Geoinformatica (2022) 26:379–395
https://doi.org/10.1007/s10707-020-00422-x

MTLM: a multi-task learning model for travel time estimation

Saijun Xu[1] · Ruoqian Zhang[2] · Wanjun Cheng[3] · Jiajie Xu[1,4]

Received: 25 April 2020 / Revised: 26 June 2020 / Accepted: 19 July 2020 /
Published online: 15 August 2020

Abstract
Travel time estimation (TTE) is an important research topic in many geographic applications for smart city research. However, existing approaches either ignore the impact of transportation modes, or assume the mode information is known for each training trajectory and the query input. In this paper, we propose a multi-task learning model for travel time estimation called MTLM, which recommends the appropriate transportation mode for users, and then estimates the related travel time of the path. It integrates transportation-mode recommendation task and travel time estimation task to capture the mutual influence between them for more accurate TTE results. Furthermore, it captures spatio-temporal dependencies and transportation mode effect by learning effective representations for TTE. It combines the transportation-mode recommendation loss and TTE loss for training. Extensive experiments on real datasets demonstrate the effectiveness of our proposed methods.

Keywords Travel time estimation · Deep learning · Trajectory data mining

Saijun Xu and Ruoqian Zhang are equally contributed co-first authors.

✉ Jiajie Xu
xujj@suda.edu.cn

Saijun Xu
sjxu@stu.suda.edu.cn

Ruoqian Zhang
rqzhang@suda.edu.cn

Wanjun Cheng
chengwj@neusoft.com

1 Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou, China

2 School of Computer Science and Technology, Soochow University, Suzhou, China

3 Neusoft Corporation, Shenyang, China

4 Institute of Electronic and Information Engineering of UESTC, Dongguan, Guangdong, China

1 Introduction

The advance of location acquisition technologies has promoted a rapid increase in the number of trajectories. A large amount of trajectory data can be used in estimating the travel time of the path. Travel time estimation is one of the most important location-based services, which can be applied in many location-based applications in the smart city, such as departure time suggestion [9], trip planning [18, 20–22, 33, 40], vehicle dispatching [34], route search [29], etc. However, travel time estimation is still challenging because of dynamic traffic conditions, different external factors (weathers, holidays, etc) and complex mobility behaviors.

Travel time estimation has been widely studied in the past. The existing solutions [7, 27] utilize trajectory data to estimate travel time. Recently, with the development of deep learning, the majority of the research [11, 26, 28, 35] design deep neural models to learn effective trajectory embeddings and capture latent features by finding appropriate representations. Then they can find (sub-)trajectories that are similar to the target trajectory based on trajectory embeddings, and estimate travel time of the target trajectory by aggregating travel time in (sub-)trajectories. However, all these TTE solutions ignore the transportation modes' effect on travel time of a given path and the transportation mode accordingly affects the travel time due to different moving speeds. Despite the impacts of transportation mode on travel time has been considered in [31], it assumes that the mode information is known for each training trajectory and the query phase. However, in reality, some or even majority of the trajectories in database may not have mode labels. In addition, the suitable transportation mode of a path may not be known by users at the query time.

Therefore, it is important to investigate the recommendation of transportation mode and estimate its corresponding travel time of a given path simultaneously. On one hand, given a path, users often need to choose appropriate transportation mode, such as taking a bus or riding, and a large number of trajectories provide corresponding knowledge for users to choose; on the other hand, users need to know the expected travel time for enhanced trip planning. [2] recommends users' transportation modes by automatically extracts additional features using a deep neural network from their movement trajectories. To estimate travel time, [26, 35] presents a deep neural model to integrate the geographic information to capture spatial correlations and stacking recurrent unit to capture the temporal dependencies. However, transportation mode recommendation and travel time estimation are close related tasks that heavily influence each other. To the best of our knowledge, there are no existing models that can capture the mutual influence of these two tasks, thus could not achieve more accurate results. Therefore, a multi-task model that integrates above two tasks in a shared space is sought after for more accurate travel time estimation.

To this end, in this paper, we propose a multi-task learning model for travel time estimation, namely MTLM, which integrates transportation-mode recommendation into travel time estimation. Specifically, first, it combines supervised learning task with an encoder part, which aims to recommend transportation mode of the given path for the user, and unsupervised learning task with a decoder part, which recovers relevant details discarded by supervised learning and retains the original data information as much as possible to reduce the risk of overfitting; second, the model incorporates representations of recommended transportation modes along with spatial patterns of trajectories and other external factors (weekdays, holidays) to estimate travel time; furthermore, MTLM is trained to simultaneously minimize the summation of supervised loss of transportation-mode recommendation, unsupervised reconstruction loss and time estimation loss by back-propagation. To sum up, the contributions of the paper are as follows:

– We propose a multi-task learning model for travel time estimation called MTLM, which recommends the transportation mode and then estimates the related travel time of a given path for the user. The model combines transportation-mode recommendation task and TTE task together by the summation of loss functions.
– We further integrate the knowledge of transportation modes, which is recommended in transportation-mode recommendation task, trajectory information and other external factors (such as holidays, weekdays or weekends), and designs a uniform deep neural model MTLM to capture spatial correlations, temporal dependencies and transportation mode influence for travel time estimation.
– We conduct extensive experiments on real datasets. The results demonstrate the advantages of our approach in recommending transportation mode and then estimating travel time of the path compared with baseline methods.

2 Related work

Travel time estimation is an important research topic which can be applied in trip planning [1, 14, 17, 24, 30], traffic speed prediction [12], trajectory activity analyses [23], etc. Existing approaches of estimating travel time in the path could be classified into two categories: segment-based and path-based approaches. Segment-based approaches [5, 19, 32] estimate the travel time on each road segment but ignore the correlations between the road segments; path-based approaches [15, 27] can resolve those issues in segment-based approaches by extracting and aggregating sub-paths from historical trajectories, but suffer from data sparsity because there are many sub-paths which cannot be found in historical trajectory data.

As mentioned above, segment-based and path-based approaches are unable to perform well in travel time estimation. Meanwhile, with the development of deep learning these years. More recent studies [26, 28, 35] utilize deep learning in travel time estimation. In [26], it presents an end-to-end estimation model that employs geo-convolution in the GPS sequence to capture spatial features and then uses LSTM to learn temporal patterns to estimate the travel time. Deeptravel proposed in [35] uses the sequence of extracted manually features in the granularity of grid cells as input to feed in Bi-LSTMs, to capture spatio-temporal patterns to estimate travel time. The WDR model [28] incorporates traffic information and is obtained by a traffic estimator during the feature extraction, then an ensemble regression model which consists of wide linear models, deep neural models and LSTM is used to estimate travel time. In [31], it proposes a transportation-mode aware deep neural model, which estimates travel time by utilizing (sub-)trajectories which are not only roughly following the target path, but also being consistent with segments of the target path in terms of transportation mode, it regards the transportation modes as known for each training trajectory and the query phase. However, in reality, some or even majority of the trajectories in database may not have mode labels.

The existing methods either have not considered the impact of transportation modes on travel time or treat the transportation mode as known for each training trajectory and the query phase. To deal with these issues, it is important to investigate how to suggest a suitable mode according to the corresponding travel time on a given path. A unified model to effectively support transportation-mode recommendation task and travel time estimation task is needed to be sought after in our work.

3 Problem formulation

This section introduces some preliminaries and gives a formal statement of the problem studied in this paper.

Definition 1 (Path) A path $P = (v_1, v_2, .., v_n)$ is a sequence of intersections, where intersection $v_i (1 \leq i \leq n)$ can be represented as a geographical location (longitude, latitude) and $(v_i, v_{i+1})(1 \leq i < n)$ is a road segment.

Definition 2 (Trajectory) A trajectory is represented as $T = \{G, m\}$, where $G = \{g_1, g_2, ..., g_n\}$ is a sequence of GPS points and each GPS point g_i is a tuple $(g_i.lng, g_i.lat, g_i.t)$ which are its longitude, latitude and timestamp respectively, m is the corresponding transportation mode of T.

The historical trajectory dataset is accordingly represented by a set of trajectories $X = \{T^{(i)} | i = 1, 2, 3, ..., N\}$.

Problem Statement Given a trajectory dataset X, a path P and a departure time d, our goal is to recommend the transportation mode and then estimate the related travel time of P, so as to make it consistent with the real trajectories.

4 The MTLM framework

In this section, we introduce the architecture of multi-task learning model MTLM, as shown in Fig.1, which consists of the data preparation layer, a transportation-mode recommendation layer and a travel time estimation layer.

4.1 Data preparation layer

In this section, we transform raw trajectory data into suitable trajectory embeddings to fully capture the spatial moving patterns in the trajectories.

First, we map the trajectory data into an area I, which is clipped by ranges of a geographic area $r_W \times r_H$ and is partitioned into $w \times w$ disjoint but equal-sized grids, then raw trajectories can be transformed as trajectory images [2]. Given a trajectory T, we compute the centroid of the GPS points in T, that is $(\frac{\sum_{i=1}^{n} g_i.lng}{n}, \frac{\sum_{i=1}^{n} g_i.lat}{n})$ and align the centroid with the center of the region I to unify the basic geographical coordinates. Each grid in the area represents a pixel in the image, the value of pixels can be calculated in the following manner: if there is a GPS point g_i in the grid $I(x, y)$, the value of $I(x, y)$ increases by 1. Finally, the raw trajectory can be portrayed as the trajectory image I.

4.2 Transportation-mode recommendation layer

In this layer, we aim to recommend an appropriate transportation mode based on the path for users by learning an autoencoder from historical data including paths and corresponding

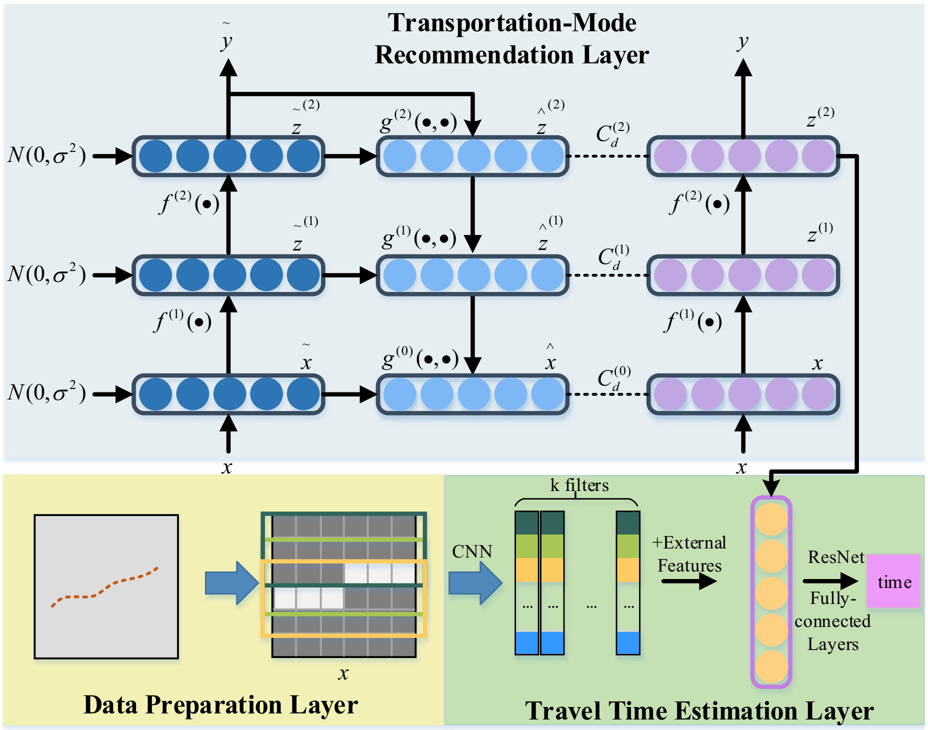


Fig. 1 The architecture of MTLM

transportation modes that users labeled. The labels in the data are different transportation modes that users choose, such as taking a bus, riding a bike, or walking. We use labeled data to supervise the autoencoder, which recommends transportation modes for users based on the path. However, even in the era of massive data, there is not much labeled data that can be used cleanly, which has led to the combination of deep learning and semi-supervised learning. Therefore, we introduce semi-supervised learning with labeled and unlabeled data simultaneously to optimize the autoencoder in the training phase.

We adopt a Ladder network [25], which is first applied to semi-supervised learning tasks in [16], to construct an autoencoder that can recommend the transportation mode of the path for users. We use the trajectory images I generated in the data preparation layer to represent the path, in this layer is denoted as x for legibility, as the inputs. Spatial moving patterns such as the spatial ranges and distributions of the path, are supposed to be captured for transportation-mode recommendation, because different transportation modes are chosen based on the topographic ranges and road conditions of the path.

Ladder network structure The structure of the Ladder network is shown in Algorithm 1. The input of the ladder network is $N + M$ trajectory images derived from raw trajectories, with N trajectory images with labels of transportation modes $\{(x(n), y^*(n))|1 \leq n \leq N\}$ and M unlabeled trajectory images $\{x(n)|N+1 \leq n \leq N+M\}$, which is denoted as $x(n)$; the output is the cost functions that include the cost between the recommended transportation mode and the actual transportation mode that users chose (Cross Entropy costs) and unsupervised costs (Reconstruction costs). The objective is to learn a function that models $P(y|x)$ by using both the labeled and the unlabeled data.

Algorithm 1 The Ladder network in transportation-mode recommendation layer.

```
   Input: $N$ trajectory images with labels of transportation modes
          $\{(x(n), y^*(n))|1 \leq n \leq N\}$ and $M$ unlabeled trajectory images
          $\{x(n)|N+1 \leq n \leq N+M\}$
   Output: Cost function $C_{semi}$ includes Cross Entropy costs and Reconstruction costs
   /* Noisy encoder */
1  $\tilde{h}^{(0)} \leftarrow \tilde{z}^{(0)} \leftarrow x(n) + noise$ // Gaussian noise on the noisy encoder.
2  for $l = 1 \rightarrow L$ do
3      $\tilde{z}_{pre}^{(l)} \leftarrow W_{1D}^{(l)} * \tilde{h}^{(l-1)} + b_{1D}^{(l-1)}$ // Convolutional operations
       /* The average value of the data in a batch */
4      $\tilde{\mu}^{(l)} \leftarrow batchmean(\tilde{z}_{pre}^{(l)})$
       /* The standard deviation of the data in a batch */
5      $\tilde{\sigma}^{(l)} \leftarrow batchstd(\tilde{z}_{pre}^{(l)})$
6      $\tilde{z}^{(l)} \leftarrow batchnorm(\tilde{z}_{pre}^{(l)}) + noise$ // Batch normalization and adding noise
7      $\tilde{h}^{(l)} \leftarrow activation(\gamma^{(l)}(\tilde{z}^{(l)} + \beta^{(l)})$ // Softmax as activation function
       // $\gamma^{(l)}$ and $\beta^{(l)}$ are parameters for shifting and scaling, which can be trained by
         using the backpropagation algorithm
8  $P(\tilde{y}|x) \leftarrow \tilde{h}^{(L)}$ // Recommendation probability in noisy encoder
   /* Clean encoder (for denoising targets) */
9  $h^{(0)} \leftarrow z^{(0)} \leftarrow x(n)$
10 for $l = 1 \rightarrow L$ do
11     $z^{(l)} \leftarrow batchnorm(W_{1D}^{(l)} * h^{(l-1)} + b_{1D}^{(l)})$ // Batch normalization
12     $h^{(l)} \leftarrow activation(\gamma^{(l)}(z^{(l)} + \beta^{(l)})$ // Softmax as activation function
   /* Final recommendation outcome: */
13 $P(y|x) \leftarrow h^{(L)}$
   /* Decoder and denoising square error costs */
14 for $l = L \rightarrow 0$ do
15     if $l = L$ then
16         $u^{(L)} \leftarrow batchnorm(\tilde{h}^{(L)})$ // Batch normalization
17     else
           /* Transposed convolution operator over an input */
18         $u^{(l)} \leftarrow batchnorm(w_{1D}^{(l)'} * \hat{z}^{(l+1)} + b_{1D}^{(l)'})$
19     $\forall i : \hat{z}_i^{(l)} \leftarrow g(\tilde{z}_i^{(l)}, u_i^{(l)})$ // Vanilla combinator
20     $\forall i : \hat{z}_{i,BN}^{(l)} \leftarrow \frac{\hat{z}_i^{(l)} - \tilde{\mu}_i^{(l)}}{\tilde{\sigma}_i^{(l)}}$ // Normalization
21 $Loss_{CE} \leftarrow -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} y_k^{*(n)} \log(p_{n,k})$ // Supervised costs(Cross Entropy)
22 $ReconsCost(z^{(l)}(n), \hat{z}^{(l)}(n)) \leftarrow \left\| \frac{\hat{z}^{(l)}(n) - \tilde{\mu}^{(l)}}{\tilde{\sigma}^{(l)}} - z^{(l)}(n) \right\|^2$ // unsupervised costs
23 $C_{semi} \leftarrow Loss_{CE} + \frac{1}{M} \sum_{n=N+1}^{N+M} \sum_{l=1}^{L} \lambda_l Reccons Cost(z^{(l)}(n), \hat{z}^{(l)}(n))$
```

The Ladder Network consists of two encoders and one decoder. One encoder applies additive noise to each layer (an extra Gaussian noise injection term in Algorithm 1) and therefore it has a regularization effect which helps generalization, called the noisy encoder; the other encoder which shares parameters with the noisy encoder is responsible for providing the clean reconstruction targets, i.e., the noiseless hidden activations, called the clean encoder. The decoder is designed to support unsupervised learning, which tries to reconstruct the original input from the internal representation and recover details discarded by the encoder. Formally, the Ladder Network is defined as follows,

$$\tilde{x}, \tilde{z}^{(1)}, ..., \tilde{z}^{(L)}, \tilde{y} = Encoder_{noisy}(x), \tag{1}$$

$$x, z^{(1)}, ..., z^{(L)}, y = Encoder_{clean}(x), \tag{2}$$

$$\hat{x}, \hat{z}^{(1)}, ..., \hat{z}^{(L)} = Decoder(\tilde{z}^{(1)}, ..., \tilde{z}^{(L)}), \tag{3}$$

where Encoder and Decoder can be replaced by convolutional operators [10] and transposed convolution operators in this case. The variables x, y, and $\tilde{y}$ are the input, the noiseless output, and the noisy output respectively. The true target is denoted as y^*. The variables $z^{(l)}$, $\tilde{z}^{(l)}$ and $\hat{z}^{(l)}$ are the hidden representation, its noisy version, and its reconstructed version at layer l. At each layer of both encoders, $z^{(l)}$ and $\tilde{z}^{(l)}$ are computed by applying a convolutional network and normalization on $h^{(l-1)}$ and $\tilde{h}^{(l-1)}$ respectively. Batch normalization [6] correction and non-linearity activation are then applied to obtain $h^{(l)}$ and $\tilde{h}^{(l)}$. We use softmax function as activation function and $a_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$. In the decoder of the Ladder network, there exists skip connections and through lateral skip connections, each layer of the noisy encoder is connected to its corresponding layer in the decoder. This enables the higher layer features to focus on more abstract and task-specific features. Hence, at each layer of the decoder, two signals, one from the layer above that is the vertical connection $u^{(l)}$, and the other from the corresponding layer that is the lateral connection $\tilde{z}^{(l)}$ in the encoder are combined to reconstruct $\hat{z}^{(l)}$ by using vanilla combinator function $g(.,.)$, which is in the following,

$$g(\tilde{z}^{(l)}, u^{(l)}) = a^{(l)}\mathcal{E}^{(l)} + b^{(l)}sigmoid(c^{(l)}\mathcal{E}^{(l)}) \tag{4}$$

where $\mathcal{E}^{(l)} = [1, \tilde{z}^{(l)}, u^{(l)}, \tilde{z}^{(l)}u^{(l)}]^T$ is the augmented input. $a^{(l)}$ and $c^{(l)}$ are trainable 1×4 weight vectors, and $b^{(l)}$ is a trainable weight.

Cost function in transportation-mode recommendation Finally, the objective function of this section is a weighted sum of supervised (Cross Entropy) costs $Loss_{CE}$ after the encoder and unsupervised costs (Reconstruction costs) during the decoder.

$$C_{semi} = Loss_{CE} + \textstyle\sum_{n=N+1}^{N+M}\sum_{l=1}^{L}\lambda_l ReconsCost(z^{(l)}(n), \hat{z}^{(l)}(n)) \tag{5}$$

where λ_l is the weight of the loss in layer l, $Loss_{CE}$ is Cross Entropy costs, which measure the relative entropy between two probability distributions and are commonly used as loss function in classification problems. It is defined as:

$$Loss_{CE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{*(n)} \log(p_{n,k}) \tag{6}$$

where the number of trajectories is N, the number of labels is K, $y_k^{*(n)}$ is the the value of the k-th dimension of the label $y^{*(n)}$ after one-hot encoding, and $p_{n,k}$ is the probability of the transportation mode of the n-th path is recommended as the k-th label.

Unsupervised costs (Reconstruction costs) is square error costs at each layer of the decoder, which are defined as the following,

$$ReconsCost(z^{(l)}(n), \hat{z}^{(l)}(n)) = \left\| \frac{\hat{z}^{(l)}(n) - \tilde{\mu}^{(l)}}{\tilde{\sigma}^{(l)}} - z^{(l)}(n) \right\|^2 \tag{7}$$

where $\hat{z}^{(l)}$ is normalized using $\mu^{(l)}$ and $\sigma^{(l)}$ which are the encoder's sample mean and standard deviation statistics of the current mini batch, respectively.

4.3 Travel time estimation layer

In this section, we aim to estimate travel time by considering the recommended transportation mode representations for the path and spatio-temporal characteristics. First, the spatial moving patterns of the path reflect the movement range and shape, we need to fully extract the spatial patterns of the historical trajectory, and then find the most similar sub-trajectory in the space for the trajectory to be predicted. After transforming raw trajectories into trajectory images, we then adopt convolution neural networks [10] in the trajectory images to capture spatial moving patterns. To be specific, given trajectory image I as inputs, the spatial representations convolved by the i-th filter is,

$$s_{i,a} = ReLU(W_{conv_i} * I_{a:a+h-1} + b_i) \tag{8}$$

where $G_{a:a+h-1} \in \mathbb{R}^{h\times w}$ is the a-th representations in the trajectory image I, $W_{conv_i} \in \mathbb{R}^{h\times w}$ denotes the parameter weights of the i-th filter and $*$ is the convolutional operation, b_i is the bias, h is the kernel size. Here we use rectifier linear unit $(ReLU)$ [10] as activation function and $ReLU(x) = max(0, x)$. Then we concatenate $s_{i,a}$ convolved by k filters and obtain the spatial representations of the a-th representations in I which is denoted as,

$$\mathcal{S}_a = \{s_{1,a}, s_{2,a}, ..., s_{k,a}\} \tag{9}$$

where $\mathcal{S}_a \in \mathbb{R}^{1\times k}$, then we scan sub-images of I in the chronological order (in height order in Fig. 1) with k 1D-CNN filters to get the spatial representations of the trajectory, denoted as $S = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{w-h+1}\}$.

Apart from spatial representations mined from trajectories, we further incorporate higher-level recommended transportation-mode features, that is $z^{(L)}$ in the clean encoder of the Ladder network and L is the number of layers in the previous layer. Besides, there are external factors which affect the travel time including whether the departure time is holidays, weekday or weekends, peak or non-peak hours or not and userID that reflects personalization information. We extract these factors according to the departure time of the trajectory and encode them into a real space $\mathbb{R}^{1\times A}$ by using the embedding mechanism [13], which not only reduces the dimension of categorical values but also represents categories in the transformed space. Then we concatenate these embedding vectors of external features to obtain low-dimensional vector E.

Finally, all the information can be concatenated as a uniform feature vector $V = \{S, z^{(L)}, E\}$ which can be utilized to estimate the travel time. Then we feed V as inputs into a deep residual neural networks [4], which capture spatio-temporal correlations in the feature vector, are easier to optimize and can gain accuracy from considerably increased depth. In the end, we further adopt two fully-connected layers to estimate the travel time of the trajectory. We use the mean absolute percentage error (MAPE) as the objective function in the training phase because it is a relative error that can estimate time of both the short paths and long paths. We define the loss function of travel time estimation as,

$$C_{TTE} = \left| \frac{\delta - \hat{\delta}}{\delta} \right| \tag{10}$$

where $\hat{\delta}$ is the estimated travel time of the trajectory and δ is the actual time.

Aiming at the influence of trajectory mis-recommendation on TTE task, on one hand, by introducing transportation mode labels, the transportation-mode recommendation task is learned in a supervised way, and the model parameters are optimized to further reduce the proportion of mis-recommendation; on the other hand, in travel time estimation task of our MTLM model, in addition to integrating the recommended transportation mode, the data preparation layer also extracts features from the trajectory for the following travel time estimation, which can reduce the impact of mis-recommendation on travel time estimation task.

4.4 Multi-task learning in training

The multi-task learning method combines a travel time estimation task and a transportion-mode recommendation task in a shared space (i.e., the high-level representation of the recommended transportation modes), to compute more effective representations and parameters for more accurate results in both tasks. We introduce the loss function in our model. This is an end-to-end model and the training goal is to minimize the loss function in multiple tasks. We combine the supervised costs for the transportation-mode recommendation accuracy after the encoder, unsupervised costs during the decoder, and loss function in the travel time estimation layer, that is,

$$C = \beta \cdot C_{semi} + (1 - \beta) \cdot C_{TTE} \tag{11}$$

where β is a weight in the combination of two losses C_{semi} and C_{TTE} to control the balance of two tasks.

In the training phase, we aim to minimize the loss function C and optimize all the parameters through back-propagation algorithm [3]. In the testing phase, in the transportation-mode recommendation layer, we skip the noisy encoder and decoder in the ladder network, and construct the recommendation only through the clean encoder, and then add the high-level features as input to the travel time estimation layer. Furthermore, we use multiple metrics to evaluate the estimation accuracy.

5 Experiments

In this section, we conduct experiments on GeoLife dataset to evaluate the performance of MTLM against several baselines for travel time estimation and compare our model under different parameter settings.

5.1 Datasets

GeoLife-labeled: We use GeoLife dataset [37–39] published by Microsoft Research. Each trajectory in GeoLife contains attributes including latitude, longitude, altitude in feet, date, time. The trajectory points in the trajectories were sampled at an interval of 1-5 seconds and 73 users have annotated labels on their trajectories with transportation mode. The transportation labels contain transportation modes annotated by users and the start and end time of using a certain transportation mode. First, we remove trajectories which users' transportation labels cannot be mapped into, because the start and end time in the labels were not in the time period of these collected trajectories. Then we split the trajectory into segments according to the start time and end time of the transportation mode marked by the user. We further partition the segment into separate trajectory segments if the interval between two consecutive GPS points exceeds 20 minutes. Finally, we obtain 6965 trajectory segments with 7 transportation modes (walk, bike, train, bus, subway, taxi, car) in Beijing.

GeoLife-unlabeled: In order to illustrate the application of our model in unlabeled data which is in unsupervised learning fashion, we extracted 4434 trajectory segments in Beijing from GeoLife dataset that users did not annotate transportation modes.

We randomly split a dataset into two folds, 80% data as the training set, 20% as the testing set on two datasets. On *GeoLife-labeled* dataset, for each transportation mode, we divide 80% of the trajectories with this transportation mode into the training set and 20% of them into the test set. In this way, we can ensure that the testing dataset and the training dataset contain the same category of transportation modes.

5.2 Implementation details

The architecture of our model MTLM is shown in Fig. 1, parameters of the model and experimental settings are described as follows:

- In the data preparation layer, ranges of longitude r_W and latitude r_H is determined by the range of the trajectory dataset, that is, $r_W = 20$, $r_H = 20$. Each trajectory segment is mapped into a region with $w \times w$ grids, we evaluate our model with different values of w and finally set $w = 100$ on *GeoLife-labeled* and *GeoLife-unlabeled* dataset.
- In the transportation-mode recommendation layer, we use three layers, the first layer is the input, the second layer in both of the encoders contains a convolutional network with kernel size 3 and the number of kernels is 64, the third layer contains a convolutional network with $h = 3$ and number of kernels is 7, the same as the number of transportation modes, then the feature maps are processed by softmax function to obtain the final recommendation outcome; in the decoder, the deconvolutional operations project the input back up to the feature maps of the corresponding layer in the encoder.
- In the travel time estimation layer, we use two layers of convolutional networks with kernel size 3 and the number of kernels 64 and 32 respectively. We fix the residual fully-connected layers as 3 and the size of each layer is 64. After residual layers, we further adopt two fully-connected layers to map the $\mathbb{R}^{64}$ to $\mathbb{R}^{1}$ to estimate the travel time. In addition, other external factors which affect the travel time, including userID, the day of the week, time slot of the departure time, dateID are embedded to $\mathbb{R}^{16}$, $\mathbb{R}^{8}$, $\mathbb{R}^{8}$, $\mathbb{R}^{8}$ respectively.
- In the training phase, we finally set the coefficient β as 0.2 on two dataset to control the balance of two tasks.

We set batch size as 64 and use Adam optimizer [8] as the optimization function. The initial learning rate is set to 0.0001. For each batch in *GeoLife-labeled*, we use half of the batch as labeled data for supervised training to construct transportation-mode recommendation costs, and the remaining half is used in unsupervised training as unlabeled data to construct reconstruction costs. The experiment is implemented by PyTorch. We train and evaluate the model on the server with one NVIDIA GTX1080 GPU and 24 CPU cores.

5.3 Evaluation metrics

MAPE is commonly used in regression problems because of its intuitive interpretation in terms of relative error. MAE is a measure of the difference between two continuous variables. For MAE, the optimal prediction will be the median target value and it is more robust when there are outliers in the data that are not conducive to the prediction results. RMSE is the square root of mean squared error, it measures the deviation between the predicted value and the real value. For RMSE, the optimal prediction will be the mean target value and it is more sensitive to outliers. Therefore, by using MAPE, MAE and RMSE, we can have a more scientific and comprehensive evaluation of the experimental results. Given $Y = \{y_1, y_2, ..., y_n\}$ as the actual values, $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ represents the predicted values. n denotes the number of the data. The metrics are defined as the following,

$$MAPE(Y, \ \hat{Y}) = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{12}$$

$$MAE(Y, \ \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - \hat{y}_i \right| \tag{13}$$

$$RMSE(Y, \ \hat{Y}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{14}$$

5.4 Baselines

- **GBDT**: Gradient boosting decision tree [36] is an ensemble model of decision trees. In each iteration, GBDT learns the decision trees by fitting the negative gradients. Several attributes of trajectories are fed into GBDT model, including the departure time, the GPS point of the origin and destination, uerID. The model is configured with 100 trees.
- **MlpTTE**: Multi-layer perceptron (MLP) with ReLU activation is applied to estimate the travel time. The input of MlpTTE is the same as GBDT and is fed into a 3-layer perceptron with ReLU activation with hidden size 64,32,1 respectively.
- **Deeptravel**: Deeptravel [35] uses the sequence of extracted manually features in granularity of grid cells as input to feed in Bi-LSTMs, so as to capture spatio-temporal patterns to estimate travel time.
- **DeepTTE**: DeepTTE [26] employs geo-convolution in the GPS sequence to capture spatial features and then uses LSTM to learn temporal patterns to estimate the travel time.

5.5 Performance comparison

The comparison results are reported in Table 1 and the results of our MTLM model are in bold in Table 1. MAPE, MAE and RMSE are evaluation metrics.

From Table 1, we can observe that: first, the GBDT has the worst performance, demonstrating the effectiveness of deep learning for travel time estimation; second, DeepTTE and DeepTravel outperform MlpTTE, indicating that effective deep neural models contribute to

Table 1 Performance comparison of evaluated approaches in metrics MAE, RMSE and MAPE

Dataset	*GeoLife-labeled*			*GeoLife-unlabeled*		
Metrics	MAE(min)	RMSE(min)	MAPE(%)	MAE(min)	RMSE(min)	MAPE(%)
MlpTTE	16.42	28.91	11.19%	25.01	55.69	13.85%
GBDT	16.90	31.96	12.21%	27.35	57.83	15.29%
DeepTravel	13.17	24.65	9.05%	21.97	49.92	12.27%
DeepTTE	12.19	21.93	8.37%	19.28	42.55	11.13%
MTLM(Ours)	**7.84**	**14.75**	**5.49%**	**12.45**	**23.87**	**7.98%**

capturing local spatial correlations and temporal patterns for travel time estimation; furthermore, our model outperforms all baseline models, which not only proves the effectiveness of incorporating the knowledge of recommended transportation mode based on path, in travel time estimation, but also proves the effectiveness of our model in capturing the spatio-temporal dependencies. In addition, our model performs well on unlabeled trajectory dataset *GeoLife-unlabeled*, which shows the feasibility of our method under unsupervised conditions.

5.6 Model analysis

In this section, we examine the parameter settings in MTLM and explore the role of the transportation-mode learning recommendation layer in TTE. To find a more appropriate parameter in MTLM, we vary different values of parameters.

5.6.1 Different grid size comparison

In the data preparation layer, the trajectory is mapped into the area to generate trajectory images as the inputs of the transportation-mode recommendation layer and travel time estimation layer, the value of w will affect both the accuracy of the travel time prediction and performance of the transportation-mode recommendation. We evaluate our model with different grid size $w \times w$ from $\{50 \times 50, 100 \times 100, 150 \times 150, 200 \times 200, 250 \times 250\}$, the estimation results are shown in Fig. 2. According to evaluation results, we find the most suitable value and finally set $w = 100$ on *GeoLife-labeled* and *GeoLife-unlabeled* dataset to get more accurate results.

5.6.2 Different convolutional kernel size comparison

In this part, we evaluate different values of convolutional kernel size h from 1 to 5. The convolutional kernels in the transportation-mode recommendation layer and in the travel time estimation layer are one-dimensional convolution operation on the trajectory image. The value of the convolutional kernel size determines the granularity of the trajectory image processing. A suitable value can fully capture the spatial movement features in the trajectories such as turning. The performance comparison is shown in Fig. 3, from the comparison results, we find that, convolution kernels that are too large or too small will decrease the performance of the model. When the kernel size is 3, it is more easily for our model to

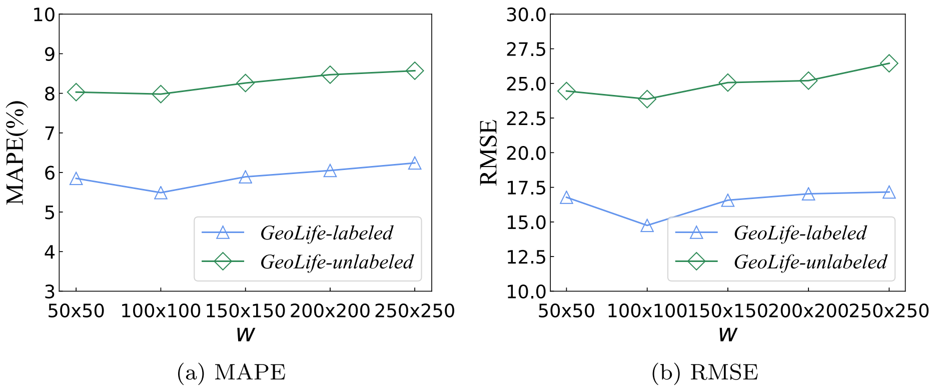


Fig. 2 Performance with different grid size in metrics MAPE, RMSE

capture spatial features in the trajectories on two datasets and the model obtains more accurate TTE results.

5.6.3 The analysis of transportation-mode recommendation layer

In this part, we evaluate the impact of transportation-mode recommendation layer on TTE task, which recommends the suitable transportation mode for users. We design a model without this layer, called NoTrans. The details are as follows: this model removes the encoder and decoder of the Ladder network. Accordingly, NoTrans contains only the data preparation layer and travel time estimation layer, especially, the knowledge of recommended transportation mode that output from the transportation-mode recommendation layer is not included in the travel time layer. The performance comparison results of NoTrans and MTLM are shown in Table 2.

From Table 2, we observe the model using transportation-mode recommendation layer performs better compared to the model removing this layer in all three evaluation metrics on two datasets. The outcome demonstrates the effectiveness of integrating knowledge of the recommended transportation modes based on path, in TTE task.

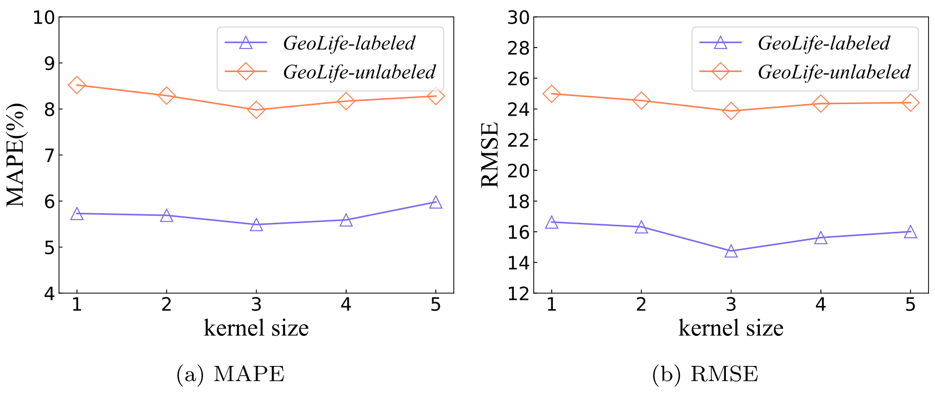


Fig. 3 Performance with different kernel size in metrics MAPE, RMSE

Table 2 Comparisons of different settings in the transportation-mode recommendation layer

Dataset	*GeoLife-labeled*			*GeoLife-unlabeled*		
Metrics	MAE(min)	RMSE(min)	MAPE	MAE(min)	RMSE(min)	MAPE
MTLM	7.84	14.75	5.49%	12.45	23.87	7.98%
NoTrans	8.36	15.31	5.79%	12.82	26.94	8.21%

6 Conclusion

In this paper, we study the recommendation of transportation mode and estimate its corresponding travel time of a given path simultaneously. Existing approaches either ignore the impact of transportation modes, or assume the mode information is known for each training trajectory and the query input. Therefore, we propose a multi-task learning model that integrates transportation-mode recommendation task and travel time estimation task, to capture the mutual influence of these two tasks for more accurate TTE results. We incorporate the knowledge of recommended transportation modes, path information and other external factors (such as holidays, weekends), in a uniform way to capture spatial correlations, temporal patterns and transportation mode effects for travel time estimation. Extensive experiments on real datasets demonstrate the effectiveness of our proposed methods.

Acknowledgments This work was supported by the National Natural Science Foundation of China under Grant Nos. 61872258, 61772356, 61876117 and 61802273, the Open Program of State Key Laboratory of Software Architecture under item number SKLSAOP1801, Dongguan Innovative Research Team Program (No.2018607201008), and Blockshine corporation.

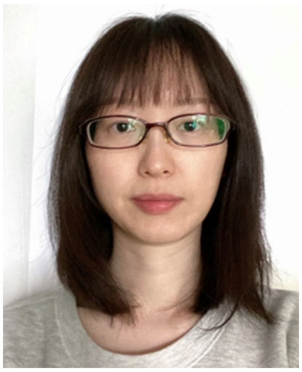
References

1. Chen X, Xu J, Zhou R, Zhao P, Liu C, Fang J, Zhao L (2020) S^2r-tree: a pivot-based indexing structure for semantic-aware spatial keyword search. GeoInformatica 24(1):3–25. https://doi.org/10.1007/s10707-019-00372-z
2. Endo Y, Toda H, Nishida K, Kawanobe A (2016) Deep feature extraction from trajectories for transportation mode estimation. In: Bailey J, khan L, washio T, dobbie G, huang JZ, Wang R (eds) PAKDD, vol 9652. Springer, Lecture Notes in Computer Science, pp 54–66
3. Gruslys A, Munos R, Danihelka I, Lanctot M, Graves A (2016) Memory-efficient backpropagation through time. In: NIPS2016, pp 4125–4133
4. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE Computer Society CVPR 2016, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
5. Hofleitner A, Herring R, Abbeel P, Bayen AM (2012) Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network. TITS 13(4):1679–1693. https://doi.org/10.1109/TITS.2012.2200474
6. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML, pp 448–456
7. Jenelius E, Koutsopoulos HN (2013) Travel time estimation for urban road networks using low frequency probe vehicle data. Transp Res B Methodol 53:64–81
8. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
9. Kisialiou Y, Sr IG, Laporte G (2018) The periodic supply vessel planning problem with flexible departure times and coupled vessels. Comput OR 94:52–64
10. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS, pp 1106–1114

11. Li Y, Fu K, Wang Z, Shahabi C, Ye J, Liu Y (2018) Multi-task representation learning for travel time estimation. In: KDD. 1695–1704. https://doi.org/10.1145/3219819.3220033, vol 2018
12. Lv Z, Xu J, Zheng K, Yin H, Zhao P, Zhou X (2018) LC-RNN: A deep learning model for traffic speed prediction. In: Lang J (ed) IJCAI 2018, ijcai.org, pp 3470–3476. https://doi.org/10.24963/ijcai.2018/482
13. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: NIPS, pp 3111–3119
14. Qian Z, Xu J, Zheng K, Zhao P, Zhou X (2018) Semantic-aware top-k spatial keyword queries. World Wide Web 21(3):573–594. https://doi.org/10.1007/s11280-017-0472-y
15. Rahmani M, Jenelius E, Koutsopoulos HN (2013) Route travel time estimation using low-frequency floating car data. In: ITSC 2013, pp 2292–2297. https://doi.org/10.1109/ITSC.2013.6728569
16. Rasmus A, Berglund M, Honkala M, Valpola H, Raiko T (2015) Semi-supervised learning with ladder networks. In: NIPS, pp 3546–3554
17. Shang S, Ding R, Yuan B, Xie K, Zheng K, Kalnis P (2012) User oriented trajectory search for trip recommendation. In: 15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings. ACM, pp 156–167. https://doi.org/10.1145/2247596.2247616
18. Shang S, Lu H, Pedersen TB, Xie X (2013a) Finding traffic-aware fastest paths in spatial networks. In: SSTD 2013, Springer, Lecture Notes in Computer Science, vol 8098, pp 128–145. https://doi.org/10.1007/978-3-642-40235-7_8
19. Shang S, Lu H, Pedersen TB, Xie X (2013b) Modeling of traffic-aware travel time in spatial networks. In: 2013 IEEE 14th International Conference on Mobile Data Management, IEEE Computer Society, pp 247–250. https://doi.org/10.1109/MDM.2013.34
20. Shang S, Guo D, Liu J, Liu K (2014) Human mobility prediction and unobstructed route planning in public transport networks. In: IEEE Computer Society IEEE MDM, pp 43–48. https://doi.org/10.1109/MDM.2014.66
21. Shang S, Liu J, Zheng K, Lu H, Pedersen TB, Wen J (2015) Planning unobstructed paths in traffic-aware spatial networks. GeoInformatica 19(4):723–746. https://doi.org/10.1007/s10707-015-0227-9
22. Shang S, Guo D, Liu J, Wen J (2016) Prediction-based unobstructed route planning. Neurocomputing 213:147–154. https://doi.org/10.1016/j.neucom.2016.02.085
23. Shang S, Chen L, Zheng K, Jensen CS, Wei Z, Kalnis P (2019) Parallel trajectory-to-location join. IEEE Trans Knowl Data Eng 31(6):1194–1207. https://doi.org/10.1109/TKDE.2018.2854705
24. Song X, Xu J, Zhou R, Liu C, Zheng K, Zhao P, Falkner N (2020) Collective spatial keyword search on activity trajectories. GeoInformatica 24(1):61–84. https://doi.org/10.1007/s10707-019-00358-x
25. Valpola H (2014) From neural PCA to deep unsupervised learning. CoRR arXiv:1411.7783
26. Wang D, Zhang J, Cao W, Li J, Zheng Y (2018a) When will you arrive? estimating travel time based on deep neural networks. In: AAAI 2018, pp 2500–2507
27. Wang Y, Zheng Y, Xue Y (2014) Travel time estimation of a path using sparse trajectories. In: KDD 2014, pp 25–34, pp https://doi.org/10.1145/2623330.2623656
28. Wang Z, Fu K, Ye J (2018b) Learning to estimate the travel time. In: KDD 2018, pp 858–866. https://doi.org/10.1145/3219819.3219900
29. Xu J, Gao Y, Liu C, Zhao L, Ding Z (2015) Efficient route search on hierarchical dynamic road networks. Distrib Parallel Database 33(2):227–252. https://doi.org/10.1007/s10619-014-7146-x
30. Xu J, Chen J, Zhou R, Fang J, Liu C (2019) On workflow aware location-based service composition for personal trip planning. Future Gener Comput Syst 98:274–285. https://doi.org/10.1016/j.future.2019.03.010
31. Xu S, Xu J, Zhou R, Liu C, Li Z, Liu A (2020) Tadnm: A transportation-mode aware deep neural model for travel time estimation. in press
32. Yang B, Guo C, Jensen CS (2013) Travel cost inference from sparse, spatio-temporally correlated time series using markov models. PVLDB 6(9):769–780. https://doi.org/10.14778/2536360.2536375
33. Yang B, Guo C, Jensen CS, Kaul M, Shang S (2014) Stochastic skyline route planning under time-varying uncertainty. In: IEEE Computer Society IEEE ICDE 2014, pp 136–147. https://doi.org/10.1109/ICDE.2014.6816646
34. Yuan NJ, Zheng Y, Zhang L, Xie X (2013) T-finder: A, recommender system for finding passengers and vacant taxis. IEEE Trans Knowl Data Eng 25(10):2390–2403
35. Zhang H, Wu H, Sun W, Zheng B (2018) Deeptravel: a neural network based travel time estimation model with auxiliary supervision. In: IJCAI 2018, pp 3655–3661. https://doi.org/10.24963/ijcai.2018/508
36. Zhang Y, Haghani A (2015) A gradient boosting method to improve travel time prediction. Transp Res Part C: Emerg Technol 58:308–324

37. Zheng Y, Li Q, Chen Y, Xie X, Ma W (2008) Understanding mobility based on GPS data. In: Youn HY, Cho W (eds) UbiComp 2008, ACM, vol 344. ACM International Conference Proceeding Series, pp 312–321. https://doi.org/10.1145/1409635.1409677
38. Zheng Y, Zhang L, Xie X, Ma W (2009) Mining interesting locations and travel sequences from GPS trajectories. In: WWW 2009. ACM, pp 791–800. https://doi.org/10.1145/1526709.1526816
39. Zheng Y, Xie X, Ma W (2010) Geolife: A, collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–39
40. Zhu S, Wang Y, Shang S, Zhao G, Wang J (2017) Probabilistic routing using multimodal data. Neurocomputing 253:49–55. https://doi.org/10.1016/j.neucom.2016.08.138

Saijun Xu received the B.S degree from Soochow University. She is currently a M.Sc. Candidate of Soochow University. Her research interest includes Spatio-temporal Data Mining.



Ruoqian Zhang is a faculty member of school of computer science and technology, Soochow university. Her research interests include spatial databases and data mining.

Wanjun Cheng received the Ph.D. degree in Computer Application Technology in Northeastern University, Shenyang, China, in 2003. He is an architect and an advanced engineer in Neusoft Corporation, Shenyang, China. His research interests include spatiotemporal data mining, software engineering and architecture, computer vision, natural language processing, information security.

Jiajie Xu received his M.S. and Ph.D. degree from Swinburne University of Technology and University of Queensland in 2006 and 2011 respectively. He is currently an associate professor of Soochow University. His research interests include Spatiotemporal Database Systems and Big Data analytics.