Multi-task Representation Learning for Travel Time Estimation

Yaguang Li*
University of Southern California
Los Angeles, United States
yaguang@usc.edu

Kun Fu
DiDi AI Labs
Beijing, China
fukunkunfu@didichuxing.com

Zheng Wang
DiDi AI Labs
Beijing, China
wangzhengzwang@didichuxing.com

Cyrus Shahabi
University of Southern California
Los Angeles, United States
shahabi@usc.edu

Jieping Ye
DiDi AI Labs
Beijing, China
yejieping@didichuxing.com

Yan Liu
University of Southern California
Los Angeles, United States
yanliu.cs@usc.edu

ABSTRACT

One crucial task in intelligent transportation systems is estimating the duration of a potential trip given the origin location, destination location as well as the departure time. Most existing approaches for travel time estimation assume that the route of the trip is given, which does not hold in real-world applications since the route can be dynamically changed due to traffic conditions, user preferences, etc. As inferring the path from the origin and the destination can be time-consuming and nevertheless error-prone, it is desirable to perform origin-destination travel time estimation, which aims to predict the travel time without online route information. This problem is challenging mainly due to its limited amount of information available and the complicated spatiotemporal dependency. In this paper, we propose a MUlti-task Representation learning model for Arrival Time estimation (MURAT). This model produces meaningful representation that preserves various trip properties in the real-world and at the same time leverages the underlying road network and the spatiotemporal prior knowledge. Furthermore, we propose a multi-task learning framework to utilize the path information of historical trips during the training phase which boosts the performance. Experimental results on two large-scale real-world datasets show that the proposed approach achieves clear improvements over state-of-the-art methods.

KEYWORDS

travel time estimation; representation learning; multi-task learning

ACM Reference Format:
Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task Representation Learning for Travel Time Estimation. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220033

1 INTRODUCTION

Real-time estimation of vehicle travel time is essential for route planning [20], ride-sharing [1] and vehicle dispatching [34]. Most existing approaches [14, 29, 31, 33] for travel time estimation assume the availability of the actual route, while in online services only the origin and the destination are given before a trip. One solution is to first infer the most likely path, such as the time dependent fastest path [8], and then to estimate its travel time. However, this approach introduces expensive computations in the route planning step, making path-based approaches less practical for fast online travel time estimation services. Besides, this path-based approach heavily depends on the route planning, and its performance suffers when the actual path deviates from the predicted one due to changing traffic conditions and different user preferences.

The recent advance in addressing this problem is to perform origin-destination (OD) travel time estimation [16, 32], which aims to estimate the travel time without the actual route. Several major challenges exist in the OD travel time estimation problem. First, because of the absence of path information, only limited raw input features, i.e., the origin, the destination and the departure time, remain available for online prediction. Besides, these raw features are usually hard to utilize for a model, e.g., it is non-trivial to measure the similarity or road network distance between two roads based on their latitudes/longitudes. Thus, to deal with the insufficiency of the information provided by raw features, it is desirable to learn a feature representation that utilizes the rich information from the underlying road network structure and the spatiotemporal properties of the problem. To the best of our knowledge, all existing works make a straightforward use of raw features to build models, and very limited attention has been paid to finding a proper representation for transportation problems, especially for travel time estimation. In [32], Wang et al. propose a nearest neighbor based method, which estimates the trip duration by averaging the scaled travel times of all historical trips with similar origins and destinations. However, this non-parametric approach is hard to generalize to cases in which no or very limited number of neighbors are available. In [16], Jindal et al. propose ST-NN, a multi-layer feed-forward neural network for travel time estimation. ST-NN first takes as input the discretized latitudes and longitudes of the origin and destination to predict the travel distance and then combines this prediction with the time information to estimate the trip duration.

[*] Work primarily done while visiting DiDi AI Labs.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220033

One common limitation of these approaches is that the underlying road network structure and the spatiotemporal properties are largely overlooked.

In this paper, we propose a novel solution for the OD travel time estimation problem. It learns better representations from limited raw features, leveraging the road network structure and spatiotemporal prior knowledge. Specifically, we model the underlying road network as a graph of links/roads and learn representation for each link considering the road network topology. Besides, to enforce the spatiotemporal smoothness prior on the learned representations, i.e., timestamps/locations that are close to each other should have similar representations, we construct the spatial and temporal graphs and utilize graph Laplacian regularization on those constructed graphs. Moreover, to learn more meaningful representations, we enforce the learned representations not only being optimized for estimating the travel time, but also capturing various path information. Specifically, we propose a multi-task learning framework which models various trip properties, e.g., the distance, the number of traveled road segments, as additional tasks during the training process. This framework not only produces more meaningful representations, but also boosts the learning performance.

In summary, the main contributions of this paper are as follow:

- We propose a novel representation learning framework for the origin-destination travel time estimation problem.
- We propose approaches to leverage the underlying road network structure as well as the spatiotemporal smoothness prior to deal with the insufficiency of the input information.
- We propose a multi-task learning approach to utilize the path information in the training phase to learn more meaningful representations, which boosts the learning performance.
- We conduct extensive experiments on two real-world large-scale trip datasets. The proposed approach clearly outperforms state-of-the-art methods for OD travel time estimation.

The rest of this paper is organized as follows. In Section 2, we discuss related work. We define the problem in Section 3, and present the proposed multi-task representation model in Section 4. In Section 5, we discuss experiment results and Section 6 concludes the paper.

2 RELATED WORK

2.1 Travel Time Estimation

There are mainly two types of approaches for travel time estimation: path-based method and origin-destination based method. Path-based methods require the route information to generate a prediction while origin-destination based methods are able to predict the travel time without route information.

Path-based travel time estimation. A straightforward approach is to first estimate the travel time on individual links and then sum up all the link travel times in the given path. The travel time of each link can be estimated using loop detector data [2, 15, 27, 31] or floating car data [9, 14, 33]. However, this method fails to consider the transition times between links, e.g., traffic lights, left/right turns. In [13, 19], the authors propose a method that considers the time spent on intersections, and in [29], Rahmani et al. propose to concatenate sub-paths to achieve more accurate travel time estimation. In [33], Wang et al. further improve this sub-path based method by first mining frequent patterns [30] and then finding the optimal way to concatenate frequent paths that balance the length and the support. The main drawback of the path-based methods is that they require estimating the path which is time consuming and error-prone.

OD travel time estimation. To mitigate this issue, recently a few works start investigating the Origin-Destination (OD) travel time estimation. In [32], the authors design a nearest neighbor based method for OD travel time estimation. This approach estimates the trip duration by averaging the scaled travel times of all historical trips with a similar origin, destination and time of the day. In [16], the authors propose the ST-NN, a multi-layer feed-forward neural network for travel time estimation. ST-NN first predicts the travel distance given an origin and a destination and then combines this prediction with the time information to estimate the travel time. One drawback of these methods is that the underlying road network structure, as well as the spatiotemporal property are largely overlooked. In this paper, we propose an approach to leverage the topological information as well as the spatiotemporal prior knowledge for OD travel time estimation. In Section 5, we will compare the proposed approach with these approaches as baselines.

2.2 Representation Learning

The performance of machine learning models heavily depends on the choice of data representation [6]. *Representation Learning* aims to learn representations from data that make it easier to extract useful information for various tasks. An overview of representation learning is available in [6]. In the case of OD travel time estimation, we want to learn representations from the raw trip data, the underlying road network structure as well as the spatiotemporal property. One benefit of explicitly dealing with representations is that we can conveniently express many general priors about the task. Specifically, in this paper we exploit the following priors in learning interpretable representations for travel time estimation.

Spatial and temporal smoothness. Consecutive or spatially nearby observations tend to be associated with the similar values. This prior was first introduced by Becker & Hinton in [4], and it can be enforced by penalizing changes in values over time or space. In [24], the authors apply the temporal smoothness prior to video modeling, and in [3], graph Laplacian is used to enforce spatial smoothness. In this work, we enforce this prior to the representation of links and spatiotemporal factors using unsupervised graph embedding and graph Laplacian regularization.

Shared factors across tasks. Many tasks of interest are explained by factors that are shared with other tasks [6]. By jointly optimizing several related tasks, we can learn representations that capture underlying factors, and achieve better empirical results. In this paper, we exploit this prior by jointly learning travel time estimation as well as many other related real-world tasks that potentially share common factors, e.g., predicting the travel distance and the number of road segments in the path.

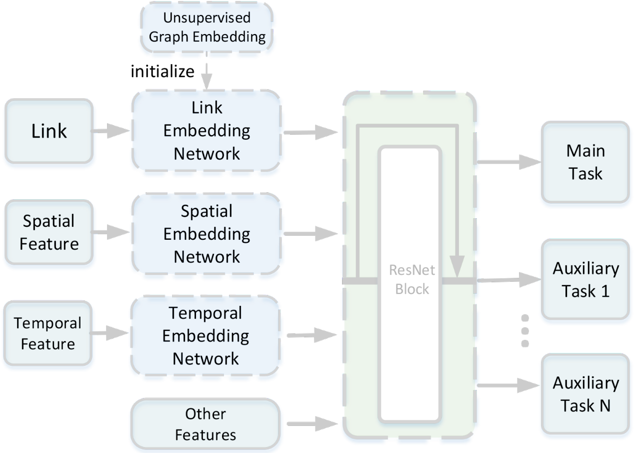


Figure 1: The system architecture of the proposed multi-task representation learning model (MURAT) for travel time estimation. The model first embeds the raw link information and spatiotemporal information into the learned spaces. Then the learned representations together with other numerical features are fed into a deep residual network which is jointly trained using supervised signals from multiple related tasks.

3 PROBLEM STATEMENT

Definition 3.1 (Path). A path P is defined as a sequence of GPS observations. Each observation consists of the location, i.e., (latitude, longitude), and the time.

Definition 3.2 (Trip). A trip $x^{(i)} = (o^{(i)}, d^{(i)}, t^{(i)}, \tau^{(i)}, P^{(i)})$ is defined as a tuple with five components where $o^{(i)}$ denotes the origin location, $d^{(i)}$ denotes the destination, $t^{(i)}$ denotes the departure time, $\tau^{(i)}$ denotes the duration and $P^{(i)}$ represents the corresponding path of this trip.

Then, the historical trip dataset with N trips can be represented as a set of trips $\boldsymbol{X} = \{\boldsymbol{x}^{(i)} | i = 1, 2, \cdots N\}$.

Definition 3.3 (Road Network). The underlying road network can be represented as an undirected graph of links/roads $\mathcal{G} = (\mathcal{V}, \boldsymbol{A})$ where $\mathcal{V}$ is the set of vertices representing links in the road network, while $\boldsymbol{A}$ denotes the connectivity among links, $\boldsymbol{A}_{ij}$ is 1 if and only if link i and link j are directly connected.

Problem 1 (OD Travel Time Estimation). *Given an origin, destination and departure time, our goal is to estimate the duration using the set of historical trip dataset $\boldsymbol{X}$ as well as the underlying road network $\mathcal{G}$.*

This problem is challenging because of (1) the complicated spatiotemporal dependency in the underlying road network and (2) the limited amount of available information when performing on-line prediction. In the following section, we describe the proposed multi-task representation learning model to deal with these challenges.

4 PROPOSED APPROACH

In this section, we introduce the proposed model, i.e., MURAT, to learn the representation for the origin-destination travel time estimation problem. First, we describe the approach to incorporate the underlying road network structure into the model. Then we show how to integrate spatiotemporal prior knowledge into the representation learning process. Finally, we introduce the proposed multi-task representation learning framework.

4.1 Representation Learning for Road Network

Travel time is strongly affected by the underlying road network structure. For example, two nearby locations in the opposite directions of the highway might have significantly different travel times with regards to the same destination. This is because of the restriction imposed by the underlying road network, preventing vehicles from moving freely in the Euclidean space. Thus, it is desirable to incorporate the road network structure into the learning process. One method is to identify the corresponding link of the location. The intuition is that nearby locations *on the same link* will have similar travel times with regards to the same destination. Note that, in the previous example, those two nearby locations in the opposite directions will reside on different links.

Given a location, we can easily match it with a link in the underlying road network using map-matching [23]. One potential way of representing the link information is to feed the link identifier either as a numerical feature or a one-hot encoded categorical feature. However, these methods fail to capture the network structure as each link is defined and learned independently. One alternative is to use unsupervised graph embedding approaches, e.g., Laplacian Eigenmap [5], DeepWalk [26], to learn a representation for each link. While these representations preserve similarities defined on the graph, it is hard for them to utilize task specific supervised signals.

Graph Laplacian Regularization. Several approaches can be used to tackle this problem. One way is to use the combination of the supervised loss and the unsupervised one as the objective. Following the practice in [5], we add the graph Laplacian as an additional unsupervised loss. Specifically, this loss serves as a regularizer that encourages adjacent links to have similar representations. Let $\mathcal{G} = (\mathcal{V}, \boldsymbol{A})$ denotes the undirected link graph, the graph Laplacian can be represented as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D} = \text{diag}(\boldsymbol{A}\boldsymbol{1})$ is the diagonal degree matrix, and $\boldsymbol{1} \in \mathbb{R}^{|\mathcal{V}|}$ denotes the all one vector. Let $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d_l}$ denotes the link embedding matrix, with $\boldsymbol{E}_{i,:} \in \mathbb{R}^{d_l}$ corresponds to the representation of the vertex v_i. Let ℓ denote the supervised loss function, e.g., the mean absolute error for the travel time, then we have the following objective function:

$$\tilde{\ell} = \ell + \alpha \text{Tr}(\boldsymbol{E}^\intercal \boldsymbol{L} \boldsymbol{E}) = \ell + \alpha \sum_{i,j} A_{ij} \|\boldsymbol{E}_{i,:} - \boldsymbol{E}_{j,:}\|^2 \tag{1}$$

where α is the parameter to control the strength of the regularization. One drawback of this approach is that for large graphs with millions of edges, e.g., the trip dataset in Beijing used in this work, this objective can become too costly to optimize as in each training step we have to calculate the large matrix multiplication and possibly the corresponding gradients.

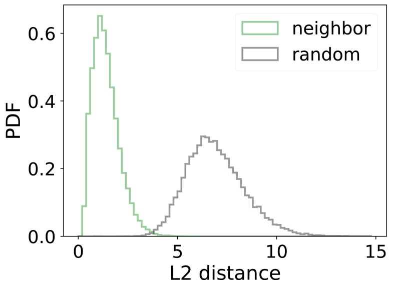
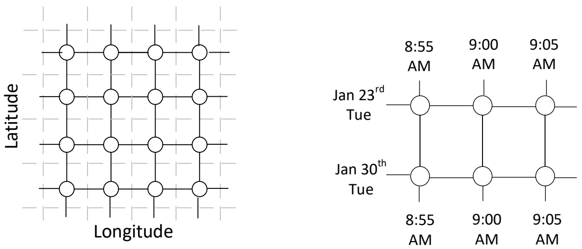


Figure 2: Distribution of distances between links calculated based on the Deepwalk embedding.



(a) Spatial graph.

(b) Temporal graph.

Figure 3: Graph-based regularization for spatiotemporal dependencies.

Unsupervised Pre-training. Inspired by the well-known practice of unsupervised pre-training [10] in which researchers used the pre-trained stacked restricted Boltzman machine to initialize a supervised classification deep neural network, we find a more efficient approach that achieves similar performance. Specifically, we first learn a representation of each link based on unsupervised graph embedding techniques, e.g., [5, 26], and then we use the learned representation as the initialization of the link embedding which will be fine-tuned later based on supervised signals. The question of why unsupervised pre-training could be helpful was extensively studied in [10] which tries to explain this phenomenon from the perspective of regularization effect and the optimization effect. Figure 2 shows the distribution of distances calculated based on the DeepWalk embedding. We observe that the $L2$ distance between embeddings of neighborhood links are much smaller than that of two random links. We hypothesize that this initialization has a similar effect to the graph Laplacian regularizer.

4.2 Spatiotemporal Representation Learning

Besides the road network structure, another important aspect of learning the representation is our prior knowledge of the spatial and temporal domain, e.g., the spatiotemporal smoothness, the periodicity and recurring nature of traffic. We integrate this prior knowledge by constructing the spatial graph and the temporal graphs in the embedding space.

Figure 3a shows an example of the spatial graph. Each node represents an equal-sized grid/region, and a node is connected to adjacent ones, i.e., nearby regions. Figure 3b shows an example of the temporal graph which aims to model the temporal smoothness as well as the weekly periodicity. In Figure 3b, each node, representing a 5 minutes time interval, is connected to its two adjacent neighbors and its counterparts at the same time of the week. In both the spatial and temporal graphs, each node is embedded as a fixed length vector, then we enforce the spatiotemporal smoothness prior and periodicity by applying the graph Laplacian regularization over these embedding vectors.

Besides, we use distributed representations for both the spatial and the temporal embeddings. For the spatial representation, the embedding of a node, i.e., a spatial grid, is represented as the concatenation of the embeddings of its latitude and longitude, while for the temporal representation, the node i.e., a temporal interval, is represented as the concatenation of the embedding of "time in day" and "day in week". Compared to representing each location/time interval separately, this distributed representation has the following main advantages: (1) fewer number of parameters, e.g., from $O(\#\text{grids})$ to $O(\sqrt{\#\text{grids}})$ for spatial representations, and (2) by sharing embeddings, it implicitly imposes the priors that locations in the same latitude /longitude, and intervals at the same time of the day or day of the week should have similar representations.

In addition to the road network structure and the spatiotemporal prior knowledge, we further enforce the model to learn more meaningful representations leveraging the path information available during the training period. This leads to the proposal of the multi-task representation learning framework.

4.3 Multi-task Representation Learning

Instead of using the path information as extra input features which are not available during testing, we extract various summaries from the path, e.g., the travel distance, the number of links in the trip, the number of traffic lights and the number of turns, and use them as auxiliary tasks to be predicted. Specifically, a multi-task learning framework is proposed to jointly learn the main task, i.e., predicting the travel time, as well as various auxiliary tasks, i.e., predicting different path summaries. We use a hard parameter sharing framework where different tasks sharing most part of the model except for having a dedicated output layer for each task.

Suppose, $\widehat{\boldsymbol{y}}^{(i)}$ denotes the prediction,and $\ell_k(\widehat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)})$ denotes the loss function of the kth task. The final learning objective is defined as a function of the individual loss functions.

$$\ell(\widehat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)}) = f_\ell([\ell_1(\widehat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)}), \cdots, \ell_k(\widehat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)})]) \qquad (2)$$

For example, the function f_ℓ can be the weighted sum function, the maximum function or other suitable aggregation functions. In the experiment, we use the weighted combination of the losses of all tasks, with the corresponding weights as hyperparameters.

To map the learned representations to target tasks, we use deep residual networks [12] with pre-activation which empirically offers better results than deep feed forward neural networks.

Figure 1 shows the architecture of the model and Algorithm 1 gives an overview of the learning process. We first initialize the link embedding network with unsupervised graph embedding, e.g., DeepWalk, learned on the link graph $\mathcal{G}_L$ and initialize the spatial/temporal embedding networks with Gaussian random noise. In

```
Algorithm 1: Multi-task Representation Learning (MURAT)
  Data: Link graph $\mathcal{G}_L$, Spatial graph $\mathcal{G}_S$, temporal graph $\mathcal{G}_T$
  Result: Learned representations: $E_L, E_S, E_T$
  begin
      $E_L \leftarrow$ GraphEmbed($\mathcal{G}_L$) // Unsupervised pre-training
      $E_S, E_T \leftarrow \mathcal{N}(0, 1)$
      for $i \leftarrow 1 \cdots N$ do
          $\mathbb{L}^{(i)} \leftarrow []$
          $\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)} \leftarrow$ sample($data, i$)
          $E_{L,\boldsymbol{x}^{(i)}} \leftarrow$ Embed($E_L, \boldsymbol{x}^{(i)}$) // Link embedding
          $E_{S,\boldsymbol{x}^{(i)}} \leftarrow$ Embed($E_S, \boldsymbol{x}^{(i)}$) // Spatial embedding
          $E_{T,\boldsymbol{x}^{(i)}} \leftarrow$ Embed($E_T, \boldsymbol{x}^{(i)}$) // Temporal embedding
          $\widehat{\boldsymbol{y}}^{(i)} \leftarrow$ ResNet($\Theta, \left[E_{L,\boldsymbol{x}^{(i)}}, E_{S,\boldsymbol{x}^{(i)}}, E_{T,\boldsymbol{x}^{(i)}}\right]$)
          // Aggregate losses from multiple tasks
          for $k \leftarrow 1 \cdots K$ do
              $\mathbb{L}^{(i)} \leftarrow \left[\mathbb{L}^{(i)}, \ell_k(\widehat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)})\right]$
          end
          $\ell^{(i)} \leftarrow f_\ell(\mathbb{L}^{(i)}) + \lambda_S \ell_{\mathcal{G}_S}(E_S) + \lambda_T \ell_{\mathcal{G}_T}(E_T)$
          $E_L, E_S, E_T, \Theta \leftarrow$ AdamOpt($\left[E_L, E_S, E_T, \Theta\right], \ell^{(i)}$)
      end
      return $E_L, E_S, E_T$
  end
```

the training process, for each sample $\boldsymbol{x}^{(i)}$, we embed its link information, spatial and temporal information into the learned spaces, i.e., $E_{L,\boldsymbol{x}^{(i)}}$, $E_{S,\boldsymbol{x}^{(i)}}$, and $E_{T,\boldsymbol{x}^{(i)}}$. Then the embedded representation together with other numerical features are fed as input into a deep residual network parameterized by Θ which generates the prediction for all tasks. The objective is defined as the aggregated loss from multiple tasks as well as the graph Laplacian regularizers for the spatial and the temporal graphs. Finally, the embeddings, as well as the weights of the residual network, are jointly optimized using the Adam Optimizer [18].

5 EXPERIMENTS

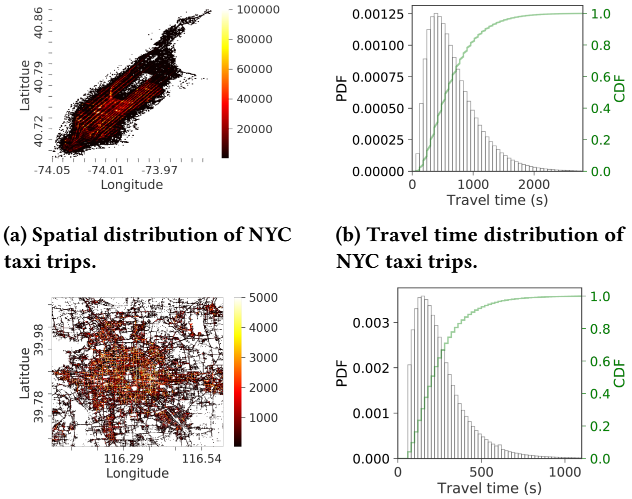
5.1 Dataset

BJS-Pickup. This dataset contains 61.4 millions of pickup trips in Beijing from May 1st 2017 to Oct 31st 2017 collected by Didi Chuxing. Each pickup trip contains source, destination, departure time, travel time and the path. We extract various path summaries including travel distance, number of roads, number of lights, turns, as auxiliary tasks. For data split, we use the data from May 1st 2017 to Oct 16th 2017 for training, data from Oct 17th 2017 to Oct 23th 2017 as validation, and the data from Oct 24th 2017 to Oct 31st 2017 is used for testing. For the road network, we use a commercial map of Beijing provided by Didi Chuxing from which we build the link graph based on the link connectivity information. Mapmatching [21] is used to get the corresponding link information for each location.

BJS-Small. We also extract a subset of the *BJS-Pickup* dataset, i.e., *BJS-Small*, for ablation study. It contains the 4.8 millions of trips in the right central part of the downtown area in Beijing which is more congested and more challenging for travel time estimation.

Table 1: Datasets used in the experiments.

Name	*BJS-Pickup*	*BJS-Small*	*NYC-Trip*
# Samples	61.4M	4.8M	21.9M
Avg. trip time	191s	335s	660s
# Links	1.1M	30K	73K



(a) Spatial distribution of NYC taxi trips.

(b) Travel time distribution of NYC taxi trips.

(c) Spatial distribution of Beijing pickup trips.

(d) Travel time distribution of Beijing pickup trips.

Figure 4: Data Statistics. *BJS-Pickup* contains trips that have smaller duration but broader spatial distribution than *NYC-Trip*.

NYC-Trip. To compare with existing approaches, we also conduct experiments on a publicly available dataset processed by [32], which contains 21.9 millions of taxi trips collected from Nov 1st 2013 to Dec 31st 2013 in New York City. Each trip records locations of the origin and the destination, departure time, along with the trip summary information including trip duration and distance. We follow the settings in [32] and use the data in November for training, and the data in December for testing. For the road network, we use the map of New York City provided by OpenStreetMap [11].

Table 1 shows the statistics of these datasets and the corresponding underlying road networks. Figure 4 plots the heatmap of pickup/drop-off locations and the travel time distributions of these two datasets.

5.2 Experimental Settings

Methods for evaluation. We compare the proposed model (MURAT) with the following methods for OD travel time estimation.

- Linear regression (LR):, which models the travel time as a linear function of the Euclidean distance and L1 distance. This simple method serves as a baseline for comparison.

- Gradient boosted machine (GBM) [17]: gradient boosting decision tree based regression implemented using LightGBM [17], The input features include time in day, day in the week, Euclidean distance, start location, end location, etc. We use maximum 500 trees with the learning rate equals to 0.1. Early stopping based on the validation dataset is used.
- Spatial temporal deep neural network (ST-NN) [16]: a deep neural network based approach which first predicts the travel distance given an origin and a destination and then combines this prediction with the time information to estimate the travel time. We implement this algorithm following the parameter settings suggested in [16]. For the experiment on the *BJS-Trip* dataset, we further tune its hyperparameters and add additional features to achieve the best performance.
- TEMP+R [32], a nearest neighbor based approach which estimates the trip duration by averaging the scaled travel times of all historical trips with similar origin and destination. The travel times of neighborhoods are scaled based on the region-based temporal speed reference.
- MURAT-NR, the variant of the proposed approach which has the same input, output and similar amount of learnable parameters to MURAT, but without explicit representation learning, i.e., we directly feed the raw features as input to the model.

All the deep neural network based approaches, including ST-NN, MURAT-NR and MURAT, are implemented using PyTorch [25]. The default experimental settings for MURAT are as follow. For the link embedding, the dimension is 40, and DeepWalk [26] is used for unsupervised pre-training. For the spatial graph, each grid is connected to 4 adjacent grids with equal weights, and the dimensions for both the "latitude" and the "longitude" are 20. For the temporal graph, each vertex corresponds to a 5-minute interval. We connect each vertex to their four neighbors with weights equal to 1. The temporal information includes "time in day" and "day in week" both of which have 20 dimensions. The deep residual network contains 5 residual network blocks [12], with 11 layers in total and each hidden layer contains 1024 units. The objective is MAE for *NYC-Trip* and MAPE for *BJS-Pickup*, optimized using Adam [18] with mini-batch size equals to 1024. The initial learning rate is 10^{-2}, and reduces to $\frac{1}{5}$ every 2 epochs. Early stopping on the validation dataset is used. In the multi-task learning, weighted linear function is used to aggregated loss function from different tasks and the best hyperparameters are chosen using the Tree-structured Parzen Estimator (TPE) [7] on the validation dataset.

Evaluation metrics. We evaluate the performance of proposed method based on three popular metrics. Suppose $\boldsymbol{x} = x^{(1)}, \cdots, x^{(N)}$ represents the ground truth, $\hat{\boldsymbol{x}} = \hat{x}^{(1)}, \cdots, \hat{x}^{(N)}$ represents the predicted values, and N denotes the number of samples, these metrics are defined as follow:
Mean Absolute Percentage Error (MAPE),

$$\mathrm{MAPE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N} \sum_{i}^{N} \left| \frac{x^{(i)} - \hat{x}^{(i)}}{x^{(i)}} \right|$$

Table 2: Performance comparison of evaluated approaches.

Dataset	Metric	LR	GBM	ST-NN	TEMP+R[1]	MURAT-NR	MURAT
NYC-Taxi	MAPE	42.43%	24.80%	24.25%	-	23.29%	**22.32**%
	MAE	213.88	178.30	149.40	145.15	145.96	**139.44**
	MARE	32.40%	27.02%	22.63%	22.10%	22.11%	**20.96**%
BJS-Pickup	MAPE	46.53%	35.82%	33.52%	-	30.37%	**26.81**%
	MAE	96.97	86.20	83.57	-	85.83	**75.60**
	MARE	37.72%	33.53%	32.51%	-	33.38%	**29.39**%

Mean Absolute Error (MAE),

$$\mathrm{MAE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N} \sum_{i}^{N} \left| x^{(i)} - \hat{x}^{(i)} \right|$$

and Mean Absolute Relative Error (MARE).

$$\mathrm{MARE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{\sum_{i}^{N} \left| x^{(i)} - \hat{x}^{(i)} \right|}{\sum_{i}^{N} |x^{(i)}|}$$

5.3 Performance Comparison

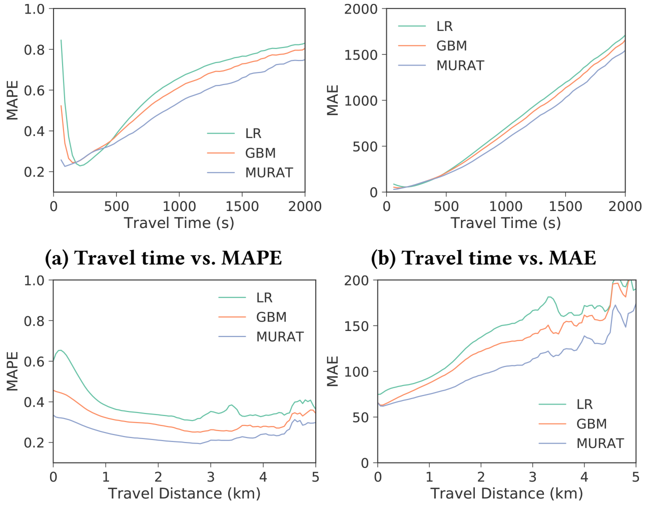
Table 2 shows the comparison of evaluated approaches for travel time estimation on both datasets. We have the following observations:

- MURAT achieves the best performance regarding all the metrics in both datasets outperforming the state-of-the-art approaches by $4\% - 10\%$ in terms of MAE. This suggests the effectiveness of proposed multi-task representation learning model.
- The performance of MURAT-NR is much worse than MURAT even with the same input/output and roughly the same amount of parameters. This demonstrates the importance of the explicit representation learning.
- The benefit of representation learning is more significant on the *BJS-Pickup* dataset than that on the *NYC-Trip* dataset. This may due to the relatively better map quality used for *BJS-Trip* (which results in better coverage and better link matching accuracy).

Performance w.r.t. Travel Time. Figures 5a and 5b show the relationship between the travel time and different metrics for evaluated approaches. As expected, the trips with large duration have higher MAPE and MAE. An interesting observation is that with the increase of travel time, MAPE first decreases dramatically and then gradually increases. This is because there is certain amount of randomness in a trip, and MAPE will be dominated by this randomness when the travel time, i.e., the denominator, is small.

Performance w.r.t. Travel distance. The relationship between the travel distance and different metrics are shown in Figures 5c and 5d. It is interesting to see that MAPE first decreases and then increases as the travel distance grows. This reflects the joint effects of the increasing denominator and the numerator, i.e., a larger travel distance usually means more uncertainties as well as a longer travel time.

[1] The numbers are copied from [32]. If near real-time trip data, i.e., trips' data from one hour ago, are available, its MAE and MARE will decrease to 142.73 and 21.73% respectively.



(a) Travel time vs. MAPE

(b) Travel time vs. MAE

(c) Travel distance vs. MAPE

(d) Travel distance vs. MAE.

Figure 5: Performance w.r.t. different trip features on the *BJS-Pickup* dataset.

Table 3: Performance comparison of approaches with different link representations on the *BJS-Small* dataset.

Metric	Raw	RandEmb	UnsupEmb	SupEmb	SupEmb+Pre
MAPE	29.05%	28.76%	28.31%	28.12%	27.85%
MAE	95.47	94.22	92.95	91.18	90.80
MARE	31.04%	30.63%	30.22%	29.64%	29.53%

5.4 Effect of Link Embedding

To investigate the effect of link embedding, we conduct experiments with the following variants of the proposed model on the *BJS-Small* dataset.

- Raw: model without link embedding;
- RandEmb: which embeds each link as a vector of random Gaussian noise.
- UnsupEmb: which uses unsupervised graph embedding, i.e., DeepWalk [26], to generate the link representation.
- SupEmb: supervised embedding where the embedding of each link is first initialized as a vector of random Gaussian noise;
- SupEmb+Pre: Supervised embedding with unsupervised pretraining using DeepWalk.

All these models have identical inputs/outputs and are adjusted to have roughly the same number of parameters by increasing/decreasing the number of hidden units. Thus, the main difference lies in the way of representing the link information.

Table 3 shows the effect of link embedding. We observe that (1) Raw, which using raw link id instead of embedding, has significantly worse performance, (2) unsupervised embedding generated by DeepWalk, which captures the graph structure, results in significantly better results than RandEmb, (3) task specific supervised embedding outperforms unsupervised embedding, and (4) by utilizing both the graph structure information and the supervised signal, SupEmb+Pre achieves the best result.

Table 4: Performance comparison of approaches with different spatiotemporal representations on the *BJS-Small* dataset.

Metric	RawST	SEmb	TEmb	STEmb
MAPE	28.15%	28.05%	27.99%	27.84%
MAE	92.76	92.44	91.79	90.80
MARE	30.16%	30.05%	29.84%	29.53%

Table 5: Effect of multi-task learning. Multi-task based approaches, i.e., ETA+Distance and All, achieve clearly better performance than single-task based approach.

Metric	ETA	ETA+Distance	All
MAPE	28.43%	28.09%	27.84%
MAE	92.71	92.18	90.80
MARE	30.14%	29.97%	29.53%

5.5 Effect of Spatiotemporal Embedding

To investigate the effect of spatiotemporal embedding, we conduct experiments with the following variants of the proposed model on the *BJS-Small* dataset:

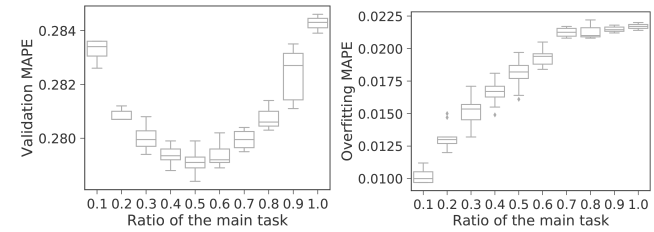
- RawST: neither the spatial nor the temporal embedding is used.
- SEmb: only the spatial embedding is used.
- TEmb: only the temporal embedding is used.
- STEmb: both the spatial and the temporal embeddings are used.

All these models have identical inputs/outputs, and are adjusted to have rougly the same number of parameters by increasing/decreasing the number of hidden units. Table 4 shows the effect of spatial and temporal embedding. We observe that: (1) using raw spatiotemporal information results in clearly worse performance, and (2) temporal embedding has a higher impact on the performance than spatial embedding. The latter observation maybe due to the fact that part of the spatial information has already been captured by link embedding.

5.6 Effect of Multi-task Learning

To investigate the effect of multi-task learning, we conduct experiments with the following variants of the proposed model on the *BJS-Small* dataset. Tasks include predicting the travel time, the travel distance, the number of links in the trip, the number of traffic lights and the number of turns.

- ETA: only the travel time is used as the supervised signal.
- ETA+Distance: the weighted combination of travel time and travel distance is used as the objective.
- All: all the supervised signals are used.



(a) Ratio vs. MAPE **(b) Ratio vs. Overfitting MAPE**

Figure 6: Effect of the ratio of the main task. Incorporating auxiliary tasks help reduce overfitting.

Table 5 shows the performance of different variants. We observe that: (1) the multi-task learning framework offers clearly better performance and (2) adding more relevant auxiliary tasks results further improves the performance.

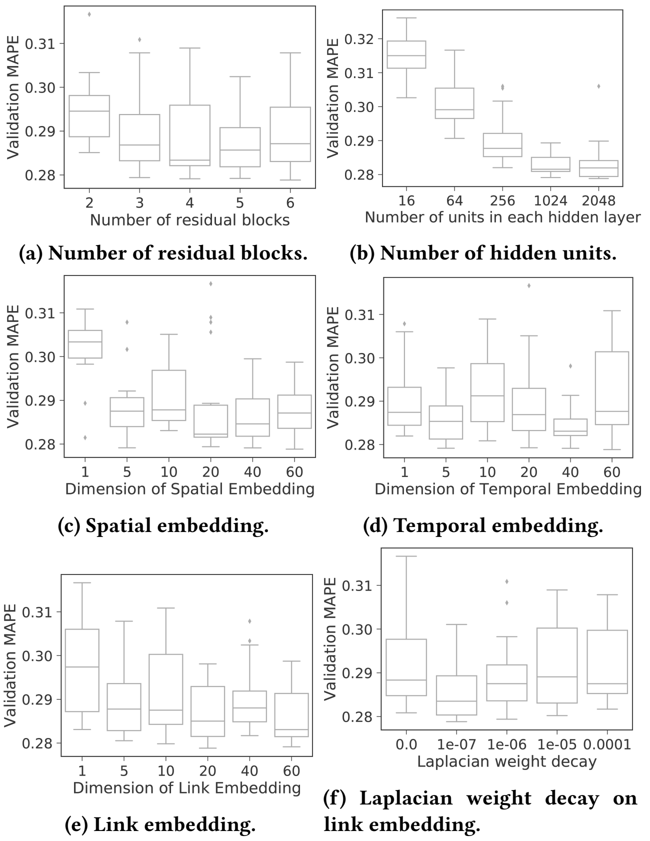
To study the effects of multi-task learning in terms of the error and overfitting of the main task, we train 200 models on the *BJS-Small* dataset with different weights of tasks. Figure 6 shows the effect of task weights. The x-axis denotes the ratio of the main task, where 1 means only using the main task and 0 means not using the main task at all. With the increase of the ratio, the error shows a U-style curve. This phenomenon of increased performance by introducing auxiliary tasks can be explained from the perspective of "shared factors across tasks" as discussed in Section 2.2. We hypothesize that these related tasks can be explained using a shared set of factors, and the multi-task learning framework exploits this commonalities among different learning tasks. Besides, as shown in Figure 6b, multi-task learning also has the effect of avoiding overfitting the main task.

5.7 Effect of Model architecture

To study the effects of different hyperparameters of the proposed model, we further train 200 models on the *BJS-Small* dataset with different combinations of hyperparameters including the number of residual blocks, the number of units in each hidden layer, the dimension of the location embedding, the dimension of the temporal embedding, the dimension of the link embedding and the weight of the graph Laplacian regularizer for link embedding.

Figure 7a shows the effect of the number of layers of the residual network. Generally, as the number of layers grows, the error first decreases, and then slightly increases. Note that, except for when the number of residual blocks is 2, there is no significant difference among the performances of different variants. This probably is due to the effect of identity mapping and batch normalization which gives the model's the ability to "skip" layers. Figure 7b shows the effect of the number of units in the residual network. We observe a clear trend of decreasing error as the number of units grows.

Figure 7c shows the effect of the dimension of spatial embedding. We observe a sharp error decrease when increasing the number of dimension from 1 to 5. After that, the performance become less sensitive to the dimension changes. This is because of the small spatial cardinality, i.e., the number of grids in each column/row. Figure7d shows the impact of the dimensions of temporal embedding. No clear trend is observed and the performance tends to be good as long as the dimension is not too small, i.e., greater than 1. This is because the cardinality of the temporal input, e.g., time in the day, day in the week, is also relatively small.



(a) Number of residual blocks. **(b) Number of hidden units.**

(c) Spatial embedding. **(d) Temporal embedding.**

(e) Link embedding. **(f) Laplacian weight decay on link embedding.**

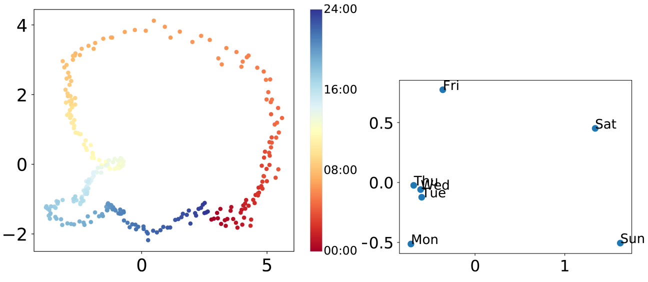
Figure 7: Effects of model parameters.

Figure 7e shows the effects of link dimension. Similar to the spatial embedding, with the increase in dimensions, the error first decreases and then becomes stable. Graph Laplacian regularization enforces the prior of local smoothness in the link embedding. As shown in Figure 7f, a proper weight decay, e.g., 10^{-7}, results in clearly improved performance.

5.8 Model Interpretation

To further understand the learned representation of the proposed model, we conduct a series of visualizations. Figure 8a shows the projection of the learned temporal embedding on the two largest principle components. Remarkably, the model automatically learns a feature representation that has a circular shape, from 00:00 to 24:00 with smooth transitions between adjacent time intervals. Besides, points tend to distribute more densely during the peak-hours, e.g., 7am-9am, 6pm-9pm, than non-peak hours.

Figure 8b shows the projection of the learned representation of day in the week. We observe that: (1) the representations of day in the week also forms a circle, from Monday to Sunday, (2) weekends are clearly separated from weekdays, and (3) Tuesday, Wednesday,



(a) Time in the day. **(b) Day in the week.**

Figure 8: Visualization of learned temporal representations. (a) The learned representation for time in day has a circular shape, from 00:00 to 24:00 with smooth transitions between adjacent time intervals. (b) Weekends are clearly separated from weekdays, where Tuesday, Wednesday, Thursday are quite close to each other, while Monday and Friday with different traffic patterns are relatively far away.

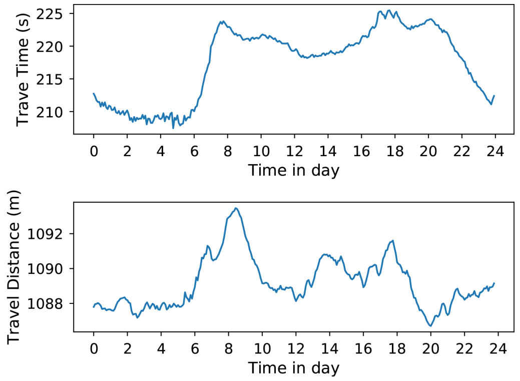


Figure 9: Learned travel time and distance patterns. In peak-hours, both the predicted travel time and travel distance increases as during peak-hours, drivers are more likely to take detours to avoid traffic congestions.

Thursday are quite close to each other, while Monday and Friday with different traffic patterns are relatively far away.

To study the spatiotemporal knowledge learned by the model, we randomly pick 10,000 origin-destination pairs from the validation dataset, and vary the departure time from 0:00 to 23:59. Then we calculate the averaged travel time and travel distances of these trips. Figure 9 shows the learned patterns about travel time and travel distance. The model generates time-varying travel times, where the travel time is larger in peak-hours and smaller in non-peak hours. One interesting observation is that the travel distances also change in different time, e.g., the two clear increases happen around 8am and 5pm. This reason is that during peak-hours, drivers are more likely to take detours to avoid traffic congestions. The results suggest that the proposed model learns a shared representation for different tasks.

6 CONCLUSION

In this paper, we proposed a novel representation learning solution, namely MURAT, to the origin-destination travel time estimation problem. Specifically, the model learned a representation that effectively captures underlying road network structures as well as spatiotemporal prior knowledge. We further introduced a multi-task learning framework to utilize path information in the origin-destination travel time estimation problem setting. When evaluated on two large-scale real-world trip datasets, our approach achieved clearly better performance than the state-of-the-art baselines.

The proposed framework is flexible to incorporate additional features. For future work, we will integrate real-time the traffic information [22] into the proposed representation learning model for origin-destination travel time estimation. In addition, it has been shown that DeepWalk is connected with normalized Laplacian [28], which provides possible directions to examine theoretical justifications for MURAT.

ACKNOWLEDGEMENT

This research has been funded in part by NSF grants CNS-1461963, IIS-1539608, Caltrans-65A0533, the USC Integrated Media Systems Center (IMSC), the USC METRANS Transportation Center and USC Annenberg Graduate Fellowship to Yaguang Li. The research is also partially supported by data retrieved from Didi Chuxing. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as NSF.

REFERENCES

[1] Mohammad Asghari, Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, and Yaguang Li. 2016. Price-aware real-time ride-sharing at scale: an auction-based approach. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 3.

[2] Mohammad Asghari, Tobias Emrich, Ugur Demiryurek, and Cyrus Shahabi. 2015. Probabilistic estimation of link travel times in dynamic road networks. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 47.

[3] Mohammad Taha Bahadori, Qi Rose Yu, and Yan Liu. 2014. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *Advances in neural information processing systems.* 3491–3499.

[4] Suzanna Becker and Geoffrey E Hinton. 1992. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature* 355, 6356 (1992), 161.

[5] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems.* 585–591.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[7] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems.* 2546–2554.

[8] Ugur Demiryurek, Farnoush Banaei-Kashani, Cyrus Shahabi, and Anand Ranganathan. 2011. Online computation of fastest path in time-dependent spatial networks. In *International Symposium on Spatial and Temporal Databases.* Springer, 92–111.

[9] Zhiming Ding, Bin Yang, Ralf Hartmut Güting, and Yaguang Li. 2015. Network-matched trajectory-based moving-object database: Models and applications. *IEEE Transactions on Intelligent Transportation Systems* 16, 4 (2015), 1918–1928.

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11, Feb (2010), 625–660.

[11] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing* 7, 4 (2008), 12–18.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision.* Springer, 630–645.

[13] Ryan Herring, Aude Hofleitner, Saurabh Amin, T Nasr, A Khalek, Pieter Abbeel, and Alexandre Bayen. 2010. Using mobile phones to forecast arterial traffic through statistical learning. In *89th Transportation Research Board Annual Meeting*. 10–14.

[14] Timothy Hunter, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. 2009. Path and travel time inference from GPS probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs* 12, 1 (2009).

[15] Zhanfeng Jia, Chao Chen, Ben Coifman, and Pravin Varaiya. 2001. The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 536–541.

[16] Ishan Jindal, Xuewen Chen, Matthew Nokleby, Jieping Ye, et al. 2017. A Unified Neural Network Approach for Estimating Travel Time and Distance for a Taxi Trip. *arXiv preprint arXiv:1710.04350* (2017).

[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3149–3157.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Mu Li, Amr Ahmed, and Alexander J Smola. 2015. Inferring movement trajectories from GPS snippets. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 325–334.

[20] Yaguang Li, Dingxiong Deng, Ugur Demiryurek, Cyrus Shahabi, and Siva Ravada. 2015. Towards fast and accurate solutions to vehicle routing in a large-scale and dynamic environment. In *International Symposium on Spatial and Temporal Databases*. Springer, 119–136.

[21] Yaguang Li, Chengfei Liu, Kuien Liu, Jiajie Xu, Fengcheng He, and Zhiming Ding. 2013. On efficient map-matching according to intersections you pass by. In *International Conference on Database and Expert Systems Applications*. Springer, 42–56.

[22] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations (ICLR '18)*.

[23] Kuien Liu, Yaguang Li, Fengcheng He, Jiajie Xu, and Zhiming Ding. 2012. Effective map-matching on the most simplified road network. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 609–612.

[24] Hossein Mobahi, Ronan Collobert, and Jason Weston. 2009. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 737–744.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Workshop*.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[27] Karl F Petty, Peter Bickel, Michael Ostland, John Rice, Frederic Schoenberg, Jiming Jiang, and YaâĂŹacov Ritov. 1998. Accurate estimation of travel times from single-loop detectors1. *Transportation Research Part A: Policy and Practice* 32, 1 (1998), 1–17.

[28] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the 24th WSDM*. ACM.

[29] Mahmood Rahmani, Erik Jenelius, and Haris N Koutsopoulos. 2013. Route travel time estimation using low-frequency floating car data. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2292–2297.

[30] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. 2014. PRESS: A novel framework of trajectory compression in road networks. *Proceedings of the VLDB Endowment* 7, 9 (2014), 661–672.

[31] Jinjun Tang, Yajie Zou, John Ash, Shen Zhang, Fang Liu, and Yinhai Wang. 2016. Travel time estimation using freeway point detector data based on evolving fuzzy neural inference system. *PloS one* 11, 2 (2016), e0147263.

[32] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A simple baseline for travel time estimation using large-scale trip data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 61.

[33] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 25–34.

[34] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2013. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge and Data Engineering* 25, 10 (2013), 2390–2403.