijian Li, and Zhaohui Wu. 2015. Understanding Taxi Service Strategies From Taxi GPS Traces. *IEEE Trans. Intelligent Transportation Systems* 16, 1 (2015), 123–135. https://doi.org/10.1109/TITS.2014.2328231

[28] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM TIST* 5, 3 (2014), 38:1–38:55. https://doi.org/10.1145/2629592