Hindawi
Complexity
Volume 2019, Article ID 7643905, 14 pages
https://doi.org/10.1155/2019/7643905

Research Article

A Destination Prediction Network Based on Spatiotemporal Data for Bike-Sharing

Jian Jiang,[1] **Fei Lin**,[1] **Jin Fan**,[1] **Hang Lv**,[1] **and Jia Wu**[2]

[1]*School of Computer Science and Technology, Hangzhou Dianzi University, 310018, Hangzhou, China*
[2]*Department of Computing, Macquarie University, Sydney, Australia*

Correspondence should be addressed to Fei Lin; linfei@hdu.edu.cn

Received 2 July 2018; Revised 27 September 2018; Accepted 30 October 2018; Published 1 January 2019

Guest Editor: Jianxin Li

Bike-sharing is a new low-carbon and environment-friendly mode of public transport based on the "sharing economy". Since 2017, the bike-sharing market has boomed in China's major cities. Bikes equipped with GPS transmitters are docked along sidewalks that can be easily accessed through smartphone apps. However, this new form of transport has also led to problems, such as illegal parking, vandalism, and theft, each of which presents a major administrative challenge. Further, imbalances in user demand and bike availability need to be overcome to ensure a convenient, flexible service for customers. Hence, predicting a cyclist's destination could be of great importance to shared-bike operators. In this paper, we propose an innovative deep learning model to predict the most probable destination for each user. The model, called destination prediction network based on spatiotemporal data (DPNst), comprises three steps. First, the data is preprocessed and a pool of likely candidate destinations is generated based on frequent item mining. This candidate set is then used to build the DPNst model: a long short-term memory network learns the user's behavior; a convolutional neural network learns the spatial relationships between the origin and the candidate destinations; and a fully connected neural network learns the external features. In the final step, DPNst dynamically aggregates the output of the three neural networks based on the given data and generates the predictions. In a series of experiments on real-world stationless bike-sharing data, DPNst returned an F1 score of 42.71% and demonstrated better performance overall than the compared baselines.

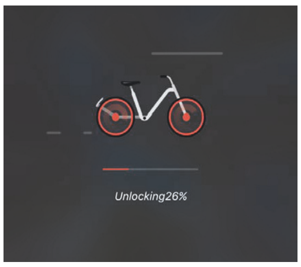
1. Introduction

Cycling is a low-carbon, environment-friendly method of transportation, and bike-sharing is the newest iteration of this popular and healthy mode of travel. Bike-sharing is based on the sharing economy, which means a community rents or shares access to good or services through online transactions. The widespread popularity of bike-sharing can be attributed to several key advantages: (1) renting and returning a bike at the roadside is convenient and affordable; (2) it can solve the last mile problem common to most mass-transit systems; and (3) it helps to alleviate traffic congestion. In fact, China's craze for bike-sharing has brought more than 2 million new bikes to its city streets [1]. In fact, Mobike, the world's largest bike operator, recently made Shanghai the world's largest bike-sharing city [2].

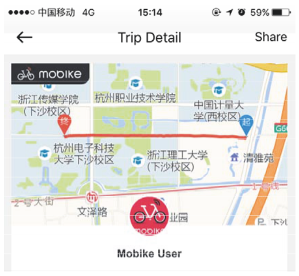
Unlike most other bike-sharing schemes around the world, China's shared bicycles can be picked up or dropped off anywhere; i.e., the systems are stationless. Each bike contains a GPS/3G module and an intelligent lock. Bicycles are locked by the rider after use and unlocked by the next rider by scanning a QR code on the frame using a mobile app. The app also records the user's riding history along with other data (see Figure 1).

Bike-sharing was invented in China and, while it may be convenient for users, it can be frustrating for city authorities. One of the major concerns is piled-up bikes on the sides of city streets. For example, Shanghai leads the world with 450,000 shared bicycles, nearly all of which have appeared in the past six months [1]. Beyond the traffic and pedestrian congestion problems too many bikes can cause, they are also symptomatic of an oversupply of bikes in one location, which often means a lack of bikes at another. The main cause of this problem is mobility, i.e., one-way bike use. Passengers rent a bike from one place and ride it to another place, but rarely return it to where they started. Operators can not necessarily

(a) A group of stationless shared bikes



(b) Unlocking a bike with the app



(c) Route history

Figure 1: Stationless bike-sharing.

(a) Excess bikes

(b) Disorderly bikes

(c) Bike damage

Figure 2: Problems with bike-sharing.

redistribute stocks in time to meet demand, which results in imbalances across the system. In addition, damage to bikes is fairly common, and these bikes need to be replaced to meet functional demand. Figure 2 illustrates some of these problems.

In response to the above concerns, several scholars have already explored some aspects of traffic flow and demand prediction in bike-sharing systems. For example, Bao et al. [2] used traffic trajectories to address bike lane planning problems. However, as yet, no studies have examined the back-end administrative issues associated with real-time cyclist behavior. To address this challenge, this paper presents a neural network that predicts cyclists' destinations. The ability to forecast likely destinations across a bike-sharing network would not only help companies with dispatch and reallocation but could also guide cyclists to park their bikes in the appropriate position. Further, such a system could help governments to supervise traffic, alleviate road congestion, and better plan urban construction projects.

The major contributions of this paper are summarized below:

(i) A method for generating likely candidate destinations in a stationless bike-sharing system. A set of candidate destinations is generated by mining frequent items with the FP-Growth algorithm. Historical user data is analyzed to identify the most likely destinations for each cyclist based on origin-destination itemsets, and these sets are used to train three destination prediction networks. This technique greatly simplifies the computational complexity of the model.

(ii) An innovative deep learning model to predict cyclists' destinations. The model, called DPNst, comprises three steps: (1) data preprocessing and candidate generation; (2) model construction; and (3) prediction. To build the model, user behavior is learned through a long short-term memory (LSTM) network [3], the spatial relationships between the origin and destination maps are learned through a convolutional neural network (CNN) [4], and the external features are learned through a fully connected neural network (FCNN) [5]. The final predictions are based on a dynamic aggregate of the output of these three neural networks.

(iii) A series of experiments that verify DPNst's performance on real-world data from Mobike's stationless bike-sharing system. The results show better performance.

The remainder of this paper is organized as follows. Section 2 establishes the problem definition and provides an overview of the model. The preprocessing methods are introduced in Section 3. Section 4 describes the method for generating the candidate destination set based on frequent item mining. The DPNst is presented in Section 5, followed by the experimental evaluation in Section 6. Related work is summarized in Section 7, and Section 8 concludes the paper.

2. Overview

This section begins by defining the problem of predicting destinations in a bike-sharing system, followed by an overview of the model's framework. The notations are defined in Table 1.

2.1. Problem Definition. Given a specific user u, time t, origin a, meteorological information m, and other external

Table 1: Notations in our paper.

Notation	Description
C	the candidate generation set
S	the minimum support of FP-Growth algorithm
$\mathbf{X}_u$	the input of user behavior sequence component trained by LSTMs
$\mathbf{X}_p$	the input of position map sequence component trained by CNNs
$\mathbf{X}_e$	the input of external feature sequence component trained by FCs
$\mathbf{X}_{ubs}$	the output of user behavior sequence component trained by LSTMs
$\mathbf{X}_{pm}$	the output of position map component trained by CNNs
$\mathbf{X}_{ef}$	the output of external feature component trained by FCs
$\widehat{\mathbf{X}}$	the output of the whole networks

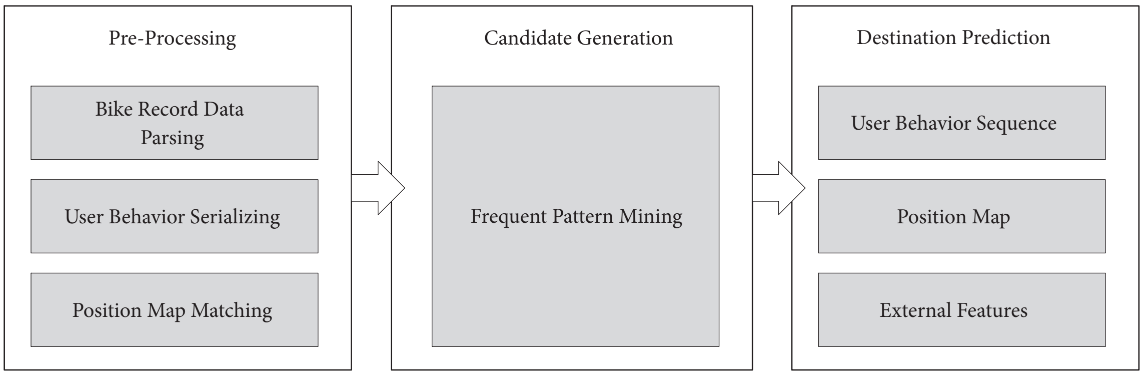


Figure 3: An overview of the framework.

information e, the probability that a bike will end its journey at destination b can be as follows.

$$P_b = f(u, t, a, m, e) \tag{1}$$

Given a set of destinations D, the predicted destination of a bicycle is the destination with the maximum probability in the following set.

$$\max_{i \in D} P_i \tag{2}$$

However, if all possible destinations were included in D, the computational complexity could be extremely high, yet ensuring that the set contains an appropriate selection of potential destinations is of great importance to the problem. This procedure is called candidate generation. Given a specific user u, time t, origin a, and a full set of positions H that includes both origins and destinations, we need to generate a manageable set of candidate likely destinations C_a^u as follows.

$$C_a^{u,t} = \{b \mid \exists a \longrightarrow b,\ User = u,\ Time = t\} \quad a, b \in H \tag{3}$$

Hence, this destination prediction problem has been converted into a recommendation problem, and finding a solution becomes a binary classification problem. If the user is likely to ride to a destination, the position is labeled with a 1, and 0 otherwise. If the user has never been to a place, the destination prediction model refers to other nearby places. Thus, generating candidates can be seen as a frequent item mining problem with the goal of identifying the most likely origin-destination itemsets for a given user.

2.2. Model Framework. The model's framework is presented in Figure 3. It consists of three main components: preprocessing, candidate generation, and destination prediction.

Preprocessing. This component is designed to process the input information, which includes bike records, map information, and meteorological information. This component has several functions: (1) to parse the bike record data and remove any outliers; (2) to match the map positions, identify the origins and destinations on the map, and extract the spatial information; and (3) to serialize the user's behavior, which converts the user's riding history into a serial format.

Candidate Generation. This component identifies the set of most likely destination candidates using the frequent pattern mining methods outlined in Section 3.

Destination Prediction. In this component, the spatial information, the user's behavior series, and external features are used to predict the user's destination from the candidates in the set. More details are provided in Section 3.

3. Preprocessing

Before constructing the model, the data needs to be preprocessed to remove as much erroneous, abnormal, and redundant data as possible to make it easier to construct a robust prediction model. However, this process must preserve the reliability and quality of the data without changing the data distribution. Hence, the procedure involves three tasks to prepare the data for further processing.

Bike Record Data Parsing. This step filters noisy data out of the dataset. Redundant records are removed. For example, if one user has multiple records covering the same period of time, the most likely route taken is retained and the others are discarded. Incomplete records are removed, such as those missing a user ID, time, origin, destination, and so on. Records with a short distance between the origin and the destination but with a long duration are regarded as invalid and are also removed. In addition, we also found some records with latitudes and longitude beyond the range of Beijing, which were removed, along with some other outlier data found using the detection methods in [6].

Once the data is cleaned, the dataset is converted from its original field format to the input format required by the training model. For the candidate generation model, we simply extract the order ID, user ID, origin, and destination information. Then, we can get the frequent items from these datasets. However, the destination prediction model requires features such as the user behavior sequence and the position map, which demands a more complex extraction process. This is described in Section 5.

Position Mapping. In this step, the latitudes and longitudes of each position are plotted onto a corresponding map. In the Mobike dataset, each location is geohashed; therefore the hashed positions needed to be decoded into latitudes and longitudes. Once we finish mapping the positions to a matrix and map, relevant features for these positions can be extracted, such as the type of location or the local weather conditions. These features are important for constructing the spatial and external feature vectors in the following models.

User Behavior Serializing. In this step, user behaviors are sorted into a series according to time. Specific users at specific times with specific origins are converted from per-line data into a corresponding behavior sequence vector as a convenient input for the subsequent prediction models. This process is applied to every user and record. More details are provided in Section 5.1.

4. Candidate Generation

The next step is to generate a pool of candidate destinations. As previously mentioned, we have framed this destination prediction problem as a recommendation problem and the solution as a binary classification problem, where the positive samples in the training set are the ground truth destinations. However, an appropriate balance between positive and negative samples is crucial. Too many negative samples would lead to a massive computational overhead. And too many positive samples will cause the imbalance of positive and negative samples and may result in prediction failure. For example, consider a city with 10,000 possible locations but only one likely destination for a specific user at a specific time given their starting point. This would result in a 1:10,000 ratio of positive to negative samples. Multiply that by a thousand orders and the number of samples becomes $1000*10{,}000 = 10{,}000{,}000$. Thus, every additional order would increase the amount of data exponentially. However, if only the 10 most likely destinations for a thousand orders were included in the candidate pool, the number of input datasets would be just $1000*10 = 10{,}000$, which vastly reduces the computational complexity.

Hence, generating a manageable candidate pool can be seen as a frequent itemset mining problem, where the goal is to identify the most common origin-destination itemsets from a user's historical data. This is discussed in more detail in the next section.

4.1. FP-Growth. To identify the most likely user destinations, we use an approach based on frequent-pattern-trees (FP-trees), i.e., the FP-Growth algorithm [7]. FP-trees are an extended prefix tree structure for storing crucial information about frequent patterns in a compact way, and FP-Growth is an efficient FP-tree based mining algorithm for mining complete sets of frequent patterns according to pattern fragment growth.

FP-Growth first compresses the input datasets, creating an FP-tree instance to represent frequent items. Then, the compressed datasets are divided into subsets of conditional datasets, each one associated with a unique frequent pattern. Each conditional dataset is then mined separately. Using this strategy, FP-Growth not only reduces the search costs, by recursively looking for shorter patterns and concatenating them into longer frequent patterns once found, but also offers good selectivity. In this problem setting, mining frequent items from historical user data with a traditional statistical method, such as the Apriori algorithm, would be both computationally intensive and, likely, less accurate. The FP-Growth algorithm, however, can extract frequent items quickly with less overhead, making it a suitable choice for identifying the most likely user destinations for the candidate set. The next section explains the candidate generation model in more detail.

4.2. Candidate Generation Model. As Algorithm 1 shows, four different itemsets are mined from the user's historical data to construct the pool of candidate destinations. Each is explained below.

User-Origin-Destination, denoted as C_{UOD}, reflects the destinations a user has most commonly traveled to with considering the origin.

User-Origin, denoted as C_{UO}, represents all the locations where a user most frequently begins their journey without considering their destination. This itemset has been included because cyclists sometimes travel a route in reverse and the origin becomes the destination.

User-Destination, denoted as C_{UD}, reflects all the locations where a user has most often returned a bike, i.e., past destinations, because users often return to the same destinations.

Origin-Destination, denoted as C_{OD}, considers all users, not just a specific user, and reflects the most common destinations for a given starting point.

```
Input: Training set $T$
Output: Candidate Set $C_{FP}$
1: Initialize the time ranges $t$ in training set $T$
2: Select the data items from training set $T$ at time $t$
3: Get $C_{UOD}$ according to users, origins and destinations with FPG at minimum
   support $S_{UOD}$
4: Get $C_{UO}$ according to users and origins with FPG at minimum support $S_{UO}$
5: Get $C_{UD}$ according to users and destination positions with FPG at minimum
   support $S_{UD}$
6: Get $C_{OD}$ according to origins and destination positions with FPG at minimum
   support $S_{OD}$
7: Get the final candidates $C_{FP} = C_{UOD} \cup C_{UO} \cup C_{UD} \cup C_{OD}$
8: Return the candidate set $C_{FP}$
```

Algorithm 1: Candidate generation algorithm.

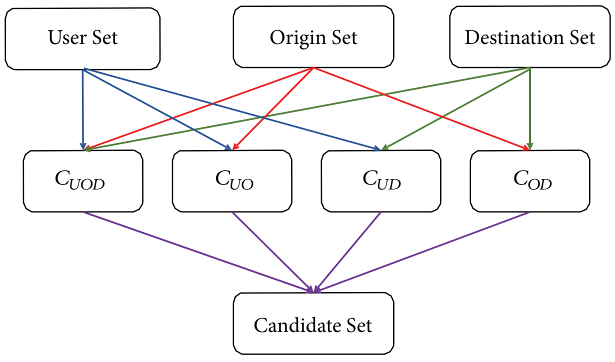


Figure 4: The candidate generation process.

With these four frequent itemsets extracted, the set of candidate destinations is constructed as follows.

$$C_{FP} = C_{UOD} \cup C_{UO} \cup C_{UD} \cup C_{OD} \quad (4)$$

Figure 4 provides a schematic overview of the process, and the algorithm is presented as Algorithm 1.

5. Destination Prediction

With the candidate destination set in hand, the next goal is to classify the likelihood that a user intends to travel to each location. Given that this is a binary classification problem, a candidate destination is labeled 1 if the destination is likely, and 0 otherwise.

5.1. Influence Factors. First of all, we need to analyze the factors that influence the users' cycling and the pattern of origins and destinations.

Users' Behavior Analysis. We use user id 2730 as an example to analyze the influence of user's behavior to the destination. As is shown in Tables 2 and 3, it can be seen from the data in May 14th that places "wx4gn0q" and "wx4gn2" appeared two times from May 11th to 13th. So, the high frequency location in historical data may be one of the destinations.

Spatial Relationship. As is shown in the Table 4, we have counted the high frequency top 10 origin-destination points. It can be found that there are strong correlations between these points and users often cycle between them. Therefore, it is necessary to learn these rules from a model.

External Factors. There are many external factors affecting traffic flow, such as weather, temperatures, and user's features. These factors have been described in [8]. Here, we use the conclusion to build models directly to learn these features.

5.2. Destination Prediction Network. Above all, the destinations are all affected by origin, the historical behavior of users, and the external features. Inspired by these factors, the model provides a detailed description of the classification tasks and the different factors considered by the three separate neural networks.

DPNst consists of three major components, as illustrated in Figure 5: a user behavior sequence model, a position map, and external features. The external features include meteorological information, riding time, and geographical features.

A user's historical behavior is first sorted into a series according to time and then is input into an LSTM network to learn the temporal rule of the origin and destination. Next, the spatial relationships between the origins and destinations are extracted and placed on a position map. This map is converted into a 2-channel image-like matrix to train the CNN and learn the spatial relationships. Lastly, the external features are input into the FCNN. The outputs of the three components are combined to produce the final results.

We adopt the parametric-matrix-based fusion method proposed in the ST-ResNet [8]. DPNst is to fuse the output from each of the component neural networks in a parametric matrix, as shown below:

$$\widehat{\mathbf{X}} = \mathbf{W}_{ubs} \circ \mathbf{X}_{ubs} + \mathbf{W}_{pm} \circ \mathbf{X}_{pm} + \mathbf{W}_{ef} \circ \mathbf{X}_{ef} \quad (5)$$

where $\circ$ is Hadamard product (i.e., element-wise multiplication) and $\mathbf{W}_{ubs}$, $\mathbf{W}_{pm}$, and $\mathbf{W}_{ef}$ are learnable parameters that adjust the degree of influence of each of the neural networks, the LSTM, the CNN, and the FCNN, respectively.

Table 2: Behaviors of user 2730 in 14th May.

user ID	time	origin	destination
3093685	2017-05-14 15:23:01	wx4gn29	wx4gn0k
2178747	2017-05-14 15:37:23	wx4gn0m	wx4gn0h
3409017	2017-05-14 17:08:20	wx4gn2h	wx4gn0r
3192545	2017-05-14 10:29:06	wx4gn21	wx4gn22
366384	2017-05-14 10:35:58	wx4gn21	wx4gn0e
164139	2017-05-14 14:40:00	wx4gn29	wx4gn2h
1682231	2017-05-14 17:40:01	wx4gn0q	wx4gn2h
3076183	2017-05-14 16:00:50	wx4gn0q	wx4gn0j
1682232	2017-05-14 21:26:15	wx4gn2h	wx4fypy
3850094	2017-05-14 16:26:45	wx4gn0q	wx4gn2h
3900595	2017-05-14 17:19:20	wx4gn0r	wx4gn0y
3093686	2017-05-14 22:00:48	wx4gn25	wx4gn29

Table 3: Behaviors of user 2730 in 11th to 13th May.

user ID	time	origin	destination
3218948	2017-05-12 21:48:31	wx4gn2m	wx4fyrf
1161301	2017-05-12 22:32:51	wx4gn2g	wx4gn2h
3530242	2017-05-12 15:18:11	wx4dzyz	wx4dzzj
2075155	2017-05-12 15:26:31	wx4dzzm	wx4epb8
94241	2017-05-11 18:57:00	wx4gn0q	wx4gn2h
759273	2017-05-12 18:18:42	wx4gn0m	wx4fyru
685779	2017-05-12 21:01:32	wx4gn0q	wx4gn2h
3622192	2017-05-13 19:54:23	wx4gn25	wx4gn0q
1229376	2017-05-13 20:16:54	wx4gn0r	wx4gn0w

Table 4: The top 10 origin and destination patterns.

origin	destination	count
wx4f9ky	wx4f9mk	681
wx4f9mk	wx4f9ky	497
wx4f9kn	wx4f9mk	437
wx4f9kn	wx4f9ms	372
wx4fg87	wx4ferq	356
wx4f9ky	wx4f9ms	356
wx4f9wb	wx4f9mu	355
wx4f9ms	wx4f9kn	345
wx4eq0c	wx4eq23	323
wx4f9mk	wx4f9kn	319

A cross combination softmax function generates the probability value of the classification prediction. The cross-entropy loss function is as follows.

$$L_\theta = -\frac{1}{m}\sum_{i}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right) \tag{6}$$

The learning process of the DPNst is shown in Algorithm 2. We first construct the training instances from the datasets. Then, DPNst is trained via backpropagation [9] and Adam [10].

5.3. The Structure of User Behavior Sequence Component. Given that bike-sharing users repeatedly rent bikes over a period of time, their historical data can be formulated as a time series, i.e., a behavior sequence with a time attribute. Typically, time series data are trained with a recurrent neural network (RNN) [11]. However, in recent years, LSTMs [3] have been successfully used to train complex time series data in a variety of applications, such as highway traffic prediction [12], traffic speed prediction [13], and tourism prediction [14]. Unlike the simple neurons in an RNN, LSTM neurons contain an input gate, an output gate, a cell, and a forget gate that determines how the information flows into and out of the neuron. Moreover, because LSTMs were specifically developed to overcome the exploding and vanishing gradient problems associated with training traditional RNNs in some scenarios, LSTMs are particularly well-suited to classification, processing, and prediction tasks with time series data that contain a time lag between important events of an unknown size or duration. Hence, the user behavior sequence component in DPNst is based on an LSTM network.

First, the data is converted into a sequence of user behaviors according to time, and the number of user-destination itemsets is counted to generate a sequence of behavior according to time. Assuming the current moment

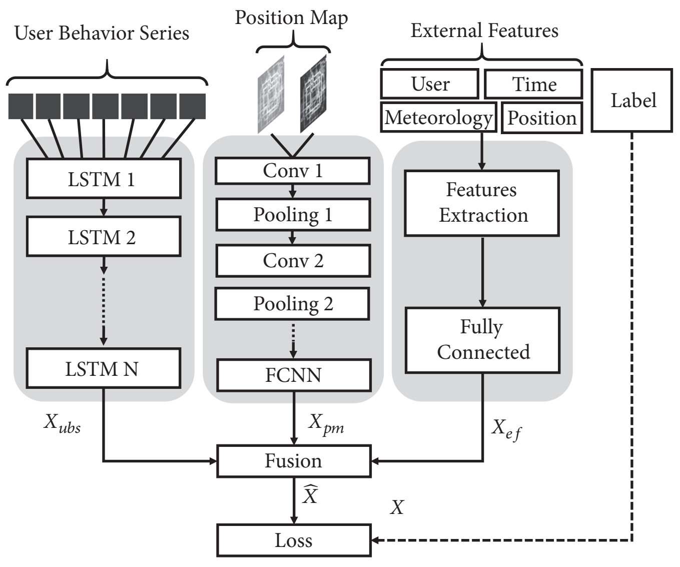


FIGURE 5: The network architecture.

```
Input: construct $\mathbf{X}_u = \{x_1, x_2, ..., x_{n-1}\}$, $\mathbf{X}_p = \{x_{ori}, x_{dest}\}$,
$\mathbf{X}_e = \{x_1, x_2, ..., x_{14}\}$ from candidate set $C_{FP}$
Output: Learned Destination Prediction Model $f$
 1: Initial the parameters $\theta$ in the networks
 2: Repeat
 3:  input $\mathbf{X}_u$ into the LSTM and get $\mathbf{X}_{ubs} = f_u(\mathbf{X}_u)$
 4:  input $\mathbf{X}_p$ into the CNN and get $\mathbf{X}_{pm} = f_p(\mathbf{X}_p)$
 5:  input $\mathbf{X}_e$ into the FCNN and get $\mathbf{X}_{ef} = f_e(\mathbf{X}_e)$
 6:  find the best $\theta$ with a cross-entropy loss function
 7: Until get the best $\widehat{X} = f(\mathbf{X}_{ubs}, \mathbf{X}_{pm}, \mathbf{X}_{ef})$
```

ALGORITHM 2: Destination prediction training algorithm.

in time is t and the number of time windows is n, these sequences are constructed as $[x_{t-n}, x_{t-(n-1)}, \ldots, x_{t-1}]$, which represents the total number of cycling trips from the origin to the destination in each time window. If there are no recorded trips in a window, the value is 0.

To identify and extract these patterns from historical behavior, a LSTM with many layers and hidden units is needed, as shown in Figure 6. A sequence is input into the first layer of the LSTM and output through a series of hidden units to the next layer. The final output is the output of the last hidden unit in the last LSTM layer. That output is then fed into a softmax activation function to generate the final prediction result.

For hidden unit h at time t, the output of h is presented as

$$\begin{aligned} f_t &= \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \\ i_t &= \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \\ \widetilde{C}_t &= \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\ o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{7}$$

where W denotes the weight matrixes, b denotes the bias vectors, i represents the input gate, f represents the forget gate, and o represents the output gate; σ is a sigmoid function.

The final prediction result is

$$\mathbf{X}_{ubs} = W * h_t + b \tag{8}$$

where h is the last hidden unit.

5.4. The Structure of Position Map Component. The biggest difference between predicting user behavior in a bike-sharing scenario and traditional time series problems is that bike-sharing data has spatiotemporal qualities. Therefore, capturing the relationships between spatial positions is very important. The relationship between spatial positions can be mapped as a two-dimensional matrix, which can, in turn, be regarded as a graph. Hence, the relationship between one point and another can be seen as the relationship between different geographical locations.

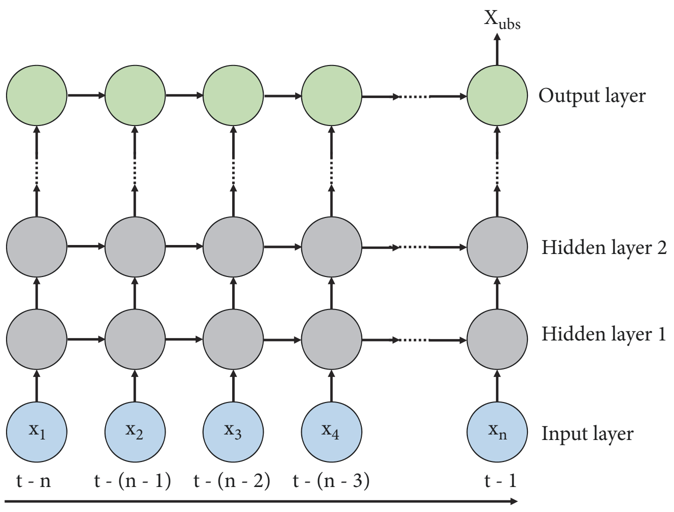


Figure 6: The architecture of the LSTM.

CNNs [4, 15] are well-suited to dealing with image information and can flexibly capture local relationships within and between images. Further, CNNs have a proven and powerful ability to hierarchically capture structural spatial information while extracting key features, and convolutional operations can be pooled to reduce complexity. Hence, CNNs constitute a highly appropriate way to extract information about the relationships between spatial locations on a map.

The first step in this component is to accurately place all the candidate destinations as positions on a map, so the CNN can extract the relationships between each position. However, given that one convolutional layer can only consider near spatial dependencies, as limited by the size of the kernels [8], the CNN in DPNst needs to contain several convolutional and pooling layers, as shown in Figure 7.

The origin and candidate destinations are parsed as a 2-channel map. Each position is marked on a 2D image; the origin is labeled as 1; the destinations are labeled 0. We handle all the datasets into maps and convert them into a tensor $\mathbf{X}^{(0)} \in \mathbb{R}^{r*I*J*2}$ where r is the numbers of maps, I is the height of the maps, and J is the width of the maps. A convolution layer follows, expressed as

$$\mathbf{X}_c^1 = f\left(W_c^1 * X_c^0 + b_c^1\right) \tag{9}$$

where $*$ denotes the first layer of the CNN, f is an activation function, e.g., ReLU, and W_c^1, b_c^1 are the learnable parameters in the first layer. Then, $\mathbf{X}_c^1$ is input into the pooling function F as follows.

$$\mathbf{X}_p^1 = F\left(X_c^1\right) \tag{10}$$

In our DPNst, we stack l convolution layers and pooling layers. Through multilayer convolution, the network could find the corresponding relationship between the origin and destination of different users. Finally, we add 2-layer FCNN to get the final result $\mathbf{X}_{pm}$.

5.5. The Structure of External Component. Beyond the relationships inherent in historical user behavior and locations, other factors, such as weather and the time of day, are also important. Even a user's personal information may affect their mode of travel and their destination. Therefore, DPNst contains an FCNN to parse these external features. The features considered follow.

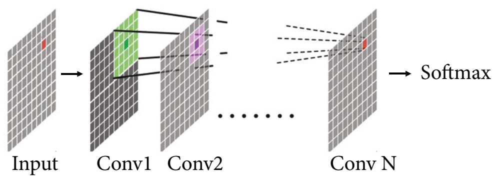


Figure 7: The architecture of the CNN (1 channel).

User Features. Each user has a unique identity in the dataset, which can be used as a basis for distinguishing specific personality traits reflected in the user's riding history. Therefore, through one-hot encoding, x_1 denotes the user ID, and the dimension represents the number of users.

Time Features. Time plays a direct role in the relationship between an origin and a destination because, naturally, users often travel to the same destination at a specific time of the day or week. Hence, the current time is constructed as a feature. We define the x_2 as month, x_3 as day, x_4 as hour, x_5 as minute, x_6 as weekday, and x_7 as weekend or public holiday (if today is not a weekday, the value is 1, else 0).

Meteorology Features. Weather affects many things, including traffic [16], a user's preferred mode of transport, and how far they are likely to ride. x_8 denotes the temperature at daytime and x_9 at night (in °C), and x_{10} denotes the Beaufort wind force scale. These features are all continuous values. x_{11} denotes the weather conditions. These values are discrete variables, such as sunny and raining. These categorical variables are encoded into a numerical vector using one-hot encoding and allocated to a categorical length vector, as shown in (11). This method can improve a model's training performance. x_{12} represents the air quality index (AQI).

$$x_{ij} = \begin{cases} 1, & \text{if } x_i \text{ is in category } j \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Position Features. These features represent the geographical characteristics of a location. x_{13} represents the distance between the origin and the candidate destination in terms of the Haversine equation, shown in (12). x_{14} is the location category, such as office, school, residence, and community service. Again, these features are defined through one-hot encoding:

$$\begin{aligned} h_{i,j} &= \sin^2(\Delta lat) + \cos(lat_i)\cos(lat_j) + \sin^2(\Delta lng) \\ d_{i,j} &= 2 \cdot R \cdot \arcsin\left(\sqrt{h_{i,j}}\right) \end{aligned} \tag{12}$$

where (lng_i, lat_i) and (lng_j, lat_j) are the latitude and longitude in radians of the two locations. Degrees are converted into radians by multiplying by $\pi/180$ as usual. $\Delta lat = (lat_i - lat_j)/2$, $\Delta lng = (lng_i - lng_j)/2$, and R is the radius of the Earth, which is about 6371 km.

The external features are constructed, then normalized, and input into a multilayer FCNN to learn their regularities. The final output is denoted as $\mathbf{X}_{ef}$.

6. Experiments

6.1. Datasets. To evaluate DPNst's performance, we conducted a series of experiments using datasets from Mobike's stationless bike-sharing scheme in Beijing combined with meteorological data from the Biendata Platform [17]. The Mobike dataset spans the period 10-24 May 2017 and contains over 3 million historical records for around 349,000 users and 485,000 bikes. The information includes order ID, user ID, bike ID, bike type, start time, and geohashed origins and destinations. The meteorological information spans the same time period and was sourced from the China Meteorological Administration website [18]. It includes weather conditions, temperatures, wind directions, Beaufort wind force scales, and other information. The statistics for each dataset are provided in Table 5.

6.2. Preprocessing. The data was preprocessed following the procedure outlined in Section 3. Then, we sampled the data, at different rates for each of the three neural networks, to reduce the computing complexity, ensuring that an appropriate balance between positive and negative records was maintained. The min-max normalization method was used to scale the data to the correct range [-1;1].

6.3. Baseline

6.3.1. Baseline of Candidate Generation. To evaluate each aspect of the candidate generation method, we constructed four baselines as follows.

TABLE 5: The details of Mobike datasets.

Datasets	**Mobike Beijing**
Time Period	10 May 2017 to 24 May 2017
Number of Users	349,693
Number of Bikes	485,465
Number of Records	3,214,096
Gird map size	(1452, 1716)
Range of Latitude	(20.01N, 40.66N)
Range of Longitude	(102.65E, 122.13E)
Datasets	**Meteorology**
Weather conditions	6 types (e.g., Sunny, Rainy)
Temperature / (°C)	$[11, 34]$
Beaufort Wind Force Scale	$[2, 5]$
Air Quality Index (AQI)	$[40, 396]$

User-Destination Count (UD). We identified the user-destination itemsets with the highest counts as candidate destinations.

User-Origin Count (UO). We scan specific users and destinations to identify the highest counting items as candidate destinations and add to UD.

Origin-Destination Count (OD). We used statistical methods to scan the origins and destinations of all users to find out the highest counting items as the candidates and add to UD and UO.

Candidate Generation Model (CGM). The most frequent itemsets for user-origin-destination were determined with the FP-Growth algorithm, using different minimum support parameters for each itemset. We will set four sets of minimum support to verify the effect of the model.

6.3.2. Baseline of Destination Prediction. Similarly, to evaluate various aspects of the destination prediction model, we constructed four further baselines as follows.

Historical Count (HC). The training set included the destinations a specific user went to the most times; the testing set included the latest data.

Naive Bayesian (NB). We use a simple naive Bayesian model to predict the destination by conditional probability by using the latest data in the training set.

To assess each of DPNst's three components, we constructed three further baselines as follows:

DPNst1: UBS. Only the LSTM was used to train the user behavior sequences.

DPNst2: UBS + PM. The LSTM was used to train the user behavior sequences and the CNN was used to train the position maps.

DPNst3-5: UBS + PM + EF. The LSTM was used to train the user behavior sequences, the CNN was used to train the position maps, and a multilayered FCNN was used to train the external features.

6.4. Hyperparameters. All models were built using Python libraries, including Numpy, Pandas, scikit-learn, and Tensorflow [19] (GPU version 1.2.1). Descriptions of the hyperparameters for both the CGM and DPNst models follow.

Hyperparameters of CGM. The FP-Growth algorithm within the CGM includes three defined hyperparameters. Minimum support S_{UOD} gauges the correlations between frequent user, origin, and destination items. Minimum support S_{UO} gauges the confidence in the correlation between frequent user and origin items. Minimum support S_{UD} gauges the correlations between frequent user and destination items. And minimum support S_{OD} gauges the correlations between frequent origin and destination items. The appropriate levels of minimum support are tuned through experimentation. The smaller the support, the higher the recall and the higher the mean numbers of candidate. So, we need to find the values that could keep balance.

Hyperparameters of DPNst. The LSTM contains 10 hidden units, with a variable number of layers. In the CNN, Conv1 contains two 5 ∗ 5 filters, and Conv2 contains four 10 ∗ 10 filters, each with a batch size of 1000. The drop-out rate was set to 0.8. The model was subsequently trained on the full set of training data for a fixed number of epochs. However, it is worth noting that hardware configurations significantly affect the optimal parameter settings. Therefore, these parameters need to be tuned to suit the specific platform configuration. The increase of LSTM layers number could learn more users behavior from the data and the increase of FCNN layers could also learn more from the external features. But the layers of CNN may not need to be higher than 3; it would be more computing cost.

6.5. Evaluation Metrics

6.5.1. Baseline of Candidate Generation. We used recall to evaluate the performance of both the candidate generation and destination prediction model, and the mean number of candidate destinations to evaluate the candidate generation model. The formulas for each metric follow:

$$Recall = \frac{TP}{TP + FN}$$
$$Mean = \frac{1}{N}\sum N_i \qquad (13)$$

where TP are the true positive samples, FN are the false negative samples, and N_i is the number of candidate destinations in the ith sample.

6.5.2. Baseline of Destination Prediction. Additionally, we used F1-scores and accuracy to evaluate the performance of the destination prediction model. The formulas follow:

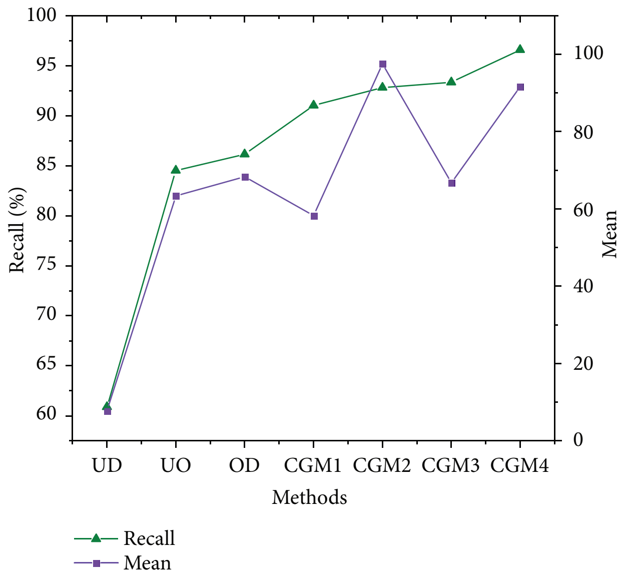


FIGURE 8: The candidate generation results.

$$Precision = \frac{TP}{TP + FP}$$
$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \qquad (14)$$

where FP are the false positive samples.

6.6. Results

6.6.1. Results of Candidate Generation. The experimental results are provided in Table 6 and Figure 8, where the results generated by each of different baselines are clear. The baseline relying solely on the frequent user-destination itemset had the lowest minimum recall at 60.89% with a mean of 7.76. Although the mean is small, the cover rate is quite low. However, after including the frequent users-origin itemsets, recall increased to 84.51% with a mean of 63.38. Moreover, after including the frequent origin-destination itemsets, recall increased even further to 86.13% with a sharp increase in the mean to 68.31.

The results for the full candidate generation model show a still more significant improvement. Recall increased to 91.02%, and the mean decreased to 58.29 with a minimum support of $S_{USD} = 1$, $S_{US} = 2$, $S_{UD} = 2$, $S_{SD} = 3$. And, as the minimum support decreased, both the recall and the mean increased. This is because decreasing the support relaxes the constraints on the frequent itemsets and more itemsets become available to satisfy the condition. This result also demonstrates the importance of choosing the appropriate parameters to balance the demands of computational complexity with the desired accuracy of the prediction model. Unfortunately, there is no standard choice and the parameters need to be tuned for each specific hardware configuration. In this paper, we chose $S_{USD} = 1$, $S_{US} = 2$, $S_{UD} = 2$, and $S_{SD} = 2$ as the best solution because these settings produced very high recall with an acceptable mean.

Table 6: Results of candidate generation.

Methods	Recall	Mean
User-Destination Count (UD)	60.89%	7.76
User-Origin Count (UO)	84.51%	63.38
Origin-Destination Count (OD)	86.13%	68.31
Our Methods		
Candidate Generation Model1 (CGM1)		
$S_{USD} = 1, S_{US} = 2, S_{UD} = 2, S_{SD} = 3$	91.02%	58.29
Candidate Generation Model2 (CGM2)		
$S_{USD} = 1, S_{US} = 1, S_{UD} = 1, S_{SD} = 3$	92.81%	97.57
Candidate Generation Model3 (CGM3)		
$S_{USD} = 1, S_{US} = 2, S_{UD} = 2, S_{SD} = 2$	**93.34%**	**66.70**
Candidate Generation Model4 (CGM4)		
$S_{USD} = 1, S_{US} = 2, S_{UD} = 2, S_{SD} = 1$	96.57%	91.58

Table 7: Results for different methods of destination prediction.

Methods	Recall	Precision	F1
Historical Count (HC)	28.96%	24.95%	26.81%
Naive Bayesian (NB)	32.14%	27.69%	29.75%
Our Methods (l means layers)			
DPNst1: UBS			
(2-l LSTM)	33.12%	27.58%	30.10%
DPNst2: UBS + PM			
(2-l LSTM + 2-l CNN)	35.46%	30.55%	32.82%
DPNst3: UBS + PM + EF			
(2-l LSTM + 2-l CNN + 2-l FCNN)	37.54%	32.34%	34.75%
DPNst4: UBS + PM + EF			
(2-l LSTM + 2-l CNN + 5-l FCNN)	31.98%	59.56%	41.62%
DPNst5: UBS + PM + EF			
(5-l LSTM + 2-l CNN + 5-l FCNN)	**35.27%**	**54.12%**	**42.71%**

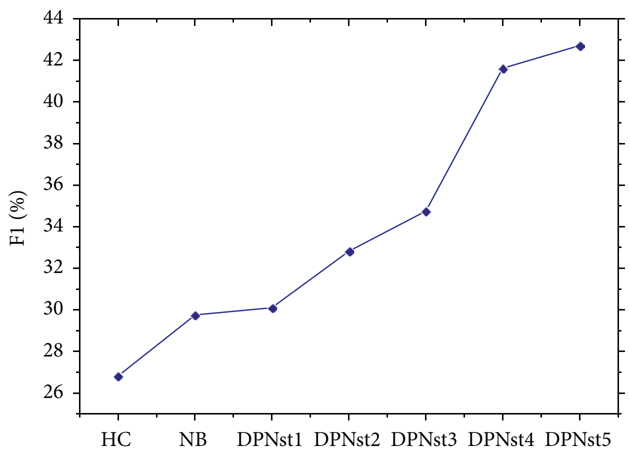


Figure 9: The destination prediction results.

6.6.2. Results of Destination Prediction. The results from the destination prediction evaluation are presented in Table 7 and Figure 9 which, again, clearly show the effect of each baseline. The baseline relying only on the highest position count resulted in the least accuracy followed by the naive Bayesian model. DPNst was the most accurate, demonstrating a 15.9% increase over the next best approach in terms of F1. DPNst was the most accurate, with a recall of 35.27%, a precision of 54.12%, and an F1 score of 42.71%, representing an increase of 15.9% in terms of F1. These results provide support for DPNst as a well-performing model in bike-sharing systems.

To further assess the model, we analyzed its performance at the component and layer level. As each 2-layer network was added, the recall, precision, and F1 scores increased. However, with a 5-layer FCNN, recall decreased to 31.98%, yet precision increased to 59.56%, and the F1 score increased to 41.62%. The best performance resulted from adding a 5-layer LSTM where recall increased to 35.27%, precision decreased to 54.12%, and the F1 score increased to 42.71%. This confirms that each component of DPNst helps to improve the model's performance.

7. Related Work

In recent years, bike-sharing has received increasing attention due to its significance as an environmentally friendly form of travel and its ability to overcome the “last mile” problems associated with other forms of mass transit. Studies on bike-sharing span both station-based and stationless systems, with many focusing on public schemes. DeMaio [20] provided an introduction to bicycle-sharing systems, including their history, impacts, models, and what the future for research in this field may hold. Etienne et al. [21] studied a statistical model of public bicycle travel based on Velib’s system in Paris, France. Midgley [22] analyzed the state-of-the-art and users’ experiences with several station-based public bicycle systems across Europe. These early studies laid the foundational concepts and working mechanisms of public bike-sharing systems.

In time, scholars began to examine some of the problems associated with bike-sharing schemes. Given that people’s bike use tends to be quite skewed and imbalanced, Pavone et al. [23] developed methods for maximizing the throughput of a mobility-on-demand urban transportation system and introduced a rebalancing policy to minimize the number of vehicles needed for rebalancing trips. This advancement provided vital inspiration for solving balance and load problems in public bike-sharing systems.

Studying user behavior patterns in bike-sharing systems helped us to understand the flow and mobility patterns in public bicycle traffic. For example, Jon Froehlich et al. [24] presented a 13-week spatiotemporal analysis of bicycle station usage in Barcelona’s bike-sharing system. Kaltenbrunner et al. [25] provided an analysis of human mobility data in urban areas based on the number of available bikes at Bicing stations C, a community bicycle program in Barcelona. Vogel et al. [26] adopted clustering and validation to analyze bike usage patterns in Vienna. Beyond insights into mobility and public bicycle flow, these studies also contributed the notion of leveraging station clustering, based on geographical position and transition patterns, as a way to reallocate bicycles to compensate for imbalanced use.

Contardo et al. [27] and Benchimol et al. [28] each presented mathematical formulations to reroute vehicles and transfer bikes. These formulations consider external features, such as vehicle capacity and the extent of the imbalance. However, simply monitoring the current number of bikes at each station and reallocating bikes after an imbalance occurs constitute a remedy for the problem, not a cure. Hence, a new set of studies that explored ways to predict potential imbalances in advance emerged.

Borgnat et al. [29] used a combination model and Velov’s dataset to predict traffic across the entire bike-sharing system at each hour of the day. Vogel et al. [26, 30] used time series analysis to forecast bike demand in Vienna, while Yoon et al. [31] used a modified ARIMA model to predict the available bikes and docks at each station by considering temporal and spatial factors. These studies provided insights into the influence of traditional market impacts on bike-sharing systems. Zheng et al. [32] predicted traffic flows at the check-in and check-out areas of New York and Washington’s public bicycle systems from a macro perspective, contributing a clustering algorithm based on k-means and a transition matrix. Conversely, Zhang et al. [33] adopted a micro perspective, using GBRT and Lasso regression to predict user behavior and travel times in Chicago’s public bicycle-sharing system. Each of these studies focussed on prediction: available bikes and docks, passenger numbers and flow at check-ins and check-outs, and so on. Their studies are the most closely related to the destination prediction problem explored in this paper.

Because stationless bike-sharing is a relatively newer business model, less research has been undertaken in this area. Jie [2] presented a data-driven approach for developing bike lane construction plans in Shanghai based on Mobike’s stationless trajectory data. This study examined mobility statistics in Mobike’s data, providing much of the inspiration for applying data mining techniques.

Further, some existing research has already been conducted on destination prediction. Natalia and Chris [34] and Patterson et al. [35] both used a Bayesian method to predict destinations for specific individuals based on their historical modes of transport. Tiesyte and Jensen [36] proposed a nearest-neighbor trajectory method that uses distance measures to identify the historical trajectory most similar to the current partial trajectory. Chen et al. [37] used a tree structure to represent historical movement patterns, which can be matched to the current partial trajectory by stepping down the tree. Zhang et al. [38] employed a Bayesian framework to model the distribution of a user’s destination based on their travel history using the DiDi Taxi dataset. Whereas each of these studies focuses on a traditional method, such as Bayesian frameworks or a nearest-neighbor method, our work incorporates deep learning to construct more intelligent models for destination prediction. Tang et al. [39] presented a system called PARecommender, which predicts traffic conditions and provides route recommendation based on generated traffic patterns.

Deep learning is an emerging, but already widely studied, field. CNNs have been successfully applied to many different kinds of problems, especially in the field of computer vision [15]. Further, as a quasi-substitute for CNNs, the capsule network was introduced Sabour et al. [40], which is a group of neurons with an activity vector that represents the instantiation parameters of a specific type of entity, such as an object or an object part. RNNs have found success in sequencing learning tasks [11], while other types of new networks have emerged to deal with spatiotemporal data. By extending a fully connected LSTM (FC-LSTM) to incorporate convolutional structures in both the input-to-state and state-to-state transitions, Shi et al. [41] proposed the convolutional LSTM (ConvLSTM) and used it to build an end-to-end trainable model for the precipitation nowcasting problem. These networks and architectures contributed to many of our ideas for constructing a destination prediction network.

Finally, the availability of massive amounts of mobility data from users, cars, and public transport systems has given rise to many urban computing techniques that solve tasks based on real-world travel demands [42]. For example, Zheng [43] mined patterns in taxi trajectories to recommend new road construction and public transport infrastructure projects. Yuan et al. [44] used traffic patterns and POI distributions to infer the different functional zones in a city.

These studies established new research methods for dealing with transportation datasets and problems.

8. Conclusion

In this paper, we proposed an innovative deep learning model called destination prediction network based on spatiotemporal data or DPNst for short. DPNst predicts the most likely destination of a cyclist in a bike-sharing system. The first step is to preprocess the data and identify the most likely candidate destinations using frequent item pattern mining. The DPNst model is then built through a series of three neural networks using this candidate set. An LSTM network [3] learns the user behavior. A CNN [4, 15] learns the spatial relationships between the origin and the candidate destinations, and an FCNN [5] learns the external features. To the best of our knowledge, this is the first proactive method to address the administrative problems associated with bike-sharing systems by predicting the most probable user destinations. A series of experiments with real-world data from Mobike show that DPNst achieves satisfactory prediction results, with an F1 score of 42.71%, and a better performance overall than the baseline methods. In future research, we hope to improve DPNst's performance and extend the model to destination prediction for taxis, private cars, and other problems that relate to traffic destination prediction.

Data Availability

The Mobike data used to support the findings of this study have been deposited in the Biendata competition platform (https://biendata.com/competition/mobike/data/).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by grants from the National Natural Science Foundation of China (No. 61602141) and Science and Technology Program of Zhejiang Province (No. 2018C04001).

References

[1] https://www.ft.com/content/5efe95f6-0aeb-11e7-97d1-5e720a26771b.

[2] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng, "Planning bike lanes based on sharing-bikes' trajectories," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017*, pp. 1377–1386, Canada, August 2017.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[5] R. J. Gibbens and F. P. Kelly, "Dynamic routing in fully connected networks," *IMA Journal of Mathematical Control and Information*, vol. 7, no. 1, pp. 77–111, 1990.

[6] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, article 29, 2015.

[7] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[8] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 1655–1661, USA, February 2017.

[9] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," *Lecture Notes in Computer Science*, vol. 1524, pp. 9–50, 1998.

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Computer Science*, 2014.

[11] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[12] F. Altche and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359, Yokohama, October 2017.

[13] Z. Cui, R. Ke, and Y. Wang, "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction," 2018.

[14] Y. Li and H. Cao, "Prediction for Tourism Flow based on LSTM Neural Network," *Procedia Computer Science*, vol. 129, pp. 277–283, 2018.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[16] F. Lin, J. Jiang, J. Fan, and S. Wang, "A stacking model for variation prediction of public bicycle traffic flow," *Intelligent Data Analysis*, vol. 22, no. 4, pp. 911–933, 2018.

[17] https://biendata.com/competition/mobike/data/.

[18] http://www.cma.gov.cn/2011qxfw/2011qsjgx/.

[19] https://www.tensorflow.org/.

[20] P. Demaio, "Bike-sharing: history, impacts, models of provision, and future," *Journal of Public Transportation*, vol. 12, no. 4, pp. 41–56, 2009.

[21] E. Come and O. Latifa, "Model-Based Count Series Clustering for Bike Sharing System Usage Mining: A Case Study with the Velib' System of Paris," *Acm Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, pp. 1–21, 2014.

[22] P. Midgley, "The Role of Smart Bike-sharing Systems in Urban Mobility," *Journeys*, vol. 2, no. 2, 2009.

[23] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, "Rebalancing the rebalancers: Optimally routing vehicles and drivers in mobility-on-demand systems," in *Proceedings of the 1st American Control Conference, ACC 2013*, pp. 2362–2367, IEEE, Washington, DC, USA, June 2013.

[24] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and Predicting the Pulse of the City through Shared Bicycling," in *Proceedings of the 21st international jont conference on Artifical intelligence*, pp. 1420–1426, Morgan Kaufmann Publishers Inc., Pasadena, California, USA, 2009 (Bulgarian).

[25] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, "Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 455–466, 2010.

[26] P. Vogel, T. Greiser, and D. C. Mattfeld, "Understanding bike-sharing systems using data mining: exploring activity patterns," *Procedia-Social and Behavioral Sciences*, vol. 20, no. 6, pp. 514–523, 2011.

[27] C. Contardo, C. Morency, and L. Rousseau, *Balancing A Dynamic Public Bike-Sharing System*, vol. 4, CIRRELT, Montreal, Canada, 2012.

[28] M. Benchimol, "Balancing the stations of a self service 'bike hire' system," *RAIRO-Operations Research*, vol. 45, no. 1, pp. 37–61, 2011.

[29] P. Borgnat, P. Abry, P. Flandrin, C. Robardet, J.-B. Rouquier, and E. Fleury, "Shared bicycles in a city: A signal processing and data analysis perspective," *Advances in Complex Systems (ACS)*, vol. 14, no. 3, pp. 415–438, 2011.

[30] P. Vogel and D. C. Mattfeld, "Strategic and Operational Planning of Bike-Sharing Systems by Data Mining – A Case Study," in *Proceedings of the International Conference on Computational Logistics*, vol. 6971, pp. 127–141, Springer Berlin Heidelberg, Hamburg, Germany.

[31] J. W. Yoon, F. Pinelli, and F. Calabrese, "Cityride: A predictive bike sharing journey advisor," in *Proceedings of the 13th International Conference on Mobile Data Management, MDM 2012*, pp. 306–311, IEEE, Karnataka, India, July 2012.

[32] Y. Li, Y. Zheng, H. Zhang, and L. Chen, "Traffic prediction in a bike-sharing system," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, Seattle, WA, USA, 2015.

[33] J. Zhang, X. Pan, M. Li, and P. S. Yu, "Bicycle-Sharing System Analysis and Trip Prediction," in *Proceedings of the 2016 17th IEEE International Conference on Mobile Data Management (MDM)*, pp. 174–179, IEEE, Porto, Portugal, June 2016.

[34] N. Marmasse and C. Schmandt, "A user-centered location model," *Personal and Ubiquitous Computing*, vol. 6, no. 5-6, pp. 318–321, 2002.

[35] D. J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," in *Proceedings of the 5th International Conference on Ubiquitous Computing*, pp. 73–89, Springer, Seattle, WA, USA, 2003.

[36] D. Tiesyte and C. S. Jensen, "Similarity-based prediction of travel times for vehicles traveling on known routes," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM GIS 2008*, pp. 105–114, ACM, Irvine, CA, USA, November 2008.

[37] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 657–676, 2010.

[38] L. Zhang, T. Hu, Y. Min et al., "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017*, pp. 2151–2159, Canada, August 2017.

[39] F. Tang, J. Zhu, Y. Cao et al., "PARecommender: A pattern-based system for route recommendation," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI 2016*, pp. 4272-4273, USA, July 2016.

[40] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Cap- sules," *NIPS Proceedings*, 2017.

[41] X. Shi, *Convolutional LSTM Network: A Machine Learning Appro- ach for Precipitation Nowcasting*, Convolutional LSTM Network, A Machine Learning Appro- ach for Precipitation Nowcasting, 2015.

[42] Y. Zheng, *Urban Computing: Concepts, Methodologies, and Appli- cations , Acm Transactions on Intelligent Systems Technology 5.3:1-55*, Acm Transactions on Intelligent Systems Technology 5.3, 1-55, 2014.

[43] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*, pp. 89–98, ACM, September 2011.

[44] J. Yuan, Y. Zheng, and X. Xie, "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 (KDD '12)*, pp. 186–194, August 2012.