DEEPTRAVEL: a Neural Network Based Travel Time Estimation Model with Auxiliary Supervision

Hanyuan Zhang†, Hao Wu†, Weiwei Sun†, Baihua Zheng‡
† School of Computer Science, Shanghai Key Laboratory of Data Science, Fudan University, China
‡ Shanghai Institute of Intelligent Electroics & Systems, Shanghai, China
† Singapore Management University, Singapore
† {hanyuanzhang16,wuhao5688,wwsun}@fudan.edu.cn, ‡ bhzheng@smu.edu.sg

Abstract

Estimating the travel time of a path is of great importance to smart urban mobility. Existing approaches are either based on estimating the time cost of each road segment or designed heuristically in a non-learning-based way. The former is not able to capture many cross-segment complex factors while the latter fails to utilize the existing abundant temporal labels of the data, i.e., the time stamp of each trajectory point. In this paper, we leverage on new development of deep neural networks and propose a novel auxiliary supervision model, namely DEEPTRAVEL, that can automatically and effectively extract different features, as well as make full use of temporal labels of the trajectory data. We have conducted comprehensive experiments on real datasets to demonstrate the out-performance of DEEPTRAVEL over existing approaches.

1 Introduction

The advances in GPS-enabled mobile devices and pervasive computing techniques have generated massive trajectory data. The large amount of trajectory data provide opportunities to further enhance urban transportation systems. The estimated travel time of a path at a certain time is an essential piece of information commuters desire to have. However, it is challenging to perform an accurate estimation as the travel time is affected by many dynamics, such as the dynamics of the traffic, the dynamics at the crossroads, the dynamics of the driving behavior and the dynamics of the travel time of same paths in the historical data.

Existing solutions mainly adopt divide-and-conquer approach to perform the estimation by decomposing a path into a sequence of segments or sub-paths. Segment-based approaches [De Fabritiis et al., 2008; Asif et al., 2014; Lv et al., 2015] estimate the travel time of each road segment individually, but ignore the additional time spent at the intersection of segments due to traffic lights and turns. Moreover, they rely on high-quality travel speed estimations/measurements, that might not be always available. Sub-path based approaches [Rahmani et al., 2013; Wang et al., 2014] try to estimate the time of the whole path by extracting the time consumption of sub-paths appeared in the historical dataset. As they can eliminate some errors accumulated by segment-based approaches, they are able to achieve a higher accuracy. However, they are still designed in an empirical and heuristic way but not training-based, which leaves the room for further improvement. In summary, existing estimation approaches could not achieve excellent accuracy because they fail to consider the path as a whole and they do not fully leverage the natural supervised labels of the data, i.e., the time stamp of each GPS sampling point that is easy to collect.

On the other hand, thanks to the recent boom of deep learning techniques, more problems can be solved by end-to-end models which significantly outperform the traditional heuristic approaches. Moreover, deep learning models have a strong representation power which enables the capturing of more latent features and the modeling of such complicated dynamics in travel time estimation problem. Motivated by this, we propose a deep model named DEEPTRAVEL which can learn directly from historical trajectories to estimate the travel time. DEEPTRAVEL considers the characteristics of trajectory data by adopting a new loss function for auxiliary supervision and is able to extract multiple features that affect the travel time.

In brief, we make three main contributions in this paper. First, we propose DEEPTRAVEL, an end-to-end training-based model which can learn from the historical dataset to predict the travel time of a whole path directly. We introduce a dual interval loss to fully leverage the temporal labels of the trajectory data which works as an auxiliary supervision. Second, we propose a feature extraction structure to extract features including spatial and temporal embeddings, driving state features, short-term and long-term traffic features. This structure can effectively capture different dynamics for estimating the travel time accurately. Last but not least, we conduct comprehensive experiments to evaluate our model with two real datasets. The results demonstrate the advantage of our model over the state-of-the-art competitors.

2 Related Work

As stated in Section 1, existing approaches on estimating the path travel time could be categorized into two clusters, segment-based approaches and sub-path-based approaches.
The former estimates the travel time of each individual road segment in the network via different methods, e.g., the loop detectors [Jia et al., 2001; Rice and Van Zwet, 2004], support vector regression [Asif et al., 2014] and stacked autoencoder [Lv et al., 2015]. Given a path, segment-based approaches treat it as a sequence of individual segments and the estimation could not achieve a high accuracy as they do not consider the interaction between road segments. In addition, they heavily rely on high quality travel speed data of each segment which might not be always available.

Inspired by the inaccuracy of road segment-based methods, sub-paths based approaches consider sub-paths instead of single segments as a way to include the interaction between road segments into the estimation. For example, [Han et al., 2011; Luo et al., 2013] mine frequent trajectory patterns; [Rahmani et al., 2013] introduces a non-parametric method and utilizes the travel time of common sub-paths between the query path and historical paths to estimate the travel time of the whole path after incorporating a list of potential biases corrections; [Wang et al., 2014] finds the optimal concatenation of trajectories for an estimation through a dynamic programming solution. They are able to improve the accuracy, as compared with segment-based approaches. However, the improvement is still limited due to the heuristical design, i.e., minimizing the estimation error of the travel time is not the target.

There are other works related to the estimation of travel time. TEMPR [Wang et al., 2016] and ST-NN [Jindal et al., 2017] study the OD travel time estimation problem that is to estimate the travel time based only on the origin and the destination without specific path while we estimate the travel time taken by a specific path. A variance was studied in ECML/PKDD 2015 challenge, i.e., estimating the travel time of the remaining trip without knowing the exact path, given the input of the initial path taken by a trip. Representative solutions include [Lam et al., 2015; Hoch, 2015]. DeepTTE [Wang et al., 2018] adopts deep learning method to predict travel time of a trajectory with intermediate GPS points. When the input is a path without any GPS point, its performance drops significantly.

On the other hand, recurrent neural network (RNN) has shown great power in modeling trajectory recently. For example, [Song et al., 2016] uses RNN to predict people’s future transportation mode; [Wu et al., 2017] models trajectory data with RNN, which achieves a better performance in predicting next movement than shallow models; [Gao et al., 2017] uses RNN with embeddings to represent the underlying semantics of user mobility patterns. In this paper, we will leverage on the power of RNN to perform travel time estimation of paths.

3 DeepTravel

To adopt neural networks in our study, we partition the whole road network into \( N \times N \) disjoint but equal-sized grids, similar to many existing approaches [de Brébisson et al., 2015; Zhang et al., 2017]. Accordingly, a travel path \( G \) started at \( t_1 \) could be represented by a sequence of grids it passes by, i.e., \( G = \{ g_1, g_2, \ldots, g_n \} \). As long as the granularity of grid cells is fine enough, \( G \) is able to capture the real movement of the path in road networks. Meanwhile, we assume sampled GPS points of the path are recorded to capture the real trajectory.
Driving state features. The driving process of vehicles can often be divided into the starting stage, the middle stage and the ending stage, and vehicles have different driving characteristics in various stages. For example, a vehicle prefers driving on the main roads/highways in the middle stage, where the speed could be very fast; while it has to move from the origin of the journey to the main roads/highways in the starting stage and it has to move from the main roads/highways to the destination in the ending stage. We use the vector \( V_{\text{dir}} \in \mathbb{R}^4 \) to represent the driving state features. It contains three 0-1 bits which represent the starting, middle and ending stages respectively and a ratio value capturing the proportion of the current path that is traveled (e.g., \([1,0,0,0,2]\) indicates a starting stage and it finishes 20% of the entire path).

Short-term and long-term traffic features. Traffic condition in a sub-region has the characteristic of continuity in terms of time dimension, e.g., a road segment that experienced traffic jam from 8:00 to 8:30 this morning is expected to have heavy traffic at 8:35, which means the traffic condition of the path right before a query is issued on the path is informative and useful. Accordingly, we use the term \( V_{\text{short}} \) to represent the short-term traffic condition features.

Given a path travel time estimation query submitted at time \( t \), we extract \( V_{\text{short}} \) from historical trajectories falling within the time window of \([t - 1 \text{ hour}, t]\). To be more specific, we partition trajectories into disjoint time-bin \( \tau_s \) of \( \delta \) minutes (e.g., 5 minutes in our experiment). Then, the traffic conditions of a certain grid \( g_i \) along these short time-bins form a sequence which reflects the temporal evolvement of the traffic condition in \( g_i \). Hence, we utilize the long short-term memory network (LSTM) [Hochreiter and Schmidhuber, 1997], a typical RNN for sequence modeling, to capture such temporal dynamics. LSTM is fed by sequences of the statistical information of each time-bin, e.g., \( \tau_1 \sim \tau_{12} \), and we set \( V_{\text{short}} \) to the last hidden state of LSTM. Notice that after partitioning the historical data into 5-minute-span time-bins, some grids may have no vehicle passing by in some time-bins. As LSTM model can handle variable length sequences, we can easily tackle this problem by skipping those time-bins with no vehicle passing by. E.g., in Figure 2, for the grid of “0-neighbor”, only \(-5\text{-min}, -25\text{-min}, \text{and} -60\text{-min} \) time-bin have historical vehicles passing by, so we can skip the remaining empty time-bins when feeding data into LSTM. Then, we design the input \( w.r.t \) the \( j \)-th time-bin \( \tau_j \) of grid \( g_i \) in the form of

\[
x_i = (j, v_j, n_j, l_{en_j}/v_j) \tag{1}
\]

\( j \) indicates the degree of closeness of \( x_i \) to the query time \( t_q \) in a linear scale, i.e., \( j = 12 \) \((j = 1)\) infers that the time-bin is one hour (5 minutes) before \( t_q \) which has the least (largest) closeness. \( v_j \) is the mean speed estimated from the samples in \( g_i \) at \( \tau_j \); \( n_j \) refers to the number of historical samples, which indicates the degree of trustworthiness (the larger the better) about the estimated speed \( v_j \) as \( v_j \) tends to be vulnerable to outliers if \( n_j \) is very small. \( l_{en_j}/v_j \) is the length of the query path \( G \) overlapped with grid \( g_i \), and \( l_{en_j}/v_j \) is a rough estimation of the average travel time of the path \( G \) spent within \( g_i \).

As mentioned before, the historical number of samples extracted at one day in a short time interval is not large which may result in data sparsity issue. Noticing the fact of spatial locality of the traffic condition, i.e., traffic conditions tend to be similar in adjacent grids, we further include the traffic feature of \( g_i \)’s neighbors’ as a solution. The \( d \)-neighbor set \( N_d \) of grid \( g_i \) is defined as the set of grid cells with their distances to \( g_i \) being \( d \), i.e., \( N_d = \{ g_j \mid \max(|x_j - x_i|, |y_j - y_i|) = d \} \), where \((x_j, y_j)\) indicates the position of \( g_j \), e.g., \((x_j, y_j) = (1, 2)\) denotes the grid located at the 1st row and 2nd column in \( N \times N \) grids. Accordingly, \( N_0 \) contains \( g_i \) itself, \( N_1 \) includes all the grid cells adjacent to \( g_i \), and so on. The final short-term traffic feature of \( g_i \) is the concatenation of \( g_i \)’s \( d \)-neighbor sets’ short-term traffic features. Figure 2 shows example \( N_d \)’s of the center grid with \( d = 0, 1, 2 \).

Previous work has shown that the long-term traffic dynamics is also important for estimating travel time [Wang et al., 2014]. The above short-term traffic feature extraction structure can be easily adjusted for supporting long-term traffic feature \( V_{\text{long}} \). In detail, we construct the sequence along the dimension of \( \text{days} \), e.g., we use the statistical information like Eq. (1) for the grid at the same time but in previous 7 days.

3.2 Prediction Layer

The prediction layer consists of two parts, namely BiLSTM and dual loss. The former is to combine feature representations of each grid to infer travel time information in hidden state vectors; while the latter is to further optimize the model.

BiLSTM. As compared with LSTM, bidirectional LSTM (BiLSTM) [Graves and Schmidhuber, 2005] utilizes additional backward information and thus enhances the memory capability. In our problem setting, we use BiLSTM to capture the information of every grid \( g_i \) in the path from the starting point to \( g_i \) and from the exit point from \( g_i \) simultaneously. We concatenate the features extracted in Section 3.1 together to get the global feature vector \( V \) of the grid, i.e.,

\[
V = [V_{\text{sp}}, V_{\text{tp}}, V_{\text{dir}}, V_{\text{short}}, V_{\text{long}}].
\]

We feed \( V \) to BiLSTM at each step and get the \( i \)-th hidden states \( h_i^+ \) and \( h_i^- \) of the forward and backward layer respectively. We then concatenate these two states to get the \( i \)-th hidden state \( h_i = [h_i^+; h_i^-] \).

Dual interval loss for auxiliary supervision. A simple way to estimate the travel time is to perform linear regression on the final hidden state \( h_n \) by employing loss function such as mean squared error \( w.r.t \) the ground truth \( t_m - t_1 \). However, it wastes much useful information (e.g., time stamps of inter-
mediate points of trajectories) that can supervise the model. To leverage such additional supervision information, we design a dual interval loss mechanism for auxiliary supervision which exactly matches the characteristic of BiLSTM.

The dual interval loss is constructed by two losses, the forward interval loss and the backward interval loss. The general idea is to force the model to learn to simultaneously predict the time interval from the start point to each intermediate GPS point \(p_j\), i.e., the forward interval, and the interval from \(p_j\) to the destination, i.e., the backward interval, as shown in Figure 3. In detail, we construct forward/backward mask vector \(M \in \{0, 1\}^n\) for activating forward/backward interval loss at some grids having supervisory information. Moreover, we construct \(T_f, T_b \in \mathbb{R}^n\) for recording the forward and backward interval ground truth. To be more specific,

\[
M_i = \begin{cases} 
1 & \text{if there is a point sampled in } g_i \\
0 & \text{otherwise}
\end{cases}
\]

\[
T_i^f = g_i.t - g_0.t \\text{ if there is a point sampled in } g_i \\
1 & \text{otherwise, a random value}
\]

\[
T_i^b = g_n.t - g_i.t \\text{ if there is a point sampled in } g_i \\
1 & \text{otherwise, a random value}
\]

\(g_i.t\) refers to the time when the vehicle leaves the grid \(g_i\) along the path, which can be derived from the corresponding trajectory data if there are GPS points sampled in the grid \(g_i\). Specifically, we define \(g_0.t\) and \(g_n.t\) as the time stamp of the first and the last GPS point respectively.

For predicting the dual intervals, instead of using the current \(h_t\) for prediction, we decide to adopt a summation operation, which adds \(h_1\) to \(h_t\), together for forward prediction and \(h_{t+1}\) to \(h_n\) for backward prediction as Figure 3 shows. Such summation operation feeds the model with the prior knowledge of the summation property of time, i.e., the time spent on a path is the summation of the time spent on two sub-paths, and there is no need to learn such summation property from the data. Moreover, the predicted time of each step, i.e., \(W^T h_t + b\), now represents the time spent only on grid \(g_i\), which forces the sum of the predicted forward interval and the backward interval hold the same across all steps, i.e., \((W^T \sum_{i=1}^{n} h_i + b)\), that’s exactly the travel time of the whole path. Thus, minimizing the dual loss can also benefit the estimation on the travel time of the entire path at each step which can be naturally regarded as the chief supervision in our task. Consequently, the forward/backward interval time estimation vectors \(\tilde{T}^f, \tilde{T}^b\) are as follows.

\[
\tilde{T}^f = W^T \left[ h_1, h_1 + h_2, \ldots, \sum_{i=1}^{n-1} h_i, \sum_{i=1}^{n} h_i \right] + b
\]

\[
\tilde{T}^b = \left[ W^T \sum_{i=2}^{n} h_i, \sum_{i=3}^{n} h_i, \ldots, h_1 + h_n, h_n \right] + b, 0
\]

Here, \(\tilde{T}^f \in \mathbb{R}^n\) represents the travel time from the starting point to each grid in the path, and \(\tilde{T}^b \in \mathbb{R}^n\) represents the travel time from each grid in the path to the ending point. For both forward and backward predictions, we use shared weight \(W, b\) because we want to restrict the task of transformation from \(h_t\) to the travel time spent on grid \(g_i\) to be the same in both forward and backward predictions. The dual interval loss is the summation of the forward and backward interval losses. We use the relative mean square error \(L\) as follows. Note, operation with \([i]^{[2]}\) indicates element-wise one.

\[
L = \frac{1}{n} \cdot \left( \left( \tilde{T}^f - T^f \right)[/\| T^f \|][2] + \left( \tilde{T}^b - T^b \right)[/\| T^b \|][2] \right)
\]

The dual interval loss not only minimizes the travel time estimation error of the whole path but also constrains the forward and backward interval estimation error of intermediate grids, which utilizes the intermediate time information of a trajectory. It has the following three advantages. First, the intermediate monitoring information to some extent increases the amount of data to help model training better. Second, adding the supervisory information in the middle can make the loss signal back-propagate more accurate and effective, which will reduce the risk of vanishing gradient for long sequences. Third, the dual loss exactly matches the BiLSTM characteristics, as each step of BiLSTM, it has the information from the starting grid to the current grid and that from the current grid to the ending grid, which can naturally be used by forward and backward interval loss. We will show the superiority of the dual interval loss in the experiment section.

### 3.3 Training

The goal of \textsc{DeepTravel} is to minimize the dual loss function \(L\), i.e., \(\min_{\theta, \varepsilon} \sum_{i=1}^{S} L^{(i)}(\theta, \varepsilon)\). Here, \(\theta\) denotes the trainable parameters in \textsc{DeepTravel}, and \(\varepsilon\) refers to the spatial and temporal embedding vectors, \(S\) is the number of training trajectories and \(L^{(i)}\) is the dual loss function of \(i\)-th trajectory data. The model is trained by employing derivatives of the loss w.r.t. all parameters through back-propagation-through-time algorithm [Werbos, 1990].

### 4 Experiments

**Datasets.** Two real trajectory datasets are used in our experimental study, namely Porto and Shanghai. The Porto dataset is a 1.8GB open dataset, generated by 442 taxis from Jan. 07, 2013 to Jun. 30, 2014. The Shanghai one is generated by 13,650 taxis from Apr. 01 to Apr. 17 in 2015 with the size of 16GB. We extract the trajectory trips occupied by passengers as valid trajectories. Table 1 reports the description and statistics of the two datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Porto</th>
<th>Shanghai</th>
</tr>
</thead>
<tbody>
<tr>
<td>trajectory number</td>
<td>420,000</td>
<td>1,018,000</td>
</tr>
<tr>
<td>sampling interval</td>
<td>15s</td>
<td>10s</td>
</tr>
<tr>
<td>area</td>
<td>16.7km x 14.4km</td>
<td>29.8km x 37.9km</td>
</tr>
<tr>
<td>grid size</td>
<td>128 x 128</td>
<td>256 x 256</td>
</tr>
<tr>
<td>travel time mean</td>
<td>762.60s</td>
<td>954.59s</td>
</tr>
<tr>
<td>travel time std</td>
<td>347.92s</td>
<td>460.71s</td>
</tr>
</tbody>
</table>

**Hyperparameters.** For hyperparameters, we split each dataset into training set, validation set and test set in the ratio of 8:1:1. The embedding size of spatial and temporal embeddings is 100 and initialized uniformly to \([-1, 1]\). We set the hidden unit as 100 for both LSTM in traffic feature extraction and BiLSTM in prediction. We train the model using Adam [Kingma and Ba, 2014] with an initial learning rate at 0.002. All the weights are uniformly initialized to \([-0.05, 0.05]\).

**Metrics.** We adopt \textit{mean absolute error} (MAE), \textit{mean absolute percentage error} (MAPE) and root-mean-squared error (RMSE).
In addition to the above existing competitors, we also propose three simple end-to-end models as baselines, namely grid-MLP, grid-CNN and grid-LSTM. grid-MLP uses multilayer perceptron (MLP) model to predict the travel time. We use a $N \times N$ matrix $M$ as the input, with each element $M_{ij}$ capturing the travel length that the vehicle passes through the grid $g_{ij}$; and we use two hidden layers with 1024 units and sigmoid as activation function. grid-CNN uses convolutional neural network (CNN) model to perform the estimation. It accepts the same input $M$ as grid-MLP. We use three convolutional layers, each having $64 \times 3 \times 3$ filters with stride 1, and 3 max-pooling layers, each in the size of $2 \times 2$. Then it is followed by a fully-connected layer with 1024 units and sigmoid activation for prediction. grid-LSTM uses LSTM to predict the travel time. We set LSTM with 100 hidden units, and feed it with the travel length of the present grid at each step. All three models adopt the mean relative squared error as the loss function. Note that DEEP TRAVEL is also an end-to-end model.

### 4.1 Overall Evaluation

The first set of experiments is to evaluate the performance of travel time estimation on the query path, with the results reported in Table 2. We observe that in general the sub-path based approaches perform better than segment based and OD based approaches. This indicates that the interaction between adjacent road segments in a path is important.

### 4.2 Performance of DEEP TRAVEL

The impact of different features. As DEEP TRAVEL takes in inputs from multiple features, we conduct the second set of experiments to study their effectiveness. We implement five different versions of DEEP TRAVEL with each taking in different feature inputs. $ST$ only uses the spatial and temporal embeddings; NaiveTraf takes in the mean historical speed corresponding to the grid as the traffic feature; Traf only uses the traffic features in our model; ST+Traf accepts both traffic features as well as spatio-temporal embeddings as input; and ST+Traf+DS takes in all the features considered by DEEP TRAVEL (DS refers to driving state feature). As listed in Table 3, ST+Traf+DS outperforms other versions. ST+Traf performs better than both ST and Traf, which means that both traffic features and spatio-temporal embeddings play important roles.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metrics</th>
<th>Porto MAE (sec)</th>
<th>Porto RMSE (sec)</th>
<th>Porto MAPE</th>
<th>Shanghai MAE (sec)</th>
<th>Shanghai RMSE (sec)</th>
<th>Shanghai MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OD based</td>
<td>TEMP [Wang et al., 2016]</td>
<td>193.67</td>
<td>314.08</td>
<td>0.2505</td>
<td>248.70</td>
<td>353.47</td>
<td>0.2513</td>
</tr>
<tr>
<td>Segment Based</td>
<td>spd-MEAN</td>
<td>245.81</td>
<td>358.32</td>
<td>0.2847</td>
<td>340.70</td>
<td>550.43</td>
<td>0.4170</td>
</tr>
<tr>
<td></td>
<td>ARIMA [Ahmed and Cook, 1979]</td>
<td>227.40</td>
<td>517.51</td>
<td>0.2757</td>
<td>315.22</td>
<td>444.42</td>
<td>0.3074</td>
</tr>
<tr>
<td></td>
<td>SVR [Asif et al., 2014]</td>
<td>241.41</td>
<td>353.35</td>
<td>0.2819</td>
<td>424.12</td>
<td>543.28</td>
<td>0.4085</td>
</tr>
<tr>
<td></td>
<td>SAE [Lv et al., 2015]</td>
<td>222.06</td>
<td>357.02</td>
<td>0.2734</td>
<td>310.47</td>
<td>413.62</td>
<td>0.3013</td>
</tr>
<tr>
<td></td>
<td>spd-LSTM [Ma et al., 2015]</td>
<td>217.37</td>
<td>334.00</td>
<td>0.2624</td>
<td>302.45</td>
<td>397.48</td>
<td>0.2945</td>
</tr>
<tr>
<td>Sub-path Based</td>
<td>RTTE [Rahmani et al., 2013]</td>
<td>169.45</td>
<td>272.22</td>
<td>0.2234</td>
<td>214.01</td>
<td>307.77</td>
<td>0.2362</td>
</tr>
<tr>
<td></td>
<td>PTTE [Wang et al., 2014]</td>
<td>159.43</td>
<td>268.11</td>
<td>0.2072</td>
<td>168.48</td>
<td>248.92</td>
<td>0.1914</td>
</tr>
<tr>
<td>End-to-End</td>
<td>grid-MLP</td>
<td>255.33</td>
<td>377.27</td>
<td>0.2933</td>
<td>423.53</td>
<td>541.19</td>
<td>0.3906</td>
</tr>
<tr>
<td></td>
<td>grid-CNN</td>
<td>250.86</td>
<td>363.17</td>
<td>0.2874</td>
<td>420.05</td>
<td>537.86</td>
<td>0.3885</td>
</tr>
<tr>
<td></td>
<td>grid-LSTM</td>
<td>180.27</td>
<td>300.98</td>
<td>0.2334</td>
<td>235.74</td>
<td>348.30</td>
<td>0.2505</td>
</tr>
<tr>
<td></td>
<td>DEEPTRAVEL</td>
<td><strong>113.24</strong></td>
<td><strong>219.25</strong></td>
<td><strong>0.1337</strong></td>
<td><strong>126.59</strong></td>
<td><strong>196.85</strong></td>
<td><strong>0.1330</strong></td>
</tr>
</tbody>
</table>

Table 2: Performance comparison of DEEP TRAVEL and its competitors.
Table 3: Performance of DEEP TRAVEL with different features.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Porto MAE (sec)</th>
<th>MAPE</th>
<th>Shanghai MAE (sec)</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
<td>129.33</td>
<td>0.1505</td>
<td>197.58</td>
<td>0.1926</td>
</tr>
<tr>
<td>NaiveTraf</td>
<td>144.41</td>
<td>0.1688</td>
<td>199.06</td>
<td>0.1940</td>
</tr>
<tr>
<td>Traf</td>
<td>132.28</td>
<td>0.1537</td>
<td>153.95</td>
<td>0.1559</td>
</tr>
<tr>
<td>ST+Traf</td>
<td>114.47</td>
<td>0.1367</td>
<td>129.44</td>
<td>0.1362</td>
</tr>
<tr>
<td>ST+Traf+DS</td>
<td>113.24</td>
<td>0.1337</td>
<td>126.59</td>
<td>0.1330</td>
</tr>
</tbody>
</table>

Table 4: The effectiveness of different loss functions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Porto MAE (sec)</th>
<th>MAPE</th>
<th>Shanghai MAE (sec)</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM_no_aux</td>
<td>130.57</td>
<td>0.1494</td>
<td>148.90</td>
<td>0.1306</td>
</tr>
<tr>
<td>BiLSTM_no_aux</td>
<td>128.85</td>
<td>0.1476</td>
<td>143.72</td>
<td>0.1475</td>
</tr>
<tr>
<td>BiLSTM_for_aux</td>
<td>116.54</td>
<td>0.1399</td>
<td>128.56</td>
<td>0.1349</td>
</tr>
<tr>
<td>BiLSTM_back_aux</td>
<td>118.85</td>
<td>0.1372</td>
<td>128.77</td>
<td>0.1355</td>
</tr>
<tr>
<td>BiLSTM_dual_aux</td>
<td>113.24</td>
<td>0.1337</td>
<td>126.59</td>
<td>0.1330</td>
</tr>
</tbody>
</table>

The effectiveness of different loss functions. In order to demonstrate the effectiveness of the proposed dual interval loss with auxiliary supervision, we compare it with other loss functions. We construct five baselines which share the same feature extraction layer as DEEP TRAVEL but different loss functions for training. To be more specific, LSTM_no_aux feeds features to an LSTM, and only uses the final hidden vector to predict the travel time (i.e., no auxiliary supervision). Like LSTM_no_aux, BiLSTM_no_aux uses the final forward and backward hidden state for prediction. Both BiLSTM_for_aux and BiLSTM_back_aux leverage the auxiliary supervision, i.e., the time stamps of intermediate GPS points, but BiLSTM_for_aux only uses the forward interval loss while BiLSTM_back_aux only optimizes the backward loss. BiLSTM_dual_aux is DEEP TRAVEL model which optimizes both forward and backward interval loss functions with auxiliary supervision.

We report the quantitative results in Table 4 and the MAPE curve in validation set w.r.t. training epochs in Figure 4. From the results, we can find that BiLSTM_no_aux performs better than LSTM_no_aux, which means that BiLSTM is able to capture correlations much better than LSTM. We also observe that all models with auxiliary supervision behave much better than models without auxiliary supervision and have a very fast convergence. This proves that the auxiliary supervision from additional interval loss benefits the back-propagation of loss signals, and the additional supervision is some kind of data augmentation which can improve the results, as analyzed in Section 3.2. Last, as we expect, BiLSTM_dual_aux performs better than BiLSTM_for_aux and BiLSTM_back_aux, which demonstrates auxiliary supervisions from forward and backward are not redundant but complementary.

Figure 4: The MAPE curve under different loss functions.

Figure 5: Performance of DEEP TRAVEL vs. length and travel time of paths.

The performance of DEEP TRAVEL vs. length and travel time of paths. Last but not least, we partition the testing trajectory set into different subsets according to the length of the trajectories and the duration of the travel time. We report the MAPE of DEEP TRAVEL under Shanghai dataset, as a representative, in Figure 5(a). In general, DEEP TRAVEL performs well (i.e., MAPE around 0.1). However, we do observe a performance drop when the path is short and the travel time is long, e.g., 4km and 35min. Firstly, this type of trajectories is abnormal as the travel time in most cases is proportional to the length of the path. For example, the paths with the length of 4km and the travel time of 35min mean the average speed is about 6.9km/h which is extremely slow, not much faster than the walking speed. By examining these trajectories from the dataset, we observe that most of them encounter unexpected congestion or stay at one place for a long time which can not be learned from historical data. The histogram in Figure 5(b) also proves that such trajectories are extremely rare.

5 Conclusion

In this paper, we present an end-to-end estimation model, namely DEEP TRAVEL, for estimating the travel time of a query path. We propose a unique feature extraction structure which takes multiple features into account. We also introduce the dual interval loss, which elegantly matches the characteristics of BiLSTM with that of trajectory data, to incorporate additional supervisory information naturally. We conduct experiments on real datasets to understand the role of different features and to demonstrate the superiority of DEEP TRAVEL.

Acknowledgments

This research is supported in part by the National Natural Science Foundation of China under grant 61772138, the National Key Research and Development Program of China under grant 2018YFB0505000 and the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative.
References


[Han et al., 2011] Jiawei Han, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.


