Bus Travel-Time Prediction Based on Deep Spatio-Temporal Model

Kaixin Zhang[1], Yongxuan Lai[1(✉)], Liying Jiang[1], and Fan Yang[2]

[1] Shenzhen Research Institute/School of Informatics, Xiamen University, Xiamen, China
laozhng@icloud.com, laiyx@xmu.edu.cn, jiangliying@stu.xmu.edu.cn
[2] Department of Automation, Xiamen University, Xiamen, China
yang@xmu.edu.cn

Abstract. Bus travel time estimation in urban city is of great importance, which reduces passengers' waiting time and improves the quality of service of bus transportation. However, the travel time estimation is affected by various factors, including spatio-temporal dependencies (e.g. traffic conditions and road networks) and external factors (e.g. weather). Moreover, the bus dwelling and transit time are predominantly affected by different factors and hence have different patterns, with a fact that there are not so much study on how to divide the dwelling and transit areas and to build independent models for them. In this paper, we propose an end-to-end deep learning framework for Bus Travel Time Estimation (called DeepBTTE) where the target path is of arbitrary length. Two independent spatio-temporal components that use 1D-CNN and LSTM are adopted to estimate the dwelling time and transit time separately, which are then combined for the final estimation. We conduct experiments to evaluate our model using a real-world dataset. The experimental results show that our approach significantly outperforms other existing methods.

Keywords: Bus travel time estimation · Spatio-temporal model · Deep learning

1 Introduction

With the development of urban transportation and data sensing technologies, a large amount of data is created every day within transportation, so there is a trend of using data for the Intelligent Transportation Systems (ITS) [13,14] where travel time estimation plays a critical role [5]. Through travel time

This research is in part supported by the Natural Science Foundation of China (61672441, 61872154, 61862051), the Shenzhen Basic Research Program (JCYJ20170818141325209, JCYJ20190809161603551), Natural Science Foundation of Fujian (2018J01097), Special Fund for Basic Scientific Research Operation Fees of Central Universities (20720200031).

Z. Huang et al. (Eds.): WISE 2020, LNCS 12342, pp. 369–383, 2020.
https://doi.org/10.1007/978-3-030-62005-9_27

estimation, it becomes possible, to reduce passenger's waiting time and bring about various ITS applications. Among different transportation modes, the bus system has the largest coverage and the lowest overall cost. Hence, bus travel and arrival time estimation are of crucial importance for the efficiency and the quality of public transportation service.

The traditional methods for predicting travel time or arrival time are based on historical data. There are research that treated the traffic trajectory as time series data, and used the Autoregressive Integrated Moving Average (ARIMA) model to predict the travel time of vehicles on the highway [4]. And Wu et al. [22] used Support Vector Machine (SVM) to predict the travel time in next time step by the data from past time steps. These methods enable relatively accurate predictions of arrival time for taxis or private cars, but are not very effective for the buses. The bus runs on a fixed route and needs to stop at various bus stations. Therefore, the dwelling time of a station has a great influence on the total travel time. Moreover, the bus dwelling and transit time are affected by different factors and hence have different patterns. Using existing methods to directly predict the travel time similar to those of taxis or private cars may lead to inaccurate results. Actually, the estimation of dwelling time and transit time of buses could be viewed as two different tasks which need further optimization and integration, yet few studies have focused on how to divide the dwelling and transit areas and how to build independent models for them [18].

To address the limitations of previous work, in this paper we propose an end-to-end framework for Bus Travel Time Estimation based on 1D-CNN and LSTM, called DeepBTTE. The primary contributions are summarized as follows:

- We use a spatio-temporal component to learn the spatio-temporal dependencies from raw GPS sequences. The component consists of two parts: 1) a convolutional layer that transforms the GPS sequences to a series of feature maps and captures the spatial dependencies from consecutive GPS points directly; 2) a recurrent layer that learns the temporal dependencies of the obtained features and the embedded external features.
- We adopt an attribute component to embed the external factors, including temperature, weather condition, day of the week, distance of the path, vehicle conditions, the habits of drivers, etc. The latent representations are fed into the model to enhance the representation capability of these external factors.
- We propose a model for the bus travel time estimation which handles arbitrary length of bus trajectories. Given any two stations p_a and p_b in bus route, the model outputs the estimation of the cost of time from p_a to p_b. Two separated components are used to estimate the transit time and the dwelling time respectively.
- We conduct extensive experiments on real-world datasets to verify the performance of the proposed model. The mean absolute percentage error on real-world dataset is 6.82%, which significantly outperforms existing methods.

The remainder of this paper is organized as follows. Section 2 describes the related works. Section 3 introduces our problem and the data preprocessing pro-

cess. Section 4 shows three components of our model. Section 5 shows the experimental setup, and Sect. 6 compares the estimate performance between our model and other methods on a real-world dataset. Section 7 presents the conclusion and future work.

2 Related Work

There are a large number of studies on the travel time estimation. Various models, such as models based on historical data, statistical models, Kalman Filtering models and models based on Deep Learning have been proposed [1]. Wall et al. [20] presented a model to predict the travel time based on historical average. Chen et al. [7] used a dynamic model for bus travel time predicting. The model consists of two elements, an artificial neural network model and a Kalman Filter based model. Balasubramanian et al. [3] proposed a bus travel time prediction model which took the periodicity of data into consideration and it also used the real-time bus information to improve the prediction accuracy.

There are some researches on bus travel time prediction based on deep learning. The followings describe the travel time prediction methods for taxis or private cars used deep learning. Existing solutions could be divided into two categories, route-based methods and OD-based methods [19]. The former predicts the travel time with path route given and the latter predicts the travel time only with the origin and destination given.

Route-based methods [2,17] can be categorized as segment-based methods and sub-path-based methods. Segment-based methods treat a path as a sequence of independent road segments and then predict the travel time of each segment individually. Segment-based methods ignore the correlation between segments and it will accumulate the prediction error of each segment, which leads to inaccurate predictions. Different from segment-based methods, sub-path-based methods consider the correlation between different segments and treat a path as a set of sub-paths. For example, Zhang et al. [24] proposed a method called DEEPTRAVEL to predict the travel time based on neural network with auxiliary supervision, which used dual interval loss mechanism to optimize loss function both forward and backward. Duan et al. [8] and Liu et al. [16] predicted the travel time based on LSTM, and the results showed that LSTM has a great ability on capturing the temporal dependencies of temporal sequences. However, these two methods didn't consider the influence of weather and other external factors, so they are limited in improving the accuracy of travel time prediction.

Jindal et al. [11] proposed a model called ST-NN(Spatio-Temporal Neural Network) based on deep neural network, which can predict the travel time when only the origin and destination are given. Then the paper developed a carpool system based on reinforcement learning (RL). In ECML/PKDD Discovery Challenge 2015 competition, Lam et al. [15] estimated the travel time of taxis in real-time based on trip matching and ensemble learning without knowing the whole path. Wang et al. [21] presented a method called DeepTTE using Attribute Component, Spatio-Temporal Component and Multi-task Learning Component

to capture the external features and spatio-temporal correlations individually, and finally combining them to predict travel time. In this paper, we propose an end-to-end deep learning framework to learn the pattern of bus travel time. In this way, bus travel time can be effectively estimated.

3 Preliminaries and Data Preprocessing

In this section, we briefly introduce the bus travel time prediction problem and the data preprocessing process.

3.1 Bus Travel Time Prediction Problem

The travel time of a bus are composed of the dwelling time at stations and transit time between stations. Therefore, we defined some concepts as follows.

Historical Trajectory. The trajectory of one bus from station i to station j is defined as $T = \{p_i, ..., p_j\}$, $p_i = \{p_i.lat, p_i.lng, p_i.in, p_i.out\}$, containing the latitude, longitude, inbound time and outbound time of station i. Moreover, for each trajectory T, we add $weather$, $temperature$, $carID$, $driverID$, $timeslotID$ and $weekday$ as external features.

Dwelling Time. The dwelling time of station i is defined as

$$p_i.dt = p_i.out - p_i.in \tag{1}$$

Transit Time. The transit time of station $i-1$ to station i is defined as

$$p_i.tt = p_i.in - p_{i-1}.out \tag{2}$$

Arrival Time. The arrival time of station i is defined as

$$p_i.at = p_i.dt + p_i.tt \tag{3}$$

Previous Dwelling Time. The previous dwelling time of station i is the dwelling time of previous time slot (length of time slot is 10 min) at station i, which is written as

$$p_i^t.pdt = p_i^{t-1}.dt \tag{4}$$

The notations and their meanings are shown in Table 1. We first construct deep models on the estimation of the transit time and dwelling time of the given path and the corresponding external factors, based on the spatio-temporal features extracted from trajectories. Then given a departure time and any two stations p_a and p_b in bus route, the model outputs the estimation of the cost of time from p_a to p_b. We assume that the travel path from p_a to p_b is the sub-path of bus route with arbitrary path length.

Table 1. Notations and their meanings

Notation	Meaning	Notation	Meaning
T	A trajectory of a bus	p_i	Station i
$p_i.lat$	The latitude of station i	$p_i.lng$	The longitude of station i
$p_i.in$	Bus arrival time at station i	$p_i.out$	Bus departure time at station i
$carID$	Identity of the vehicle	$driverID$	Identity of the driver
$timeslotID$	The time slot of day	$weekday$	The day of week
$weather$	Weather information	$p_i.tt$	Transit time of station
$p_i.dt$	Dwelling time of station i	$p_i.at$	Arrival time of station i
$p_i.pdt$	Previous dwelling time of station i		

3.2 Data Preprocessing Process

Trajectory Data Processing. The raw bus-to-departure data includes all inbound and outbound information. Before performing feature extraction, we need to drop duplicate and invalid data. Then, group the data by date, direction(uplink or downlink) and $carID$. For each trajectory, the dwelling time and transit time of each station are calculated. We fill in the dwelling time and transit time according to two situations. In weekdays, we populate them with weekdays' average value. And on weekends, we fill them with weekends' average value. If more than 50% station data of a trajectory is lost, then drop this trajectory.

External Feature Processing. External features include weather condition, $driverID$, $carID$, $timeslotID$ and $weekday$. We fill the weather data with the latest hour data if there is any missing data. And we choose temperature and weather description as two features from weather information which have great impact on bus travel time. We also associate each trajectory with a driver according to the schedule data.

4 Model Description

As shown in Fig. 1, this model consists of three components. The attribute component is used to process the external factors (e.g. weather) and basic information (e.g. start time). The output of the attribute component will be part of the input of other components. The spatio-temporal component is the main component that learns the spatial correlations and temporal dependencies from the GPS sequences. Finally, the fusion estimation component forecasts the bus travel time of a given path based on the previous two components.

4.1 Attribute Component

The travel time of a bus is affected by many factors. For instance, in the peak hour there are many vehicles on the road and numerous passengers waiting at the station, so buses have to spend more time to finish a route. But in non-peak hour, the travel time is shorter. So the travel time is time-varying. Furthermore,

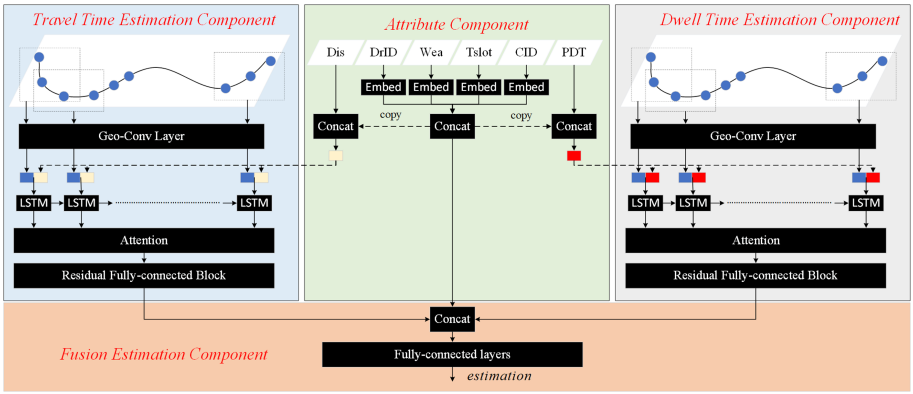

Fig. 1. Overall Framework. Dis denotes distance, DrID denotes driverID, Wea denotes weather, Tslot denotes time slot, CID denotes carID, PDT denotes previous dwelling time, Concat denotes concatenation operation.

the travel time is also affected by many external factors, like weather, driver habit, bus running information and day of week. We use a simple and efficient component to incorporate such factors into our model, which we call attribute component.

As shown in Fig. 1, we use weather, driverID, carID, weekday to represent external factors respectively, the timeslot represents the start time. These attributes are categorical values that can not be fed into the neural network directly. So we need to use the embedding method [9] to transform each categorical attribute into the low-dimensional real vector. More specifically, the embedding method maps each categorical value $x \in [V]$ to a real space $R^{E\times 1}$, where V represents the vocabulary size of categories value and E represents the dimension of embedding space. Usually, $E << V$. The embedding method effectively reduces the input dimensions and thus reduces the computational complexity.

Besides the embedded attributes, other important attributes (e.g. travel distance and dwelling time of the previous time slot) are incorporated. We use $\Delta d_{p_a} \rightarrow d_{p_b}$ to denote the total distance traveling from station p_a to p_b along the path, i.e., $\Delta d_{p_a} \rightarrow d_{p_b} = \sum_{i=a}^{b-1} Dis\,(p_i, p_{i+1})$ where Dis is the geographical distance between two bus stations. Similarly, we use $\Delta t_{p_a} \rightarrow t_{p_b}$ to denote the total dwelling time of previous time slot between p_a and p_b, i.e., $\Delta t_{p_a} \rightarrow t_{p_b} = \sum_{i=a}^{b-1} Time\,(p_i, p_{i+1})$ where $Time$ is the total dwelling time of previous time slot between two bus stations. Then, we concatenate the obtained embedded vector together with $\Delta d_{p_1} \rightarrow d_{|T|}$ and $\Delta t_{p_1} \rightarrow t_{|T|}$, respectively. We denote the concatenation as the output of attribute component.

4.2 Spatio-Temporal Component

The spatio-temporal component has two sub-components. Dwelling time estimation component is used to predict the total dwelling time of each station, and the transit time estimation component is used to predict the total transit time between stations. These two components have the same structure, they are composed of two parts. The first part is a geo-convolutional neural network which transforms raw station GPS sequences to a series of feature maps. The second part is the recurrent neural network which learns the temporal correlations of obtained feature maps from the previous part.

Geo-Conv Layer. As we mentioned before, a historical trajectory T is a sequence of GPS points $\{p_1, ..., p_{|T|}\}$ where each p_i represents one bus station which contains the corresponding longitude, latitude, arrival and departure time. Extracting spatial dependencies from GPS sequences is critical to dwelling time and transit time estimation. The convolutional neural network has been widely used for capturing spatial relationships. A typical convolutional network consists of multiple convolutional filters and the filters learn the spatial relationships in the input by applying the convolution operation. Since the spatial dependency cannot be obtained directly from GPS coordinates, Zhang et al. [25] found a way to transform the GPS coordinates to capture the spatial feature. They first partitioned the area where all GPS records are located into $I \times J$ grids and then map each GPS coordinate into a grid cell, and finally, they used 2D-CNN to extract the spatial correlation from the number of the grid cell. However, in our case, directly mapping the GPS coordinates into grid cells is unsuitable. Firstly, the GPS coordinates of each bus station are fixed and limited, and they don't need to be mapped. Secondly, if two stations are mapped to the same cell we can't distinguish them. Thus, we use a Geo-Conv layer which can capture the spatial dependency in geo-location sequences while retains the location information.

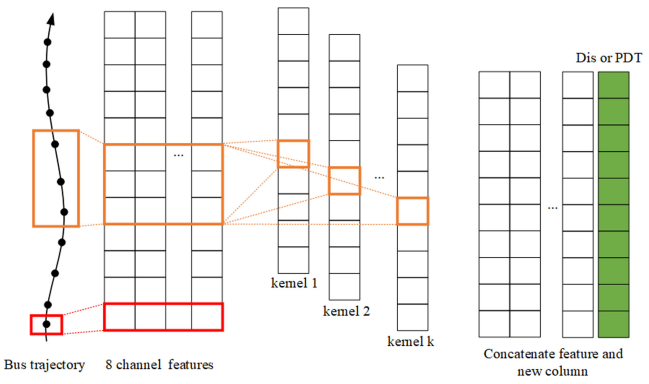

Fig. 2. Illustration of Structure of the Geo-Conv Layer. Dis denotes distance, PDT denotes previous dwelling time.

The architecture of Geo-Conv layer is shown in Fig. 2. We first use a non-linear mapping to transform each station GPS point p_i in the sequence into vector. $loc_i \in R^8$.

$$loc_i = tanh\left(W_{loc} \cdot [p_i.lat \circ p_i.lng]\right) \tag{5}$$

where $\circ$ denotes the concatenate operation and W_{loc} is a learnable weight matrix. Thus, the output sequence $loc \in R^{8\times|T|}$ indicates the non-linear mapping for each GPS point. loc_i can be seen as an 8-channel input. Each channel represents the geographical features of the original GPS sequence. We use a 1D-CNN [21], with parameter matrix $W_{conv} \in R^{k\times 8}$, to extract spatial features where k is the kernel size of a filter. It applies the convolutional operation on the sequence loc, along with a one-dimensional sliding window. The i-th dimension of its output is denoted as:

$$loc_i^{conv} = \sigma\left(W_{conv} * loc_{i:i+k-1} + b\right) \tag{6}$$

where $*$ denotes the convolutional operation, b is the bias term, $loc_{i:i+k-1}$ is the subsequence of loc from i to $i+k-1$ and the σ is an activation function.

The sub-sequence from station p_i to p_{i+k-1} is defined as the i-th local path. The loc_i^{conv} represents the spatial feature of the i-th local path. We get the feature map of local paths with shape $R^{c\times(|T|-k+1)}$, where c is the number of filters.

However, in our task, the transit time and dwelling time are highly correlated with the total distance of the path and the total dwelling time of previous time slot, respectively. We can't extract the geometric distance and previous dwelling time from latitude/longitude through 1D-CNN. Therefore, we further append a column to the previously obtained feature map in Geo-Conv layer. As shown in Fig. 2, in transit time estimation component, the i-th element of new appended column is the distance of the i-th local path, i.e., $\sum_{j=i+1}^{j+k-1} Dis\left(p_{j-1}, p_j\right)$. Similarly, in dwelling time estimation component, the i-th element of new column is the previous dwelling time of the i-th local path, i.e, $\sum_{j=i+1}^{j+k-1} Time\left(p_{j-1}, p_j\right)$. Finally, we obtain the feature map of shape $R^{(c+1)\times(|T|-k+1)}$ by Geo-Conv layer. These feature maps are denoted as loc^d and loc^t.

Recurrent Layer. The feature maps loc^d and loc^t capture the spatial dependencies from all local paths. To further capture temporal dependencies, we introduce the recurrent layers. The recurrent neural network (RNN) is widely used for capturing the temporal dependency in sequential learning. The RNN can 'memorize' the history in the processed sequence. When processing sequences in each time step, it updates its memory (i.e., hidden state) according to the current input and the hidden state of the previous step. The output of the RNN is the hidden state sequence of the input sequence at all time steps.

In both sub-components of spatio-temporal component, the feature map (i.e., loc^d and loc^t) by the Geo-Conv layer is set as the input of the recurrent neural networks. As mentioned before, the length of feature map is $|T|-k+1$. To further improve the estimating ability of recurrent layers, we incorporate the attributes'

information (i.e., $attr^d$ and $attr^t$) obtained from the attribute component. A simple updating rule of the RNN can be expressed as:

$$h_i = \sigma\left(W_x \cdot loc_i^f + W_h \cdot h_{i-1} + W_a \cdot attr\right) \tag{7}$$

where h_i represents the hidden state after the processing of the i-th local path and σ_{rnn} is the activation function. W_x, W_h and W_a are learnable parameters' matrix of spatial features. However, the RNN usually fails when processing the long sequence due to the vanishing gradient and exploding gradient problem [10]. To overcome this issue, we use Long Short-term Memory [10] layer to replace it. Compared with RNN, the LSTM uses the input gate and forget gate to control the in/out information flow. Such gates enable LSTM to retain important information and filter out the unimportant information which effectively mitigates the gradient vanishing/exploding problem.

Now, we obtain the spatio-temporal sequence $\left\{h_1, h_2, ..., h_{|T|-k+1}\right\}$ from raw data by utilizing the Geo-Conv layer and the recurrent layer.

Estimation. Since the length of each bus trajectory is variable, the length of spatio-temporal feature sequence $\{h_i\}$ is also variable. To estimate the transit time/dwelling time of the entire path directly, we need an attention model that can calculate sets of hidden states of different sizes. The attention mechanism is essentially the weighted sum of sequence $\{h_i\}$, where the weights are parameters learned by model. Thus, we have that:

$$h_{att} = \sum_{i=1}^{|T|-k+1} \alpha_i \cdot h_i \tag{8}$$

where α_i is the weight for the i-th local path, and the summation of all weights equals to 1. In our case, the weight of each local path is related to the spatial information of the local paths, as well as external features such as start time slot, the weekday and the weather condition. The attention mechanism can be expressed as:

$$\begin{aligned} z_i &= \langle \sigma_{att}\left(attr\right), h_i \rangle \\ \alpha_i &= \frac{e^{z_i}}{\sum_j e^{z_j}} \end{aligned} \tag{9}$$

where $attr$ represents the external factors captured by attribute component, $\{h_i\}$ represents the spatial-temporal feature of local path, $\langle \cdot \rangle$ represents the inner operator and the σ_{attr} is a non-linear mapping which maps $attr$ to a vector with the same length as h_i. Substituting Eq. (9) into Eq. (8), we obtain the attention based vector h_{attr}.

Finally, we pass h_{attr} to several Residual fully-connected layers with equal size. In our model, we use σ_{f_i} to denote the i-th layer. For the first layer, the output of this layer is $\sigma_{f_1}(x)$. For the rest of the residual fully-connected layers, we use x to denote the output of i-th layer. Then, the output of the $(i+1)$-th express as $x \oplus \sigma_{f_{i+1}}(x)$ where the $\oplus$ is the element-wise add operation. At the last layer, we use a single neuron to obtain the estimation.

4.3 Fusion Estimation Component

In fusion estimation component, we combine the previous components and estimate the total arrival time of the input path:

$$total = \hat{t_{transit}} \circ \hat{t_{dwell}} \circ attr \tag{10}$$

where *total* represents the input of fusion estimation component, $\hat{t_{transit}}$ represents the estimation of transit time, $\hat{t_{dwell}}$ represents the estimation of dwelling time, and *attr* represents the embedded vector of external features. We pass *total* to several fully-connected layers. At the last layer, we use a single neuron to obtain the estimation.

5 Experimental Setup

5.1 Data Description

We evaluated our model on a real-world dataset. This experiment is based on the bus-to-departure data and the corresponding bus schedule data of Xiamen 22 bus from September 1, 2018, to February 28, 2019. The bus-to-departure data contains 301,130 records. After data preprocessing, as mentioned in Data Preprocessing Process, we obtain 9523 trajectories information. The route map of 22 bus is shown in Fig.3, where the total route length is about 14 km. Due to the departure interval of 22 bus is 10 min, we set the length of time slot as 10 min. Each trajectory information contains the longitude, latitude, dwelling time, transit time of each station. Finally, we intercept arbitrary-length paths from complete trajectory information. The shortest trajectory contains 10 stations and the longest trajectory contains 25 stations (Number of bus station on Route 22).

The trajectories in Xiamen dataset are associated with the corresponding weekday, timeslot, driverID, and carID.

5.2 Parameter Setting

The parameters we used in our experiment are described as follows:

- In attribute component, we embed driverID to R^3, carID to R^3, weekID to R^3, timeslotID to R^4, temperature to R^3 and corresponding weather condition to R^3.
- We set the same parameters of Geo-conv for both sub-components of spatio-temporal component. The number of filters $c = 32$ and we use the ELU as the activation function. The ELU is defined as $ELU(x) = e^x - 1$ for $x \leq 0$ and $ELU(X) = x$ for $x > 0$. For kernel size k, which represents the length of local path, we set the size of k to 3 to evaluate our model.
- We set the same parameters of the recurrent layer for both sub-components. We use *tanh* as the activation function and set the size of the hidden vector as 128.

Fig. 3. Route of No. 22 Bus in Xiamen City.

- In fusion estimation component, the ReLU function is as the activation function of the fully-connected layers. We set the number of the fully-connected layers as 2 and the size of each layer as $[64, 32]$.

In the Xiamen dataset, we divided the trajectories generated in February 2019 into two parts, one for evaluation and the other for test. Except for the trajectories generated in February 2019, the rest of trajectories are used as the training set. We adopt an Adam optimization algorithm to train the parameters. The learning rate of Adam is le^{-4}, the batch size during training is 64 and the epochs of training are 50.

Our model is implemented with PyTorch 1.1.0, a widely used Deep Learning Python library. We train/test our model on the server with one NVIDIA TITAN X GPU.

6 Results and Analyses

6.1 Performance Comparison

Because there is little research on bus arrival time estimation, to demonstrate the strength of our model, we use Root Mean Square Error (RMSE) [6], Mean Absolute Error (MAE) [6], and Mean Absolute Percentage Error (MAPE) [12] as metrics to compare our model with other baseline methods, including:

- **AVG:** AVG simply calculates the average transit time and dwelling time of each station on the training set, and estimates the arrival time of given trajectory based on the average transit time/dwelling time calculated on training set.

- **GBDT:** Gradient Boosting Decision Tree (GBDT) [23] is a powerful ensemble method. In our case, the input of GBDT is the same as the input of our model, including all the inputs of attribute component and the raw GPS sequences. We use a GBDT to estimate the arrival time. Because of the lengths of GPS sequences are variable, GBDT can't handle the sequences directly. Thus, we transform each GPS sequence to fixed length of 25.
- **MLPBTTE:** Multi Layer perception, a 5-layer Perception is used to estimate the time of arrival. The activation function of Mlp is ReLU, the input of Mlp is the concatenation of GPS sequences and embedded external features. The size of hidden layers in Mlp is fixed as 128.
- **Single Spatio-Temporal Model:** In the case of not estimating the dwelling time and transit time separately, a single spatio-temporal component is used to estimate the bus arrival time. We concatenate the output of spatio-temporal component and attribute component and the concatenation is passed to a residual fully-connected layer to obtain the estimation.
- **RnnBTTE:** RnnBTTE is a simplified model of our model. We replace the LSTM in the recurrent layer with RNN and use mean pooling to process the output of the recurrent layer into a 128-dimensional feature vector. We concatenate the feature vector and the output of attribute component. Finally, the concatenation is passed to the fusion estimation component to obtain the estimation.

Table 2. Performance comparison

	RMSE	*MAE*	*MAPE*
AVG	602.3859	423.2324	25.7433
GBDT	506.1589	389.7922	21.6339
MlpBTTE	281.6376	200.4457	10.3773
RnnBTTE	195.2742	144.9747	7.5667
DeepBTTE	**172.5082**	**127.8042**	**6.8258**
Single Spatio-Temporal Model	203.4458	147.6104	7.7000

As shown in Table 2, simply using the average transit time and dwelling time of each station leads to a very inaccurate result. The ensemble method GBDT is much better than AVG. MlpBTTE shows a better performance than AVG and GBDT. However, it does not consider the spatio-temporal dependencies in data. The RnnBTTE uses the RNN in recurrent layers to capture the spatio-temporal dependencies in data. It achieves 7.57% on Xiamen dataset which is much better than above mentioned methods. Our DeepBTTE model further significantly outperforms RnnBTTE and other methods. The mean absolute percentage error of Xiamen dataset is only 6.82%.

6.2 Effect of Separate Estimation

We use single spatio-temporal model to estimate bus arrival time. Compared with DeepBTTE, single spatio-temporal model makes an overall estimation of arrival time. As shown in Table 2, Our DeepBTTE model outperforms single spatio-temporal model. More specifically, the bus dwelling time and transit time are affected by different factors and hence have different patterns. In our model, we use two independent components to estimate dwelling and transit time separately which can enhance the estimation performance.

6.3 Effect of External Factors

To evaluate the effectiveness of different external factors, We devise a series of controlled experiments on Xiamen dataset. For each experiment, we eliminate one attribute. As shown in Table 3, we find that weekdays affect the estimation significantly. This conforms to reality that the traffic condition of 22 bus line is completely different on weekdays and weekends. The average arrival time on weekends is longer than that of weekdays. Eliminating the weather causes an error growth of 0.47%. This also confirms to reality that the average arrival time will be longer under bad weather condition. Eliminating the driver and vehicle information have little effect on the results.

Table 3. Effect of different attributes

	RMSE	*MAE*	*MAPE*
Eliminate carID	175.0516	131.7047	6.9885
Eliminate driverID	173.0027	128.1601	6.8483
Eliminate weekday	194.3426	145.1981	7.5775
Eliminate weather	181.0990	136.6376	7.3041
Without eliminate	**172.5082**	**127.8042**	**6.8258**

6.4 Training and Predicting Time

Table 4 shows the training and predicting time of each method (i.e., AVG, GBDT, MlpBTTE, RnnBTTE and DeepBTTE), the GPU we used to train our model is a TITAN X. Despite the training time of RnnBTTE and DeepBTTE are longer, it is acceptable in offline training. In practical applications, we can obtain a lot of offline resources for pre-training. During the test phase, estimating about 1000 paths takes DeepBTTE 0.0015 s, RnnBTTE 0.0018 s, MlpBTTE 0.0005 s on GPU.

Table 4. Comparative training time and testing time

	Training time (*s*)	*Predicting time* (*s*)
GBDT	3505	0.0132
MlpBTTE	161	0.0005
RnnBTTE	306	0.0018
DeepBTTE	355	0.0015
Single Spatio-Temporal Model	231	0.0016

7 Conclusion and Future Work

In this paper, we study the problem of estimating bus travel time for any given path in one route. We propose an end-to-end framework based on 1D-CNN and LSTM network. This model can effectively capture the spatio-temporal dependencies from raw GPS sequences. Our model also considers various external factors such as weather condition and time which may affect the arrival time. We evaluate our model on a real-world dataset. The results show that our method achieves a high estimation accuracy and outperforms the other methods significantly. For future work, we will enhance the performance of our model and add additional data as model inputs, such as GPS points of intersections and traffic lights and the number of passengers getting on and off at each station.

References

1. Altinkaya, M., Zontul, M.: Urban bus arrival time prediction: a review of computational models. Int. J. Recent Technol. Eng. (IJRTE) **2**(4), 164–169 (2013)
2. Asif, M.T., et al.: Spatiotemporal patterns in large-scale traffic speed prediction. IEEE Trans. Intell. Transp. Syst. **15**(2), 794–804 (2013)
3. Balasubramanian, P., Rao, K.R.: An adaptive long-term bus arrival time prediction model with cyclic variations. J. Public Transp. **18**(1), 6 (2015)
4. Billings, D., Yang, J.S.: Application of the ARIMA models to urban roadway travel time prediction - a case study. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006 (2006)
5. Birr, K., Jamroz, K., Kustra, W.: Travel time of public transport vehicles estimation. Transp. Res. Proc. **3**, 359–365 (2014)
6. Chai, T., Draxler, R.R.: Root mean square error (RMSE) or mean absolute error (MAE)? - arguments against avoiding RMSE in the literature. Geosci. Model Dev. **7**(3), 1247–1250 (2014)
7. Chen, M., Liu, X., Xia, J., Chien, S.I.: A dynamic bus-arrival time prediction model based on APC data. Comput.-Aided Civ. Infrastruct. Eng. **19**(5), 364–376 (2004)
8. Duan, Y., Lv, Y., Wang, F.Y.: Travel time prediction with LSTM neural network. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1053–1058. IEEE (2016)
9. Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1019–1027 (2016)

10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Jindal, I., Qin, Z.T., Chen, X., Nokleby, M., Ye, J.: Optimizing taxi carpool policies via reinforcement learning and spatio-temporal mining. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 1417–1426. IEEE (2018)
12. Khair, U., Fahmi, H., Hakim, S.A., Rahim, R.: Forecasting error calculation with mean absolute deviation and mean absolute percentage error. J. Phys.: Conf. Ser. **930**, 012002 (2017)
13. Lai, Y., Lv, Z., Li, K.C., Liao, M.: Urban traffic coulomb's law: a new approach for taxi route recommendation. IEEE Trans. Intell. Transp. Syst. **20**, 3024–3037 (2018)
14. Lai, Y., Yang, F., Zhang, L., Lin, Z.: Distributed public vehicle system based on fog nodes and vehicular sensing. IEEE Access **6**, 22011–22024 (2018)
15. Lam, H.T., Diaz-Aviles, E., Pascale, A., Gkoufas, Y., Chen, B.: (Blue) taxi destination and trip time prediction from partial trajectories. arXiv preprint arXiv:1509.05257 (2015)
16. Liu, Y., Wang, Y., Yang, X., Zhang, L.: Short-term travel time prediction by deep learning: a comparison of different LSTM-DNN models. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–8. IEEE (2017)
17. Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y.: Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. **16**(2), 865–873 (2014)
18. Ma, J., Chan, J., Ristanoski, G., Rajasegarar, S., Leckie, C.: Bus travel time prediction with real-time traffic information. Transp. Res. Part C: Emerg. Technol. **105**, 536–549 (2019)
19. Veres, M., Moussa, M.: Deep learning for intelligent transportation systems: a survey of emerging trends. IEEE Trans. Intell. Transp. Syst. **21**, 3152–3168 (2019)
20. Wall, Z., Dailey, D.: An algorithm for predicting the arrival time of mass transit vehicles using automatic vehicle location data. In: 78th Annual Meeting of the Transportation Research Board, pp. 1–11. Citeseer (1999)
21. Wang, D., Zhang, J., Cao, W., Li, J., Zheng, Y.: When will you arrive? Estimating travel time based on deep neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
22. Wu, C.H., Ho, J.M., Lee, D.T.: Travel time prediction with support vector regression. Intell. Transp. Syst. **5**, 276–281 (2003)
23. Lai, Y., Yang, X., Cao, Q., Cao, H., Wang, T., Yang, F.: A bus running length prediction method based on gradient boosting. Big Data Res. **5**, 58–78 (2019)
24. Zhang, H., Wu, H., Sun, W., Zheng, B.: DeepTravel: a neural network based travel time estimation model with auxiliary supervision. arXiv preprint arXiv:1802.02147 (2018)
25. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X.: DNN-based prediction model for spatio-temporal data. In: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 92. ACM (2016)