STDR: A Deep Learning Method for Travel Time Estimation

Jie Xu(✉), Yong Zhang, Li Chao, and Chunxiao Xing

Department of Computer Science and Technology,
Research Institute of Information Technology,
Beijing National Research Center for Information Science and Technology,
Institute of Internet Industry, Tsinghua University, Beijing 100084, China
xujl5@mails.tsinghua.edu.cn, {zhangyong05,li-chao,
xingcx}@tsinghua.edu.cn

Abstract. With the booming traffic developments, estimating the travel time for a trip on road network has become an important issue, which can be used for driving navigation, traffic monitoring, route planning, and ride sharing, etc. However, it is a challenging problem mainly due to the complicate spatial-temporal dependencies, external weather conditions, road types and so on. Most traditional approaches mainly fall into the sub-segments or sub-paths categories, in other words, divide a path into a sequence of segments or sub-paths and then sum up the sub-time, yet which don't fit the real-world situations such as the continuously dynamical changing route or the waiting time at the intersections. To address these issues, in this paper, we propose an end to end Spatial Temporal Deep learning network with Road type named STDR to estimate the travel time based on historical trajectories and external factors. The model jointly leverages CNN and LSTM to capture the complex nonlinear spatial-temporal characteristics, more specifically, the convolutional layer extracts the spatial characteristics and the LSTM with attention mechanism extracts the time series characteristics. In addition, to better discover the influence of the road type, we introduce a road segmentation approach which is capable of dividing the trajectory based on the shape of trajectory. We conduct extensive verification experiments for different settings, and the results demonstrate the superiority of our method.

Keywords: Travel time estimation · CNN · LSTM · Road network

1 Introduction

Nowadays, with the explosive growth of the location-enabled devices, the importance and usage of geospatial information have attracted more and more attention from researchers in many applications. In this paper, we mainly focus on the estimation of travel time, which can bring societal and environmental benefits, and is useful for driving navigation, traffic monitoring, route planning, ride sharing and so on [1–5]. However, evaluating an accurate travel time is a challenging problem affected by the following aspects: (1) The road traffic condition continuously dynamically changes during the process of vehicles moving. (2) The driver needs to slow down or wait for a

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 156–172, 2019.
https://doi.org/10.1007/978-3-030-18579-4_10

while at the intersection, and the waiting time is a random variable, modeling the waiting time is not easy. (3) Different road segments may exhibit very different behaviors due to external requirements, for instance, residential areas speed limits are distinct from industrial areas. Although the problem has been widely studied in the past, however, traditional travel time estimation approaches mostly adopt divide-and-conquer approach [6–8]. Those methods mainly decompose a path into a sequence of sub-segments or sub-paths, and then sum up the multiple sub-segments or sub-paths travel time into a whole. Nonetheless, those methods are model-driven and can't handle the delay time caused by the turnings or intersections very well, and the estimated travel time errors, which are affected by external factors such as weather or road types, will also accumulate.

Generally speaking, solving this kind of optimal problem is not easy. Fortunately, recently deep learning has achieved considerable achievements in computer vision, machine translation, image generation, natural language processing field and road trajectories [9, 10]. Deep learning approaches have a strong capability to learn more latent features and simulate complicated dynamics trajectory problem [4, 11–13]. Motivated by aforementioned knowledge, in this paper, we propose an end-to-end Spatial Temporal Deep learning network with Road types (STDR) by using convolutional neural networks (CNN) and long short-term memory (LSTM) to jointly capture complex spatial and temporal nonlinear correlations. The core idea lies in transforming the trajectory data into vector space, and applying the neural network on them. The contributions of this work are summarized as follows.

First, to capture the spatial features of a trajectory, we embed the GPS points into corresponding vectors rather than working on them directly. The vectors can preserve the original correlation of different points, and then are fed into the CNN to learn the spatial dependencies.

Second, we introduce LSTM with attention mechanism to model the time series. The attention mechanism can judge which segment has a higher weight for estimating the whole trajectory. Meanwhile, the influences of external factors (e.g., weather and time metadata) are also concentrated into the LSTM input, which can significantly improve the predicting accuracies.

Third, since the trajectory sampling frequency is fixed, the travel distances with different velocities at the same time interval are diverse. For example, the driving distance on the highway is longer than the distance on an overpass at the same time interval. Therefore, we partition the trajectory into many sub-segments based on the road types, then the sub-segments are encoded into vectors and concatenated with the output of LSTM for training the model together.

Fourth, extensive comprehensive experiments on the real datasets are conducted, and results show the advancement of our approach.

The remainder of this paper is structured as follows. Section 2 describes a review of literature, and Sect. 3 introduces the details of our algorithm. We conduct experiments and evaluate the results in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Related Work

Existing researches can be roughly classified into two categories, i.e., the traditional approaches with sub-segments or sub-paths, and the deep learning approaches. In this section, we introduce the literature and summarize the key technologies.

2.1 Traditional Travel Time Prediction Approaches

Wang et al. [7] proposed an efficient and simple model that leverages plenty of historical trips without using the intermediate trajectory points to evaluate the travel time between source and destination. Comparing with the most existing approaches, it retrieves the neighboring trips with a similar origin and destination to approximately estimate the travel time. However, the method outperformed many online methods. To deal with traffic time series which were usually sparse, dependent and heterogeneous (e.g., some segments may have morning and afternoon peak hours, while others may not), Yang et al. [14] proposed Spatial-Temporal Hidden Markov models (STHMM). The dependencies and the correlations among different time series were modeled while considering the topology of the road network. Wang et al. [6] modeled different drivers' travel times with a three-dimension tensor, the frequent trajectory patterns were extracted from historical tips to decrease the candidates of concatenation and suffix-tree-based indexes, then an object function was devised and proved to model the tradeoff between the length of a path and the number of trajectories traversing the path. The object function was then solved by a dynamic programming solution. Wen et al. [8] proposed a novel probability-based method by constructing a temporally weighted spatial-temporal distribution patterns to estimate the logistical transport time. In order to explore location and time relationship, they designed frequent spatial connections, in which area-based spatial-temporal probabilistic distribution can be identified by kernel density estimation. Then the transportation time between two locations in the area can be estimated. Similarly, Jabari et al. [15] established a mixture asymmetric probabilistic statistical framework, i.e., a novel data-driven methodology of Gamma mixture densities, to model complexity multi-modal urban travel time distributions, experiments also demonstrated their methods can further solve the data sparsity.

2.2 Deep Learning Travel Time Prediction Approaches

To cope with the insufficiency of the input information, Li et al. [11] constructed a more smooth and meaningful multi-task representation learning by leveraging the underlying road network structure and spatial-temporal prior knowledge. Jindal et al. [13] first predicted the distance between the origin and destination, and then estimated the travel time based on the above predicted distance. The advance of ST-NN was that it only take advantage of the raw trips data without demanding further feature engineering. However, the road network structure, i.e., the spatial and temporal relationship, was neglected. In order to solve the inaccuracies caused by the divide-and-conquer methods, Wang et al. [16] proposed a novel end-to-end deep learning framework to estimates the travel time of the whole path directly, they used a geo-convolutional and LSTM layer to capture the spatial and temporal features meantime.

In addition, they also introduced a multi-task learning to balance the effect of the entire path and each local path. However, they didn't consider the influences brought by the driver's habit and external road types. Zhang et al. [17] proposed an end-to-end training-based model named DEEPTRAVE to predict the travel time of a whole path directly, and designed an auxiliary supervision with dual interval loss mechanism to fully leverage the temporal existing historical labeling information. They utilized a feature extraction structure to effectively capture different dynamics, such as short-term and long-term traffic features, for estimating the travel time accurately. Cui et al. [18] presented a deep stacked bidirectional and unidirectional LSTM (SBULSTM) neural network architecture, which investigated both spatial features and bidirectional temporal dependencies from historical data. This mechanism can effectively handle the missing value for input data, and can also address the passing information from a back-propagation direction.

3 Methodology

In this section, we formally depict the preliminaries and define the notions of the problem, then present the details of our method.

3.1 Preliminaries

Definition 1. A road network $G = (V, E)$ is a directed graph, the $V = \{v_i(x_i, y_i)\}$ represents a set of vertices, each v_i incorporates latitude x_i and longitude y_i, the $E = \{e_j(v_m, v_n)\}$ represents a set of edges, each e_j is comprised of two directly connecting vertices.

Definition 2. A trajectory T is a sequence of GPS points generated from LBS (Location Based Service) devices, which can be denoted as $T = \{p_1, p_2 \ldots\ldots p_n\}$. Each GPS point p_i contains 5-tuple $(t_{pi}, x_i, y_i, velocity_i, head_i)$, where t_{pi} denotes the timestamp, x_i and y_i denote the latitude and longitude respectively, $velocity_i$ is the vehicle speed, $head_i$ is the angle of driving direction.

Problem: Given a trajectory T and departure time t_o, our goal is to estimate the travel time for T by using a series of historical trajectories on road network G.

3.2 Framework of STDR

As presented in Fig. 1, we provide the details for our proposed spatial-temporal network framework, which is comprised of three major components.

Spatial Component: We first leverage the road network embedding method to embed the GPS points into corresponding vectors, thus we can use the convolutional layer with many filters to extract the spatial characteristic [9], after that the output matrix vectors of convolutional operation are used as the input of LSTM.

Temporal Component: We manually collect some external features from external datasets, such as time of day, weather conditions, etc., and then embed them as a vector

X*ext*, and concentrate with vectors C_{IN} which are obtained from above spatial component, moreover feed them into two stacked LSTM with attention mechanism to extract the temporal characteristic with attention mechanism.

Road Type Component: We introduce a segmentation approach which is capable of dividing the trajectory into sub-segments based on the road types, then calculate each sub-segments average distance, which is further encoded into vectors. To enable the full connected operations on the irregular vectors matrix, we pad zero where it is necessary. Then the vectors further combine the output of the LSTM component, and furthermore go through a fully connected layer at the end of the network for joint prediction.

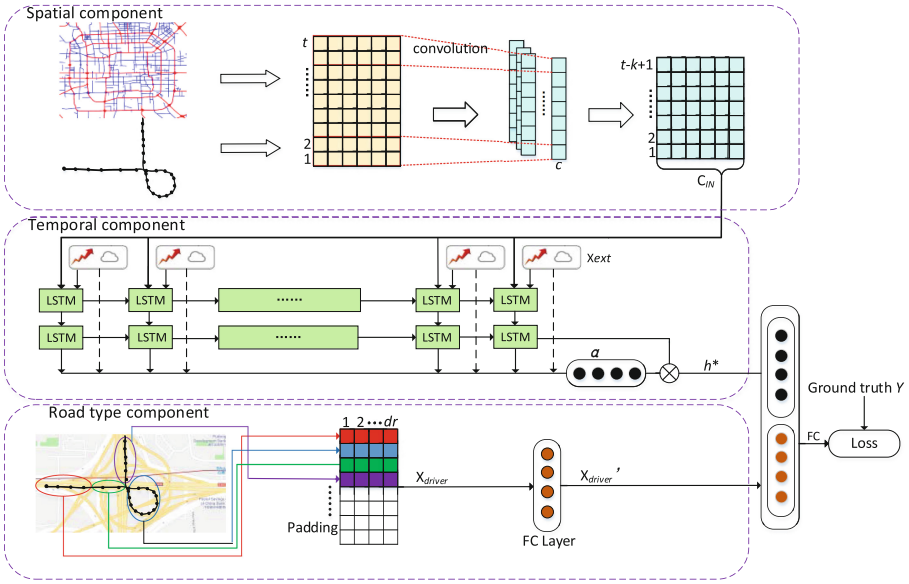

Fig. 1. The framework of STDR

3.3 Trajectory Embedding Representation

In [16], the geographical features they extracted are multiplied by latitude and longitude of two points, which don't consider the global road network dependencies among different vertexes. Therefore, we first need to convert the GPS points into vectors rather than working on them directly [9]. How to transfer the trajectory can be recognized as the problem of embedding very large road networks into low-dimensional vector spaces, aiming to capture and preserve the original network structure. The characteristics of vertexes are dependent on both the local and global network structure. Therefore, how to

simultaneously preserve above structures is a tough problem. In this paper, we use a road network embedding method [19], which can effectively handle the distance and angle in the road network space by using the Minkowski *pth*-order metrics:

$$L_p(x,y) = \left[\sum_{i=1}^{k} |x_i - y_i|^p\right]^{1/p} \tag{1}$$

Transforming a road network $G = (V, E)$ into a vector space (R^L, D') is a mapping $V \to R^L, E \to D'$, where L is the vectors' dimension and D' is one of Minkowski metrics [19, 22]. The vertex is extended as follows: Let V_i be a subset of V, x be a point, and $Dist(x, V_i) = \min_{y \in Vi}\{Dist(x, y)\}$, here $Dist(x, V_i)$ is the distance from point x to its closet neighbor in V_i. Then let set $R = \{V_{1,1}, \ldots V_{1,k,} \ldots V_{\beta,1} \ldots V_{\beta,k}\}$ be a subset of V, where k is set as O(log n) and β is set as O(log n), the original space can be embedded into a O($\log^2$ n) dimensional space. Specially, let $E(v) = (R_{V_{1,1}}(v), \ldots, R_{V_{1,k}}(v), \ldots R_{V_{\beta,1}}(v) \ldots R_{V_{\beta,k}}(v))$, in which $R_{V_{i,j}}(v) = Dist(v, V_{i,j})$, thus for single point v, the finally output is vector $E(v) \in R^{\beta * k}$ with the dimension $\beta * k$ (the following is denoted by $L = \beta * k$ for the rest paper) on the road network. For dynamic insertion adjustment in one specified reference subset $V_{a,b}$, the new vertex V_N is modified by $R_{V_{a,b}}(V_N) = \min(Dist(V_N, P_i) + Dist(P_i, V_{a,b}), Dist(V_N, P_j) + Dist(P_j, V_{a,b}))$, where $Dist(P_i, V_{a,b})$ is distance between P_i and $V_{a,b}$, as presented in Fig. 2.

After the trajectory embedding, we can obtain t vectors $\in R^L$, where t is the number of trajectory points.

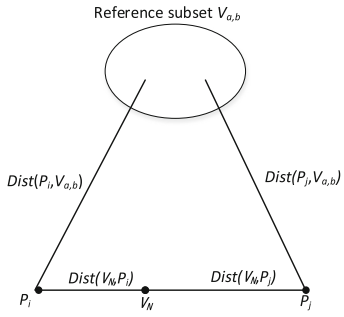

Fig. 2. Dynamic embedding for trajectory points

3.4 Trajectory Spatial Characteristics Captured by Convolutional Layer

Inspired by the successful applications of CNN on images, we also employ it to extract the GPS spatial characteristics. Compared with image having two-dimensions spatial structure, trajectory has only one-dimensional spatial structure like the sentence in word2vec classification model, where words display in sequence. Thus we use the one-dimension convolutions method to learn pixel-level spatial correlation features by considering the trajectory points as an image of width t and height 1.

As in Fig. 1, let $x_i \in R^L$ [9] be the L dimensional road network embedding vector corresponding to the i-th point in the trajectory sequence, the trajectory with length t is presented as:

$$x_{1:t} = x_1 \oplus x_2 \oplus \ldots \oplus x_t \tag{2}$$

where $\oplus$ thus the trajectory can be converted as a vector matrix, with the size $L * t$. A convolution operation involves a filter $w \in R^{L*k}$, whose kernel window size is k, and is overlaid across the GPS points vectors ranging from i to $i + k - 1$. Next, it performs an element-wise product, and then adds them together and obtains one new feature. For example, the transformation from a window of $x_{i:i+k-1}$ is defined as follows:

$$c_i = \delta(w_i \cdot x_{i:i+k-1} + b) \tag{3}$$

where w_i is a weight parameter and b is a bias and $\delta(\cdot)$ is a non-linear function. Thus, a feature vector is generated from one filter, which is successively applied to GPS points $\left(\mathbf{x}_{1:k}, \mathbf{x}_{2:k+1} \ldots x_{t-k+1:t}\right)$ where stride equals 1, with the index ranging from 1 to $t-k+1$. Finally, for each trajectory connected by c filters, we get the output vectors $C_{IN} \in R^{c*(t-k+1)}$ presented by Formula 4.

$$C_{IN} = [c_1, c_2, \ldots, c_{(t-k+1)}] \tag{4}$$

3.5 Trajectory Temporal Characteristics Captured by LSTM

A successful approach for solve the time sequential problems is RNN (Recurrent Neural Network), which can remember the previous historical sequence by using a transition function and leveraging an internal memory to process the dynamic temporal behavior. RNN has proven the ability to model variable length sequence. However, traditional RNN may also face the gradients exploding or vanishing because the time sequences is too long, thereby the LSTM is designed [20] since it can decide whether or not to abandon the previous hidden states depending on the time restrictions. Generally speaking, LSTM extends RNN by adding three gate (i.e., one input gate, one forget gate, one output gate) and a memory cell. The forget gate is employed to abandon some irrelevant information and can effectively solve the vanishing or exploding gradient problem. The input gate and output gate are utilized to control the input and output vectors. The output is the last hidden state of LSTM. At each time interval t, LSTM takes the output of convolutional layer as an input, and then all information is accumulated to the memory cell, each cell in LSTM is defined as follows:

$$f_t = \delta(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

$$i_t = \delta(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \tag{8}$$

$$o_t = \delta(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t \circ tanh(C_t) \tag{10}$$

where f_t, i_t, o_t represent the forget gate, input gate, and output gate, W_f, W_i, W_c, W_o are the weight parameters matrices, b_f, b_i, b_c, b_o are the biases values, $\delta(\cdot)$ denotes the nonlinear activation function, the *tanh* denotes the hyperbolic tangent function. Furthermore, the multi-layers LSTM is more efficient than a single LSTM layer [20].

Intuitively, the travel time can be affected by many complex external factors, such as weather conditions and time metadata (i.e., time-of-hour, day-of-week). For instance, traffic on rainy days is usually more congested than usual, and the road is more prone to have high-level crowd [3], etc. Note that these external factors cannot be directly fed into a neural network, we embed the weather conditions [4] as $X \in R^{16}$, time-of-hour as $X \in R^{24}$, day-of-week as $X \in R^{7}$, etc., by using the hot coding. Then the individual vectors are concatenated with the output of CNN, and fed into LSTM units.

Attention Mechanism Model: The traditional LSTM cannot detect which segment has a greater weight for estimating the whole trajectory time. For example, the impacts of expressway and speed limited road on the whole estimated time are different from each other. In order to address this issue, we design an attention mechanism [21] that can describe the key part of segments among whole trajectory segments. Let $Xext \in R^{e*(t-k+1)}$ represents the embedding of external factors, we append it to C_{IN} as LSTM input presented by Formula (11). Furthermore, Let $H \in R^{c\times(t-k+1)}$ be a matrix comprised of hidden vectors $[h_1, h_{2,\ldots,} h_{t-k+1}]$ that the LSTM produced in Formula (10). As in Figs. 1 and 3, the attention vector takes hidden vectors H and X*ext* as input to compute the probability distribution of source trajectory input. By utilizing this mechanism, it is possible for finally prediction to capture somewhat global information rather than solely to derive from hidden states. A vector including attention weights and a weighted hidden representation of GPS points are denoted as α, h^* respectively. The details are presented from Formula 12 to 14.

$$C_{IN} = W_{in} C_{IN} + W_{ext1} Xext \tag{11}$$

$$M = tanh(\begin{bmatrix} W_h H \\ W_{ext2} Xext \end{bmatrix}) \tag{12}$$

$$\alpha = softmax(\omega^T M) \tag{13}$$

$$h^* = H\alpha \tag{14}$$

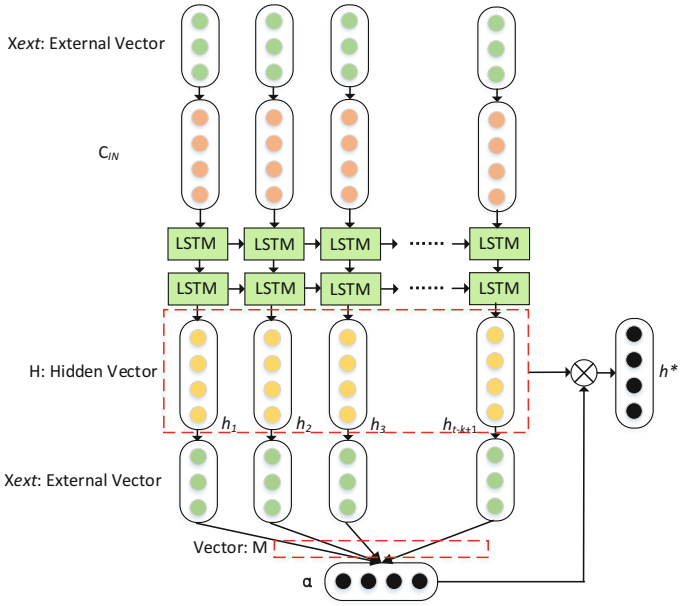

Fig. 3. The architecture of LSTM attention mechanism

where $W_{in} \in R^{c*c}$, $W_{ext1} \in R^{c*e}$, $W_h \in R^{c*c}$, $W_{ext2} \in R^{e*e}$, $\omega^T \in R^{1*(c+e)}$, $M \in R^{(c+e)*(t-k+1)}$, $\alpha \in R^{(t-k+1)}$, $h^* \in R^c$. Finally, the output of the temporal com ponent is h^*.

3.6 The Detail of Road Types Embedding

In fact, different road types have different effects on the travel time. For example, driving on the overpass is more time-consuming than on the highway at the same time interval. As in Fig. 4, a sample trajectory passes on the auxiliary road, urban expressway road, overpass road, urban expressway road sequentially, different parts of the trajectory exhibit diverse driving speeds marked out by ovals. For instance, the blue oval denotes the overpass road, whose sampling locations are close to each other, while the red oval denotes urban expressway road, whose sampling locations are far away from each other.

In summary, we propose a trajectory segmentation approach [22, 23]. Trajectory segmentation is a fundamental task which tries to partition a trajectory into several segments based on a set of optimization goals. We aim to find the characteristic points where the shape of a trajectory changes rapidly. The segmentation includes two desirable properties: preciseness and conciseness [23], we leverage the concept of Minimal Description Language (MDL) to find the optimal tradeoff between preciseness and conciseness.

The MDL is comprised of two components: $L(H)$ and $L(D|H)$. $L(H)$ is regarded as the hypothesis description with the length of the data described in bit, which measures

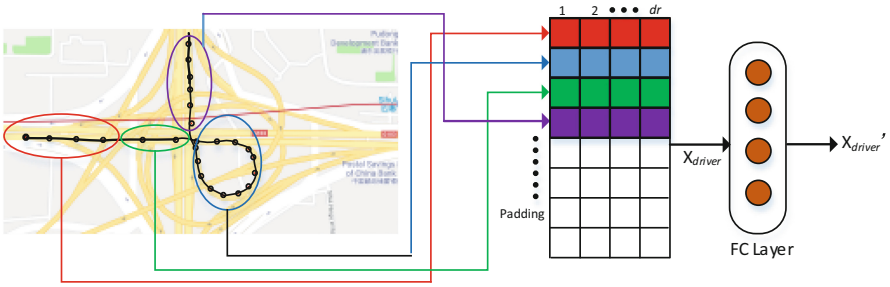

Fig. 4. The framework of road type component (Color figure online)

the degree of conciseness, while the $L(D|H)$ is regarded as the length of the description of data under the hypothesis H, which measures preciseness. In our paper, $L(H)$ is simply equal to $\log_2 x$. Furthermore, the $L(H)$ represents the total length of the Euclidean distance between all p_j and p_{j+1}, while the $L(D|H)$ represents the sum of the difference between a trajectory and its trajectory partitions. At last, a list of characteristic points is picked out by minimizing $L(H) + L(D|H)$. The trajectory is partitioned into segments by these characteristic points.

Then we obtain the average distance of every segment, and embed them as vector $X_{driver} \in R^{8*dr}$ groups padding whatever it is necessary. Finally, the X_{driver} is connected with FC layer to yield vector $X_{driver'} \in R^{dr}$, which concatenates LSMT to jointly train the model, thus the Formula 14 is rewritten as:

$$h^* = W_{h*}h^* + W_{driver}X_{driver'} \tag{15}$$

where $W_{h*} \in R^{c*c}$, $W_{driver} \in R^{c*dr}$.

3.7 Prediction Component

The next step is to estimate the travel time by integrating the output of LSTM and the output of road type component. We feed the $h*$ into the fully connected network to get the final estimated value $\tilde{Y}_t$, which is calculated as follows:

$$\tilde{Y}_t = \tanh(W_{of} \cdot h^* + b_{of}) \tag{16}$$

where W_{of} and b_{of} are learnable parameters, $tanh(x)$ is a hyperbolic tangent function, which ensures the output values are in $[-1 \sim 1]$. The loss function we used is defined by minimizing the mean squared error between the estimated time and the true time:

$$L(\theta) = \left|Y_t - \tilde{Y}_t\right|^2 \tag{17}$$

where θ is the set of all learnable parameters needed to be trained. We continuously adjust the parameter sets using back propagate by Tensorflow until loss function converges.

4 Experiments and Discussions

4.1 Dataset

Datasets: We test the method on the Beijing road network including about 330,000 vertices and 440,000 edges. We use two GPS trajectory datasets named **Taxi** and **Ucar** [24]. The **Taxi** data contains about 180,000 trajectories generated by more than 7,000 public taxis, **Ucar** data contains about 480,000 trajectories generated by more than 6,000 private taxis in November 2015. Each sampling point includes timestamp, latitude, longitude, vehicle speed, and direction. The abnormal records are first filtered out, and then the map matching algorithm is employed to relocate the deviated sample points. The data is divided into two subsets: we use the first 24 days data as the training data, the rest days as the test data.

Meteorological data: We record the Beijing weather data from Beijing Meteorological Bureau, the data include rainfall, temperature, wind velocity and so on. The weather conditions are divided into 14 types: cloudy, sunny, heavy rain, middle rain, light rain, heavy snow, light snow, dense fog, little fog, overcast, hail, frost, smog and haze, and sand storm.

4.2 Parameters Setting

Our model is implemented with Python 2.0. The model is deployed on the server with Core i7-4790 CPU, 16 GB RAM, NVIDIA GTX1080 GPU. We adopt Adam optimization algorithm with mini-batch size equals to 512 to train the parameters, the initial learning rate is set as 0.01.

4.3 Baseline Algorithms for Comparison

To demonstrate the validity of our model, we compare it with 5 baseline methods including:

ARIMA: ARIMA means Auto Regressive Integrated Moving Average, which is a typical and well known statistical model that depicts a suite of different standard temporal attributions.

XGBoost: XGBoost is an efficient, flexible machine learning technique for regression, classification and sorting tasks by assembling multiple weak learning under the gradient boosting framework, usually referring to decision trees. It belongs to ensemble learning.

SimpleTTE [7]: SimpleTTE presents a Simple Travel Time Estimating method that leverages the neighboring trips from the large amount of historical data. Being different from the traditional approaches, SimpleTTE indexes all the neighboring points with similar original and destination, and calculates the absolute temporal speed reference under irregularities traffic condition. After that, the travel time is scaled and estimated from the similar trips with the original demands.

Multi MASK [11]: Multi MASK is a multitasking representation learning model for time estimation, which does not hypothesize that the travel route is predetermined, but

utilizes the underlying road network with time and space prior knowledge. This method also engenders a meaningful representation that retains the various travel attributes.

DeepTTE [16]: DeepTTE is an end-to-end deep learning framework approach for estimating travel time of the whole path directly, the method presents a geo-convolution operation to capture the spatial correlations, and leverages stacking LSTM layers to capture the temporal dependencies as well. In addition, the relationships between the local and the global tradeoff are determined by the multitasking they presented. We apply this algorithm following the parameter settings deployed in [16].

4.4 Evaluation Metrics

We evaluate the performance of the proposed method based on three popular metrics. Assume $y_1, y_2, \ldots, y_n$ denote the ground truth, $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n$ denote the estimated value, and n denotes the numbers of samples points. Here, Mean Absolute Percentage Error (MAPE), Mean Relative Error (MRE), and Mean Absolute Error (MAE) are employed as evaluation metrics, their definitions are as follows:

$$MAPE = \sum_{t=1}^{n} \left| \frac{y_t - \tilde{y}_t}{y_t} \right| * \frac{1}{n} \tag{18}$$

$$MRE = \frac{\sum_{t=1}^{n} |y_t - \tilde{y}_t|}{\sum_{t=1}^{n} |y_t|} \tag{19}$$

$$MAE = \frac{\sum_{t=1}^{n} |y_t - \tilde{y}_t|}{n} \tag{20}$$

4.5 Performance Comparisons

Table 1 shows the comparisons between baseline algorithms and our presented method.

Table 1. Performance comparisons with baselines

Model	Taxi			Ucar		
	MAPE(%)	MRE(%)	MAE(s)	MAPE(%)	MRE(%)	MAE(s)
ARIMA	35.49	32.18	257	32.32	30.1	243
XGBoost	34.45	31.18	234	32.99	28.2	227
SimpleTTE	26.93	25.71	213	22.75	22.3	219
Multi MASK	23.35	22.63	207	21.53	19.45	186
DeepTTE	19.37	18.52	191	17.61	16.8	164
Ours STDR	**16.19**	**15.54**	**155**	**15.04**	**13.31**	**136**

From the Table 1, we can see that the MAPE, RMSE, and MAE of the ARIMA and XGboost perform poor results, which demonstrates that the simple traditional prediction method cannot effectively describe the large scale complex spatial-temporal data. The DeepTTE and Multi MASK both outperform ARIMA and XGBoost, the comparisons reveal that deep learning methods can work well. Furthermore, since the DeepTTP considers utilizing the convolutional operation to capture the spatial characteristic, it shows the better result than Multi Mask. Next, it is interesting the SimpleTTE method displays an accept medium performance between the traditional and deep learning methods. However, it is just an approximate method, which is more fit for the ideal situation, such as the highway or urban expressway with little speed changing. Finally, our algorithm significantly outperforms above mentioned methods with the lowest MAPE (16.19% and 15.04%), MRE (15.54% and 13.31%), and MAE (155 and 136) on two datasets respectively, which verifies the superiority and feasibility of our approach. The reason is that our algorithm further exploits LSTM attention mechanism and takes account of the influence of road type in the whole trajectory, these settings can better preserve the spatial-temporal characteristics of the original trajectory.

4.6 Comparison with Different Variants

To investigate the effectiveness of different components in Fig. 1, we compare our STDR with 4 different varieties including: (1) LSTM and road type without CNN component. (2) Only LSTM component, neither the CNN nor the road type is used. (3) CNN and LSTM components without road type. (4) LSTM without attention mechanism. All these models have identical inputs and the parameters are roughly the same. The results are presented in Table 2.

Table 2. Evaluation of our method and its variants

Model	Taxi			Ucar		
	MAPE (%)	MRE (%)	MAE (s)	MAPE (%)	MRE (%)	MAE (s)
①Only LSTM component	32.14	31.82	234	29.47	26.53	225
②LSTM + road type without CNN	29.25	26.72	207	25.78	24.38	183
③CNN + LSTM without road type	22.48	21.68	188	22.16	21.63	167
④STDR without attention	20.31	17.56	171	18.86	15.74	158
Our STDR	**16.19**	**15.54**	**155**	**15.04**	**13.31**	**136**

From Table 2, we have the following observations. Firstly, it is not surprising that only LSTM component exhibits the lowest performance as expected. Secondly, considering the ②LSTM + road type without CNN and the ③CNN + LSTM without road type, the impacts of CNN (MPAE is 22.48%) is more obvious than the road type (MAPE is 29.25%). The reason for this phenomenon is that some road type

characteristics can be also extracted by CNN in spatial component. Thirdly, both two variants (i.e., the ③CNN + LSTM without road type and ④STDR without attention) outperform the two variants without CNN (i.e., the ②LSTM + road type without CNN, and the ①only LSTM component), which demonstrates the significant role of CNN in spatial trajectory data mining. Fourthly, the ④STDR without attention is weaker than our STDR, the explanation is that the error of various types roads will continuously accumulate as the trajectory sequence grows, which confirms the effectiveness of attention mechanism. Finally, the performance of our STDR achieves the best when all aspects are considered.

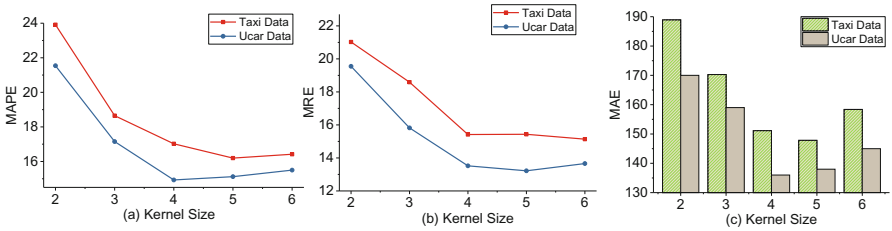

Fig. 5. Impacts of kernel size

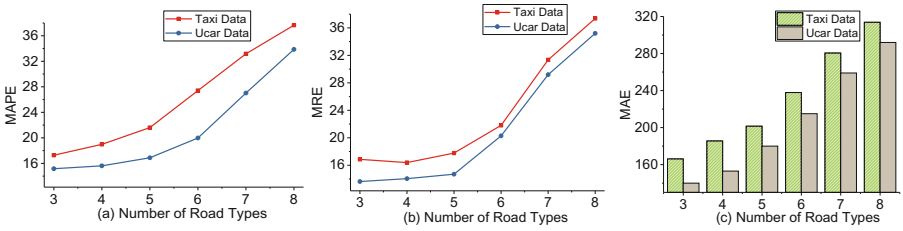

Fig. 6. Impacts of the number of road types

4.7 Impacts of Kernel Size and the Number of Road Types

We first evaluate the performance of kernel size of the convolutional filter. From Fig. 5, we observe that the MAPE of both data decreases as the kernel size grows, this discloses that the large kernel size can better capture the far away spatial dependences on the trajectory. However, when the kernel size exceeds 4, the effect becomes doubtful and the MAPE even rises. The cause is that although a larger filter can capture more information, it also imports much unnecessary noise, damaging the original road network correlation, such as two contradictory features generated by two successive turnings.

The impacts of the number of road types are exhibited in Fig. 6. As we can see, the results reveal that when the number of road type is smaller than 6, the corresponding average distance on the road is about 14 km according to historical trajectories statistics, the MAPE, MRE, and MAE are nearly unchanged, the reason may be related

to that the Beijing's layout of streets is in the shape of regular block layout. However, as the number of road types increases continually, the growth ratio rises dramatically, this indicates that more influencing factors produced by a long trajectory on the road can easily result in inaccurate predictions.

4.8 Impact of Weather Conditions

In this paper, we estimate the travel time by utilizing the weather forecasting as the approximate weather at future time interval $n + 1$. But in fact, weather forecasting is not fairly accurate all the time due to the technology and so on. To investigate the effectiveness of the weather component, first we remove it and compare it with STDR, then pick out five typical kinds weather, i.e., cloudy, sunny, rain, snow, and fog. The comparisons are shown in Table 3 and Fig. 7.

Table 3. Experimental results without weather conditions

	Taxi			Ucar		
Model	MAPE (%)	MRE (%)	MAE (s)	MAPE (%)	MRE (%)	MAE (s)
STDR without weather conditions	23.71	19.48	196	21.53	16.21	177
Our STDR	**16.19**	**15.54**	**155**	**15.04**	**13.31**	**136**

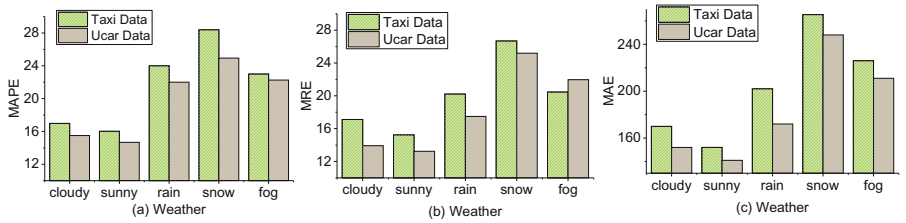

Fig. 7. Impact of weather conditions

In Table 3, we can see that the performance of STDR without weather conditions decrease by 31.7% comparing with our STDR (w.r.t. Taxi's MAPE), this indicates the weather has a significant influence on travel time. Next, from Fig. 7, we can discover that the MAPE on cloudy and sunny days are almost the same, suggesting that good weather condition has few impacts on travel time. On the rainy and foggy days, the results are relatively poor, and are generally worse than cloudy and sunny days, this is because bad weather can affect people's attention and result in slow response. The outcome on snowy days is the worst, the cause is that drivers need to be much longer waiting time at intersections due to the wet slippery road, this also conforms to our intuitive sense.

5 Conclusions

In this paper, we propose a novel deep learning end-to-end model based on CNN and LSTM for estimating travel time by using real historical traffic data. The method first embeds trajectory into low dimension vectors with the road network, then employs CNN to capture the spatial characteristic, further utilizes LSTM with attention mechanism to capture the time sequence characteristic. What's more, we import the road segmentation to fully depict the influence of road type. To validate the effectiveness of the proposed STDR, extensive experiments with 5 baselines are conducted. The results, in terms of MAPE, MRE, and MAE, demonstrate the superiority of our methodologies. In the future we plan to work on three interesting directions: (1) Incorporate the social network for the estimating travel time model. (2) Apply machine learning to interdisciplinary areas such as smart transportation and economics disciplines. (3) Extend and apply the framework to other trajectory problems.

Acknowledgements. This work was supported by NSFC(91646202), National Key R&D Program of China (SQ2018YFB140235), and the 1000-Talent program.

References

1. Yi, X., Zhang, J., Wang, Z., Li, T., Zheng, Y.: Deep distributed fusion network for air quality prediction. In: SIGKDD, pp. 965–973 (2018)
2. Zhang, S., Wu, G., Costeira, J.P., Moura, J.M.: FCN-rLSTM: deep spatio-temporal neural networks for vehicle counting in city cameras. In: ICCV, pp. 3687–3696 (2017)
3. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI, pp. 1655–1661 (2017)
4. Yao, H., et al.: Deep multi-view spatial-temporal network for taxi demand prediction. In: AAAI, pp. 2588–2595 (2018)
5. Amirian, P., Basiri, A., Morley, J.: Predictive analytics for enhancing travel time estimation in navigation apps of Apple, Google, and Microsoft. In: SIGSPATIAL, pp. 31–36 (2016)
6. Wang, Y., Zheng, Y., Xue, Y.: Travel time estimation of a path using sparse trajectories. In: SIGKDD, pp. 25–34 (2014)
7. Wang, H., Kuo, Y. H., Kifer, D., Li, Z.: A simple baseline for travel time estimation using large-scale trip data. In: SIGSPATIAL, pp. 61:1–61:4 (2016)
8. Wen, R., Yan, W., Zhang, A.N.: Adaptive spatio-temporal mining for route planning and travel time estimation. In: Big Data, pp. 3278–3284 (2017)
9. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)
10. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
11. Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y.: Multi-task representation learning for travel time estimation. In: KDD, pp. 1695–1704 (2018)
12. Wang, J., Gu, Q., Wu, J., Liu, G., Xiong, Z.: Traffic speed prediction and congestion source exploration: a deep learning method. In: ICDM, pp. 499–508 (2016)
13. Jindal, I., Chen, X., Nokleby, M., Ye, J.: A unified neural network approach for estimating travel time and distance for a taxi trip. In: CoRR, abs/1710.04350 (2017)

14. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio temporally correlated time series using Markov models. In: VLDB, pp. 769–780 (2013)
15. Jabari, S.E., Freris, N.M., Dilip, D.M.: Sparse travel time estimation from streaming data. In: CoRR, abs/1804.08130 (2018)
16. Wang, D., Zhang, J., Cao, W., Li, J., Zheng, Y.: When will you arrive? Estimating travel time based on deep neural networks. In: AAAI, pp. 2500–2507 (2018)
17. Zhang, H., Wu, H., Sun, W., Zheng, B.: DeepTravel: a neural network based travel time estimation model with auxiliary supervision. In: IJCAI, pp. 3655–3661 (2018)
18. Cui, Z., Ke, R., Wang, Y.: Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. In: arXiv, abs/1801.02143 (2018)
19. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. GeoInformatica **7**(3), 255–273 (2003)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
21. Wang, Y., Huang, M., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: EMNLP, pp. 606–615 (2016)
22. Zheng, Y.: Trajectory data mining: an overview. ACM TIST **6**(3), 29:1–29:41 (2015)
23. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD, pp. 593–604 (2007)
24. Ta, N., Li, G., Zhao, T., Feng, J., Ma, H., Gong, Z.: An efficient ride-sharing framework for maximizing shared route. TKDE **30**(2), 219–233 (2018)