Aggressive Driving Saves More Time? Multi-task Learning for Customized Travel Time Estimation

Ruipeng Gao[1*], **Xiaoyu Guo**[1], **Fuyong Sun**[1], **Lin Dai**[1], **Jiayan Zhu**[2], **Chenxi Hu**[2] and **Haibo Li**[2]
[1]Beijing Jiaotong University
[2]Beijing DiDi Infinity Technology and Development Co., Ltd.
{rpgao, guoxiaoyu, fysun12, dailin}@bjtu.edu.cn, {zhujiayan, huchenxi, lihaibo}@didiglobal.com

Abstract

Estimating the origin-destination travel time is a fundamental problem in many location-based services for vehicles, e.g., ride-hailing, vehicle dispatching, and route planning. Recent work has made significant progress to accuracy but they largely rely on GPS traces which are too coarse to model many personalized driving events. In this paper, we propose Customized Travel Time Estimation (CTTE) that fuses GPS traces, smartphone inertial data, and road network within a deep recurrent neural network. It constructs a link traffic database with topology representation, speed statistics, and query distribution. It also uses inertial data to estimate the arbitrary phone's pose in car, and detects fine-grained driving events. The multi-task learning structure predicts both traffic speed at public level and customized travel time at personal level. Extensive experiments on two real-world traffic datasets from Didi Chuxing have demonstrated our effectiveness.

1 Introduction

Thanks to the explosion of sharing economy, we can easily hail a ride at anywhere and anytime in most urban cities. Such ride-hailing platforms, e.g., Uber, Lyft, and Didi Chuxing, benefit our everyday travel and ensure efficient use of vehicles. However, the riding experience differs a lot among drivers, and sometimes they may even adopt aggressive driving behaviors to arrive earlier. To get rid of such dangerous events[1], we are curious about how much time they can save for their specific driving behaviors.

Origin-destination travel time estimation is pivotal to many location-based services, including ride-hailing, vehicle dispatching, and route planning. Focusing on individual drivers, the central problem is the balancing art between large-scale crowdsourced traffic data and your own driving behavior. A solution must consider both general traffic speed at public level and your specific driving patterns at personal level.

Recently, a series of efforts have been undertaken to address this problem. The route-based solutions [Sevlian and Rajagopal, 2010; Pan *et al.*, 2012; Wang *et al.*, 2016b] estimate the driving time on each road segment and intersection, then summarize them as the origin-destination travel time. However, precisely modeling of dynamic transportation systems is difficult, especially via sparse and low-quality crowdsourced traffic data. The data-driven solutions [Rahmani *et al.*, 2013; Wang *et al.*, 2016a; Wang *et al.*, 2018a; Zhang *et al.*, 2018; Li *et al.*, 2018; Wang *et al.*, 2018b] are mainly based on machine learning techniques, which formulate a multivariate time series prediction problem on spatial-temporal traffic data. However, they always lack the considerations of road map topology, general traffic condition, or individual driving patterns.

In this paper, we propose a novel multi-source heterogeneous data fusion approach that can predict both general traffic speed on each road link and travel time for each individual, via one multi-task learning model. Such a data fusion approach entails a series of non-trival challenges. First, how to extract the most important features from multiple types of traffic data. Second, how to identify aggressive driving events for different drivers, despite noisy inertial data from smartphones with arbitrary poses in car. Finally, how to balance the general traffic speed and personalized driving behaviors.

Our solution consists of several components to deal with the above challenges, producing accurate predictions simultaneously for both general traffic speed and customized travel time. It utilizes GPS trajectories, road network, query amount, and auxiliary information (e.g., weather and holiday) to learn general traffic features at public level, and fuses them within a Recurrent Neural Network (RNN) to predict future traffic speed on each road link. To analyze personalized driving behaviors, we explore a Principal Component Analysis (PCA) algorithm to estimate arbitrary phone's pose in car, then harness inertial data to identify aggressive driving events for each driver. Such general traffic features and individual driving behaviors are further fused within a multi-task learning structure for customized travel time estimation.

Specifically, we make the following contributions:

- We produce a large-scale multi-source heterogeneous traffic dataset collected via mobile crowdsensing, including geographic vehicle trajectories, smartphone inertial data, road network, and auxiliary information.

[*]Contact Author

[1]According to the statistics, approximately one-third of all traffic fatalities in the US occur due to "aggressive driving." [BMV, 2019]

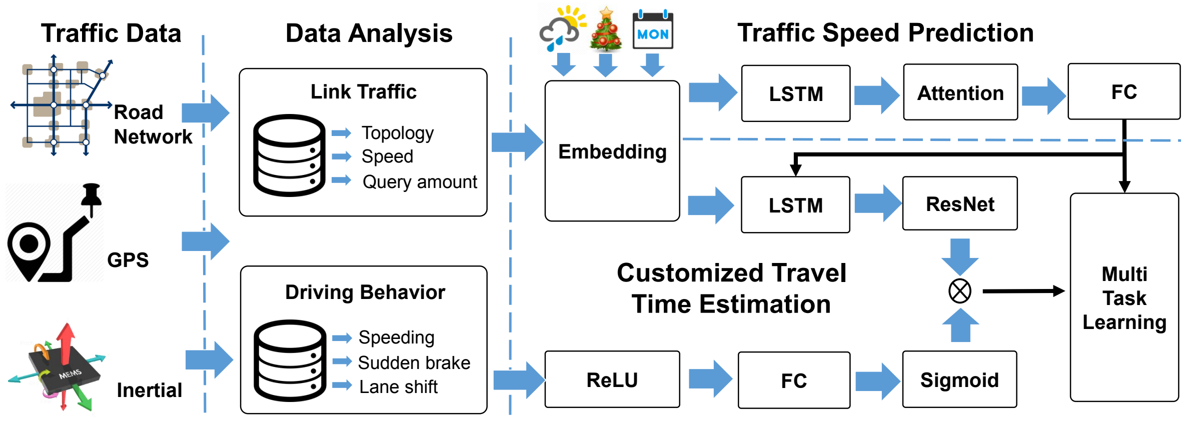


Figure 1: CTTE takes road network, GPS trajectories, and inertial data as inputs. In data analysis stage, we produce two datasets on link traffic and driving behavior. We further propose a multi-task learning structure to predict both traffic speed and customized travel time.

- We construct two traffic databases: 1) link traffic database, with link topology representation, speed statistics, and query distribution; and 2) driving behavior database, focusing on three aggressive driving events, i.e., speeding, sudden brake, and lane shift.
- We fuse GPS traces, smartphone inertial data, and road network within a deep recurrent neural network. We also propose a multi-task learning structure to estimate both traffic speed on each link and customized travel time for each individual.
- We conduct extensive experiments on two real-world datasets collected by DiDi platform. Results have shown our effectiveness compared with the state-of-the-art.

2 Overview

In this section, we first present several important definitions in our problem, and depict the overview of our method.

2.1 Definitions

Definition 1. **Road Network.** Given a fixed region on the map, the road network is defined as the set of underlying road links (i.e., road segments). Each road link includes its geographical location, length, direction, speed class, lane number, and other attributes. Note that this road network model is different from others, which define it as a directed/undirected graph with nodes and edges. Given a vehicle trajectory, we observe that those nodes (mainly road intersections) sometimes cause drivers a very long queuing time, especially in rush hours. Thus, we treat road intersections also as links.

Definition 2. **Path and Trajectory.** A driving path is defined as a sequence of points, and each point contains the location (e.g., latitude and longitude), time in the day, and road link index indicating which road segment it locates on. In addition, a vehicle trajectory $x^{(i)}$ is defined as a tuple with three components, i.e., $x^{(i)} = (u^{(i)}, P^{(i)}, \lambda^{(i)})$, where i is the trajectory ID, $u^{(i)}$ is the driver's ID, $P^{(i)}$ is the driving path, and $\lambda^{(i)}$ is the auxiliary information for this trajectory, including the day index in the week, holiday index, and weather index.

Definition 3. **Aggressive Driving Events.** We consider three aggressive driving events as dangerous behaviors: 1) speeding; 2) sudden brake; 3) frequent lane shift. When sensing with a smartphone, such events will cause distinct signal patterns in both GPS trajectories and inertial data.

Given the above definitions, we conceive the customized travel time estimation problem as:

Definition 4. **Problem Statement.** During the training phase, we learn: 1) how to estimate the traffic speed on each link via the road network and GPS trajectories, and 2) how to analyze driving behavior for each driver via inertial data. During the test phase, given a driver ID, an origin, a destination, and a departure time, our goal is to estimate the travel time for this specific driver, with the path generated by other route planning techniques.

2.2 Model Architecture

Our method utilizes GPS trajectories, smartphone inertial data, and road network (assumed already available) as inputs, and formulates customized travel time estimation with a multi-task learning structure (Figure 1).

The data analysis stage produces two databases: the link traffic and the driving behavior. The link traffic database contains link topology representation from road network, link speed statistics in each time segment, and query amount on each link. We also identify three aggressive driving events, i.e., speeding, sudden brake, and lane shift.

Next, we fuse the multi-source heterogeneous traffic data within a deep recurrent neural network, and explore a multi-task learning structure for both traffic speed prediction and customized travel time estimation.

3 Data Analysis

In this section, we manage and analyze the large-scale heterogeneous traffic data, and produce two specific datasets.

3.1 RAW Traffic Data

Our raw data was collected from the Didi Chuxing mobile application, including: 1) road network information; 2) vehicle GPS trajectories; and 3) smartphone inertial data from

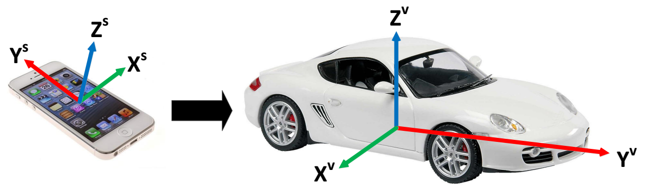


Figure 2: Smartphone pose estimation, with (X^s, Y^s, Z^s) for the phone's coordinate system and (X^v, Y^v, Z^v) for the vehicle's.

IMU (Inertial Measurement Unit) sensors, with 3-axis accelerations by accelerometer and 3-axis angular speed by gyroscope, both in the smartphone's coordinate system.

Notice that the smartphones' coordinate system is not always the same as the vehicle's (shown in Figure 2), we explore a Principal Component Analysis (PCA) algorithm to estimate the arbitrary phone's pose in the car, i.e., estimating the vehicle's coordinate system in the phone's coordinate system. Step 1) When the car is static, the gravity direction (i.e., Z-axis of the vehicle) can be computed via a low-pass Butterworth filter to remove high frequency components. Step 2) We use the gravity direction to deduct 3-axis accelerations onto the horizontal plane, and the forward direction (i.e., Y-axis of the vehicle) is caused by vehicle accelerating and decelerating, which can be computed as the maximum acceleration direction by PCA algorithm. Step 3) The rest X-axis of the vehicle is calculated as the cross product of the other two axis directions. Finally, we transform the motion data from the phone to the vehicle.

3.2 Link Traffic

Link Topology

Road topological relations is crucial for both traffic speed prediction and travel time estimation. Some advanced transportation systems have already assigned several successive road segments with the same traffic light lifetime, thus vehicles pass them with a high speed and do not need to stop frequently. However, representing the road topology is not trivial. Simple numerical or one-hot encoded categorical features can not reflect the entire road topology, especially for complex road networks. Graph Laplacian Regularization method [Li *et al.*, 2018] enhances the loss with a graph Laplacian factor, thus adjacent links are likely to be assigned with similar representations, but its optimization step fails for large-scale road networks.

Inspired by the unsupervised graph embedding approach in DeepWalk [Perozzi *et al.*, 2014], we explore a road topology representation method for large-scale road networks. First, instead of using the RandomWalk algorithm which generates random node sequences to "sample" the graph, we leverage a map-matching method [Jagadeesh *et al.*, 2004] to attach each GPS point onto a specific road link and merge continuous links, thus each trajectory can produce a sequence of link IDs.

Next, we use the SkipGram language model [Mikolov *et al.*, 2013] which maximizes the co-occurrence probability among words in a sentence. Given a link sequence $(l_1, l_2, ..., l_N)$, we represent each link as an $N \times 1$ vector by one-hot encoding, and define a neural network to compute

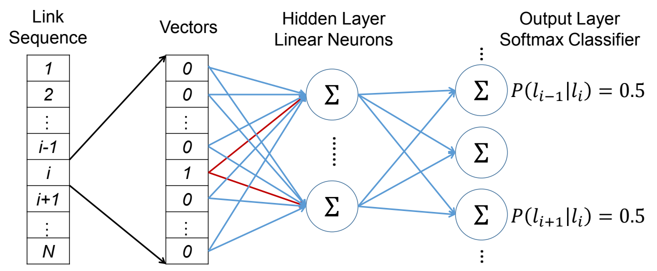
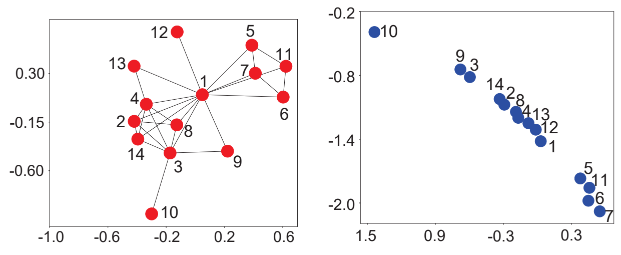


Figure 3: Illustration of the SkipGram algorithm for a link sequence.



(a) Example road network (b) Topology representation

Figure 4: Topology representation on example road network.

the probability of each other link that it is adjacent to link l_i (shown in Figure 3).

In this model, the neural network has only one hidden layer without the activation function, and the output layer uses softmax function to ensure the output vector as a probability distribution. We only keep the weight matrix in hidden layer as link representations. However, when representing a road network with millions of links, the large-scale weight coefficients make the posterior distribution learning (i.e., $P(l_{i-1}|l_i)$ and $P(l_{i+1}|l_i)$) extremely difficult. To speed up its training process, we use the Hierarchical softmax function in DeepWalk. We assign all road links as a sequence of tree nodes, and transform the prediction problem into maximizing the probability of a specific path in the hierarchy, i.e.,

$$P(l_{i-1}|l_i) = \prod_{k=1}^{\lceil \log N \rceil} P(l_{a_k}|l_i) \tag{1}$$

where the path in tree structure from root to node l_{i-1} is denoted as $(l_{a_0}, l_{a_1}, ..., l_{a_{\lceil \log N \rceil}})$. In this equation, $P(l_{a_k}|l_i)$ can be learned by a binary classifier:

$$P(l_{a_k}|l_i) = \frac{1}{1 + e^{-r(l_{a_k}) \cdot r(l_i)}} \tag{2}$$

where $r(.)$ is the representation vector of each link.

Finally, we use the stochastic gradient descent (SGD) algorithm for optimization, with the learning rate of 2.5% and decreasing linearly. Figure 4 shows the representation example of a road network with 14 links.

Link Traffic Speed

Historical traffic speed statistics can reflect rush hours and traffic conditions on each link, thus has a direct impact when estimating the travel time. After map matching process, the vehicle trajectory is comprised of a series of GPS points, each

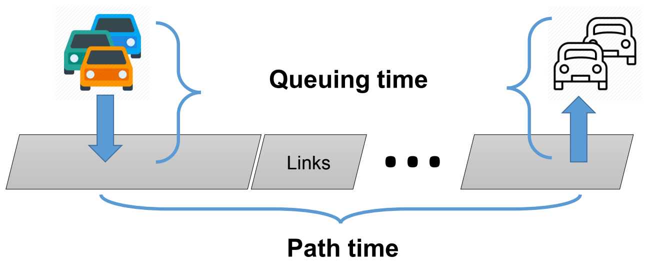


Figure 5: Illustration of the origin-destination travel time, which consists of the path time on links and the queuing time at inlet/outlet.

with a timestamp and a road link index, thus we can easily compute the average traffic speed on each link.

Suppose a trajectory segment with GPS points $p_{k:k+n}$ locates on link l_i, starting from time t to $t+T$. The link traffic speed for l_i is computed as $l_i^v = \frac{l_i^{length}}{T}$. Notice that vehicles enter the main road on the first link and exit on the last link, actual driving length on these two links are not complete. Thus, we remove the GPS points on these two links. Finally, we summarize all historical trajectories and compute the average traffic speed on each link for each time interval (5 minutes in this paper).

Link Query Amounts

As shown in Figure 5, the origin-destination travel time consists not only the path time when driving on road links, but also the queuing time when entering/exiting the main road, especially when we visit a POI (Point of Interest).

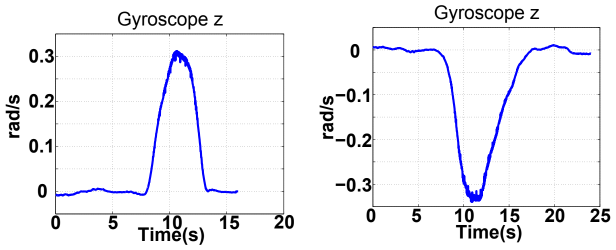
By Queuing Theory, the queuing time can be computed via queuing length (amount) and driving speed. Thus, we also analyze the historical query distribution on each link at each time interval. For example, a trajectory starting from link l_A at time t_A and ending from link l_B at time t_B can produce two records of link query: $q_{l_A,IN}^{t_A}$ and $q_{l_B,OUT}^{t_B}$, where IN and OUT denote query direction to links.

3.3 Driving Behavior

Driving behaviors differ significantly among drivers, especially for rookie and aggressive drivers. Rookie drivers may drive relatively slow, wait longer when queuing, and make violent brakes. Aggressive drivers may shift lanes frequently to overtake others. Thus, driving behavior analysis for each individual is crucial to assess driving performances and estimate personalized travel time more precisely.

Measuring the driving behavior is not trivial. Although some recent applications (e.g., Waze) have already used drivers' prefered routes to provide better path navigation, they largely rely on GPS trajectories with relatively poor accuracy and low sampling rate, e.g., $\sim 5m$ position errors at $1Hz$ for commercial smartphones[2], thus are too coarse to model many fast driving events such as lane shifts and sudden brakes. Our intuition comes from the inertial data which describes both linear accelerations and rotations of a smartphone at fine-grained level.

[2]https://www.gps.gov/systems/gps/performance/accuracy



(a) Left turn (b) Right turn

Figure 6: Gyroscope signals used for turn detection.

Lane Shifts

First, we leverage the PCA algorithm to transform the motion information from the smartphone to the vehicle (elaborated in Section 3.1), and obtain vehicle's forwarding acceleration a^Y and angular speed ω^Z for driving behavior analysis. a^Y corresponds to vehicle's velocity and can be used to detect speeding events and sudden brakes, while ω^Z corresponds to vehicle's rotations and can be used to detect lane shifts. For example, a lane shift on the left corresponds to a first left turn (Figure 6(a)) then immediately a right turn (Figure 6(b)) within a short time.

In addition, the gyroscope readings are known to be accurate within a short time, but suffer from a linear drift. We further explore a complementary filter algorithm [Shen *et al.*, 2018] to fuse ω^Z with the long-term vehicle orientation observations from GPS traces, thus eliminate gyroscope drifts and adjust detection thresholds of lane shifts and turns for each trajectory.

Speeding and Sudden Brakes

In real-world traffic dataset, we observe that there are many GPS invisible environments near buildings, bridges, and trees. Besides using GPS for speed measurement under open sky, we also need to recover vehicle's speed in case lacking of GPS signals. Thus, we devise a highly accurate odometry from only inertial data.

Since the accuracy of the forwarding acceleration signal a^Y is always noisy and varies among smartphones, we should calibrate raw acceleration readings first. We propose a deep Recurrent Neural Network (RNN) model to map a sequence of forwarding accelerations to available GPS velocity, i.e.,

$$f_\Phi : a_{t:t+T}^Y \mapsto \Delta\hat{v}_T \tag{3}$$

where T is the window length, and Φ represents the parameter in the RNN model. Next, we approximate the actual vehicle velocity difference Δv_T from the GPS trajectory, thus our objective is to minimize the total prediction errors, i.e.,

$$\Phi^* = \arg\min_\Phi \sum \|\Delta\hat{v}_T - \Delta v_T\|_2^2 \tag{4}$$

During implementation, we find the standard RNN model is always hard to train, thus adopt BiLSTM [Chen *et al.*, 2018] network to eliminate accumulated prediction errors via the backward information. Finally, we leverage the inferred vehicle velocity to identify speeding and sudden brake events.

4 Applications

In this section, we first present our link speed prediction model via link traffic database and auxiliary information, then use it to further infer customized travel time for each individual.

4.1 Link Speed Prediction

Our link traffic database contains most useful traffic features for links: the link topology captures adjacency relations, the historical speed reflects busy hours, and the query amount measures queuing time. We also implement a web crawler to extract other auxiliary attributes such as the weather, the holiday, and the day in week information.

With such heterogeneous data inputs from multiple sources, we incorporate them and use the embedding method [Gal and Ghahramani, 2016] to transform those categorical attributes into low-dimensional vectors, thus can feed them into the neural network.

To further capture the temporal dependencies among road links, we apply the recurrent neural network (RNN) to learn long-term temporal patterns. RNN has been widely used in sequential learning on natural language processing, machine translation, and speech recognition. In our model, the input vector for each link i is constructed as the concatenation of all attributes, i.e.,

$$x_i = relu(W_x \cdot [x_i^{topology} \circ x_i^{speed} \circ x_i^{query} \circ x_i^{auxiliary}]) \quad (5)$$

where $x_i^{topology}$, x_i^{speed}, x_i^{query}, and $x_i^{auxiliary}$ denote the embedded vector of topology, speed, query amount, and auxiliary information for each link, respectively. W_x is the weight matrix.

Next, we input the concatenated vector into a LSTM [Yao *et al.*, 2017] structure as the RNN implementation, and obtain the current hidden variable h_t as:

$$h_t = LSTM(x_i, h_{t-1}) \quad (6)$$

We also use the soft attention mechanism [Liang *et al.*, 2018] to capture the weights of a sequence of LSTM's output. Finally, the output of attention is connected to an FC layer, then compared with speed statistics to calculate the speed prediction loss L_{speed}.

4.2 Customized Travel Time Estimation

In addition to predicting current traffic speed, estimating the origin-destination travel time for each individual is also meaningful in many intelligent transportation systems and applications. However, driving skills vary obviously among different drivers, thus we should also consider their driving behaviors for customized travel time estimation.

Given the path of a trajectory, we first build an LSTM network to capture the temporal and spatial (topological) features of travel time. The hidden state output of LSTM is connected to a ResNet (Residual Neural Network [He *et al.*, 2016]) module for decoding.

To identify the aggressive driving events for each driver, we use smartphone inertial data with pre-tuned filters to generate different channels of driving behaviors (details elaborated in Section 3.3). We further concatenate such driving behavior data and process them sequentially with a $ReLU$ function, an FC layer, and a $sigmoid$ function to calculate a driver-personalized scale factor.

Next, we use the scale factor to amend the output of ResNet by multiplication. Note that this driver-personalized scale factor is better than simple driver ID used in other methods [Wang *et al.*, 2018b; Wang *et al.*, 2018a], since it can match similar driving behaviors among different drivers, and remain effective even for new users with few historical data.

Finally, to train our model, we formulate a multi-task learning problem, and our objective is to minimize the combination of both the speed loss and time loss, i.e.,

$$L_{time} + \alpha \cdot L_{speed} \quad (7)$$

where the coefficient α weights the speed loss item. During training, we leverage the mean square error (MSE) as the loss function for speed prediction, and use the mean absolute percentage error (MAPE) for travel time estimation.

5 Experiments

We evaluate our proposed method on two large-scale real-world traffic datasets, which is collected crowdsourcingly from DiDi platform. We also compare it with the latest existing approaches for effectiveness.

5.1 Datasets and Baseline Algorithms

Our traffic datasets are gathered in Beijing and Shanghai, the largest two cities in China with millions of vehicles. The time period is both three weeks, from Aug. 6th to Aug. 26th, 2018. We have defined the same data format for each dataset, consisting of heterogeneous sensory data from multiple sources. Each dataset contains GPS trajectories, inertial data, road network, and auxiliary information (weather index, holiday index, the day in the week). In addition, the GPS points on each trace have been projected onto the road link via a Map Matching algorithm [Jagadeesh *et al.*, 2004]. Table 1 depicts the details for each dataset.

In this experiment, we implement our model in Python using PyTorch toolbox, and train the model on 1080Ti with 32GB memory. A typical training process takes about 30 minutes on link typology representation, and about 17 hours on the multi-task learning for traffic speed prediction and travel time estimation.

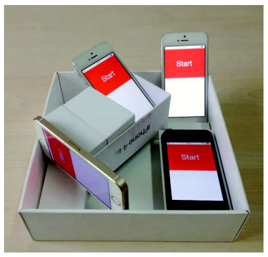
We further compare with Support Vector Regression (SVR) [Chun-Hsin Wu *et al.*, 2004] and DeepTTE [Wang *et al.*, 2018a]. SVR uses SVM for regression, which has been widely used in sequence prediction. DeepTTE is the state-of-the-art approach for travel time estimation and it is an open source project. We leverage the mean absolute percentage error (MAPE), the mean absolute error (MAE), and the mean square error (MSE) to measure the accuracy.

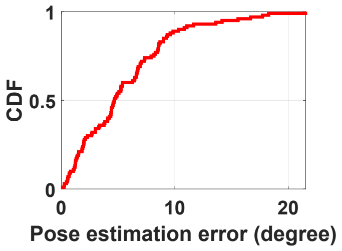
Dataset	Beijing	Shanghai
Time	8.6-8.26, 2018	8.6-8.26, 2018
Traces	3.9GB (410,882)	2.6GB (270,716)
Links	513KB (12,600)	350KB (8,500)
Inertial data	2.3GB	1.22GB

Table 1: Dataset information

Aggressive driving event	Precision	Recall
Speeding	91%	86%
Sudden brake	96%	92%
Lane shift	99%	95%

Table 2: Precision and recall of aggressive driving detection


(a) Mould



(b) Pose estimation

Figure 7: Phone pose estimation for four iPhones in a mould.

5.2 IMU Data Processing

Smartphone pose estimation. To measure the pose estimation accuracy, we use a mould to hold four iPhones with different poses (shown in Figure 8(a)), and measure their ground truth poses via a protractor. The Cumulative Distribution Function (CDF) curve of pose estimation errors is shown in Figure 8(b), with the 90-percentile error at 10 degrees.

Aggressive driving detection. To collect the ground truth, we take videos to record the aggressive driving events during driving. We evaluate the detection precision and recall in Table 2. Both the precision and recall of three aggressive driving events are over 86%. Sudden brakes are easily identified via static GPS locations and extreme inertial accelerations, while lane shifts are reliably identified by gyroscope.

5.3 Link Speed Prediction

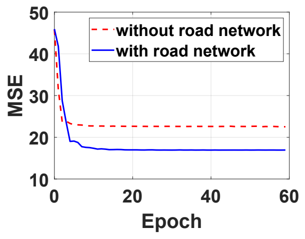
Figure 8(a) and Figure 8(b) show the MSE and MAE results on link speed prediction, respectively. We observe that the road network is effective to link speed prediction accuracy, reducing the MSE from 22.5 to 16.9, and reducing the MAE from 2.4 to 2.3.

Effect of attention. With the road network, our attention mechanism further reduces MSE from 16.9 to 16.2, and reduces MAE from 2.3 to 2.1. Figure 9 presents the weights of each attention channel, where we divide the time period with a 5-minutes interval, $t0$ presents the same time period in last week, and $t1 - t12$ denote 12 continuous time periods in last hour. Since our temporal feature is fine-grained, the time period in last 5 minutes ($t12$) has the largest weight.

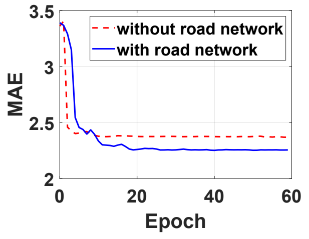
5.4 Travel Time Estimation

Road network	IMU	MSE	MAE	MAPE
ID	no	220290.5	280.4	23.9%
Topology	no	179017.5	255.5	23.1%
Topology	yes	177818.1	251.6	22.6%

Table 3: Travel time estimation with different inputs



(a) MSE



(b) MAE

Figure 8: Link speed prediction accuracy, with/out road network.

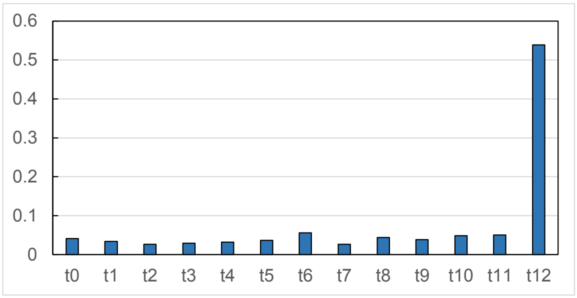


Figure 9: Attention weights.

MAPE w.r.t. travel time. Figure 10 shows the error bar graph on MAPE value for each trajectory with its travel time period. The travel time period is relatively wide, varying from 1 minute to more than 70 minutes among trajectories. As consistent with our common sense, longer travel time always causes worse prediction results but lower variances. An interesting observation is that extreme short travel time (e.g., less than 10 minutes) also leads to large prediction errors, due to variety of environmental interferences besides the travel time.

Effect of network structure. We evaluate the effects of ResNet module, road network representation, and IMU data in our neural network.

1) *ResNet*. With topology embedding of road network, we test the effect of ResNet in Figure 11. Since a neural network with deep layers is always difficult to converge, the ResNet module helps to converge our model much faster.

2) *Road network representation*. Table 3 shows the comparison between ID embedding and our topology embedding on road network. The ID embedding result is based on the initial embedding API in Pytorch. We observe that our topology embedding method for road network helps to converge the loss to a much lower level, e.g., reducing the MAPE with 0.8% (first two rows).

3) *IMU data*. In implementation, we concatenate the inertial features and input them with an FC layer and a $sigmoid$ function to get a scale factor, then amend the output of neural network with this scale factor (multiplication). Since the IMU data is very sparse at current DiDi platform, it only reduces the MAPE loss with 0.5% (details shown in Table 3). However, this experiment proves the effectiveness of IMU data, and we will test more fine-grained IMU data in the future for more improvements.

Effects of hyper-parameters. Figure 12(a) and Figure 12(b) show the effects on two hyper-parameters in our model, i.e., the dropout value and hidden state size, respec-

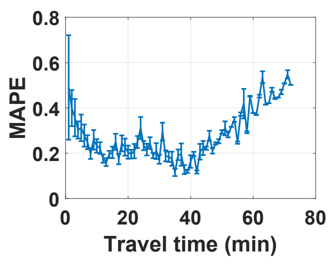


Figure 10: MAPE w.r.t. time.

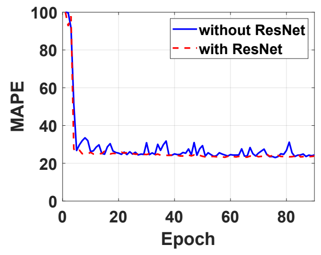
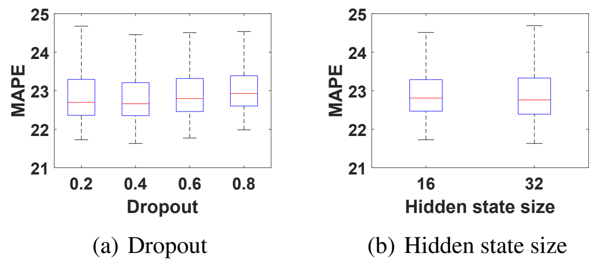


Figure 11: ResNet effect.



(a) Dropout (b) Hidden state size

Figure 12: Effects of two hyper-parameters.

	MSE	MAE	MAPE
SVR	15081.9	100.7	83.1%
DeepTTE	939957.4	392.5	25.8%
CTTE	177818.1	251.6	22.6%

Table 4: Performance on Beijing dataset

	MSE	MAE	MAPE
SVR	16226.0	103.1	100.6%
DeepTTE	675110.2	445.8	29.1%
CTTE	245917.4	304.7	24.3%

Table 5: Performance on Shanghai dataset

tively. We have set the dropout value with $\{0.2, 0.4, 0.6, 0.8\}$, and the number of LSTM hidden state size with 16 and 32. The experiment for each hyper-parameter set is repeated by 10 times. The corresponding MAPE value is all around 23%. Such experiment shows the robustness of our model with such hyper-parameters.

Comparison with SVR and DeepTTE. Table 4 and Table 5 depict the final accuracy of travel time estimation on two real-world datasets in Beijing and Shanghai, respectively. We observe that the SVR achieves the least MSE and MAE, but its MAPE is almost 4x than ours; the DeepTTE achieves similar MAPE as ours, but with an extreme high MSE; our approach CTTE produces the least MAPE, and maintains both MSE and MAE at very low level. Thus, our method is superior at all three loss metrics.

6 Related Work

Traffic speed prediction. Traffic speed prediction plays a fundamental role in intelligent transportation system. There are generally two kinds of methods for traffic speed prediction: the parametric and the non-parametric approaches. ARIMA [Williams and Hoel, 2003] is a classic parametric method and models the traffic in a stationary process. However, those parametric methods are known to be not suitable for large-scale data due to the heavy computation complexity. Some supervised learning approaches, e.g., SVR [Chun-Hsin Wu *et al.*, 2004] and LR [Ristanoski *et al.*, 2013], formulate the traffic speed prediction issue as a regression problem. Currently, CNN based models [Wang *et al.*, 2016c; Ma *et al.*, 2015] are used for traffic flow prediction based on large-scale historical traffic data, but they don't consider the link road topology information.

Travel time estimation. Travel time estimation is very important to location-based services for vehicle navigation applications. The existing approaches can be classified into two categories, the route-based solutions and the data-driven solutions. The first [Sevlian and Rajagopal, 2010; Pan *et al.*, 2012; Wang *et al.*, 2016b] estimates total travel time as the time summation on each road segment and intersection. The second [Rahmani *et al.*, 2013; Wang *et al.*, 2016a; Wang *et al.*, 2018a; Zhang *et al.*, 2018; Li *et al.*, 2018; Wang *et al.*, 2018b] formulates the travel time estimation as a multivariate time series prediction problem. However, they fail to consider general traffic conditions and personalized driving behaviors. Some recent work [Wang *et al.*, 2018b] begins to use the personalized information, but it is simply driver ID and largely relies on GPS data which are too coarse to model many fast driving events, e.g., lane shifts.

Driving behavior analysis. Thanks to the widely used smartphones for driving navigation, there have been several approaches to use the smartphone's inertial data to monitor the driving behaviors, e.g., dangerous driving alert [Lindqvist and Hong, 2011], traffic accidents detection [Mohan *et al.*, 2008], and road landmark detection [Gao *et al.*, 2017]. Among them, driving speed is the only critical factor to dangerous driving, while we also consider rotations. Besides the three aggressive driving events in this paper, we aim to identify other pivotal events, e.g., not remaining aloof. We plan to explore computer vision algorithms to detect the distance to a predecessor car via a dashboard camera.

7 Conclusion

In this paper, we propose CTTE, which takes GPS traces, smartphone inertial data, and road network to produce customized travel time estimation. It addresses one interesting problem to the ubiquitous vehicle location-based services: how much time your aggressive driving behavior saves. CTTE enables public traffic speed monitoring on each road link, and personalized travel time estimation based on unique driving patterns. We have conducted extensive experiments in large-scale real-world datasets at Beijing and Shanghai, and the results demonstrate the effectiveness of our method compared with the state-of-the-art.

Acknowledgments

This work is supported in part by NSFC 61702035 and DiDi GAIA Collaborative Research Funds for Young Scholars.

References

[BMV, 2019] Aggressive driving. https://www.in.gov/bmv/files/Drivers_Manual_Chapter_5.pdf, 2019.

[Chen *et al.*, 2018] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of AAAI*, pages 6468–6476, 2018.

[Chun-Hsin Wu *et al.*, 2004] Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281, 2004.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*, pages 1027–1035, 2016.

[Gao *et al.*, 2017] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, and G. Luo. Smartphone-based real time vehicle tracking in indoor parking structures. *IEEE Transactions on Mobile Computing*, 16(7):2023–2036, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of ECCV*, pages 630–645, 10 2016.

[Jagadeesh *et al.*, 2004] G. R. Jagadeesh, T. Srikanthan, and X. D. Zhang. A map matching method for gps based real-time vehicle location. *Journal of Navigation*, 57(3):429–440, 2004.

[Li *et al.*, 2018] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *Proceedings of KDD*, pages 1695–1704, 2018.

[Liang *et al.*, 2018] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *Proceedings of IJCAI*, pages 3428–3434, 2018.

[Lindqvist and Hong, 2011] Janne Lindqvist and Jason Hong. Undistracted driving: A mobile phone that doesn't distract. In *ACM HotMobile*, pages 70–75, 2011.

[Ma *et al.*, 2015] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187 – 197, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[Mohan *et al.*, 2008] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Using mobile smartphones for rich monitoring of road and traffic conditions. In *ACM SenSys*, pages 357–358, 2008.

[Pan *et al.*, 2012] B. Pan, U. Demiryurek, and C. Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604, 2012.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, pages 701–710, 2014.

[Rahmani *et al.*, 2013] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. Route travel time estimation using low-frequency floating car data. In *Proceedings of IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2292–2297, 2013.

[Ristanoski *et al.*, 2013] Goce Ristanoski, Wei Liu, and James Bailey. Time series forecasting using distribution enhanced linear regression. In *Advances in Knowledge Discovery and Data Mining*, pages 484–495. Springer Berlin Heidelberg, 2013.

[Sevlian and Rajagopal, 2010] Raffi Sevlian and Ram Rajagopal. Travel time estimation using floating car data. *CoRR*, abs/1012.4249, 2010.

[Shen *et al.*, 2018] Sheng Shen, Mahanth Gowda, and Romit Roy Choudhury. Closing the gaps in inertial motion tracking. In *Proceedings of MobiCom*, pages 429–444, 2018.

[Wang *et al.*, 2016a] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. A simple baseline for travel time estimation using large-scale trip data. In *Proceedings of ACM SIGSPATIAL*, 2016.

[Wang *et al.*, 2016b] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong. Traffic speed prediction and congestion source exploration: A deep learning method. In *Proceedings of IEEE ICDM*, pages 499–508, 2016.

[Wang *et al.*, 2016c] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong. Traffic speed prediction and congestion source exploration: A deep learning method. In *Proceedings of ICDM*, pages 499–508, 2016.

[Wang *et al.*, 2018a] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of AAAI*, 2018.

[Wang *et al.*, 2018b] Zheng Wang, Kun Fu, and Jieping Ye. Learning to estimate the travel time. In *Proceedings of KDD*, pages 858–866, 2018.

[Williams and Hoel, 2003] Billy M. Williams and Lester A. Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of Transportation Engineering*, 129:664–672, 2003.

[Yao *et al.*, 2017] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of WWW*, pages 351–360, 2017.

[Zhang *et al.*, 2018] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. In *Proceedings of IJCAI*, 2018.