Nei-TTE: Intelligent Traffic Time Estimation Based on Fine-Grained Time Derivation of Road Segments for Smart City

Jing Qiu, Lei Du, Dongwen Zhang, Shen Su, and Zhihong Tian

***Abstract*—With the development of the Internet of Things and big data technology, the intelligent transportation system is becoming the main development direction of future transportation systems. The time required for a given trajectory in a transportation system can be accurately estimated using the trajectory data of the taxis in a city. This is a very challenging task. Although historical data have been used in existing research, excessive use of trajectory information in historical data or inaccurate neighbor trajectory information does not allow for a better prediction accuracy of the query trajectory. In this article, we propose a deep learning method based on neighbors for travel time estimation (TTE), called the Nei-TTE method. We divide the entire trajectory into multiple disjoint segments and use the historical trajectory data approximated at the time level. Our model captures the characteristics of each segment and utilizes the trajectory characteristics of adjacent segments as the road network topology and speed interact. We use velocity features to effectively represent adjacent segment structures. The experiments on the Porto dataset show that the experimental results of our model are significantly better than those of the existing models.**

***Index Terms*—Gated recurrent unit (GRU), intelligent transportation systems, travel time estimation (TTE), trajectories, time series.**

I. Introduction

With the rapid growth in data volume, we have entered the era of big data. The Internet of Things (IoT) and big data technology have provided strong support for the concept of smart cities that has been proposed in recent years [1], [2], [11], [30]. With the increase of the urban population and the variety of travel modes, congestion is more prevalent, making traveling more cumbersome. Therefore, the construction of intelligent transportation system to alleviate traffic pressure is the main development direction of smart cities.

Smart cities generate large amounts of data every day for many applications, such as [7] extracting the free relationship between concepts in smart city data, and with the in-depth research and development of network security technologies [3], [4] and IoT technologies [5], [6], [25], the security of the IoT has increased, and applications based on the IoT technology are also rapidly reaching. GPS and timing equipment are installed in many taxis, using IoT technology to regularly upload driving trajectory data to taxi companies. In this process, the sensor information of the vehicle is collected periodically to obtain GPS and other information, which is converted into formatted data through data analysis and submitted to the cloud platform for storage. The company records taxi trips and publishes these data regularly. This trajectory information can reflect the driving status information of the corresponding area at any time of the day. We can use this information to plan or forecast travel times and provide guidance for travel in smart cities. The question is how to effectively use big data technology to analyze taxi data for efficient transportation planning, forecasting travel times and trends, and perform other tasks for precise decision-making. It is an urgent problem that needs to be solved. In this article, we focus on travel time forecasting. There has been many works [8], [9], [10] in this area, and they have achieved good results. A lot of work uses different methods on travel process and data mining (time-space data and other data). Under the premise of ignoring uncontrolled device sampling errors, research in this field usually considers the following points.

First, the spatial data correspond to the trajectory information of the taxi and consists of a sequence of GPS points. This sequence of points is taken from real roads [12]. Since the characteristics of the road network have a great influence on the traffic state, capturing the topology of the road network has great significance for travel time prediction. In the existing road segment-based methods [14], [15], there is no consideration of the traffic lights and speed changes on intersecting road segments. The actual topology information can reflect these situations well. Since a subpath based method [10], [16] takes into account some of the intersecting road segment information (that is, the topology of the intersection), it achieves better results.

Manuscript received June 19, 2019; revised August 24, 2019 and September 6, 2019; accepted September 18, 2019. Date of publication September 26, 2019; date of current version January 17, 2020. This article was supported in part by the National Key Research and Development Plan under Grant 2018YFB0803504 and Grant 2018YEB1004003, in part by the Guangdong Province Key Research and Development Plan under Grant 2019B010137004, and in part by the National Natural Science Foundation of China under Grant 61871140, Grant 61872100, Grant 61572153, and Grant U1636215. Paper no. TII-19-2578. *(Corresponding author: Zhihong Tian.)*

J. Qiu, S. Su, and Z. Tian are with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: qiujing@gzhu.edu.cn; johnsuhit@gmail.com; tianzhihong@gzhu.edu.cn).

L. Du and D. Zhang are with the Department of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050018, China (e-mail: leidu_close@163.com; zdwwtx@hebust.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2943906

However, due to the deviation of the GPS device sampling, the trajectory of the driving cannot be corresponding to the real road segment, and the real topology information corresponding to the road network cannot be obtained from the trajectory point road segment. Road network topology features are difficult to extract. We use speed features to represent the surrounding road network topology information. For example, the speed of the vehicle will decrease while turning and the speed of the vehicle will come close to zero when waiting for traffic lights. Besides, the topology of adjacent sections has similarities.

Second [15], the traffic conditions are dynamic and, therefore, are highly correlated with the time interval information. The time interval information indicates the start times of the current trajectories or road segments. The different start times affect the arrival time. In the same position, there may be different traffic conditions during a longer time interval, for instance, the time between the work peaks and the time after the peak period may be used as the start time of the current route, which will be very different. People move from one place to another, always habitually review their current position, and ignore the position that has been already traveled. Therefore, in order to make accurate time predictions, it is necessary to use data in a more fine-grained time period. For example, here we use the timestamp for calculating each point as the arrival time.

Finally, the road conditions at the same time of the same road segment are more similar, so historical trajectory data can be used to provide auxiliary information for the current query trajectory [10], but with the increase of the driving distance, the same can be matched. There are fewer and fewer driving trajectories, and the historical trajectory data that can be used by the current trajectory are reduced. This data sparse problem causes the credibility of the prediction result to be reduced. Besides, in order to obtain more accurate prediction results, a more detailed division of time is required, which makes the data sparse problem more serious. Wang *et al.* [13] used the start and end positions of a given query trajectory, by querying the historical trajectories with the same start and end positions as an auxiliary example of the current trajectory, because the traffic conditions are greatly affected by the dynamics of time. Although this simple method has achieved good results, there is still error in not using the latest fine-grained driving data, and only the position information of the start and end points is used. There may be a situation around the distance that causes errors in the results.

To solve the above problem, we propose a neighbor-based deep learning method for estimating the travel time, called the Nei-TTE method. The contributions in this article are mainly as follows.

1) We set the exact start time for each segment of the entire trajectory. The entire trajectory is divided into multiple disjoint road segments, each has a start time. At fine-grained time intervals, the historical data that are similar to the spatial segment of the road segment are extracted. The traffic condition depends greatly on the time factor, thus, to fully represent the temporal feature, we represent the time factor using a detailed expression, which is used to mine the temporal features that are relevant to this task.
2) We characterize the topological information of the area around the road. A novel method is used to effectively represent the topology information of a road segment. The road network topology results are not easy to obtain accurately and the topology of the road segment and its adjacent segments have great similarities, and the speed features are used to represent them. Using the features of adjacent segments as auxiliary information increases the sample size, making real-time traffic data more accurate.
3) We performed experiments on a large-scale real dataset and achieved a *mean absolute percentage error* (MAPE) of 0.070%, which shows that our method is significantly better than the existing method.

The rest of this article is structured as follows: Section II describes the related work, Section III illustrates the problem predefinition, Section IV introduces our model, Section V shows the experimental description, and Section VI summarizes this article.

II. Related Works

There are two main ways to estimate the travel time. First are the segment-based methods, e.g., [15] and [17]. In this part of the work, since the characteristics of the intersections of road segments are not taken into consideration and the intersections of road segments are affected by many factors, such as traffic lights and speed changes, it is difficult to obtain an accurate prediction. Second are the subpath-based methods, such as [10] and [16]. The subpath-based methods take into account some of the intersections of some road segments and have better results than the road segment methods. However, they will introduce data sparsity. The longer the trajectory, the fewer vehicles will be driven. If the query trajectory is not included in the historical trajectory, or there are only a few trajectories, the sparsity of the data will result in reduced reliability of the prediction results.

Many methods have achieved good performance using deep learning methods and there are some research on deep learning related to this article. For example, flow prediction [18], [19] and trajectory modeling [12] achieved good results compared with the traditional method. Song *et al.* [20] proposed a method called DeepTransport, which can automatically simulate or predict people's future activities and their transportation modes. In the work of [8], the road segment and the subpath-based method are combined in a multitasking manner to calculate and sum the two loss functions in a certain proportion and estimate the travel time, but the time feature cannot be accurately used. The temporal characteristics of each subpath are used in our proposed method (the start time of the subpath is the temporal characteristic of the subpath). For example, the time of the entire path is the temporal characteristic of driving on the first subpath and the start time of the path is the time of the first path. If the other subpaths use the same temporal characteristics, the actual time does not coincide here, as the driving time increases, the error will further increase [9]. Using the trajectory point data, all the trajectory points are converted into a grid map in a certain proportion. The query trajectory points are mapped to the corresponding grid to retrieve the information of the surrounding grid and capture

the surrounding information. It uses too much historical data to represent the current subpath information; it cannot make an accurate estimate of the current information, because the traffic conditions are particularly time-dependent and dynamic, and cannot express the long-term characteristics of the state of compression as the current state. For example, the current road segment characteristics may not need to consider the early peaks experienced an hour ago. In our model, we have detailed the temporal features and filter the historical trajectory data to find the nearest time interval according to the current time (set to 3 min in the experiment). In the historical trajectory, accurate information about the current query segment or subpath can be obtained by accurately obtaining the location information and the time information so that accurate road segment information can be obtained, but the sample size may be too small due to the very accurate search, and the reliability of the path information may be reduced. When specifically querying the start and end positions of the entire path by using the historical trajectory with the same starting and ending points for training, some of the paths cannot be accurately predicted due to too few samples, and it is possible to travel in a circle from point A to point B. We improved on the method proposed by using the information for the road segment and the adjacent road segments to increase the number of samples and prevent data sparseness.

There are also many ways to process and utilize the trajectory data [21]. Liu *et al.* [22] proposed a spatial temporal recurrent neural network (ST-RNN) to simulate the temporal and spatial context characteristics in each layer. Gao *et al.* [23] proposed a semisupervised deep learning model called TULER for mining the nonsemantic features of user movement patterns from spatiotemporal data. Wu *et al.* [12] used the improved RNN to process variable length sequences and solve the topology constraints of the structure modeling on trajectories. Zhang *et al.* [18] proposed an end-to-end model called ST-ResNet, which uses the residual neural network framework to simulate the time proximity, period and trend properties of population traffic. These demonstrate the powerful representation of deep neural networks, so we use deep neural networks to build our models.

III. Premises

In this section, we describe the taxi data processing used in the experiment and the definition of the problem.

The raw data contain many driving paths, each path has Timestamps, Polylines, and DriverIDs (the other properties are not used). A polyline consists of a series of consecutive GPS points (each point has lat and lon), i.e., $G = \{g_0, g_1 \ldots g_n\}$, the timestamp indicates the start time of the path, and the polylines indicate the driving positions of the path. We divide the path into multiple disjoint segments, i.e., $\mathrm{Seg} = \{s_0, s_1, \ldots s_{n-1}\}$. The start and end points of each road segment are determined by the direction of travel of the route, e.g., for s_0, the starting point is g_0, and the ending point is g_1. The driving distance and speed of the road segment are calculated based on this information. Note that the time interval between the points in the dataset is 15 s.

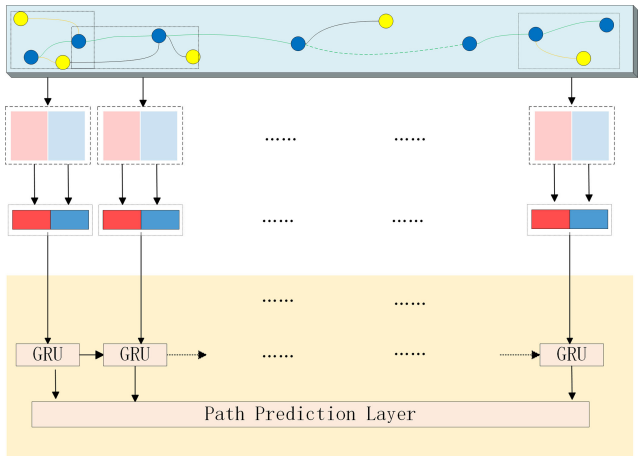


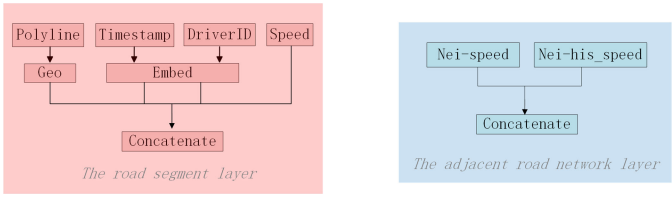
Fig. 1. Nei-TTE architecture. The blue points correspond to the GPS sequence of the current path (the line connecting the blue points represents the entire path), and the light red and blue frames represent the segment and the adjacent segment layers, respectively, which output to the dark red box and blue box after feature extraction.

We use the features of this road segment, the characteristics of the adjacent road networks, and the historical information as input to train the model, and then predict the path travel time.

We address the structure as being similar to sentences and words. The entire path contains a number of segments, each of which corresponds to a word. Since each path has different results and cannot correspond to the same road segment, we define the road segment and adjacent road segments based on [13]. As mentioned in the introduction, the trajectory points cannot reasonably correspond to real road segments. That is, there is no guarantee that the segments of each trajectory can be analogized to the words in a sentence. We extract the link information in the nearest space for this segment and the adjacent segments. There are three advantages to this approach. First, the surrounding road segments can be searched as if they had a relationship probability similar to that between words. Second, the use of the road segments can better find the nearest road segment to prevent problems in path matching the start and end points (such as path A–C–B may appear in A–B path detection). Finally, since the road network topology is difficult to capture, the travel trajectory of the surrounding road segments is used to effectively represent structural features.

IV. Model Description

In this section, we will introduce the architecture of the Nei-TTE model, as shown in Fig. 1, which includes the road segment layer, adjacent road network layer, and prediction layer. The road segment layer is used to process the time information (e.g., the timestamp), spatial information (e.g., the start and end locations of the current road segment), and basic information for a given path (e.g., the DriverID). The adjacent road network layer is used to capture the topological features of the adjacent segments of the road segment. The outputs of the road segment and adjacent road network layer are concatenated together and used as input



(a) (b)

Fig. 2. (a) Segment layer. (b) Adjacent road segment layer.

to the prediction layer. The prediction layer uses a time-series model to estimate the travel time of the entire path.

A. Road Segment Layer

We use the road segment layer to learn the segment feature. It includes processing the spatiotemporal features and other external features of the road segment, as shown in Fig. 2(a). For the spatial features, we map the original GPS points corresponding to the location of the road segment into the spatial road segments and capture the spatial feature representation of the road segment. To fully mine the time features of the road segment, we make full use of time stamps and, then, extract the historical trajectory information closest to the current time as the feature representation of the road segment. We use the historical trajectory information that is closest to the space in this segment to express its characteristics.

1) Driver ID Embedding: Each path has a DriverID, that is, the DriverID is the same outside of the divided path segment. This feature cannot be ignored in the existing travel time estimation solutions, because each driver has his own driving habits that are not easily changed, the driving habits of different drivers will be very different. Embedding [24] can improve the quality of the word vectors and the training speed and can effectively reduce the dimensions. Many existing methods (e.g., [9]) use embedding to map the driver ID to a lower dimension. Here, our article uses the same technique.

2) Spatio-Temporal Feature: First, we represent the geo component. As defined in PREMISES, we divide each path into segments of length n – 1, where n is the length of the path. For spatial features, *convolutional neural networks* (CNNs) or nonlinear functions are generally used to mine their features. CNN widely apply the features used in images to extract space and achieve good results. In the work of [15] and [22], the GPS points of the path are mapped to an $I \times J$ grid map and, then, the points in the path features are captured as images. Direct mapping of the path points to the grid map does not take into account the spatial information (the two-dimensional data cannot represent the three-dimensional space well). In [8], the spatial description is captured by a nonlinear function, and the spatial features of the subpath are captured using convolution operations. In our model, we use nonlinear functions and simplify this operation to extract spatial features. Although the CNN is effective on images, using it will increase the training time of the model and to potentially capture the spatial characteristics of the road segment fully.We use the geo-component to map the road segment location information to the road space. For each road segment, the start and end point position information is stitched and mapped to the multidimensional image by a function. The space information of the road segment is indicated as a result, as shown in formula (1). $< >$ represents the concatenated spatial feature of the road segments. Tanh is an activation function for the result and has the following advantages: first, the value range is [–1, 1], and the mean is zero. This is effective for standardizing the data when the features are significantly different as all the information is effectively captured

$$V_{\text{spa}} = \tanh\left(W_{\text{spa}} \left\langle \text{start}_{lon}, \text{end}_{lat} \right\rangle\right). \tag{1}$$

W_{spa} is a parameter matrix that is used to learn the spatial characteristics of the segment in a function. The output is n-dimensional, i.e., n channels represent the feature space. The same procedure is applied to the segments that are divided by the entire path and the result is input into the Geo component. To obtain more detailed spatial features, k is set as 32. The output is expressed as V_{spa}.

The entire path is divided by the same time interval to obtain each road segment, and each road segment starts with a timestamp. This timestamp indicates the start time of the current road segment. In [9], a day is divided into different time intervals, and the time stamp of each point is converted into the corresponding time interval which is embedded. In [8], the timeID and weekID for each path are used to represent the start time and the day of the week, respectively, and the two attributes are directly embedded to represent the time characteristics of the path. In processing the time features, according to the length of the current time, the time proximity, period, and trend attributes are proposed to fully analyze the time features. It can be seen that the dynamics of traffic conditions are largely influenced by time factors. We provide the start time for each segment of the path. Providing the model with the exact start time of each segment can improve the accuracy of the time estimate. Since the path lengths are different, in general, the longer the path is, the more time it takes to travel. If the start time of each road segment is the same as the entire path, the problem is equivalent to estimating the time required for different sections at the same starting time. Then, it adds up to get the entire path prediction time and actual time to evaluate. In the real world, we always use the current time to make predictions. When going to a strange place, we may use the navigation system to guide us. Estimate the time required by selecting a route based on the current location. After a while, check the navigation system again to determine whether our location is on the right route. If it is correct, we will re-estimate the time required at the current location; otherwise, the location will be adjusted. In our case, we introduced the temporal features for a more accurate time estimate. We put the timestamp in a standard time format and extract the month, day, hour, minute, and second. The same day of a year has similar traffic conditions. The working days and nonworking days are very different, and different working days will be different. For example, the daily rush hour will be in the same time interval with small time fluctuations.

Since the dynamic traffic has particularly large changes, the entire embedded timestamp cannot capture more detailed features. For example, a traffic jam or traffic accident that happened a few minutes in the past may not affect the current traffic conditions, so the details of the division need to be made at the current time instead of embedding the entire timestamp. The effect of extraction time feature on time estimation is concatenated as V_{time} after embedding. This better captures the impact of the time characteristics on the current traffic.

3) Speed: The distance and travel speed of the road segment can be calculated and measured using its start and end points to determine the driving status and other conditions. The driving distance of the road segment can effectively reflect the driving situation. For example, if the driving distance is short, there may be a segment with a speed limit. If the driving distance is long, the driving state is better during this period. However, the model cannot effectively simulate this feature, so we calculate the feature and add the driving distance to the road segment feature. Although the speed may cause errors during sampling, it can also be a good indication of the original road conditions. For example, if the speed is extremely slow, there may be road congestion. The topological information of the road largely affects the driving state, which affects the time estimation result. Since the trajectory points in the data do not correspond exactly to the actual road segments, it is very difficult to extract the topology information of the road segments. The focus of this article is only to predict the time required for a given path. As mentioned above, the speed feature can be used for the topological information. When the vehicle turns, the speed is slower; when driving on a section with speed limits, the speed usually does not exceed the maximum speed limit. Simply considering the speed of a segment cannot reflect the road characteristics during the corresponding period, so the historical information must be added. If a long period of historical time is introduced to calculate the speed, it will cause a deviation in the speed value prediction resulting in an error in the predicted time. For example, there may be a very large difference in speed within 1 h of calculating the start or end of a shift. We apply the data in the current road segment space to the historical trajectory data, which can effectively represent the features in the current road segment space. Using these data can not only represent the topology information well but also prevent the sample from being underutilized, making the segment features inaccurate. We consider the average speed in the current time, the representation vector for the first 3 min of the current time, and the representation vector for the same historical time period. We put all the features of the road segment together as the output of this layer, shown as

$$V_{\text{seg}} = \langle V_{\text{spa}}, V_{\text{time}}, V_{\text{speed}}, V_{\text{his-speed}}, V_{\text{dist}} \rangle . \tag{2}$$

Note that this is a segment of the path.

Introducing a long period of historical information will interfere with the current road segment features, such as using the historical information from 1 h before the current road segment time. To improve this part of the error, we use the first 3 min before the current historical information (the same as the adjacent road network layer).

B. Adjacent Road Network Layer

Only considering the feature of this segment does not capture the feature of the roads around the segment. Most of the work already uses the surrounding information. In [13], for the query path, the start and end points are fixed and the estimate is based on the path with the same start and end points in the historical trajectory. There is a part of the path that does not have a historical path with the same start and end position as the reference. The topology of a link has a great influence on the road conditions. In [12], the improved RNN is used to model the topology constraints on the trajectory [9]. The trajectory points are mapped to the grid, and the grid information around the current grid is used as auxiliary information and extract these features using CNN. The adjacent road segments have the similar road condition as the road segment, the data can provide auxiliary information and improve the accuracy of the road segment features. Therefore, this method can introduce the trajectory features of nonadjacent segments. For example, if traffic jams occur on a segment of the road, the adjacent road segments will also be affected by it; at the same time, accidents in adjacent segments will also affect the road conditions of this segment. If there are two disjoint paths, the traffic conditions of the two sections may vary greatly due to the fact that they are not affected by the accident. Since the topology information has appropriate factors for traffic conditions but cannot be extracted efficiently, the speed feature can effectively represent the topology information in another way. Therefore, we use the speed in the historical trajectory in the space to represent the surrounding topology information. The data of the adjacent road segments are used as the historical trajectory data. The interval is the same as in the speed feature (3 min). The average speed from the first 3 min to the current time is used as the auxiliary neighbor feature $V_{\text{nei\_speed}}$, and the average historical speed of the period is extracted as $V_{\text{nei\_his\_speed}}$. We use V_{nei}^{i} to represent the result of $V_{\text{nei\_speed}}$ and $V_{\text{nei\_speed}}$ concatenated.

C. Prediction Layer

The RNN is a key technology for processing time-series data. It has made important breakthroughs in the fields of natural language processing and machine translation. A sentence is modeled as an input to learn the semantic and structural information of the sentence and is effectively translated into another language [26]. There are many variants; here, we cover two of them, the *long short-term memory* (LSTM) [27] and *gated recurrent unit* (GRU) [28]. Both of these methods alleviate the RNN problem of gradient disappearance caused by backpropagation. The GRU will forget the data and input the data synthesis to update the gates. Compared with the LSTM, the model is simpler, and due to fewer parameters, it does not easily cause overfitting problems in the model.

In the prediction layer, we use GRU to predict the time of the entire path. Many existing travel time estimation problems use LSTM as the main part of the model. In our model, the characteristics of the road segment are extracted by the segment layer. The adjacent road segments capture the features around the road segment, and their output is used as the input for the

GRU as well as the features output by the segment layer and the output from the adjacent segment layer. The V^i_{seg} feature output by the segment layer, and the V^i_{nei} feature output by the adjacent segment layer, represent the ith segment, i∈ path of length n – 1. The modified formula can be expressed as

$$h_i = \theta_{\text{relu}}\left(W_{\text{seg}} \bullet \left\langle V^i_{\text{seg}}, V^i_{\text{nei}} \right\rangle\right) \tag{3}$$

where W_{seg} is a parameter matrix that is used to simulate the road segment features and adjacent topological features. h_i is the hidden state, which is the output of a cell unit of a GRU. θ_{relu} is an activation function; we use the *rule* function in the experiment.

Since the length of each path is not the same, we use an averaging calculation to set each path to the same length. Then, the full connection method is used to calculate the travel time of the entire path for the estimate. The residual network can be better used for model fitting due to the introduction of residual terms to prevent model degradation. The residual term is then added to the fully connected layer. As shown in (4), the prediction of the ith path is processed as an average

$$\text{Path}_i = \frac{1}{n-1}\sum_{i=0}^{n-1} h_i. \tag{4}$$

D. Training

We now show the training process of the model. As with most of the work, to enable the model to have good matching capabilities on short and long paths, we use the MAPE as the primary evaluation criterion. The goal of our model is to add the estimated time of all the paths to the real time loss and optimize the loss by a back-propagation algorithm. The objective function is as follows:

$$\min(w)\sum_{i=0}^{k} F\left(\text{Path}_i, \text{Path}_i^{\text{true}}\right). \tag{5}$$

In this function, k represents the number of all the paths, and F represents the function used for the evaluation of the model training results, where the two parameters represent the estimated time of the real and predicted paths, respectively, Path_i is the estimated time of our model, and w is the optimize parameters.

We use three measurement methods to evaluate the estimation results, including the *mean absolute error* (MAE), MAPE, and *rooted mean squared error* (RMSE).

V. Experiments

In this section, we will describe the real-world datasets used in our experiments, then compare them with the proposed baseline method, and, finally, verify the validity of the modules used in the experiments.

A. Data Description

The experimental dataset comes from the Kaggle competition, public dataset, and has a size of 1.8 G; it is linked online.[1] The data range from July 1, 2013 to June 30, 2014 and consist of the trajectories for all 442 taxis running in the city of Porto.

The trajectories in the Porto dataset are associated with the corresponding Timestamp, DriverID, and Polyline (contains a list of GPS coordinates, i.e., *lon* and *lat*]. We removed the data with a polyline length greater than 250 (preventing too few samples, accounting for 0.00578% of the total data), and randomly selected trajectory 50 W from the remaining data for the experiment, as shown in Table I. At the same, we also extract the timestamp corresponding to each trajectory point in different trajectories. Experimental data sample distribution are shown in Fig. 3.

TABLE I
Experimental Dataset Description

Dataset	Porto
Trajectory number	50 w
Travel time mean	708.60 s
Travel time std	432.26 s
interval	15 s

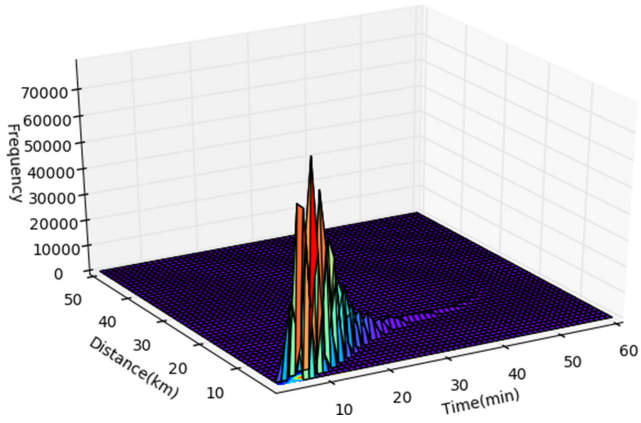


Fig. 3. Experimental data sample distribution.

B. Parameters

We set the DriverID to R^{16}, the timestamp after the split, the month, day, hour, minute, and second are set to R^4, R^5, R^5, R^6, and R^6 In the Geo component, the activation function of the nonlinear function is tanh, and the output is R^{32}. In the prediction layer, we use GRU, the hidden layer size is 128, and two layers of fully connected networks with residuals are used. The size of each layer is set to 128, and the rule method is used as the activation function.

We performed a fivefold cross-validation using the experimental data. The optimization algorithm for training used in our article is Adam algorithm [29], the batch size is 512, and the model is trained for 50 epochs. The learning rate of Adam is 1e–4.

C. Performance Comparison

We give a comparison with the existing methods [the results of the Stacked AutoEncoders (SAEs) and path travel time estimation (PTTE) come from Deep-TTE] and our own baseline methods.

[1][Online]. Available: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

TABLE II
PERFORMANCE COMPARISON FOR THE PORTO DATASET

	MAPE (%)	MAE (s)	RMSE (s)
SAEs	0.273	222.06	357.02
PTTE	0.207	159.43	268.11
Deep-TTE	0.133	113.24	219.25
TTE-FULL	0.202	174.17	322.04
TTE-LSTM	0.087	97.44	226.36
Nei-TTE-time	0.085	81.69	173.35
Nei-TTE	0.070	53.66	107.80

SAEs: Segment-based approach, and in terms of the SAEs [14]; this is first time an automatic encoder was used as a building block to represent a deep architecture model of the predicted traffic flow characteristics, and it achieved good performance in traffic flow forecasting tasks.

PTTE: Subpath-based approach, PTTE [16], uses a tensor to simulate the travel time of different drivers on different road segments in different time periods. Combining the geospatial data from the trajectory and map data, a dynamic programming solution is designed to find the optimal trajectory cascade.

Deep-TTE: End-to-end approach, deep-TTE [9] is a supplementary supervision model in which the trajectories points are mapped onto the grid and the information is used to query the path trajectory points around the point estimate of the time path.

TTE-FULL and **TTE-LSTM:** In addition to the above methods, we propose our own two baseline models, the TTE-FULL and TTE-LSTM. The input of these two models is the same as the original input. In the TTE-FULL method, we modified the LSTM in the model to have two fully connected layers in the stack in order to be similar to the LSTM network. The size of the fully connected layer is also set to 128. In the TTE-LSTM method, we use the LSTM as a two-layer Bi-LSTM (the LSTM model is implemented with Pytorch), and both models add residuals in the path time prediction part using the two layers of full connections with residuals. We separately adjusted the two models to optimize them.

As can be seen from Table II, the subpath-based approach is generally superior to the segmentation-based approach because the subpath-based approach takes into account traffic lights and speed changes, such as PTTE, which is superior to SAE. The end-to-end deep-TTE method works better than PTTE. Since the adjacent trajectory points are used to increase the sample, the data reliability is improved, and the time information of each trajectory is used as an aid. Furthermore, our method is better than the Deep-TTE method, and the error rate is 0.070. The TTE-FULL has higher error rate compared with the TTE-LSTM and the Nei-TTE because the trajectory data are not considered. As for the latter two, although they both use the variant RNN structure, the GRU has fewer parameters than the LSTM to get the model to an optimal state faster. We formerly analyze the error rate and driving distance of our proposed model. As shown in Fig. 4, our model can predict the long trajectory better. Finally, we perform the binning operation by predicting the time and real

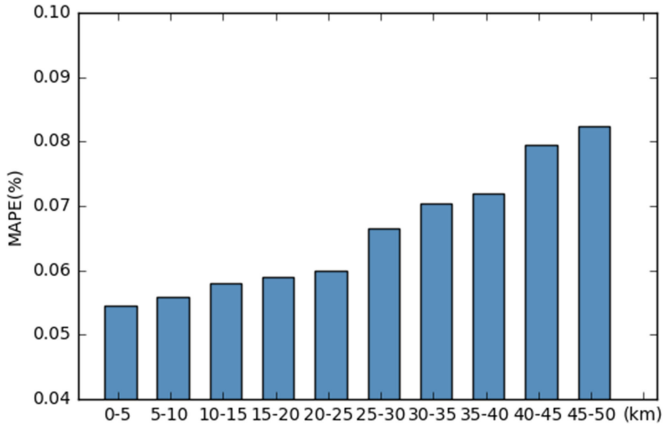


Fig. 4. Comparison of MAPE and travel distance.

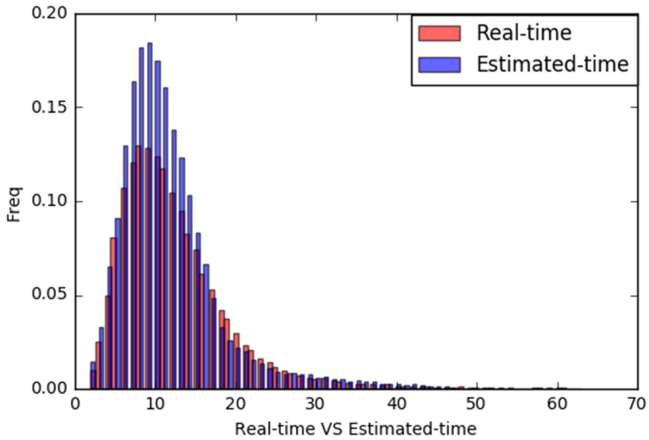


Fig. 5. Real time versus estimated time.

time and counting the frequency of each time segment. As shown in Fig. 5, it is further verified that our model can predict long trajectories.

D. Feature Effectiveness

We show the performance of the proposed properties. It is mainly tested in two aspects, including the start time characteristics of each road segment and the speed characteristics of the road segments and adjacent road segments. We add the characteristics of these two parts separately and, then, compare them. Nei-TTE-time contains only time features while Nei-TTE contains all features.

We calculate real start timestamp for each trajectory point in each path, and use this result for training and prediction. As the result, the MAPE of our model (Nei-TTE) is only 0.085. Compared with the subpath-based method whose MAPE is 0.207, there is a significant improvement. Even though Deep-TTE also uses the information of the trajectory point as the auxiliary information, its MAPE is still 0.133. It can be seen that if the start time of the trajectory is used and the time of the entire path is directly estimated, a large error will result. At the same time, as

the path time and distance increase, the error will become larger. Each road segment has a start time that can be used to accurately obtain the current time period and road condition information in the road space. Since the time feature is the start time feature of the entire path, losing part of the feature due to not specifying the start time may result in error, we provide the same start time for each segment. In this way, we can increase the time prediction precision of each segment.

We further increase the trajectory information adjacent to each trajectory segment. Compared with the previous results, MAPE increased by 0.015. This shows that we can use the trajectory information of the adjacent road segments of a certain road segment as a valid example to compensate for the data sparse problem caused by the long travel distance, so that the trajectories of different lengths can be predicted. The results show that this feature can have the prediction problem of long trajectories.

VI. Conclusion

In this article, we proposed a travel time estimation model based on deep learning, called Nei-TTE. By using the historical data for the road segments at fine-grained time intervals, the speed feature can be effectively used to simulate the topology information of the surrounding space. We also assigned an accurate start time to each road segment to predict the time using an innovative concept. We experimented on large-scale real-world datasets. The results showed that our method has achieved good estimation results and is significantly better than the existing models. It has important guiding significance for the application of intelligent transportation in the IoT.

References

[1] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, and L. T. Yang, "Trust-based communication for the industrial Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 16–22, Feb. 2018.

[2] C. Zhu, L. Shu, V. C. M. Leung, S. Guo, Y. Zhang, and L. T. Yang, "Secure multimedia big data in trust-assisted sensor-cloud for smart city," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 24–30, Dec. 2017.

[3] Z. Tian *et al.*, "Real time lateral movement detection based on evidence reasoning network for edge computing environment," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4285–4294, Jul. 2019.

[4] X. Du and H. H. Chen, "Security in wireless sensor networks," *IEEE Wireless Commun. Mag.*, vol. 15, no. 4, pp. 60–66, Aug. 2008.

[5] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang, and Z. Tian, "Towards a comprehensive insight into the eclipse attacks of tor hidden services," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1584–1593, Apr. 2019.

[6] Z. Tian, S. Su, W. Shi, X. Du, M. Guizani, and X. Yu, "A data-driven method for future internet route decision modeling," *Future Gener. Comput. Syst.*, vol. 95, pp. 212–220, 2019.

[7] J. Qiu, Y. Chai, Y. Liu, Z. Gu, S. Li, and Z. Tian, "Automatic non-taxonomic relation extraction from big data in smart city," *IEEE Access*, vol. 6, pp. 74854–74864, Nov. 2018.

[8] D. Wang *et al.*, "When will you arrive? Estimating travel time based on deep neural networks," in *Proc. Nat. Conf. Artif. Intell.*, 2018, pp. 2500–2507.

[9] H. Zhang *et al.*, "DeepTravel : A neural network based travel time estimation model with auxiliary supervision," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3655–3661.

[10] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos, "Route travel time estimation using low-frequency floating car data," in *Proc. 16th Int. Conf. Intell. Transp. Syst.*, 2013, pp. 2292–2297.

[11] J. Zhang, W. Cao, Z. Qin, L. Zhu, Z. Yu, and K. Ren, "When privacy meets economics: enabling differentially-private battery-supported meter reporting in smart grid," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–9.

[12] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3083–3090.

[13] H. Wang, Y. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, Art. no. 61.

[14] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[15] D. Wang, W. Cao, M. Xu, and J. Li, "Etcps: An effective and scalable traffic condition prediction system," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2016, pp. 419–436.

[16] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of apathusing sparse trajectories," in *Proc. 20th Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 25–34.

[17] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration : A deep learning method," In *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 499–508.

[18] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31th AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.

[19] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[20] X. Song, H. Kanasugi, and R. Shibasaki, "Deeptransport: Prediction and simulation of human mobility and transportation mode at a city wide level," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 2618–2624.

[21] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, Art. no. 92.

[22] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: a recurrent model with spatial and temporal contexts," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 194–200.

[23] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1689–1695.

[24] T. Mikolov *et al.*, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[25] Z. Zhang, Z. Qin, L. Zhu, J. Weng, and K. Ren, "Cost-friendly differential privacy for smart meters: Exploiting the dual roles of the noise," *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 619–626, Mar. 2017.

[26] J. Qiu, Y. Liu, Y. Chai, Y. Si, S. Su, and L. Wang, "Dependency-based local attention approach to neural machine translation," *Comput., Mater. Continua*, vol. 58, no. 2, pp. 547–562, Mar. 2019.

[27] M. T. Asif *et al.*, "Spatiotemporal patterns in large-scale traffic speed prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 794–804, Apr. 2014.

[28] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," presented at the ICLR: 3rd Int. Conf. Learning Rep., San Diego, CA, USA, 2015.

[30] L Zhu, M Li, Z Zhang, and Z Qin, "ASAP: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2850780.