information

Article

Vehicle Location Prediction Based on Spatiotemporal Feature Transformation and Hybrid LSTM Neural Network

Yuelei Xiao [1,2,*] and Qing Nian [1]

1 School of Modern Posts, Xi'an University of Post and Telecommunications, Xi'an 710061, China; nq_imagination@163.com
2 Shanxi Provincial Information Engineering Research Institute, Xi'an 710075, China
* Correspondence: xiao_yuelei@163.com; Tel.: +86-02987303602

Received: 12 December 2019; Accepted: 1 February 2020; Published: 4 February 2020

Abstract: Location prediction has attracted much attention due to its important role in many location-based services. The existing location prediction methods have large trajectory information loss and low prediction accuracy. Hence, they are unsuitable for vehicle location prediction of the intelligent transportation system, which needs small trajectory information loss and high prediction accuracy. To solve the problem, a vehicle location prediction algorithm was proposed in this paper, which is based on a spatiotemporal feature transformation method and a hybrid long short-term memory (LSTM) neural network model. In the algorithm, the transformation method is used to convert a vehicle trajectory into an appropriate input of the neural network model, and then the vehicle location at the next time is predicted by the neural network model. The experimental results show that the trajectory information of an original taxi trajectory is basically reserved by its shadowed taxi trajectory, and the trajectory points of the predicted taxi trajectory are close to those of the shadowed taxi trajectory. It proves that our proposed algorithm effectively reduces the information loss of vehicle trajectory and improves the accuracy of vehicle location prediction. Furthermore, the experimental results also show that the algorithm has a higher distance percentage and a shorter average distance than the other predication models. Therefore, our proposed algorithm is better than the other prediction models in the accuracy of vehicle location predication.

Keywords: location prediction; spatiotemporal feature shadowing; feature static processing; semantic feature mapping; long short-term memory (LSTM); hybrid LSTM

1. Introduction

With the rapid development and popularization of global positioning technologies and mobile communication networks, the location information of a moving object can be obtained by many devices, such as a mobile phone and global position system (GPS)-based equipment. The location information is the key information to analyze the behavior of the moving object. Based on the location information, the location predication of the moving object can be performed, which is of great significance in many location-based services (LBS). In recent years, the location prediction of a moving object was concerned by many scholars [1,2] and widely applied to many scenarios [3–5], such as package delivery and advertising recommendation systems. For example, while a package is being delivered, customers are eager to know where the courier is, and which place he would visit next. For advertising recommendation systems, places where customers will go can be predicted, and then various kinds of ads can be recommended to these customers.

The existing location prediction methods mainly include the location prediction methods based on the Markov chain (MC) model [6–16], the location prediction methods based on frequency pattern (FP) mining [17–26], and the location prediction methods based on recurrent neural network (RNN) [27–29]. The location prediction methods based on the MC model are to calculate the probability of the next location of a moving object by establishing the transfer-probability matrix of the moving object. These methods have the problems of transfer-state-space explosion and low prediction accuracy. For the location prediction methods based on FP mining, the moving pattern of a moving object is explored by mining its historical location trajectories, and then its location is predicted. However, this kind of method does not consider the sequence relationship of trajectory points in time and space. In the location prediction methods based on RNN, the trajectory sequence data of a moving object is processed through the middle recursion layer of the RNN, and then the location of the moving object is predicted. However, the RNN [28] cannot deal with long sequence dependence very well, thus it will lead to the gradient explosion and disappearance problems in [30] when dealing with long sequence data.

In recent years, a long short-term memory (LSTM) model [31] was proposed. It can deal with long sequence dependence very well and has obtained certain results in many fields, such as machine translation [32], information retrieval [33], and image processing [34]. Therefore, some researchers proposed some location prediction methods based on the LSTM model [35–37]. The above location prediction methods usually start with clustering trajectories into frequent regions or stay points, or simply partitioning trajectories into grids, and then predict the probability of each candidate region, point or grid, causing large trajectory information loss and low prediction accuracy. In an intelligent transportation system, the next location of a vehicle generally needs to be accurately predicted according to the past locations of the vehicle, which needs small trajectory information loss and high prediction accuracy. This is important for traffic congestion prediction. Hence, the above location prediction methods are unsuitable for vehicle location prediction.

To solve the problem, we proposed a vehicle location prediction algorithm in this paper. The algorithm includes a spatiotemporal feature transformation method and a hybrid LSTM neural network model. The former is used to convert a vehicle trajectory into an appropriate input of the latter. Then, the vehicle location at the next time is predicted by the latter. The experimental results indicate that our proposed algorithm effectively reduces the information loss of vehicle trajectory and improves the accuracy of vehicle location prediction, and is better than the other prediction models in the accuracy of vehicle location predication.

The remainder of this article is organized as follows. Section 2 provides an overview of the existing location predication methods. In Section 3, we provide the rationale of our proposed algorithm. Section 4 describes in detail the experimental process, experimental results, and analysis. Finally, we conclude the paper in Section 5.

2. Related Works

Recently, studies on the location prediction of a moving object have gained increasing popularity. In the aspect of the location prediction methods based on the MC model, Gambs et al. [6] proposed a mobile Markov chain (MMC) model to predict the location of a moving object by fusing the historical trajectory points of the moving object. In [7], Chen et al. proposed a hidden Markov model (HMM) to predict the location of a moving object. Mathew et al. [8] proposed a hybrid method to predict a moving object's location based on the HMM. This method clusters the moving object's historical trajectory points, and trains each category using the HMM. Li et al. [9] proposed a location prediction method based on a mixed multi-step Markov model, which can solve the problems of insufficient trajectory information utilization and low prediction accuracy in the first-order Markov model, and the problem of sharp expansion of state space in the multi-step Markov model. Karatzoglou et al. [10] investigated the performance of Markov chains with respect to modeling semantic trajectories and predicting future locations, and proposed semantic-enhanced multi-dimensional Markov chains on semantic trajectories to predict future locations. In [11], Yang et al. applied the density-based spatial clustering of applications with noise (DBSCAN) algorithm to cluster trajectory points to obtain stay points, and then used a variable-order Markov model to predict the next location. Lin et al. [12] proposed a Markov location prediction method based on moving behavior similarity and user clustering. In this method, they proposed a moving behavior similarity calculation method that considers both user transfer characteristics and user region characteristics. Then they clustered users according to the moving behavior similarity, and used a first-order Markov model to predict the location of the clustered user groups. Song et al. [13] proposed a location prediction algorithm based on the Markov model and trajectory similarity for moving objects, aiming at the problem of poor prediction accuracy of low-order Markov models and high prediction sparsity of multi-order Markov models. This algorithm takes the trajectory similarity as an important factor of location prediction, and combines the similarity factor with the Markov model to get the final prediction result. Li et al. [14] put forward a location prediction algorithm for a moving object, which is based on the movement tendency of the moving object. This algorithm not only uses the Markov model to model the moving object's historical activity trajectory, but also takes the moving object's movement tendency as an important factor of location prediction. It takes all the historical stay regions as candidate regions for a future location. In [15,16], a novel division method was proposed for preprocessing trajectory data. In this method, the feature points of an original trajectory are extracted according to the change of trajectory structure, and then important locations are further extracted by using an improved density peak clustering algorithm to cluster these feature points. Finally, to predict the next location of a moving user, a multi-order fusion Markov model based on an Adaboost algorithm was proposed.

In the aspect of the location prediction methods based on FP mining, Chen et al. [17] used the idea of data mining and the method of spatial region division to analyze and identify a user's trajectory points, and mined the user's movement patterns to predict the user's movement behavior. Koren et al. [18] proposed a matrix decomposition method that decomposes a user's behavior matrix to determine the next probable location of the user, which considers the geographical location and other features of the user. Zheng et al. [19] mined the location features related to the user's activity, then used the method of matrix decomposition to mine the user's location. Morzy et al. [20] introduced a data mining approach to predict the location of a moving object. They mined the location database of moving objects to discover movement rules. Then, they matched the trajectories of the moving object with these discovered movement rules to build a location probabilistic model of the moving object. Morzy et al. [21] presented a new method for predicting the location of a moving object. This method uses the past trajectories of the moving object, and compounds them with those discovered movement rules in the moving object's trajectory database. Deng et al. [22] proposed a method to dynamically analyze the trajectory pattern of a moving object and predict the trajectory point of the moving object, which is based on that trajectory data of massive users. In this method, an improved pattern mining model is used to extract the trajectory frequent pattern of the moving object. Then, a dynamic pattern tree (DPT) algorithm is used to store and query the trajectory frequent pattern of the moving object, and a location prediction algorithm is used to calculate the best matching degree and get the predicted location of the moving object. Jeung et al. [23] forecasted the future location of a user by using predefined motion functions and linear or nonlinear models, then extracted the movement patterns of the user by using an Apriori algorithm. Yavas et al. [24] utilized an improved Apriori algorithm to find association rules with support and confidence. These association rules reveal the co-occurrences of locations. Li et al. [25,26] proposed two kinds of trajectory patterns: the periodic behavior pattern and the swarm pattern. Trajectories are first clustered into reference spots, and a Fourier-based algorithm is utilized to detect periods. Then periodic patterns are mined by hierarchical clustering.

In the aspect of the location prediction methods based on RNN, Liu et al. [27] extended the original RNN [28], and proposed a spatiotemporal recurrent neural network (ST-RNN) to predict the next location of a moving object. This method applies the spatiotemporal features, but ignores the dependence problem of long trajectory sequence, which will lead to the decrease of prediction accuracy. Molegi et al. [29] proposed a location prediction method, called spatiotemporal features-based recurrent neural network (STF-RNN). This method is to input the spatiotemporal features directly into the network, and extract the internal representation of the spatiotemporal features through the network self-learning. However, the traditional RNN models cannot deal with the trajectory sequence of large amounts of historical trajectory data, which will lead to the problems of gradient disappearance, gradient explosion and historical information loss during parameter training [30], while the LSTM model [31] can solve these problems. The LSTM model has obtained some achievements in many fields, such as machine translation [32], information retrieval [33] and image processing [34]. Wu et al. [35] proposed a spatiotemporal semantic network algorithm based on the LSTM model to predict the location of a moving object. First, a spatial-temporal-semantic (STS) feature extraction algorithm is used to convert the trajectory to location sequences with fixed and discrete points in the road networks. Then, an LSTM-based model is constructed to predict the further location of the moving object. Gao et al. [36] proposed a location distributed representation model (LDRM) to solve the dimension disaster, which is caused by too many locations in the trajectory data of a moving object. In this model, a high-dimensional one-hot vector which is difficult to deal with is reduced to a low-dimensional location vector which contains the moving object's motion mode. Then, the model is combined with the LSTM model to predict the location of the moving object. In [37], Xu et al. proposed a short-term and long-term memory (ST-LSTM) model based on the spatiotemporal features, aiming at the problem that the existing location prediction studies ignore the correlation between time and space.

Additionally, the above location prediction methods often start with clustering trajectories into frequent regions or stay points, or simply partitioning trajectories into grids, and then predict the probability of each candidate region, point or grid, causing large trajectory information loss and low prediction accuracy. With the continuous expansion of urban size, the scale of urban traffic network is growing, and the number of vehicles is also increasing. As a result, traffic congestion has gradually evolved into an unavoidable problem. In an intelligent transportation system, the next location of a vehicle generally needs to be accurately predicted according to the past locations of the vehicle, which needs small trajectory information loss and high prediction accuracy. This is important for traffic congestion prediction. Hence, the above location prediction methods cannot be applied in vehicle location prediction.

3. Our Proposed Algorithm

Our proposed algorithm consists of a spatiotemporal feature transformation method and a hybrid LSTM neural network model, as shown in Figure 1. In this algorithm, the original trajectory data of a vehicle is preprocessed at first. Then, the spatiotemporal feature transformation method is used to convert the preprocessed vehicle trajectory data into an input $\boldsymbol{X}$ that is suitable for the hybrid LSTM neural network model. Finally, $\boldsymbol{X}$ is run through the hybrid LSTM neural network model, and an output $\boldsymbol{Y}$ is obtained.

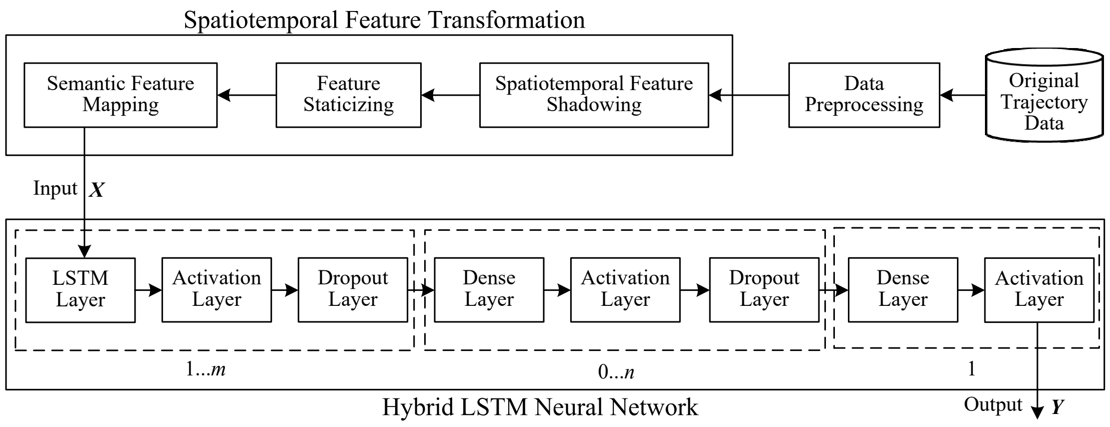

Figure 1. Our proposed algorithm.

In Figure 1, the spatiotemporal feature transformation method includes a spatiotemporal feature shadowing process, a feature staticizing process and a semantic feature mapping process. The hybrid LSTM neural network model consists of one or more LSTM combination layers, zero or more Dense combination layers, and one output combination layer. The LSTM combination layer includes an LSTM layer, an Activation layer, and a Dropout layer. The Dense combination layer includes a Dense layer, an Activation layer, and a Dropout layer. The output combination layer includes a Dense layer and an Activation layer. The Dropout layer means that some neural units are temporarily discarded from the neural network according to a certain probability during the training process, preventing over-fitting.

3.1. Spatiotemporal Feature Transformation

3.1.1. Spatiotemporal Feature Shadowing Process

The original trajectory of a vehicle is usually composed of a series of trajectory points, and its expression is $L = \{p_i | i = 1, 2, 3, \cdots, N\}$, where p_i denotes the i-th trajectory point of the original vehicle trajectory and $p_i = (lon_i, lat_i, t_i)$, lon_i is the longitude of p_i, lat_i is the latitude of p_i, t_i is the time of p_i, and N is the number of trajectory points of the original vehicle trajectory. As a result of the different sampling intervals and possible loss of trajectory points, the trajectory points of the original vehicle trajectory may be uneven on the time dimension. If the original trajectory data of the vehicle is directly imported into a location predication model, the uncertainty of the model will increase, and the prediction accuracy of the model will decrease. Therefore, the original trajectory data of the vehicle needs to be shadowed according to its spatiotemporal features, making the trajectory points of the shadowed vehicle trajectory even on the time dimension.

To solve this problem, a proper interval should be selected firstly. The time interval is mainly determined by the demand for specific services, and should be generally larger than the average sample rate [35]. If a time interval Δt is selected, then the number of trajectory points of the shadowed vehicle trajectory will be as follows:

$$N' = \left\lfloor \frac{t_N - t_1}{\Delta t} \right\rfloor \tag{1}$$

Correspondingly, the original vehicle trajectory $L = \{p_i | i = 1, 2, 3, \cdots, N\}$ will be transformed into the shadowed vehicle trajectory $L' = \left\{ {p'}_j \middle| j = 1, 2, 3, \cdots, N' \right\}$, where ${p'}_j$ denotes the j-th trajectory point of the shadowed vehicle trajectory and ${p'}_j = ({lon'}_j, {lat'}_j, {t'}_j)$, ${lon'}_j$ is the longitude of ${p'}_j$, ${lon'}_j$ is the latitude of ${p'}_j$, ${t'}_j$ is the time of ${p'}_j$. If ${t'}_j = t_i$, then ${lon'}_j = lon_i$ and ${lat'}_j = lat_i$, i.e., ${p'}_j = p_i$. Otherwise, ${lon'}_j$ and ${lat'}_j$ are calculated respectively as follows:

$${lon'}_j = \frac{lon_{i+1} - lon_i}{t_{i+1} - t_i}({t'}_j - t_i) + lon_i \tag{2}$$

$${lat'}_j = \frac{lat_{i+1} - lat_i}{t_{i+1} - t_i}({t'}_j - t_i) + lat_i \tag{3}$$

where t_i and t_{i+1} are the time of the previous trajectory point p_i and the time of the next trajectory point p_{i+1}, respectively. In other words, the previous trajectory point p_i and the next trajectory point p_{i+1} are linearly shadowed to ${p'}_j$ at ${t'}_j$.

3.1.2. Feature Staticizing Process

Stationary time series data is of great significance to time series modeling [38]. Therefore, time series data should be checked first to see if it is non-stationary. If it is non-stationary, then it needs to be transformed into stationary time series data before it is modeled. Time series graph is the simplest method to check whether time series data is stationary [39]. Unit root test is a test method commonly used in the statistical test. It is more accurate and important for testing the stability of time series data [40].

In this paper, we first checked whether the shadowed vehicle trajectory $L' = \left\{p'_j \middle| j = 1,2,3,\cdots,N'\right\}$ is stationary by observing its time series graph. Then we used the unit root test method of Augment Dickey-Fuller (ADF) [40] to further test whether it is stationary. The ADF test method consists of the following three models [40], as shown in Equations (4)–(6), respectively.

$$\Delta X_t = \delta X_{t-1} + \sum_{i=1}^{m} \beta_i \Delta X_{t-i} + \varepsilon_t \tag{4}$$

$$\Delta X_t = \alpha + \delta X_{t-1} + \sum_{i=1}^{m} \beta_i \Delta X_{t-i} + \varepsilon_t \tag{5}$$

$$\Delta X_t = \alpha + \beta t + \delta X_{t-1} + \sum_{i=1}^{m} \beta_i \Delta X_{t-i} + \varepsilon_t \tag{6}$$

where t denotes the time, X denotes the tested variable, ΔX denotes the one-order difference of X, α denotes the constant term, βt denotes the trend term, $\beta_i \Delta X_{t-i}$ denotes the lagging difference term, m denotes the number of lagging difference terms, and ε denotes the residual term. The original assumption of the ADF test method is $H_0 : \delta = 0$. An ADF test starts with the third model, then the second model, and the first model. If the ADF test rejects $H_0 : \delta = 0$, then the ADF test can be stopped. Otherwise, the ADF test continues until the first model is tested.

If the shadowed vehicle trajectory $L' = \left\{p'_j \middle| j = 1,2,3,\cdots,N'\right\}$ is non-stationary, then it needs to be statically processed. In this paper, $L' = \left\{p'_j \middle| j = 1,2,3,\cdots,N'\right\}$ was statically processed by using one-order difference, which makes it stationary. The expression of one-order difference is as follows:

$$\Delta X_t = X_{t+1} - X_t \tag{7}$$

Correspondingly, the shadowed vehicle trajectory $L' = \left\{p'_j \middle| j = 1,2,3,\cdots,N'\right\}$ would be transformed into a one-order difference vehicle trajectory $L'' = \left\{\Delta p'_k \middle| k = 1,2,3,\cdots,N'-1\right\}$, where $\Delta p'_k$ denotes the k-th one-order difference trajectory point of the one-order difference vehicle trajectory and $\Delta p'_k = (\Delta lon'_k, \Delta lat'_k, \Delta t'_k)$. $\Delta lon'_k$ denotes the one-order difference between the longitude of p'_{k+1} and the longitude of p'_k, and $\Delta lon'_k = lon'_{k+1} - lon'_k$. $\Delta lat'_k$ denotes the one-order difference between the latitude of p'_{k+1} and the latitude of p'_k, and $\Delta lat'_k = lat'_{k+1} - lat'_k$. $\Delta t'_k$ denotes the one-order difference between the time of p'_{k+1} and the time of p'_k, and $\Delta t'_k = t'_{k+1} - t'_k$.

3.1.3. Semantic Feature Mapping Process

After the spatiotemporal feature shadowing process and the feature staticizing process, the one-order difference vehicle trajectory $L'' = \left\{\Delta p'_k \middle| k = 1,2,3,\cdots,N'-1\right\}$ cannot be directly imported into a location prediction model. It is necessary to reconstruct $L'' = \left\{\Delta p'_k \middle| k = 1,2,3,\cdots,N'-1\right\}$ semantically to meet the location prediction model. During time series modeling, it is usually necessary to use the last time step as an input variable and use the next time step as an output variable. Therefore, to predict the location of the next time, we proposed the semantic feature mapping process of $L'' = \left\{\Delta p'_k \middle| k = 1,2,3,\cdots,N'-1\right\}$, as shown in Figure 2.

From Figure 2, the prediction of trajectory points of $L' = \left\{p'_j \middle| j = 1,2,3,\cdots,N'\right\}$ is realized through the prediction of one-order difference trajectory points of $L'' = \left\{\Delta p'_k \middle| k = 1,2,3,\cdots,N'-1\right\}$. For example, for an input $\Delta p'_1 = p'_2 - p'_1$, there is an output $\Delta p'_2 = p'_3 - p'_2$ that is predicted, so $p'_3 = \Delta p'_2 - p'_2$. In other words, p'_3 is predicted by using p'_1 and p'_2.

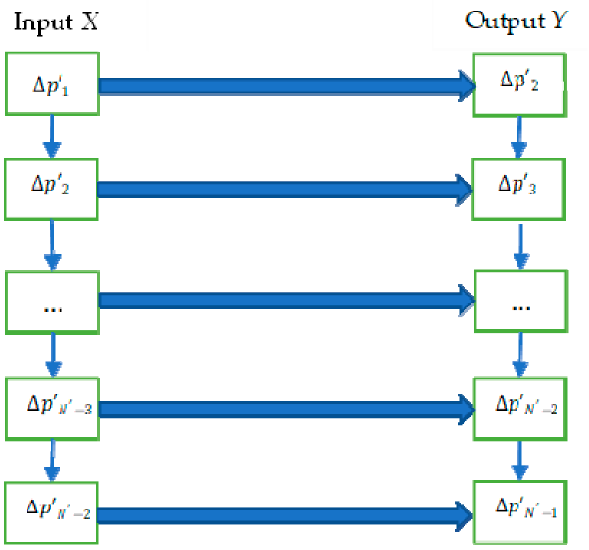

Figure 2. The semantic feature mapping process.

3.2. Hybrid LSTM Neural Network

3.2.1. LSTM Layer

The LSTM layer consists of a series of LSTM units, called LSTM model [31]. Compared with other neural networks, the LSTM model has three multiplicative units, i.e., input gate, output gate, and forget gate. The input gate is used to memorize some information of the present, the output gate is used for output, and the forget gate is used to choose to forget some information of the past. The multiplicative unit is mainly composed of a sigmoid function and a dot product operation, where the sigmoid function has a range of [0, 1] and the dot product operation can determine how much information is transmitted. If the value of the dot product operation is 0, then no information is transmitted. All information is transmitted if the value of the dot product operation is 1. In the forget gate, all the information from the previous hidden state and the current input is passed to the sigmoid function, which has a range of [0, 1]. Close to 0 means discard, and close to 1 means keep. The LSTM unit [31] is shown in Figure 3.

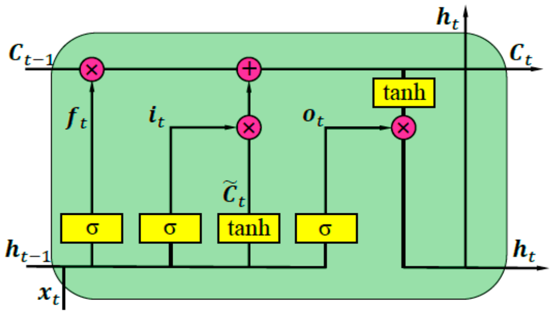

Figure 3. The long short-term memory (LSTM) unit.

In Figure 3, f_t denotes the forget gate, i_t denotes the input gate, and o_t denotes the output gate, which together control the cellular state C_t that stores historical and future information. x_t and h_t respectively represent input and output information of the LSTM unit. The processing expressions of the LSTM unit [31] are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{9}$$

$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C) \tag{10}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{C}}_t \tag{11}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{12}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \tag{13}$$

Equation (8) is a forget gate expression, which takes the weighted sum of cell state at time t, the cell output at time $t-1$, and the input at time t as the input of activation function to get the output of the forget gate. The activation function is generally a logistic sigmoid function. Equation (9) is an input gate expression, where the parameters and principles are the same as those of Equation (8). Equation (10) calculates the candidate value of memory unit at the time t, where the activation function can be a tanh function or a logistic sigmoid function. Equation (11) combines past and present memories. Equation (12) is an output gate expression, where the parameters and principles are identical with those of Equation (8). Equation (13) is an expression for cell output, where the activation function is generally a tanh function. $\boldsymbol{W}$ is a weight vector matrix, and $\boldsymbol{b}$ is a bias vector.

3.2.2. Activation Layer

The Activation layer is mainly used to improve the fitting ability of the model. In Figure 1, each combination layer contains an Activation layer. The activation layer of the output combination layer can use specific activation functions, such as logistic sigmoid function for binary classification and SoftMax function for multiple classification [41]. There are many activation functions that can be used to the Activation layers of the LSTM and Dense combination layers, such as tanh function, sigmoid function, Rectified Linear Unit (ReLU) function, linear function, hard-sigmoid function, softsign function, and softplus function [41–43].

The tanh function is completely differentiable [41]. Its function image is anti-symmetric and its symmetric center is at the origin, so its output is a zero center. The mathematical expression of the tanh function [41] is as follows:

$$f(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{14}$$

However, like many traditional activation functions of neural networks, the tanh function has the vanishing gradient problem. Once the value of the tanh function reaches saturation area, the gradient begins to vanish, and the learning speed becomes slower. To alleviate these problems, researchers proposed a variant of the tanh function, called softsign function [42]. The softsign function is anti-symmetric, decentralized and fully differentiable, and returns a value between -1 and 1. The difference is that the slope of the softsign function is smaller than that of the tanh function in the activation area, and the softsign function enters the saturation area later than the tanh function. The mathematical expression of the softsign function [42] is as follows:

$$f(x) = \frac{x}{1+|x|} \tag{15}$$

The linear function is a simple and typical activation function for regression [43], and the mathematical expression of the linear function [43] is as follows:

$$f(x) = x \tag{16}$$

3.2.3. Dense Layer

The Dense layer mainly classifies feature vectors and maps the samples from the feature space to the labels [44]. It consists of two parts: the linear part and the nonlinear part. The linear part mainly performs linear transformation and analyzes the input data. The calculation method of this part is linear weighted sum, whose mathematical expression [44] is as follows:

$$\boldsymbol{Z} = \boldsymbol{W} * \boldsymbol{X} + \boldsymbol{b} \tag{17}$$

where $\boldsymbol{Z}$ is the output vector of the linear part. It can be expressed as $\boldsymbol{Z} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_m]^T$. $\boldsymbol{X}$ represents the input vector of the linear part, which is expressed as $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n]^T$. $\boldsymbol{W}$ is a weight vector matrix of $m \times n$. $\boldsymbol{b}$ is a bias vector, which is expressed as $\boldsymbol{b} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_m]^T$.

The nonlinear part performs nonlinear transformation, namely corresponding function transformation. This operation has two functions as follows. (1) Data normalization. That is to say, no matter what the previous linear part does, all the values of the nonlinear part will be limited to a certain range. (2) Breaking the linear mapping relation before. In other words, if the Dense layer has no non-linear part, it is meaningless to add multiple neural networks in the model.

4. Experiments and Results Analysis

4.1. Experimental Environment and Dataset

The experimental environment in this paper is as follows: Intel (R) Core (TM) i7-4790 3.6 GHz dual-core processor with 8 GB memory. The operating system is 64 bit Windows 10, the programming language is Python, and the deep learning framework is Keras. The experimental dataset is the taxi trajectory data from August 3rd to 30th in Chengdu City, Sichuan Province, China. The dataset contains more than 1.4 billion trajectories of 14,000 taxis. Each taxi trajectory is from 06:00:00 to 23:59:59 every day and has five fields, which are respectively Taxi ID, Latitude, Longitude, Passenger Carrying Status (1 for carrying passengers, 0 for no passengers), and Time. The taxi trajectory of Taxi ID 4 on 8 August 2014 is shown in Table 1.

Table 1. The taxi trajectory of Taxi ID 4.

Taxi ID	Latitude	Longitude	Passenger Carrying Status	Time
4	30.642221	103.985691	1	2014/8/8 6:00:06
4	30.642635	103.986682	1	2014/8/8 6:00:15
...	...	...	...	...
4	30.656147	104.060689	1	2014/8/8 23:59:49
4	30.655033	104.060724	1	2014/8/8 23:59:59

In this paper, we selected the taxi trajectory of Taxi ID 4 on 8 August 2014 to train and test the above vehicle location predication algorithm.

4.2. Experimental Performance Evaluation

The goal of this paper is to predict the next trajectory point of a taxi, i.e., the specific location of the taxi at the next time. Therefore, the most effective evaluation method for the above vehicle location predication algorithm is to calculate the relative distance between the shadowed trajectory point and the predicted trajectory point of the taxi. The relative distance between two trajectory points is calculated by using the Haversine method [45], and the Haversine formula is as follows:

$$D = 2R * \arcsin(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})}) \tag{18}$$

where D denotes the relative distance between two trajectory points, R denotes the earth radius, ϕ_1 and ϕ_2 denote the latitudes of the two trajectory points respectively, and λ_1 and λ_2 denote the longitudes of the two trajectory points, respectively.

For the i-th predicted trajectory point, if D_i denotes the relative distance between the i-th shadowed trajectory point and the i-th predicted trajectory point, then the average distance for all predicted trajectory points is as follows.

$$D_{aver} = \frac{1}{n}\sum_{i=1}^{n} D_i \tag{19}$$

where D_{aver} denotes the average distance for all predicted trajectory points, n denotes the number of all predicted trajectory points. The average distance for all predicted trajectory points can be used to evaluate the above vehicle location predication algorithm.

Additionally, a distance percentage for all predicted trajectory points was proposed to evaluate the above vehicle location predication algorithm, its expression is as follows:

$$P = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{20}$$

where P denotes the distance percentage for all predicted trajectory points, $a_i = 1$ if $D_i \le d$, $a_i = 0$ if $D_i > d$, and d denotes a specified relative distance. For all predicted trajectory points, the smaller the average distance and the higher the distance percentage, the better the prediction performance of the above vehicle location predication algorithm.

According to the taxi trajectory of Taxi ID 4 on 8 August 2014, for most $\Delta lon'_k$ and $\Delta lat'_k$, it is not a non-zero value until the fourth decimal place. The relative distance is 11m for each unit of the fourth decimal place in longitude or latitude, therefore d is set to 11m when training the model.

4.3. Experimental Process

4.3.1. Spatiotemporal Feature Transformation Process

For the taxi trajectory of Taxi ID 4 on 8 August 2014, the sampling frequency distribution of different sampling intervals is shown in Figure 4.

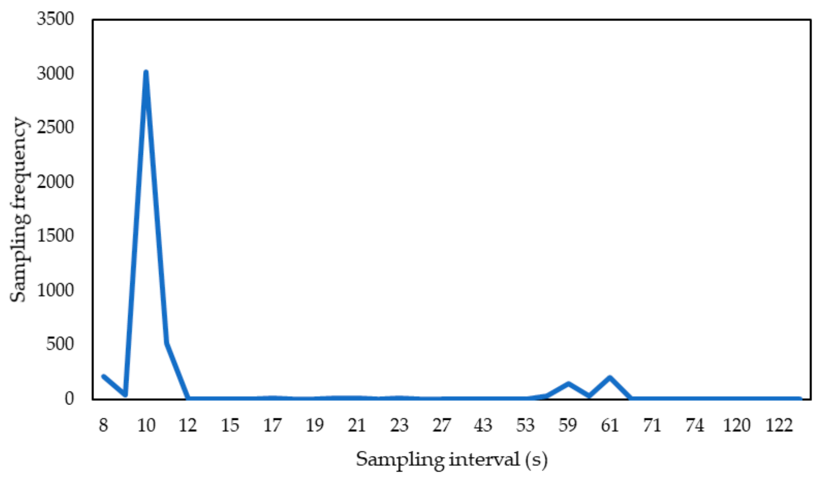

Figure 4. The sampling frequency distribution of different sampling intervals.

According to Figure 4, the sampling frequency of 10 seconds is the highest. Considering that the speed of the taxi is fast, and in order to minimize the trajectory information loss of the taxi, we set the time interval Δt to 10 seconds in this paper. As a result, the number of trajectory points of the shadowed taxi trajectory is 6479 according to Equation (1). The longitudes and latitudes of trajectory points of the shadowed taxi trajectory are calculated by Equations (2) and (3), respectively. It makes that the original taxi trajectory is transformed into the shadowed taxi trajectory.

For the longitudes and latitudes of trajectory points of the shadowed taxi trajectory, their time series graphs are shown in Figures 5 and 6.

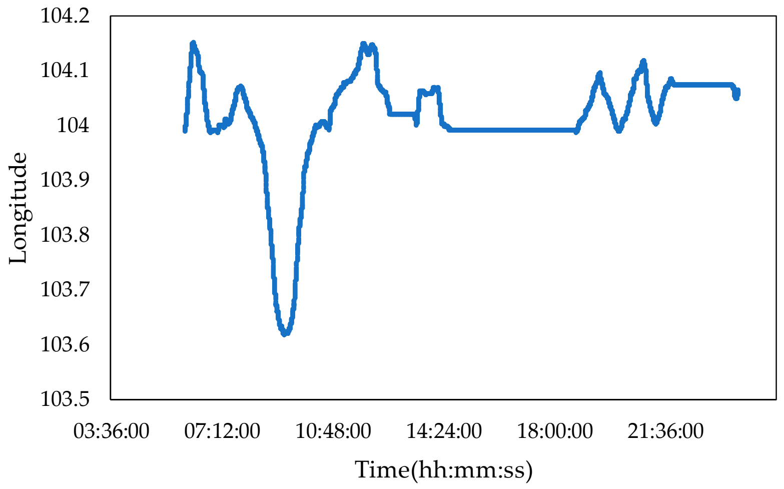

Figure 5. The time series graph of the longitudes of trajectory points of the shadowed taxi trajectory.

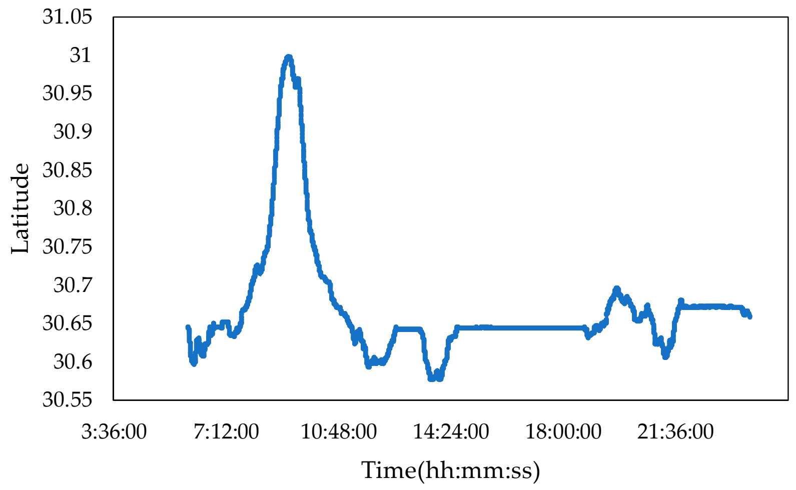

Figure 6. The time series graph of the latitudes of trajectory points of the shadowed taxi trajectory.

From Figures 5 and 6, the longitudes and latitudes of trajectory points of the shadowed taxi trajectory are non-stationary, and we used the ADF test method to further test them. The ADF test results of them are shown in Table 2.

Table 2. The Augment Dickey-Fuller (ADF) test results of the longitudes and latitudes of trajectory points of the shadowed taxi trajectory.

Model	Variable	t-Statistic	*p*-Value	1% Level	5% Level	10% Level
The first model	Longitude	0.133675	0.7247	−2.565319	−1.940873	−1.616667
	Latitude	0.120147	0.7206	−2.565320	−1.940873	−1.616667
The second model	Longitude	−2.575786	0.0981	−3.431182	−2.861792	−2.566946
	Latitude	−2.144854	0.2271	−3.431183	−2.861793	−2.566947
The third model	Longitude	−2.810999	0.1932	−3.959455	−3.410498	−3.127016
	Latitude	−2.357264	0.4022	−3.959456	−3.410499	−3.127016

According to Table 2, the longitudes and latitudes of trajectory points of the shadowed taxi trajectory are non-stationary because their t-Statistic values are greater than the threshold at all levels and their p-values are not very close to 0. To make the shadowed taxi trajectory stationary, we applied one-order difference to it, generating a one-order difference taxi trajectory.

For the one-order difference taxi trajectory, the time series graph of the one-order longitude difference is shown in Figure 7, and the time series graph of the one-order latitude difference is shown in Figure 8.

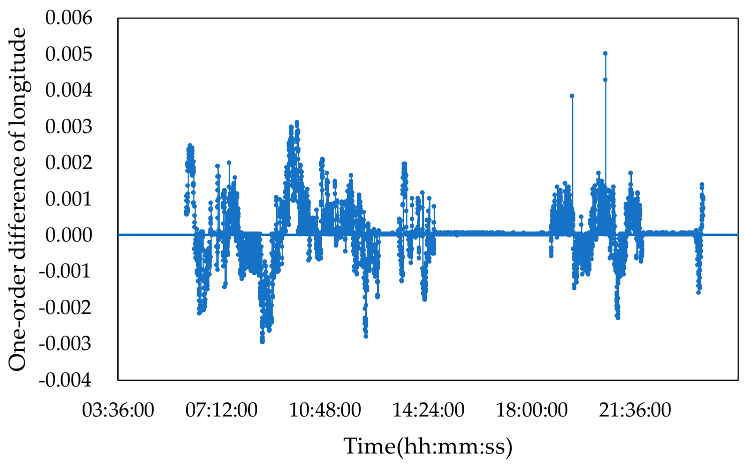

Figure 7. The time series graph of the one-order longitude difference.

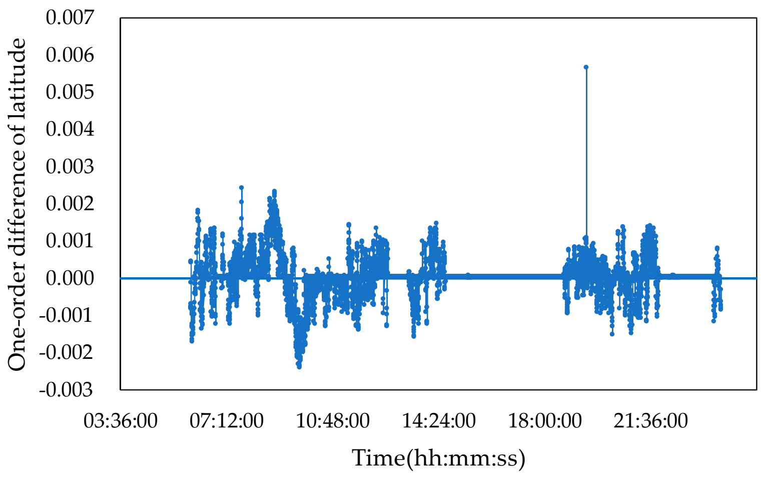

Figure 8. The time series graph of the one-order latitude difference.

From Figures 7 and 8, for the one-order difference taxi trajectory, the one-order longitude difference and the one-order latitude difference are stationary, and we use the ADF test method to further test them. The ADF test results of them are shown in Table 3.

Table 3. The ADF test results of the one-order longitude and latitude differences of the one-order difference taxi trajectory.

Model	Variable	t-Statistic	*p*-Value	1% Level	5% Level	10% Level
The first model	Longitude	−8.562614	0.0000	−2.565319	−1.940873	−1.616667
	Latitude	−8.739988	0.0000	−2.565320	−1.940873	−1.616667
The second model	Longitude	−8.563044	0.0000	−3.431182	−2.861792	−2.566946
	Latitude	−8.740014	0.0000	−3.431183	−2.861793	−2.566947
The third model	Longitude	−8.565636	0.0000	−3.959455	−3.410498	−3.127016
	Latitude	−8.749874	0.0000	−3.959456	−3.410499	−3.127016

According to Table 3, for the one-order difference taxi trajectory, the one-order longitude and latitude differences are stationary because their t-Statistic values are less than the threshold at all levels and their *p*-values are 0.

Finally, to predict the taxi location of the next time, the semantic feature mapping process of the one-order difference taxi trajectory was performed according to Figure 2.

4.3.2. Hybrid LSTM Neural Network Modeling Process

To transform the one-order difference taxi trajectory into supervised learning problems and evaluate the prediction accuracy of the model, we divided the trajectory points of the one-order difference taxi trajectory into a training set and a testing set, where the first 80% (i.e., 5182 trajectory points) of the one-order difference taxi trajectory is the training set and the last 20% (i.e., 1296 trajectory points) of the one-order difference taxi trajectory is the testing set. The hybrid LSTM neural network modeling process is as follows:

(1) Network Structure Tuning

When tuning the network structure in Keras, the hybrid LSTM neural network structure shown in Figure 1 selects one or more LSTM combination layers, zero or more Dense combination layers, and one output combination layer, where each LSTM combination layer only includes an LSTM layer, each Dense combination layer only includes a Dense layer, and the output layer only includes a Dense layer. The parameters of the LSTM layer are set to the default values of Keras [46], and the parameters of the Dense layer are set to the default values of Keras [46]. Additionally, the batch_size is set to 1, the number of neurons is set to 5, the time of epoch is set to 300, the optimizer is set to Adam and its learning rate is set to 0.0008. The number of neurons, the time of epoch, the optimizer and its learning rate are further tuned in the subsequent experiments. Figure 9 is the distance percentage of different LSTM layer number, and Figure 10 is the distance percentage of different Dense layer number.

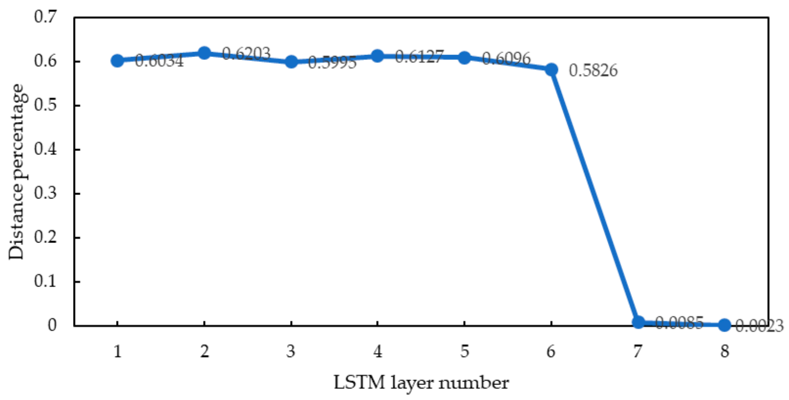

Figure 9. The distance percentage of different LSTM layer number.

From Figures 9 and 10, we can see that the distance percentage is the maximum (i.e., 0.6203) when the number of LSTM layers is 2, and we can see that the distance percentage is the maximum (i.e., 0.6211) when the number of Dense layers is 5. Therefore, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers and one output combination layer in the subsequent experiments.

(2) Neurons Number Tuning

The number of neurons in the neural network has a certain impact on the output of the model, and an appropriate number of neurons can improve the prediction accuracy of the model. When tuning the number of neurons in Keras, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers, and one output combination layer, where each LSTM combination layer only includes an LSTM layer, each Dense combination layer only includes a Dense layer, and the output layer only includes a Dense layer. The above parameter values except the number of neurons are applied here. Figure 11 is the distance percentage of different neurons number.

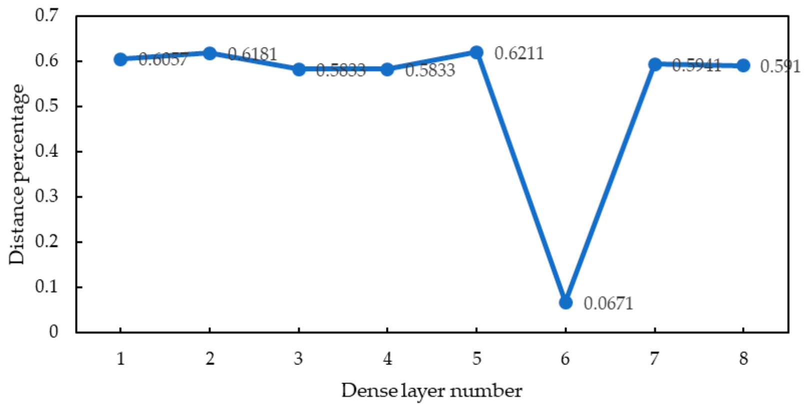

Figure 10. The distance percentage of different Dense layer number.

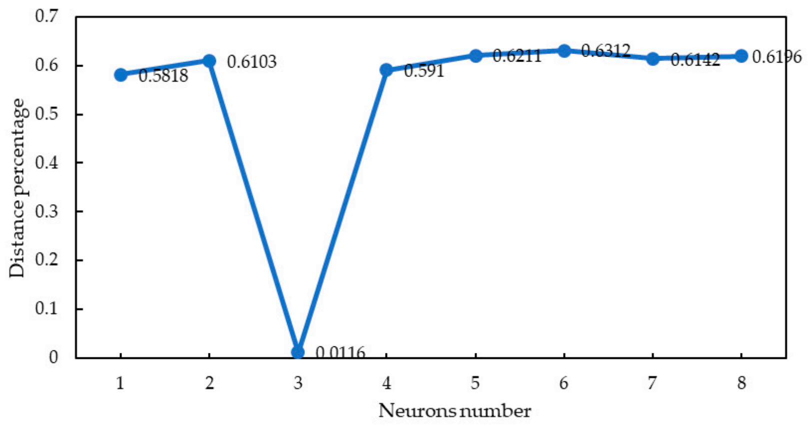

Figure 11. The distance percentage of different neurons number.

From Figure 11, when the number of neurons is 6, the distance percentage is the maximum (i.e., 0.6312). Therefore, the number of neurons is set to 6 in the subsequent experiments.

(3) Optimizer Tuning

In deep learning, in order to minimize the given objective function, the descending gradient can be optimized. Different optimizers can be used to further optimize the target problem. The commonly used optimizers include Adam, RMSProp, Adagrad, Adamax, Nadam, Adadelta, Stochastic Gradient Descent (SGD), etc. [46]. When tuning the optimizers in Keras, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers, and one output combination layer, where each LSTM combination layer only includes an LSTM layer, each Dense combination layer only includes a Dense layer, and the output layer only includes a Dense layer. The above parameter values except the optimizers and their learning rates are applied here. Figure 12 is the distance percentage of different optimizers and learning rates.

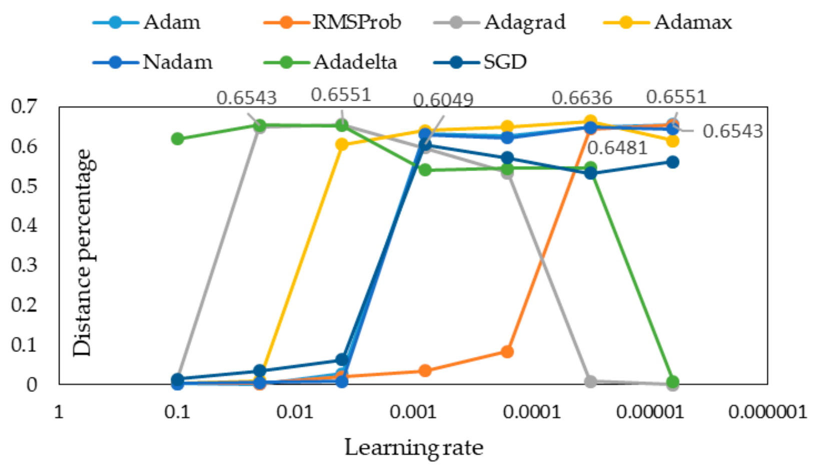

Figure 12. The distance percentage of different optimizers and learning rates.

According to Figure 12, when the optimizer is Adamax and its learning rate is 0.000032, the distance percentage is the maximum (i.e., 0.6636). Therefore, the optimizer is set to Adamax and its learning rate is set to 0.000032 in the subsequent experiments.

(4) Activation Function Tuning

When tuning the activation functions in Keras, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers, and one output combination layer. In order to further improve the prediction accuracy of the model, we added an Activation layer to each of the above combination layers, i.e., each LSTM combination layer includes an LSTM layer and an Activation layer, each Dense combination layer includes a Dense layer and an Activation layer, and the output layer includes a Dense layer and an Activation layer. The activation function of the Activation layer of each LSTM combination layer is the same as that of the Activation layer of each Dense combination layer, called intermediate activation function. The activation function of the Activation layer of the output combination layer is called output activation function. When tuning the intermediate activation function in Keras, the output activation function is set to the linear function shown as Equation (16), and the above parameter values are applied here. Table 4 is the distance percentage of different intermediate activation functions.

Table 4. The distance percentage of different intermediate activation functions.

Intermediate Activation Function	Distance Percentage
Relu	0.6636
Sigmoid	0.6512
Tanh	0.6597
Softplus	0.6597
Softsign	0.6644
Hard_sigmoid	0.6497

From Table 4, when the intermediate activation function is Softsign, the distance percentage is the maximum (i.e., 0.6644). Therefore, when tuning the output activation function in Keras, the intermediate activation function is set to Softsign, and the above parameter values are applied here. Table 5 is the distance percentage of different output activation functions.

Table 5. The distance percentage of different output activation functions.

Output Activation Function	Distance Percentage
Sigmoid	0.00001
Tanh	0.6528
Softplus	0.00001
Linear	0.6651
Hard_sigmoid	0.00001

From Table 5, when the output activation function is the linear function shown as Equation (16), the distance percentage is the maximum (i.e., 0.6651). Therefore, the output activation function is set to the linear function shown as Equation (16) in the subsequent experiments.

(5) Dropout Tuning

When tuning the dropout rate in Keras, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers and one output combination layer. An appropriate dropout rate can sometimes improve the predication accuracy of the model, therefore we added a Dropout layer to each of the combination layers, i.e., each LSTM combination layer includes an LSTM layer, an Activation layer, and a Dropout layer, each Dense combination layer includes a Dense layer, an Activation layer, and a Dropout layer, and the output layer includes a Dense layer and an Activation layer. The intermediate activation function is set to Softsign, the output activation function is set to the linear function shown as Equation (16), and the above parameter values are applied here. Figure 13 is the distance percentage of different dropout rates.

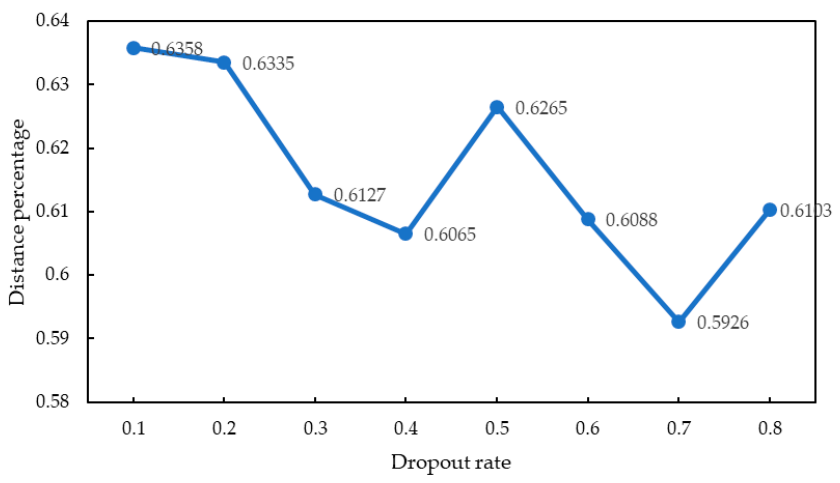

Figure 13. The distance percentage of different dropout rates.

According to Figure 13, when the dropout rate is 0.1, the distance percentage is the maximum (i.e., 0.6358), less than 0.6651 as shown in Figure 14. Hence, the dropout rate in each Dropout layer should be set 0, i.e., no neural units need to be discarded for the one-order difference taxi trajectory.

(6) Epoch Time Tuning

In Figure 12, when the optimizer is Adam and its learning rate is 0.0000064, the distance percentage is raised to 0.6551. When the optimizer is RMSprop and its learning rate is 0.0000064, the distance percentage is raised to 0.6543. When the optimizer is Adagrad and its learning rate is 0.004, the distance percentage is raised to 0.6551. When the optimizer is Adamax and its learning rate is 0.000032, the distance percentage is raised to 0.6636. When the optimizer is Nadam and its learning rate is 0.000032, the distance percentage is raised to 0.6481. When the optimizer is Adadelta and its learning rate is 0.02, the distance percentage is raised to 0.6543. When the optimizer is SGD and its learning rate is 0.0008, the distance percentage is raised to 0.6049.

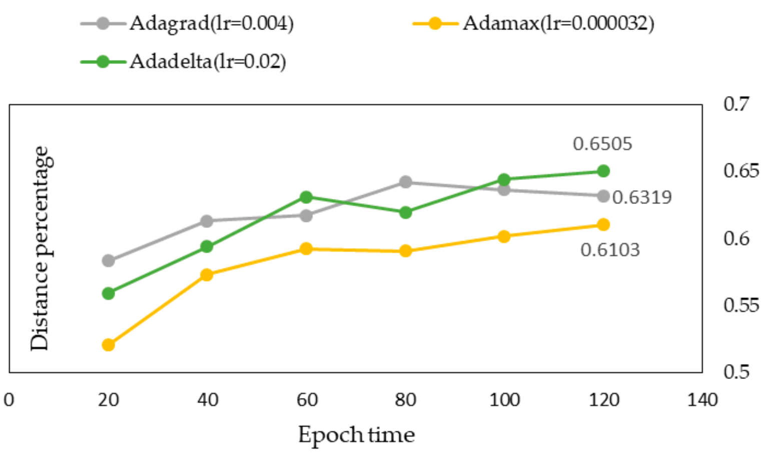

Figure 14. The distance percentage of different epoch times.

Generally, when the learning rate is low, the learning speed will be slow. Therefore, for the Adagrad with a learning rate of 0.004, the Adamax with a learning rate of 0.000032, and the Adadelta with a learning rate of 0.02, their distance percentages are raised to a higher value more quickly. In order to improve the training performance of the model, we further tuned the epoch time for them. When tuning the epoch time in Keras, the hybrid LSTM neural network structure shown in Figure 1 only selects two LSTM combination layers, four Dense combination layers, and one output combination layer, where each LSTM combination layer includes an LSTM layer, an Activation layer and a Dropout layer, each Dense combination layer includes a Dense layer, an Activation layer, and a Dropout layer, and the output layer includes a Dense layer and an Activation layer. The intermediate activation function is set to Softsign, the output activation function is set to the linear function shown as Equation (16), and the above parameter values except the epoch time are applied here. Figure 14 is the distance percentage of different epoch times.

According to Figure 14, when the epoch time is 120, the distance percentage for the Adadelta with a learning rate of 0.02 is higher than that for the Adagrad with a learning rate of 0.004 and that for the Adamax with a learning rate of 0.000032, and is raised to 0.6505. This is very close to the maximum distance percentage 0.6636 above, while the epoch time is reduced from 300 to 120. Therefore, the optimizer is set to Adadelta and its learning rate is set to 0.02, and the epoch time is set to 120 finally.

4.4. Experimental Results and Analysis

According to the above experimental process, for the one-order difference taxi trajectory, the optimal hybrid LSTM neural network structure is composed of two LSTM combination layers, four Dense combination layers and an output combination layer, where each LSTM combination layer includes an LSTM layer, an Activation layer, and a Dropout layer, each Dense combination layer includes a Dense layer, an Activation layer, and a Dropout layer, and the output layer includes a Dense layer and an Activation layer, the batch_size value is 1, the epoch time is 120, the neurons number is 6, the optimizer is Adadelta and its learning rate is 0.02, the intermediate activation function is Softsign, the output activation function is linear, the dropout rate in each Dropout layer is 0, and the other parameters are the default values in Keras.

Our experimental environment is based on a common CPU computer, and the CPU time for the above model is 480 seconds. Generally, a high-performance GPU computer runs tens to hundreds of times faster than a common CPU computer. Hence, the above model can be executed on a high-performance GPU computer or a GPU cluster to meet the actual application requirements of the model. Through the above experimental process, the final distance percentage is raised to 0.6505. The original, shadowed, and predicted trajectories of the taxi for the testing set are shown in Figure 15.

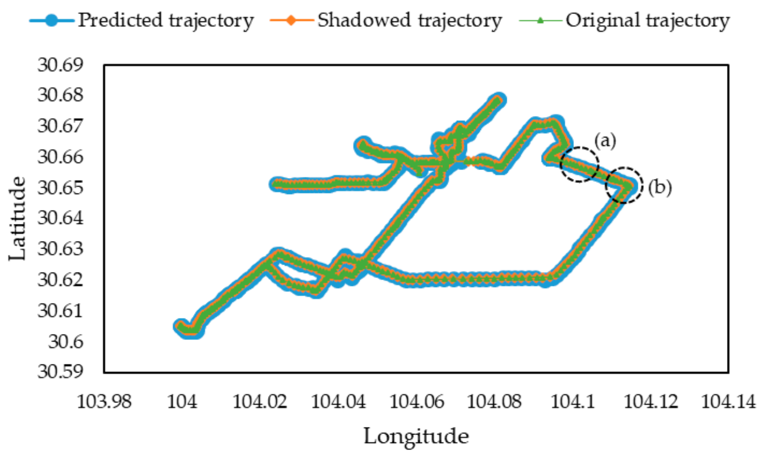

Figure 15. The original, shadowed, and predicted trajectories of the taxi for the testing set.

From Figure 15, the original, shadowed, and predicted trajectories of the taxi for the testing set are very similar, where five-minute taxi trajectories without a turning part are marked by (a) and five-minute taxi trajectories with a sharp turning part are marked by (b). Figure 16 is the original, shadowed, and predicted trajectories of the taxi for the marked (a), and Figure 17 is the original, shadowed, and predicted trajectories of the taxi for the marked (b).

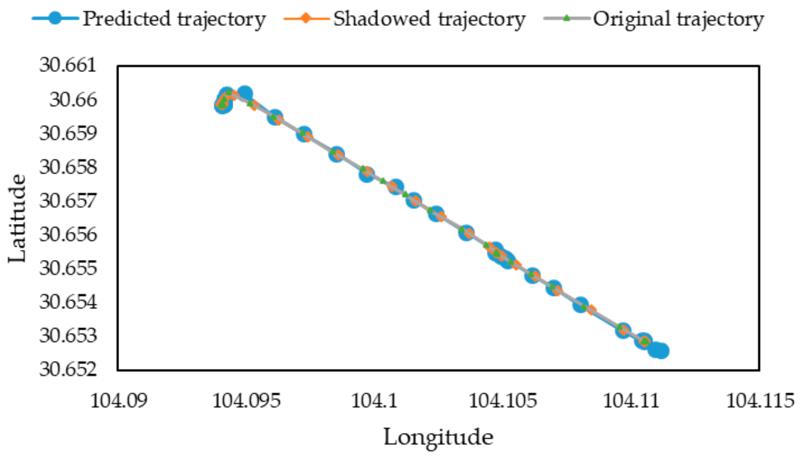

Figure 16. The original, shadowed, and predicted trajectories of the taxi for the marked (a).

According to Figures 16 and 17, the original and shadowed trajectories of the taxi are very close, regardless of a non-turning part or a turning part. It means that the trajectory information of the original taxi trajectory is basically reserved by the shadowed taxi trajectory. For the non-turning part, the trajectory points of the predicted taxi trajectory are close to those of the shadowed taxi trajectory. The trajectory points of the predicted taxi trajectory are more off those of the shadowed taxi trajectory after the turning part. The reason is that the driving direction of the taxi needs to be corrected by several time steps during location predication process.

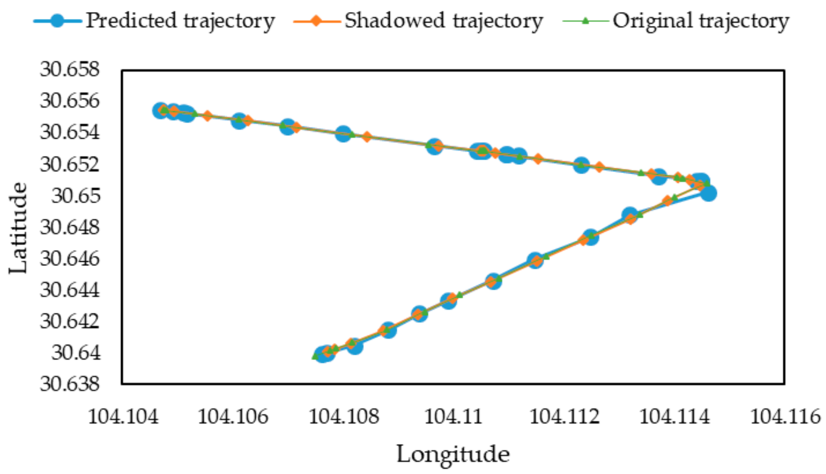

Figure 17. The original, shadowed, and predicted trajectories of the taxi for the marked (b).

Additionally, for the testing set, the distance percentage and average distance of the random forest (RF) regression, the K nearest neighbor (KNN) regression, the multi-layer perceptron (MLP), the RNN, and our proposed algorithm were further compared, as shown in Table 6. When training the RF regression, n_estimators = 100, bootstrap = true, and the other parameters are the default values. When training the KNN regression, n_neighbors = 5, weights = uniform, algorithm = auto, leaf_size = 30, and the other parameters are the default values. When training the MLP, the network structure includes five Dense layers and one output layer, where the batch_size value is 1, the epoch time is 120, the number of neurons is 6, the optimizer is Adadelta and its learning rate is 0.02, the activation function is Softsign, and the other parameters are the default values. When training the RNN, the network structure includes two SimpleRNN layers, five Dense layers, and one output layer, where the batch_size value is 1, the epoch time is 120, the number of neurons is 6, the optimizer is Adadelta and its learning rate is 0.02, the activation function is Softsign, and the other parameters are the default values.

Table 6. The distance percentage and average distance of different predication algorithms for the testing set.

Algorithm	**Distance Percentage**				**Average Distance (m)**
	***d* = 11 m**	***d* = 20 m**	***d* = 30 m**	***d* = 40 m**	
RF Regression	0.6065	0.6968	0.7662	0.8465	17.48
KNN regression	0.5949	0.7045	0.7940	0.8565	16.87
MLP	0.6219	0.7068	0.7878	0.8603	20.30
RNN	0.5918	0.6906	0.7785	0.8488	18.50
Our proposed algorithm	0.6505	0.7392	0.8164	0.8843	14.96

From Table 6, we can see that our proposed algorithm is obviously better than the other prediction algorithms in the accuracy of location predication.

In Table 6, feature staticizing was used in these prediction algorithms. If feature staticizing was not used in our proposed algorithm, then the distance percentage and the average distance of the algorithm for the testing set is shown in Table 7.

Table 7. The distance percentage and average distance of our proposed algorithm with no feature staticizing for the testing set.

Algorithm	**Distance Percentage**				**Average Distance (m)**
	***d* = 11 m**	***d* = 20 m**	***d* = 30 m**	***d* = 40 m**	
Our proposed algorithm with no feature staticizing	0.0278	0.1019	0.7330	0.8040	32.02

From Tables 6 and 7, we can see that our proposed algorithm with feature staticizing is obviously better than our proposed algorithm with no feature staticizing in the accuracy of location predication. Hence, feature staticizing is effective in our proposed algorithm.

5. Conclusions

The existing location predication methods have large trajectory information loss and low prediction accuracy, so they are unsuitable for vehicle location predication of an intelligent transportation system. To solve the problem, we proposed a vehicle location prediction algorithm based on spatiotemporal feature transformation and hybrid LSTM neural network in this paper. In this algorithm, a spatiotemporal feature transformation method is used to convert a vehicle trajectory into an appropriate input of a hybrid LSTM neural network model, and then the vehicle location at the next time is predicted by the model. The spatiotemporal feature transformation method includes a spatiotemporal feature shadowing process, a feature staticizing process, and a semantic feature mapping process. The spatiotemporal feature shadowing process makes the trajectory points of the vehicle trajectory even on the time dimension, increasing the certainty of the model and reducing the information loss of the vehicle trajectory. The feature staticizing process is used to test whether the vehicle trajectory is stationary, and then makes the vehicle trajectory stationary. The semantic feature mapping process is used to reconstruct the vehicle trajectory semantically to meet the location prediction of the model. Therefore, the spatiotemporal feature transformation method can reduce the information loss of the vehicle trajectory and improve the predication accuracy of the model. The hybrid LSTM neural network model consists of one or more LSTM combination layers, zero or more Dense combination layers, and one output combination layer. The LSTM combination layer includes an LSTM layer, an Activation layer, and a Dropout layer. The Dense combination layer includes a Dense layer, and an Activation layer and a Dropout layer. The output combination layer includes a Dense layer and an Activation layer. The hybrid LSTM neural network structure and its parameters can be optimized for the vehicle trajectory, further improving the prediction accuracy of the model.

After the experiments of our proposed algorithm and the other prediction models for a taxi trajectory, the experimental results show that the trajectory information of the original taxi trajectory is basically reserved by the shadowed taxi trajectory, and the trajectory points of the predicted taxi trajectory are close to those of the shadowed taxi trajectory. Hence, our proposed algorithm effectively reduces the information loss of vehicle trajectory and improves the accuracy of vehicle location prediction. Additionally, the experimental results also show that our proposed algorithm has a higher distance percentage and a shorter average distance than the other predication models. Therefore, our proposed algorithm is better than the other prediction models in the accuracy of vehicle location predication. In this study, the influence of the actual road network on the algorithm is not considered. We will further validate and improve our proposed algorithm to meet the demand for vehicle location prediction in the actual road network.

Author Contributions: Methodology, Y.X.; software, Q.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (61741216, 61402367), the Shaanxi Science and Technology Co-ordination and Innovation Project (2016KTTSGY01-03), the Special Scientific Research Project of Education Department of Shaanxi Province (17JK0704) and the New Star Team Project of Xi'an University of Posts and Telecommunications.

Acknowledgments: The authors would like to thank the anonymous reviewers for their contribution to this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kolodziej, K.W.; Hjelm, J. *Local Positioning Systems: LBS Applications and Services*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2006; pp. 101–158.
2. Li, W.; Zhao, X.; Sun, C. Prediction of trajectory based on modified Bayesian inference. *J. Comput. Appl.* **2013**, *33*, 1960–1963. [CrossRef]
3. Bao, J.; Zheng, Y.; Mokbel, M. Location-based and preference-aware recommendation using sparse geo-social networking data. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, New York, NY, USA, 6–9 November 2012; pp. 199–208.
4. Bao, J.; He, T.; Ruan, S.; Li, Y.; Zheng, Y. Planning bike lanes based on sharing-bikes' trajectories. In Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Halifax, NS, Canada, 13–17 August 2017; pp. 797–806.
5. Xiao, Y.; Yin, Y. Hybrid LSTM neural network for short-term traffic flow prediction. *Information* **2019**, *10*, 105. [CrossRef]
6. Gambs, S.; Killijian, M.O.; Cortez, D.P.; Miguel, N. Next place prediction using mobility Markov chains. In Proceedings of the First Workshop on Measurement, Privacy, and Mobility, Bern, Switzerland, 10 April 2012; pp. 1–6.
7. Chen, Z.; Wen, J.; Geng, Y. Predicting future traffic using hidden Markov models. In Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 8–11 November 2016; pp. 1–6.
8. Mathew, W.; Raposo, R.; Martins, B. Predicting future locations with hidden Markov models. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; pp. 911–918.
9. Li, S.; Qiao, J.; Lin, S. Hybrid multi-step Markov location prediction based on GPS trajectory data. *J. Northeast. Univ. (Nat. Sci.)* **2017**, *38*, 1686–1690.
10. Karatzoglou, A.; Köhler, D.; Beigl, M. 13. Semantic-enhanced multi-dimensional Markov chains on semantic trajectories for predicting future locations. *Sensors* **2018**, *18*, 3582. [CrossRef]
11. Yang, J.; Xu, J.; Xu, M.; Zheng, N.; Chen, Y. Predicting next location using a variable order Markov model. In Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming, Dallas, TX, USA, 4 November 2014; pp. 37–42.
12. Lin, S.; Li, S.; Qiao, J.; Yang, D. Markov location prediction based on user mobile behavior similarity clustering. *J. Northeast. Univ. (Nat. Sci.)* **2016**, *37*, 323–326.
13. Song, L.; Meng, F.; Yuan, G. Moving object location prediction algorithm based on Markov model and trajectory similarity. *J. Comput. Appl.* **2016**, *36*, 39–43.
14. Li, W.; Xia, S.; Feng, L.; Zhang, L.; Yuan, G. Location prediction algorithm based on movement tendency. *J. Commun.* **2014**, *2*, 46–53.
15. Yang, Z.; Wang, H. Location prediction method of mobile user based on Adaboost-Markov model. *J. Comput. Appl.* **2019**, *39*, 675–680.
16. Wang, H.; Yang, Z. Next location prediction based on an Adaboost-Markov model of mobile users. *Sensors* **2019**, *19*, 1475. [CrossRef]
17. Chen, C.; Gong, H.; Lawson, C.; Bialostozky, E. Evaluating the feasibility of a passive travel survey collection in a complex urban environment: Lessons learned from the New York City case study. *Transp. Res. Part A Policy Pract.* **2010**, *44*, 830–840. [CrossRef]
18. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]

19. Zheng, V.; Zheng, Y.; Xie, X.; Yang, Q. Collaborative location and activity recommendations with GPS history data. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 1029–1038.
20. Morzy, M. Mining frequent trajectories of moving objects for location prediction. In Proceedings of the 5th International Workshop on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany, 18–20 July 2007; pp. 667–680.
21. Morzy, M. Prediction of moving object location based on frequent trajectories. In Proceedings of the 21th International Symposium on Computer and Information Sciences, Istanbul, Turkey, 1–3 November 2006; pp. 583–592.
22. Deng, J.; Wang, Y.; Dong, Z. Dynamic trajectory pattern mining facing location prediction. *Appl. Res. Comput.* **2017**, *34*, 2984–2988.
23. Jeung, H.; Liu, Q.; Shen, H.T.; Zhou, X. A hybrid prediction model for moving objects. In Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE 2008), Cancun, Mexico, 7–12 April 2008; pp. 70–79.
24. Yavaş, G.; Katsaros, D.; Ulusoy, Ö.; Manolopoulos, Y. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng.* **2005**, *54*, 121–146. [CrossRef]
25. Li, Z.; Ding, B.; Han, J.; Kays, R.; Nye, P. Mining periodic behaviors for moving objects. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 25–28 July 2010; pp. 1099–1108.
26. Li, Z.; Han, J.; Ji, M.; Tang, L.A.; Yu, Y.; Ding, B.; Lee, J.G.; Kays, R. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 37. [CrossRef]
27. Liu, Q.; Wu, S.; Wang, L.; Tan, T. Predicting the next location: A recurrent model with spatial and temporal contexts. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 194–200.
28. Lipton, Z.C. A Critical Review of Recurrent Neural Networks for Sequence Learning. Available online: https://arxiv.org/pdf/1506.00019v2.pdf (accessed on 11 December 2019).
29. Al-Molegi, A.; Jabreel, M.; Ghaleb, B. STF-RNN: Space-time features-based recurrent neural network for predicting people's next location. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence, Computational Intelligence and Data Mining (CIDM 2016), Athens, Greece, 6–9 December 2016; pp. 1–7.
30. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
31. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
32. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the 2014 Annual Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
33. Palangi, H.; Deng, L.; Shen, Y.L.; Gao, J.; He, X.; Chen, J.; Song, X.; Ward, R. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 694–707. [CrossRef]
34. Visin, F.; Kastner, K.; Cho, K.; Matteucci, M.; Courville, A.; Bengio, Y. ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks. Available online: https://arxiv.org/pdf/1505.00393v3.pdf (accessed on 11 December 2019).
35. Wu, F.; Fu, K.; Wang, Y.; Xiao, Z.; Fu, X. A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. *Algorithms* **2017**, *10*, 37. [CrossRef]
36. Gao, Y.; Jiang, G.; Qin, X.; Wang, Z. Location prediction algorithm of moving object based on LSTM. *J. Front. Comput. Sci. Technol.* **2019**, *13*, 23–34.
37. Xu, F.; Yang, J.; Liu, H. Location prediction model based on ST-LSTM network. *Comput. Eng.* **2019**, *45*, 1–7.
38. McKinney, W.; Perktold, J.; Seabold, S. Time series analysis in Python with statsmodels. In Proceedings of the 10th Python in Science Conference, Austin, TX, USA, 11–16 July 2011; pp. 96–102.
39. Li, W.; Liu, Y.; Yang, J. Test and empirical analysis of time series stationarity based on R language. *China Bus. Trade* **2017**, *16*, 150–153.
40. Mo, D. Carry out unit root test using SPSS. *J. Hezhou Univ.* **2009**, *25*, 133–135.

41. Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks Book*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 31–38.
42. Xavier, G.; Yoshua, B. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
43. Xavier, G.; Yoshua, B.; Bengio, Y. Deep sparse rectifier neural networks. *J. Mach. Learn. Res.* **2011**, *15*, 315–323.
44. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
45. Li, Z.; Li, J. Quickly calculate the distance between two points and measurement error based on latitude and longitude. *Geomat. Spat. Inf. Technol.* **2013**, *36*, 235–237.
46. Keras Documentation. Available online: https://keras.io/ (accessed on 11 December 2019).