CoDriver ETA: Combine Driver Information in Estimated Time of Arrival by Driving Style Learning Auxiliary Task

Yiwen Sun, *Graduate Student Member, IEEE*, Kun Fu, *Member, IEEE*, Zheng Wang, *Member, IEEE*, Donghua Zhou, *Fellow, IEEE*, Kailun Wu, *Member, IEEE*, Jieping Ye, *Fellow, IEEE*, and Changshui Zhang, *Fellow, IEEE*

***Abstract*— Estimated time of arrival (ETA) is one of the most important services in intelligent transportation systems (ITS). Precise ETA ensures proper travel scheduling of passengers as well as guarantees efficient decision-making on ride-hailing platforms, which are used by an explosively growing number of people in the past few years. Recently, machine learning-based methods have been widely adopted to solve this time estimation problem and become state-of-the-art. However, they do not well explore the personalization information, as many drivers are short of personalized data and do not have sufficient trajectory data in real applications. This data sparsity problem prevents existing methods from obtaining higher prediction accuracy. In this article, we propose a novel deep learning method to solve this problem. We introduce an auxiliary task to learn an embedding of the personalized driving information under multi-task learning framework. In this task, we discriminatively learn the embedding of driving preference that preserves the historical statistics of driving speed. For this purpose, we adapt the triplet network from face recognition to learn the embedding by constructing triplets in the feature space. This simultaneously learned embedding can effectively boost the prediction accuracy of the travel time. We evaluate our method on two large-scale real-world datasets from Didi Chuxing platform. The extensive experimental results on billions of historical vehicle travel data demonstrate that the proposed method outperforms state-of-the-art algorithms.**

***Index Terms*— Estimated time of arrival, personalized driver information, transfer knowledge.**

Manuscript received January 2, 2020; revised July 12, 2020 and September 12, 2020; accepted November 9, 2020. Date of publication December 9, 2020; date of current version May 3, 2022. This work was supported in part by the National Key Research and Development Program of China (No. 2018AAA0100701) and in part by the National Natural Science Foundation of China under Grant 61751307. The Associate Editor for this article was L. Li. *(Corresponding author: Changshui Zhang.)*

Yiwen Sun, Kailun Wu, and Changshui Zhang are with the Institute for Artificial Intelligence, Tsinghua University (THUAI), Beijing 100084, China, and also with the State Key Laboratory of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: syw17@mails.tsinghua.edu.cn; wukl14@mails.tsinghua.edu.cn; zcs@mail.tsinghua.edu.cn).

Kun Fu and Zheng Wang are with DiDi AI Labs, Beijing 100084, China (e-mail: fukunkunfu@didichuxing.com; wangzhengzwang@didichuxing.com).

Donghua Zhou is with the College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao 266590, China, and also with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: zdh@mail.tsinghua.edu.cn).

Jieping Ye is with DiDi AI Labs, Beijing 100084, China, and also with the Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yejieping@didichuxing.com).

Digital Object Identifier 10.1109/TITS.2020.3040386

I. Introduction

ESTIMATED time of arrival (ETA), considered as the vehicle travel time estimation between origin and destination locations, is one of the most challenging problems in intelligent transportation systems (ITS) [1]. It attracts more attention in recent years, as online ride-hailing mobile applications, such as DiDi and Uber, are used by millions of people per day and the travel time is one of the key considerations in the decision-making process on the ride-hailing platforms, including route planning, navigation, carpooling, vehicle dispatching and scheduling [2], [3]. An accurate ETA is essential to build a highly efficient transportation platform and to constantly enhance user experience.

As one of the core problems in practical transportation system which is shown in Fig. 1, travel time estimation has been widely studied in the past [2]–[15]. The existing works can be summarized into two categories. The first category is the route-based method. This type of methods splits a route into several road segments. The overall travel time is obtained by summing over the travel time of all segments and the delay time at all intersections. The segment travel time and interaction delay time can be estimated based on diverse sources of spatio-temporal (ST) data [4], [5] by using different methods, such as tensor decomposition [6], dynamic Bayesian network [8], pattern matching [9] and gradient boosted regression tree [10].

The second category is the data-driven method which directly estimates the overall travel time along the entire route based on historical travel trajectories. This kind of method becomes more popular in both academia and industry due to its good performance and robustness to the time-varying traffic condition. Thus, how to effectively use the emerging data from the transportation platforms becomes more and more demanded. Recently, learning to estimate the travel time based on large-scale travel data has achieved promising results and been successfully applied in real applications [2]. It formulates ETA problem as a regression task. The parameters

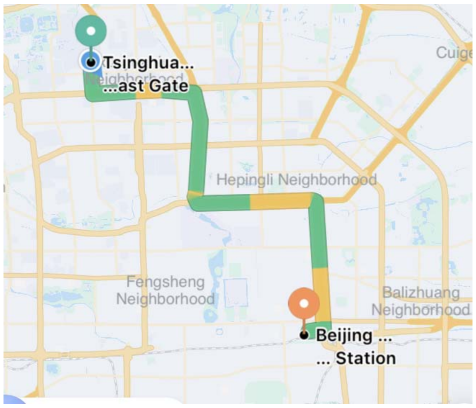


Fig. 1. An example of Estimated Time of Arrival (ETA): the green pin and orange pin represent the origin and destination of the trip, respectively; ETA is the travel time along the given path for a specific vehicle.

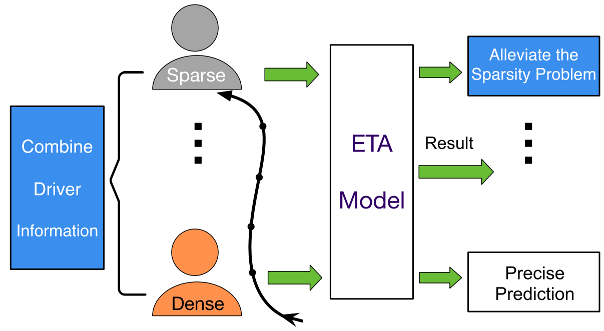


Fig. 2. The conceptual framework of *CoDriver ETA*. Sparse drivers, such as those doing part-time job or those newly joined, take a large part in ride-hailing platforms. By transferring knowledge from dense drivers, the ETA model improves personalized travel time adaptation on sparse drivers.

are trained with the collected historical trajectories. With the rapidly increasing scale of trajectory data collected by GPS devices and the boom of deep learning [16], neural network methods recently achieve promising performance [2], [3], [11]. The Wide-Deep-Recurrent (WDR) model [2] combines the wide-deep model from the recommendation field and the recurrent model from natural language processing field, giving a road-segment level modeling of ETA. It achieves the state-of-the-art performance in ETA. However, it still suffers from the data sparsity problem.

The data sparsity problem of ETA comes from two aspects: (1) many links, namely the road-segments, are scanned by very few probe cars or even not scanned during a day. ETA system lacks the traffic information of these links to make an accurate prediction; (2) many drivers do not have sufficient trajectory data and thus their embedding vectors [17], [18] are under-fitting. Such drivers usually include the newly joined ones and the part-time ones who only share their car during commuting.

In this article, we focus on solving the driver data sparsity problem. Personalized information is a key feature to ETA models. For example, different drivers may spend distinct time passing through the same link due to their driving style difference. A driver may also drive faster in his/her familiar areas. We refer to those lacking trajectories as sparse drivers and those owning plenty of trajectories as dense drivers. Sufficient training data ensures that dense drivers have good personalized adaptation of travel time, leading to more accurate prediction. However, sparse drivers often suffer poor personalized adaptation.

The novelty of the work can be concluded as follows. Although some methods [2], [3], [11] based on deep learning have achieved good prediction results for ETA. However, they do not make effective response to the driver data sparsity problem we just discuss. Therefore, in the face of such sparse data case, the prediction accuracy may be significantly affected. To alleviate the driver data sparsity problem, we propose a novel method – *CoDriver ETA*, which stands for combining the information of sparse drivers and dense drivers. We assume that drivers sharing similar driving styles should be closer in the embedded space. We adopt the framework of multi-task learning [19] and improve the triplet loss [20] to measure the distance between driving preference of different drivers via an auxiliary task. With such a mechanism, our ETA model can transfer knowledge from the dense drivers to the sparse drivers, and thus enhance the prediction performance. (See Fig. 2 for a conceptual demonstration of our method.) A series of extensive experiments have proved that *CoDriver ETA* can significantly alleviate the sparsity problem corresponding to drivers. The proposed framework is highly scalable because it can be applied to all learning-based ETA models using driver embedding.

Our main contributions can be summarized as follows:

- To our best knowledge, our proposed method is the first one to address the driver data sparsity problem. It significantly improves the travel time prediction precision for the drivers with insufficient data.
- We transfer knowledge from dense drivers to sparse drivers under multi-task learning framework and adapt the triplet loss to measure the distance between different drivers' driving preference. The learnt representations better capture the similarities among drivers.
- We evaluate our method on large-scale real-world data, which collects over 100 millions of trajectories from 350 thousands of drivers. The quantitative results validate that *CoDriver* can improve the ETA performance. The experimental results demonstrate that sparse drivers do benefit from the knowledge transferred from dense drivers.

The rest of the paper is organized as follows. Section II presents the related works. Section III analyzes the driver data sparsity problem and introduces the proposed method, *CoDriver ETA*. Section IV shows experimental results on large-scale real-world datasets. Finally, we conclude this article and discuss the future work in Section V.

II. Related Work

We first review the previous work on ETA. Since our method leverages the framework of multi-task learning and transfer learning, we introduce them as well.

ETA system is regarded as one of the fundamental modules in ITS. Making ETA more and more accurate may indirectly assist the research development of other ITS tasks, such as autonomous driving and intelligent vehicles better coping with the real environment [21]–[26]. The first category method to solve ETA problem is the route-based ETA. Learning-based methods for route-based ETA attracted wide attentions. Statistical models are used to find the correlation between current travel time and historical data. Reference [4] proposes to build a statistical model to estimate the mean and covariance of travel time in link level. [8] uses dynamic bayesian network to learn the arterial traffic from probe vehicle's trajectory data. Reference [7] uses least-square minimization to estimate the travel time of each individual road link. Reference [6] first decomposes trips into link, time and driver dimensions and then applies tensor decomposition algorithm to learn link travel time. Reference [9] uses pattern matching to get the similar historical data to predict the temporal-spatial evolution of traffic. Reference [10] proposes a gradient boosted regression tree method which combines simple regression trees with poor performance to predict ETA. References [12], [13] focus on inferring the current traffic speed and the flow volume to estimate the travel time of individual road links indirectly.

The second category is the data-driven method that is most popular in both academia and industry. Reference [15] computes travel time from a weighted sum of the nearest neighbors, which share sub-paths with the query route, and then further adjust the result by temporal dynamics. Reference [14] proposes a time-dependent landmark graph to model the dynamic traffic pattern. The above methods rely on traditional machine learning algorithms. It is difficult for them to fully mine spatio-temporal dependency on massive data. Deep learning is known for learning effective representations from large-scale data on complicated tasks, such as object recognition, audio classification and speech recognition [16], [27]–[29]. In traffic field, several recent works are inspired to apply deep learning models to fit traffic data [30]–[33]. Reference [30] uses deep belief networks to predict the traffic flow. Reference [31] equips 3D spatial-temporal convolution to neural networks to forecast the link traffic. For ETA problem, [3] proposes an end-to-end model which adopts geo-convolution operation and the Long-Short Term Memory network (LSTM) [34] to estimate travel time using sampled GPS points. However, their models are limited in production scenarios because the GPS point sequence cannot be obtained before the users actually finish the trip. Reference [11] proposes a multi-task representation learning model for origin-destination (OD) travel time estimation. Such an OD ETA aims to estimate the travel time without any given path. Reference [2] proposes Wide-Deep-Recurrent (WDR) model to jointly train wide linear models, deep neural networks and recurrent neural networks, making more effective use of different types of feature information. The Long-Short Term Memory network (LSTM) [34], a variant of recurrent neural network, plays a key role in the model. It introduces capability to represent the road segment in a more fine-grained manner. As the state-of-the-art method, WDR is selected as the main baseline in this article.

Though travel time is significantly affected by personalized factors, only a few methods consider the driver character. References [2], [3], [11] embed drivers into latent spaces but often suffer from the driver data sparsity problem. References [35], [36] transform the road network into the sequence of generalized images, and use convolutional neural network (CNN) to capture the spatial dependence for ETA. However, these two methods also can not effectively deal with the driver data sparsity problem. To our best knowledge, *CoDriver ETA* which is proposed in this article is the first method that could address the driver data sparsity problem. Our method is inspired by multi-task learning (MTL) [19] to transfer knowledge among drivers. MTL is a paradigm of machine learning, which learns several tasks simultaneously with the aim of a mutual benefit. The information from auxiliary tasks can help to learn the related main task more effectively while using a shared representation [30]. MTL has been successfully used in transportation research. Reference [30] proposes a deep architecture that consists of a deep belief network (DBN) and a multi-task regression layer for traffic flow prediction. Transfer learning aims at improving performance in the target task by using knowledge from the source domain and task [37]. When labels of both the source and target domain are available and both the source and target tasks are learnt simultaneously, this case of transfer learning is similar to MTL [37].

III. Methodology

Our goal is to alleviate the driver data sparsity problem. That means, a basic ETA model is needed to finish the travel time regression task. We choose the state-of-the-art method WDR [2] as our basic model, upon which we develop a novel multi-task learning mechanism to transfer knowledge from dense drivers to sparse drivers. However, readers should note that our method is not limited to neural network models. Instead, our method can be applied to any model using the technique of driver embedding.

Next, we will introduce the basic ETA model, the driver data sparsity problem and our method in a sequence.

A. Basic ETA Model

We first give the definition of ETA learning problem. Essentially, it is a regression task:

Definition 1: **ETA learning**. Suppose we have a database of trips, $D = \{p_i, s_i, e_i, x_i\}_{i=1}^{N}$, where p_i is the trajectory path for the $i\text{-}th$ trip order sample, s_i is the departure time, e_i is the arrival time, x_i is the driver-id and N denotes the number of samples. The ground-truth travel time is given by $y_i = e_i - s_i$. Given a query $q = (p_q, s_q, d_q)$, our goal is to estimate the travel time y_q with the given path p_q, departure time s_q and driver d_q.

Same to WDR, our basic ETA model consists of three part: (1) the wide part is like a factorization machine [38] which memorizes the historical patterns by constructing a second order cross-product and an affine transformation of the inputs; (2) the deep part is a stack of fully-connected layers which improves the feature generalization; (3) and the recurrent part is a standard LSTM [34] which provides a fine-grained

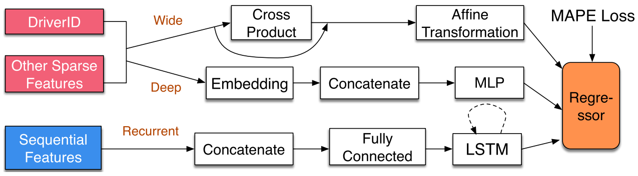


Fig. 3. The structure of our basic ETA model. Same to WDR [2], the model consists of three part: (1) the wide part memorizes the historical patterns by constructing a second order cross-product and an affine transformation of the sparse features; (2) the deep part embeds the sparse features and utilize the MLP to improves the feature generalization; (3) and the recurrent part provides a fine-grained modeling on the link level by a standard LSTM [34]. The concatenated sequential features is fed into a stack of fully-connected layers followed by a LSTM. The whole model is trained under MAPE loss.

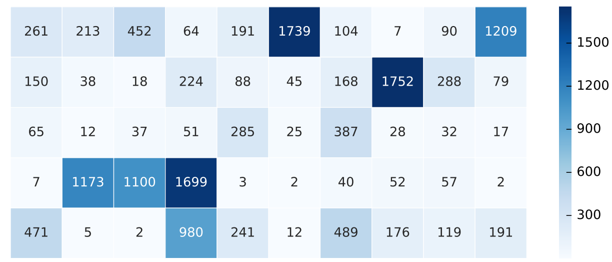


Fig. 4. Heatmap of the driver occurring frequency counted on a 8 months real world dataset (darker color means higher frequency). Among the 50 randomly selected drivers, 7 drivers have less than 10 trajectories. This makes their driver embedding vectors in bad under-fitting.

modeling on the link level by learning the dependency between different parts of the given path. LSTM is a variant of RNN, which can process sequential input data and better store the information in hidden state through three gates and a memory cell [34]. The input dimensions of the wide and deep parts are fixed, while the input length of the recurrent part varies as the path length. Our basic ETA model is able to deal with variable-length paths.

Embedding [17] is a popular technique in deep learning. In the deep part of the basic ETA model, we construct embedding tables for the discrete features including driver ID, starting time slice and day of week. For example, for driver ID x, we look up the corresponding embedding table $E_d \in \mathbb{R}^{n\times m}$, where n is the total driver number and m is the dimension of embedding space. We then use x_{th} row of E_d as the driver $= x$'s distributional representation. The day of week and starting time slice of s_i is w and t. We finally get three embedding vectors: $E_d(x, :)$ for the driver, $E_t(t, :)$ for starting time slice (288 slices per day) and $E_w(w, :)$ for day of week. The embedding vectors of discrete features are concatenated: $[E_d(x, :), E_t(t, :), E_w(w, :)]$ and is then fed into the following module.

The activation functions are chosen as the Relu [28]. The parameters of the three parts are jointly trained under the Mean Absolute Percentage Error (MAPE) loss, as described below:

$$L_{main} = \mathrm{MAPE}(y, y') = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|y_i - y_i'\right|}{y_i}, \tag{1}$$

where y_i' is the predicted ETA, and y_i is the ground-truth travel time. Fig. 3 gives the detailed structure of our basic ETA model.

B. Driver Data Sparsity Problem

Driver ID is an important feature for the ETA model because it allows the personalized adaptation of the travel time. With more historical trajectories, the model can better learn a driver's driving style and thus make more accurate adjustment of the predicted ETA. However, the ride-hailing platforms cannot collect sufficient data for every driver. For example, some part-time drivers only share their car during the commuting. Even for full-time drivers, they also lack data in the starting stage. For such sparse drivers with insufficient data, the ETA precision on them is typically lower than on the dense drivers.

To provide an intuitive observation, we randomly select 50 drivers from a real world dataset which collects 8 months data on DiDi platform, and visualize their occurring frequency in a heatmap. The details of the dataset will be introduced in the experimental part, specifically Section IV-A. As shown in Fig. 4, only 6 drivers have more than 1,000 samples and their personalized travel time adaptation is well learnt. Frequencies of over half of the drivers do not reach 100. Even for 7 drivers of them, less than 10 samples are collected.

We also plot histograms of the driver frequency (a.k.a order number) to quantitatively demonstrate the data sparsity problem of drivers. We separately do the statistic on two kinds of trajectory data: *pickup* represents the stage when a driver reaches a passenger, and *trip* represents the stage when a driver delivers a passenger to the destination. Fig. 5 shows that drivers concentrate on the low order number intervals, meaning that data sparsity problem occurs in most of the drivers.

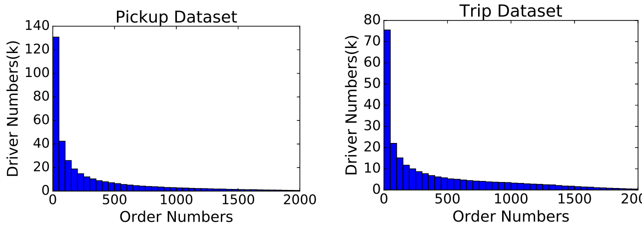
In one epoch, the training iteration of the embedding parameters for a driver equals to its occurring frequency in the dataset. So for extremely sparse drivers, their embedding vectors are updated by very few iterations and ended in badly under-fitting status. Our solution is to increase the training intensity of sparse drivers under the guidance of dense drivers. To reach this, we need to measure the similarity between drivers via driving style.

C. Driving Style Statistics

The average speed is the most essential metric to reflect the personalized driving style. We take this statistic to measure the driver similarity. Namely, drivers with similar driving styles should be close in the dimension of average speed.

We denote S as the total driving distance and T as the total driving time of a driver across the whole dataset. The S is computed by summing up all the link lengths the driver passing through: $S = \sum_{i,j} l_{i,j}$, where $l_{i,j}$ is the length of the j_{th} link in the i_{th} trajectory. Similarly, we have $T = \sum_i y_i$, where y_i is the ground-truth time of the i_{th} trajectory. The driver's average speed is then calculated as follows:

$$\bar{v} = \frac{S}{T} = \frac{\sum_{i,j} l_{i,j}}{\sum_i y_i}. \tag{2}$$



(a) (b)

Fig. 5. Statistics of driver order numbers. The histograms are separately on two kinds of data: (a) pickup and (b) trip. Most of the drivers locate in the low frequency intervals and the median order numbers are 107 and 201, respectively.

Using average speed to measure driver similarity has two advantages. Firstly, it is directly related to ETA. When the driving distance is fixed (path is given), the travel time is only determined by the average speed. The average speed is a summary indicator of many factors of driving style. For example, a driver who does not like overtaking other cars on the road and tends not to rush the yellow light at crossing will have lower average speed. Generally, his/her travel time should be longer than the risky drivers'.

Secondly, average speed can be computed from the current ETA dataset without any additional information. As a comparison, if we use nasty brake frequency to measure driving style, we need to additionally collect the inertial sensor signal, which undermines the compatibility of the existing data.

D. Improving Triplet Loss

We assume that drivers with similar driving style are closer in the embedded space. To achieve this, we take triplet loss to build an auxiliary task. Triplet loss is initially proposed for face recognition and clustering [20]. A triplet consists of three samples: an anchor sample, a positive sample which comes from the same category as the anchor sample, and a negative sample whose category is other than the anchor sample's. The loss punishes the situation that Euclidean distance between negative sample and anchor sample is smaller than that between positive sample and anchor sample. It achieves excellent face recognition performance and could cluster those images from one person together.

For our driver embedding case, since there is no category label for the drivers, we propose a variant of triplet loss. Suppose we have an anchor driver $x_i^{(a)}$, we randomly select two other drivers from data and assign one as positive driver $x_i^{(p)}$ and the other as negative driver $x_i^{(n)}$. The assignment should satisfy the below inequation:

$$|\bar{v}_{x_i^{(a)}} - \bar{v}_{x_i^{(p)}}| \leqslant |\bar{v}_{x_i^{(a)}} - \bar{v}_{x_i^{(n)}}|, \tag{3}$$

which means that the anchor driver $x_i^{(a)}$ is closer to the positive driver $x_i^{(p)}$ than to the negative driver $x_i^{(n)}$, under the metric of average speed. According to our assumption, the embedding vectors of them should also satisfy a similar distance inequation:

$$\|E_d(x_i^{(a)}, :) - E_d(x_i^{(p)}, :)\|_2 < \|E_d(x_i^{(a)}, :) - E_d(x_i^{(n)}, :)\|_2. \tag{4}$$

Under this constraint, we ensure that more similar drivers are closer in the embedding space, and thus play more similar roles when computing ETA predictions. Usually, the distance gap is required to be larger than a small margin. Moreover, we also complete L_2 normalization of each embedding vector. We can change Eq. 4 into a soft version:

$$\|\widetilde{E}_d(x_i^{(a)}, :) - \widetilde{E}_d(x_i^{(p)}, :)\|_2^2 + \alpha < \|\widetilde{E}_d(x_i^{(a)}, :) - \widetilde{E}_d(x_i^{(n)}, :)\|_2^2, \tag{5}$$

where α is a hyper-parameter controlling the distance margin for the metric learning loss and $\widetilde{E}_d \in \mathbb{R}^{n\times m}$ is L_2 row normalization of E_d. Our final triplet loss is in the form of:

$$L_{aux} = \frac{1}{N}\sum_{i=1}^{N}[\|\widetilde{E}_d(x_i^{(a)}, :) - \widetilde{E}_d(x_i^{(p)}, :)\|_2^2 - \|\widetilde{E}_d(x_i^{(a)}, :) - \widetilde{E}_d(x_i^{(n)}, :)\|_2^2 + \alpha]_+, \tag{6}$$

where the operator $[z]_+ = max(z, 0)$ and N is the number of possible triplets in the training set. By adding the minimizing this auxiliary loss, we force the sparse drivers to be close to their similar dense drivers in the embedding space. Thus they have a chance to achieve good personalized prediction adaptation though lacking trajectory data.

In practice, we compute the driver average speeds before training ETA model. During the training, we construct the triplets in a batch manner. We first randomly select a mini-batch of samples $\mathcal{B}_a$ – just the same as the common neural network training. The samples are duplicated and shuffled twice to generate $\mathcal{B}_p$ and $\mathcal{B}_n$. Note that these 3 mini-batches contain the same samples but have different order. We then align $\mathcal{B}_a$, $\mathcal{B}_p$ and $\mathcal{B}_n$, and check whether the triplets satisfy Eq. 3. If the inequation does not hold, we accordingly exchange the samples in $\mathcal{B}_p$ and $\mathcal{B}_n$ until each of the triplets is valid. Such an in-place construction leads to an acceleration of data loading.

E. CoDriver ETA

We propose *CoDriver ETA* to combine the information of dense drivers and sparse drivers, as shown in Fig. 6. Following the multi-task learning framework, it consists of a main task and an auxiliary task. The main task is to learn a basic ETA model as introduced in Section III-A. This model captures the spatial and temporal correlations between the complicated traffic factors, and finally outputs a prediction of travel time. We choose 20 as the dimension of embedding space for the discrete features. For the hidden state sizes of LSTM and MLP regressor, we choose 128. The global inputs include driver ID, time slice and day of week. For each link, we have 4 features including link ID, link length, estimated link speed and its corresponding passing time. The estimated link speed is an average of the probe car speeds in the last 10 minutes (if no car scanned the link, we use a default speed).

The auxiliary task is to transfer knowledge of driving style from dense drivers to sparse drivers. Its input triplets are

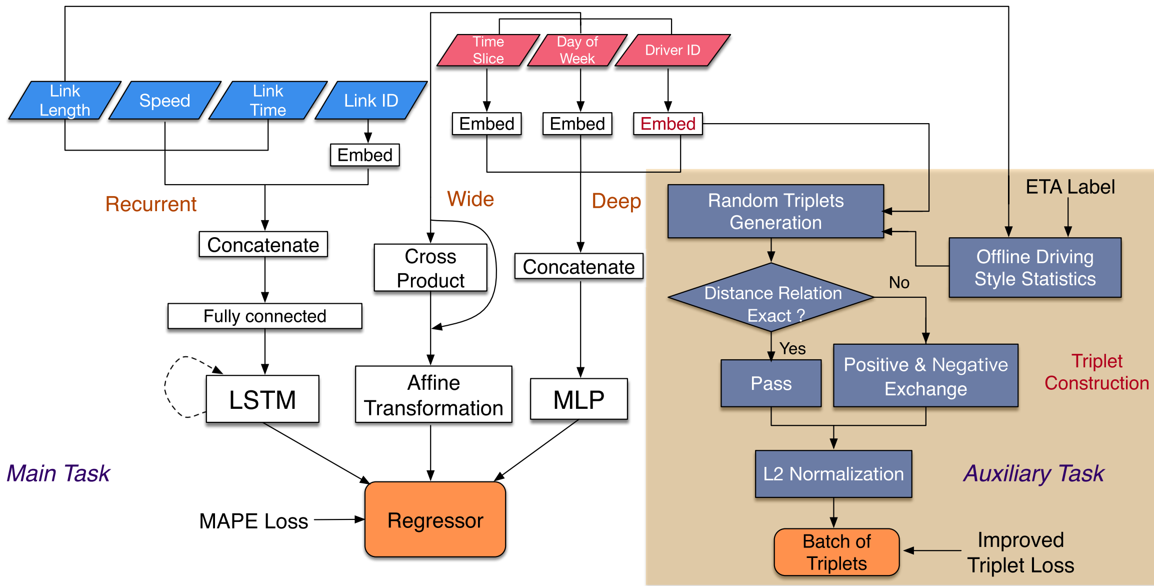


Fig. 6. The overall architecture of *CoDriver ETA*. Following the MTL framework, the proposed model can be divided into the main task and the auxiliary task. The main task is to learn a basic ETA model as introduced in Section III-A. The three parts(wide, deep and recurrent) learns the spatial correlations and temporal dependencies from global sparse features as well as sequential features and outputs a prediction of travel time. The auxiliary task is to transfer knowledge of driving style from dense drivers to sparse drivers. We complete the offline driving style statistics and then realize triplet construction using improved triplet loss.

extracted from the driver embedding vectors of the main task. It can be seen as a regularization to encourage the representations of drivers in similar style to be together, and the representations of drivers in different style to be farther away. We use a hyper-parameter β to balance the trade-off between the main task and the auxiliary task. β is the weighting coefficient of the objective function of the two tasks. That is to say, our objective function is:

$$L = (1 - \beta) \cdot L_{main} + \beta \cdot L_{aux}. \tag{7}$$

The model parameters are optimized under this loss. Fig. 6 summarizes the architecture of *CoDriver ETA*.

IV. Experiment

A. Dataset

We collect and organize two massive real-world floating-car datasets in Beijing of more than 8 months in 2018 on DiDi platform. These two datasets represent two types of working status of the drivers. The first is the *pickup dataset* which contains the orders when the driver responds to a passenger's request until the driver picks up the passenger. The second is the *trip dataset* which contains the orders when the passenger is on the journey from his/her origin to destination. Both the two massive spatio-temporal datasets contain a variety of complicated road types and driving habits of a large number of drivers.

Notice that we need to remove the abnormal cases with extremely short travel time (<60s) and extremely high travel speed (>120km/h) before experimenting with these datasets. We divide these two massive real-world floating-car datasets into training sets (first 8 months), validation sets (next 2 weeks) and test sets (last 2 weeks), and summarize their statistics in Table I. From the table, we can see that the sample numbers of orders in both datasets are huge, and the number of drivers in each dataset is also very large. The statistics of driver order number and driver data sparsity problem are discussed in detail in Section III-B.

TABLE I
STATISTICS OF TWO MASSIVE REAL-WORLD FLOATING-CAR DATASETS

	duration	pickup dataset	trip dataset
training set	8 months	111.0M	105.5M
validation set	2 weeks	4.0M	4.5M
test set	2 weeks	4.1M	3.9M
unique drivers	-	355.7k	249.2k

B. Comparison Methods

On both two massive datasets, we compare our method with several competitors. According to the results in [2], the performance of non-deep learning method is obviously worse than that of deep learning-based method. We choose a representative method: route-ETA among non-deep learning methods. The main competitor in this article is the WDR model proposed by [2]. This method achieves state-of-the-art performance for the ETA task in the literature. We also evaluate WDR-no-driver in order to explore the effect of driver embedding on the accuracy of overall prediction.

- Route-ETA is simple and effective, and thus is widely used in many location-based services. This solution first splits a path into several road links along a given route. It requires a traffic monitoring service to provide estimated link speed and delay time at each intersection. The travel time in each link is estimated by dividing its length by the real-time traffic speed on the road link. The overall ETA is a weighted sum of the travel time of each link and delay time at each intersection.

- Wide-Deep-Recurrent (WDR) is a deep learning-based ETA model. Via combining the wide, deep, and recurrent models, WDR can make use of different types of feature information. The model details are introduced in Section III-A.
- WD-FFN is also a deep learning-based method. The difference between WD-FFN and WDR is that WD-FFN adopt the feed-forward network (FFN) to mine the information from sequential features. FFN is simpler than RNN in terms of computational complexity. WD-FFN is a common and advanced method.
- Resnet is another state-of-the-art deep learning model which is first proposed in the field of computer vision [39]. Resnet is utilized to predict the travel time end to end [3] and for OD ETA [11]. Sufficient experiments prove the ability of this deep neural network in spatio-temporal data mining [3], [11] for ETA.
- WDR-no-driver is a variant of WDR. The only difference is that WDR-no-driver discards the driver ID for input, while the other model parts such as the amount of learnable parameters and output are almost the same. Naturally, there is no need to embed driver ID. WDR-no-driver reflects the difference in prediction performance between with and without driver ID information for WDR, which further reflects the extent to which WDR can mine driver information.

C. Experimental Settings

In our comparative experiments, deep neural network based solutions, including WDR, WDR-no-driver and *CoDriver ETA*, are all implemented in Python using PyTorch toolbox [40]. We parallelize the training on 4 NVIDIA Tesla P40 GPU cards. The number of iterations of WDR and *CoDriver ETA* is set to 7M consistently with mini-batch size equals to 256. The typical training duration of WDR and *CoDriver ETA* are 19 hours and 22 hours on *pickup*. On *trip*, the average duration of training process corresponding to WDR and *CoDriver ETA* are 25 hours and 26 hours. The hyperparameters α and β of *CoDriver ETA* are chosen on the validation dataset. The α for both *pickup* and *trip* is 0.01. We set β for *pickup* to 0.45 and for *trip* to 0.38.

The WDR and *CoDriver ETA* are trained using BP. The objective is the MAPE loss for WDR and L in Section III-E. We choose Adam [41] to optimize the WDR and *CoDriver ETA*. The advantages of Adam is that this stochastic gradient descent method has an adaptive step size and momentum [41]. We set the initial learning rate of Adam to 0.0002. The update of adaptive statistics is in a lazy manner, namely, the momuntums are accumulated only if the corresponding embedding vectors are used in the mini-batch. Otherwise, the momuntum of embedding parameter will be frequently moving averaged with zeros.

1) Evaluation Metrics: To evaluate the performance of proposed method and other competitors, we adopt three classical and popular metrics. Suppose that y_i' represents the our final estimation of travel time, y_i represents the ground-truth label for each sample, and N denotes the number of samples, the three metrics are as follows (the lower the better):

Mean Absolute Percentage Error (MAPE) which is expressed in formula 1,

Mean Absolute Error (MAE):

$$\mathrm{MAE}(y,\ y') = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - y_i'\right|, \tag{8}$$

and Mean Square Error (MSE):

$$\mathrm{MSE}(y,\ y') = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2. \tag{9}$$

Among them, the most important metric is MAPE which is relative and reasonable. MAPE is also chosen as the loss function of WDR and main task of *CoDriver ETA*.

TABLE II
THE COMPETING RESULTS ON THE PICKUP DATASET

	MAPE	MAE	MSE
route-ETA	25.963%	75.030	13471.375
WD-FFN	20.717%	56.215	8270.623
Resnet	20.610%	56.410	8384.804
WDR-no-driver	19.895%	55.699	8194.264
WDR	19.386%	54.700	8080.265
CoDriver ETA (ours)	*19.166%*	*53.503*	*7708.510*
CoDriver ETA + WDR (ours)	**19.136%**	**53.390**	**7665.027**

TABLE III
THE COMPETING RESULTS ON THE TRIP DATASET

	MAPE	MAE	MSE
route-ETA	15.635%	150.346	60862.980
WD-FFN	11.999%	111.688	36360.641
Resnet	12.028%	111.503	36086.929
WDR-no-driver	12.095%	113.075	36665.827
WDR	11.744%	108.924	*34399.237*
CoDriver ETA (ours)	**11.587%**	**108.272**	34548.308
CoDriver ETA + WDR (ours)	*11.681%*	*108.524*	**34231.267**

D. Comparison and Analysis of Results

1) Competing Results: We present the competing results on *pickup* and *trip*, as shown in Table II and Table III. We mark in tables results of the best method under the corresponding metric by **black bold**, and results of the second best method by *italic*.

The results show that our *CoDriver ETA* and the combined method outperform all competitors regarding to all the metrics on both datasets. *CoDriver ETA* outperforms the state-of-the-art method, i.e., WDR, by 1.1% and 1.3% in terms of MAPE on *pickup* and *trip*. *CoDriver ETA* outperforms WDR by 2.2% and 0.6% in terms of MAE on *pickup* and *trip*, respectively. On *pickup*, compared with WDR, *CoDriver ETA*'s prediction performance is significantly better regarding to all the metrics in the sparse part of the test set, but slightly inferior regarding to all the metrics in the dense part. We will illustrate this point in the following detailed local experimental results. We try to adopt *CoDriver ETA* in the sparse part and WDR in the dense part as another method. The thresholds for deciding which method to adopt are chosen as 230 on *pickup* and 500 on *trip* on the validation dataset. We observe that this combined algorithm(*CoDriver ETA* + WDR) has been further upgraded on the basis of *CoDriver ETA* regarding to all the metrics. On *trip*, compared with WDR, *CoDriver ETA*'s performance is significantly better regarding to all the metrics in the sparse part, and also slightly better regarding to MAPE and MAE in the dense part. *CoDriver ETA* is slightly inferior to WDR regarding to MSE in the dense part. So on *trip*, *CoDriver ETA* + WDR outperforms WDR regarding to all metrics and outperforms *CoDriver ETA* in terms of MSE. *CoDriver ETA* + WDR outperforms WDR by 5.1% and 0.5% in terms of MSE on *pickup* and *trip*. The above results show that *CoDriver ETA* + WDR is also a potential method in production scenarios.

We observe that the deep learning-based methods (WDR, WDR-no-driver, WD-FFN, Resnet, *CoDriver ETA*) outperform the non-deep learning route-ETA. This is consistent with the results shown in [2]. WDR outperforms the WDR-no-driver by 2.6% and 2.9% while *CoDriver ETA* outperforms the WDR-no-driver 3.7% and 4.2% in terms of MAPE on *pickup* and *trip*, separately. This demonstrates that *CoDriver ETA* has obvious advantages over WDR in mining driver information. Two advanced deep learning-based methods for ETA, WD-FFN and Resnet is similar to WDR-no-driver on *pickup* and *trip* in terms of prediction accuracy. The proposed *CoDriver ETA* outperforms these two methods evidently.

2) Detailed Local Experimental Results: To show whether *CoDriver ETA* address the driver data sparsity, we explore the performance of the proposed model in the sparse and dense parts of the dataset respectively. Therefore, we design a series of detailed local experiments on *pickup*. For the sparse part, we select a subset of the test set to test WDR as well as *CoDriver ETA* and compare their performance. The drivers in the sparse subset have no more than a certain number of orders in the training set. Similarly, for the dense part, the drivers in the dense subset have no less than a certain number of orders. We select several representative certain values and record the values, the remaining rate and three metrics of the two methods in Table IV and Table V. The remaining rate refers to the sample number proportion of the subset to the entire test set. In order to show the performance comparison more intuitively, we draw the results in Fig. 7.

From the tables and graphs, we can summarize the following results. First, the driver data sparsity problem does have a great impact on WDR model. It is necessary to pay attention to and alleviate this problem. All three metrics corresponding to both methods become better as the degree of sparsity decreases. Second, our *CoDriver ETA* is obviously superior to WDR in the sparse part, showing that our method addresses the driver data sparsity problem effectively. Third, the performance of WDR on the dense subset is significantly better than that in the overall dataset regarding to all the three metrics. This again shows that this sparsity problem deserves attention. Fourth, compared with WDR, *CoDriver ETA*'s performance is significantly better in the sparse part, but slightly inferior in the dense part in terms of three metrics. As a result, the combined algorithm is a little better than *CoDriver ETA* in Table II.

3) Dense to Sparse Experimental Results: In addition to the above detailed local experiments, we also show the superiority of *CoDriver ETA* in alleviating the driver data sparsity problem from another angle. We extract randomly 10 dense drivers with orders exceeding 1000 on *pickup*. In the training set, we only keep 90 orders for each driver. Then the 10 drivers are artificially turned into sparse drivers. We train WDR and *CoDriver ETA* in this training set, and see the performance of the two models on these 10 drivers in the test set. WDR trained with complete dataset is as a ceiling to observe the degree of *CoDriver ETA*'s alleviation for the driver data sparsity problem. We show the comparison regarding to three metrics in Fig. 8. The horizontal axis shows the number of orders for the driver. Fig. 8 shows that the performance of *CoDriver ETA* with 90 orders is significantly better than WDR with 90 orders in terms of all metrics. The metrics of *CoDriver ETA* with 90 orders are mostly slightly higher than WDR (the ceiling) and in some cases even slightly lower than the ceiling.

TABLE IV

DETAILED LOCAL EXPERIMENTAL RESULT COMPARISON ON SPARSE PART OF DATASET

Max order	MAPE		MAE		MSE	
(Remain rate)	WDR	*CoDriver*	WDR	*CoDriver*	WDR	*CoDriver*
10(4.33%)	23.815%	**20.244%**	74.340	**61.034**	13093.939	**9963.047**
20(5.09%)	23.653%	**20.273%**	73.461	**60.776**	12888.062	**9873.862**
30(5.81%)	23.435%	**20.248%**	72.359	**60.370**	12600.946	**9751.099**
40(6.55%)	23.204%	**20.225%**	71.335	**60.065**	12312.406	**9631.499**
50(7.29%)	22.968%	**20.201%**	70.321	**59.775**	12056.244	**9538.958**
70(8.76%)	22.541%	**20.143%**	68.560	**59.298**	11643.865	**9430.857**
90(10.22%)	22.178%	**20.072%**	67.021	**58.808**	11230.429	**9264.514**
110(11.56%)	21.914%	**20.030%**	65.881	**58.479**	10940.222	**9169.192**
130(12.92%)	21.699%	**19.997%**	64.864	**58.146**	10694.241	**9091.039**
150(14.21%)	21.511%	**19.962%**	64.069	**57.919**	10484.219	**9015.702**
170(15.48%)	21.342%	**19.923%**	63.288	**57.624**	10300.212	**8948.125**
190(16.76%)	21.210%	**19.896%**	62.674	**57.421**	10136.224	**8885.229**
210(17.95%)	21.095%	**19.866%**	62.103	**57.191**	9982.705	**8815.967**
230(19.14%)	21.000%	**19.844%**	61.653	**57.038**	9863.928	**8773.139**
250(20.32%)	20.920%	**19.830%**	61.244	**56.897**	9753.483	**8729.330**

TABLE V

DETAILED LOCAL EXPERIMENTAL RESULT COMPARISON ON DENSE PART OF DATASET

Min order	MAPE		MAE		MSE	
(Remain rate)	WDR	*CoDriver*	WDR	*CoDriver*	WDR	*CoDriver*
1000(40.64%)	**18.779%**	18.819%	**52.268**	52.516	**7253.875**	7337.521
1100(36.02%)	**18.774%**	18.815%	**52.283**	52.538	**7240.760**	7324.598
1200(31.59%)	**18.753%**	18.795%	**52.240**	52.493	**7191.038**	7274.175
1300(27.32%)	**18.744%**	18.785%	**52.247**	52.504	**7166.228**	7250.805
1400(23.37%)	**18.733%**	18.779%	**52.251**	52.527	**7167.288**	7257.489
1500(19.75%)	**18.721%**	18.764%	**52.313**	52.580	**7180.187**	7269.233
1600(16.42%)	**18.721%**	18.762%	**52.386**	52.658	**7214.820**	7305.790
1700(13.19%)	**18.714%**	18.752%	**52.448**	52.728	**7241.567**	7332.876
1800(10.44%)	**18.710%**	18.745%	**52.527**	52.815	**7306.360**	7394.687
1900(8.06%)	**18.714%**	18.754%	**52.608**	52.915	**7367.987**	7461.961
1950(6.99%)	**18.734%**	18.771%	**52.724**	53.002	**7393.562**	7477.222

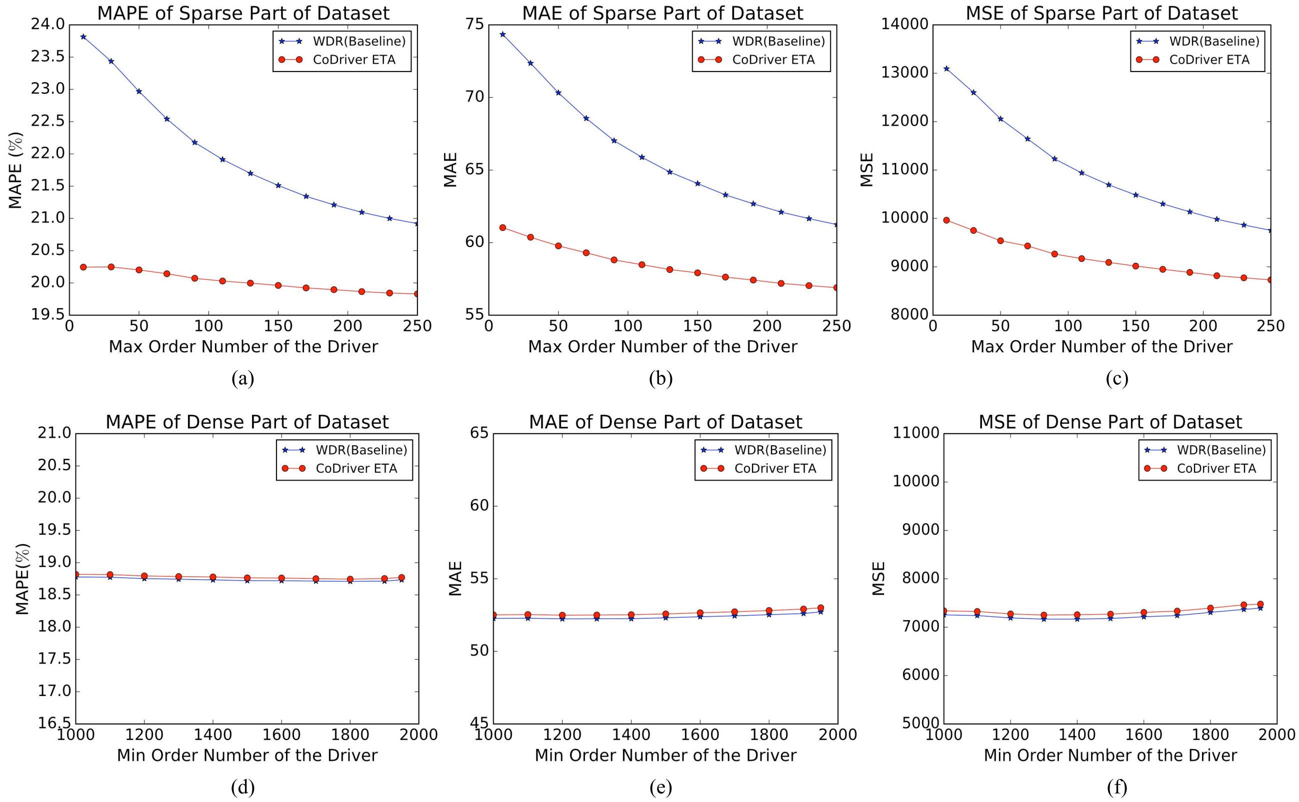


Fig. 7. Detailed local experimental result comparison on sparse and dense part of dataset. *CoDriver ETA* is obviously superior to WDR in the sparse part and slightly inferior to WDR in the dense part. (a) Comparison regarding to MAPE of sparse part. (b) Comparison regarding to MAE of sparse part. (c) Comparison regarding to MSE of sparse part. (d) Comparison regarding to MAPE of dense part.(e) Comparison regarding to MAE of dense part. (f) Comparison regarding to MSE of dense part.

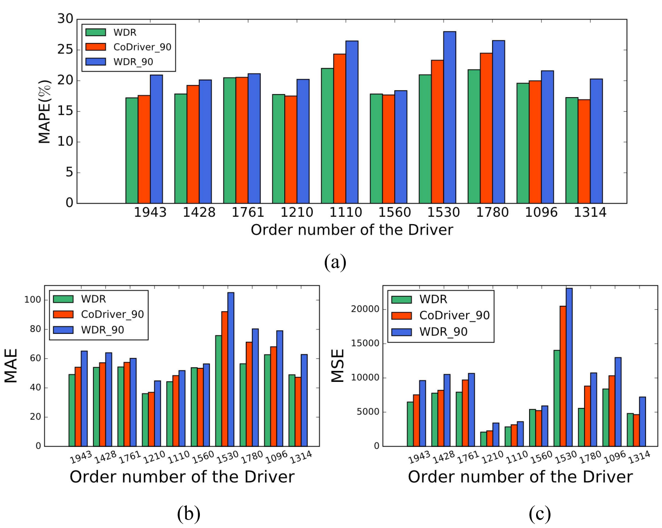


Fig. 8. Dense to sparse experimental result comparison. The performance of *CoDriver ETA* with 90 orders is significantly better than WDR with 90 orders in terms of all metrics and mostly slightly worse than WDR (the ceiling). (a) Comparison regarding to MAPE. (b) Comparison regarding to MAE. (c) Comparison regarding to MSE.

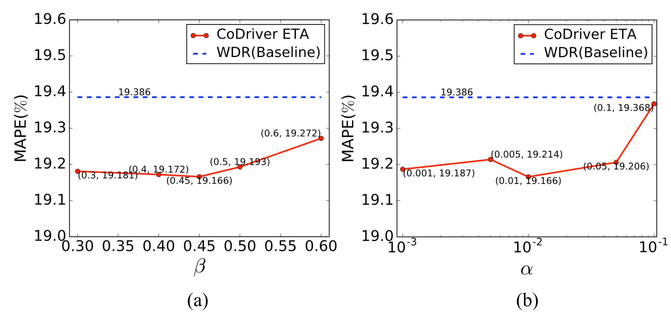


Fig. 9. The effects of main hyperparameters on *CoDriver ETA*. (a) Combination coefficient β of multi-task learning. (b) Margin α of improved triplet loss.

The results show that *CoDriver ETA* has a distinct improvement in response to the driver data sparsity problem.

4) Effects of Hyperparameters: We evaluate *CoDriver ETA* on *pickup* under different combination coefficient β (when $\alpha = 0.01$) and different margin α (when $\beta = 0.45$). The result in Fig. 9 shows that *CoDriver ETA* is pretty robust for a wide range of β and α. The error rates are high only when $\alpha = 0.1$. Moreover, *CoDriver ETA* under all these hyperparameters is better than WDR.

5) Visualizations of Driver Embedding: In order to explore the distribution of driver embedding in WDR and *CoDriver ETA* models, we conduct a series of visualizations. We randomly select about 1000 orders on the test set. After that, the driver embedding vectors of two algorithms on two datasets are reduced to two dimensions through t-SNE [42]. We show the results of dimensionality reduction in Fig. 10.

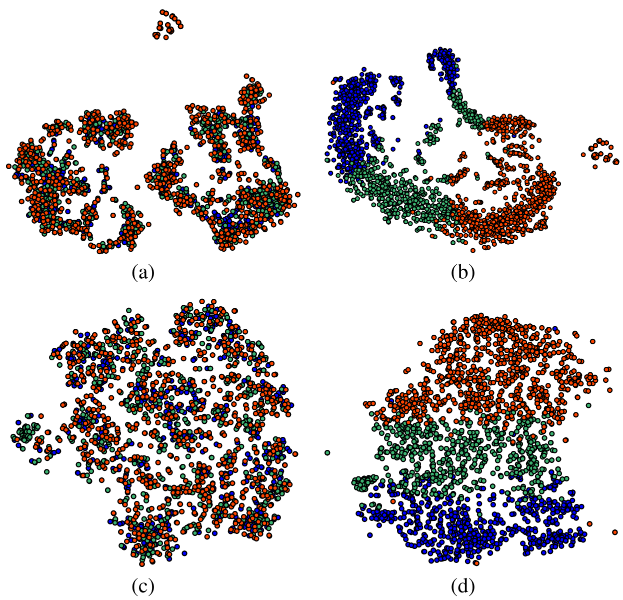


Fig. 10. Visualization of driver embedding space after t-SNE [42] on *pickup* and *trip*. (a) Driver embedding of WDR on *pickup*. (b) Driver embedding of *CoDriver ETA* on *pickup*. (c) Driver embedding of WDR on *trip*. (d) Driver embedding of *CoDriver ETA* on *trip*.

In the figure, the average speeds of the drivers corresponding to the red point are relatively fast, the average speeds of the drivers corresponding to the green points are general, and the average speeds of the drivers corresponding to the blue points are relatively slow. The threshold to distinguish the three colors is chosen by the tertiles of the average speeds of drivers.

Fig. 10(a) and Fig. 10(c) show that the red dots, green dots and blue dots in visualization results of the WDR model are mixed together. It shows that the WDR model can not learn the driving style of the driver's average speed and embody it in the driver's embedding during the test phase. In contrast, Fig. 10(b) and Fig. 10(d) show that the red dots, green dots and blue dots in visualization results of the *CoDriver ETA* are clearly distributed in three regions. The proposed model can achieve the effect that drivers with similar driving style are closer in the embedded space. In this way, the driver information are combined by driving style learning auxiliary task.

V. Conclusion and Future Work

In this work, we discuss the driver data sparsity problem in ETA in detail. It is worthy of attention for it has a great impact on performance of the learning-based method using driver embedding. To address this problem, we propose a novel deep learning model: *CoDriver ETA*. The core idea is to combine the information of sparse drivers and dense drivers. We adopt MTL architecture so that the driver embedding can be optimized by both ETA main task and the driving style learning auxiliary task. We characterize personalized driving styles with no additional information. Then we improve the triplet loss to measure the distance between different drivers' driving styles. With such a mechanism, proposed model can transfer knowledge from the dense drivers to the sparse drivers, and thus enhance the prediction performance. A series of experimental results demonstrate that our method alleviates the driver data sparsity problem obviously. We evaluated on two billions of real-world floating-car datasets from Didi Chuxing platform, our *CoDriver ETA* outperforms the state-of-the-art baselines.

Because our method is based on deep learning which is known as the black-box model, the weak interpretability is the limitation of *CoDriver ETA*. With regard to the future efforts, how to mine the spatial-temporal dependencies effectively with satisfactory interpretability is of importance and may lead to great progress. Moreover, though *CoDriver ETA* is proposed for ETA learning, it is flexible enough to be applied for other tasks in transportation research. Our second future work is to solve other important ITS problems, especially the prediction problem, using the ideas put forward in this article.

Acknowledgment

The authors would like to sincerely thank Jiang Lu and Ziang Yan for the discussion with them.

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[2] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 858–866.

[3] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[4] E. Jenelius and H. N. Koutsopoulos, "Travel time estimation for urban road networks using low frequency probe vehicle data," *Transp. Res. B, Methodol.*, vol. 53, pp. 64–81, Jul. 2013.

[5] W.-C. Lee, W. Si, L.-J. Chen, and M. C. Chen, "HTTP: A new framework for bus travel time prediction based on historical trajectories," in *Proc. 20th Int. Conf. Adv. Geograph. Inf. Syst. (SIGSPATIAL)*, 2012, pp. 279–288.

[6] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2014, pp. 25–34.

[7] X. Zhan, S. Hasan, S. V. Ukkusuri, and C. Kamga, "Urban link travel time estimation using large-scale taxi data with partial information," *Transp. Res. C, Emerg. Technol.*, vol. 33, pp. 37–49, Aug. 2013.

[8] A. Hofleitner, R. Herring, P. Abbeel, and A. Bayen, "Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1679–1693, Dec. 2012.

[9] H. Chen, H. A. Rakha, and C. C. McGhee, "Dynamic travel time prediction using pattern recognition," in *Proc. 20th World Congr. Intell. Transp. Syst.*, 2013, pp. 1–17.

[10] F. Zhang, X. Zhu, T. Hu, W. Guo, C. Chen, and L. Liu, "Urban link travel time prediction based on a gradient boosting method considering spatiotemporal correlations," *ISPRS Int. J. Geo-Inf.*, vol. 5, no. 11, p. 201, Nov. 2016.

[11] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1695–1704.

[12] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic estimation and prediction based on real time floating car data," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 197–203.

[13] X. Zhan, Y. Zheng, X. Yi, and S. V. Ukkusuri, "Citywide traffic volume estimation using trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 272–285, Feb. 2017.

[14] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 220–232, Jan. 2013.

[15] H. Wang, Y. H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst., (ACM SIGSPATIAL GIS)*. New York, NY, USA: Association for Computing Machinery, 2016, p. 61.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[17] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[18] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Proc. Interspeech*, 2013, pp. 3771–3775.

[19] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[21] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 163–168.

[22] D. Li and H. Gao, "A hardware platform framework for an intelligent vehicle based on a driving brain," *Engineering*, vol. 4, no. 4, pp. 464–470, Aug. 2018.

[23] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4224–4231, Sep. 2018.

[24] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5999–6008, Jul. 2018.

[25] H. Gao, W. Bi, X. Wu, Z. Li, Z. Kan, and Y. Kang, "Adaptive fuzzy region-based control of euler-Lagrange systems with kinematically singular configurations," *IEEE Trans. Fuzzy Syst.*, early access, May 15, 2020, doi: 10.1109/TFUZZ.2020.2994991.

[26] H. Gao *et al.*, "EEG-based volitional control of prosthetic legs for walking in different terrains," *IEEE Trans. Autom. Sci. Eng.*, early access, Dec. 27, 2020, doi: 10.1109/TASE.2019.2956110.

[27] Y. LeCun *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Netw., Stat. Mech. Perspective*, vol. 261, p. 276, Jan. 1995.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[29] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, no. 1, pp. 1–40, Jan. 2009.

[30] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[31] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan, "Deep spatial–temporal 3D convolutional neural networks for traffic data forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3913–3926, Oct. 2019.

[32] Y. Sun, Y. Wang, K. Fu, Z. Wang, C. Zhang, and J. Ye, "Constructing geographic and long-term temporal graph for traffic forecasting," 2020, *arXiv:2004.10958*. [Online]. Available: http://arxiv.org/abs/2004.10958

[33] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI*, vol. 33, 2019, pp. 922–929.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] T.-Y. Fu and W.-C. Lee, "Deepist: Deep image-based spatio-temporal network for travel time estimation," in *Proc. ACM CIKM*, 2019, pp. 69–78.

[36] W. Lan, Y. Xu, and B. Zhao, "Travel time estimation without road networks: An urban morphological layout representation approach," in *Proc. IJCAI*, Aug. 2019, pp. 1772–1778.

[37] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[38] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[40] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.

[42] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, Oct. 2014.

Yiwen Sun (Graduate Student Member, IEEE) received the B.E. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University. His research interests include machine learning, deep learning, sequence learning, spatio-temporal data mining, and intelligent transportation systems.

Kun Fu (Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Automation, Tsinghua University, Beijing, China, in 2011 and 2017, respectively. He is currently a Researcher with DiDi AI Labs. His research interests include machine learning, deep learning, and intelligent transportation systems.

Zheng Wang (Member, IEEE) received the Ph.D. degree from the Department of Automation, Tsinghua University, China, in 2011. He was a Research Faculty with the University of Michigan, Ann Arbor, MI, USA. He is currently a Researcher and the Principle Engineer of DiDi AI Labs. His research interests include big data, machine learning, and data mining, with applications in transportation. His papers have been selected for the KDD Best Research Paper Runner-Up in 2013 and the SocialCom Best Paper Award in 2013. He has served on Senior Program Committee/Program Committee for many conferences, including NIPS, ICML, KDD, ICLR, IJCAI, and AAAI.

Donghua Zhou (Fellow, IEEE) received the B.Eng., M.Sci., and Ph.D. degrees in electrical engineering from Shanghai Jiaotong University, China, in 1985, 1988, and 1990, respectively. He was an Alexander von Humboldt Research Fellow with the University of Duisburg, Germany, from 1995 to 1996, and a Visiting Scholar with Yale University, USA, from 2001 to 2002. In 1996, he joined Tsinghua University, where he was promoted as a Full Professor in 1997. He was the Head of the Department of Automation, Tsinghua University, from 2008 to 2015. He is currently the Vice President of the Shandong University of Science and Technology and a joint Professor with Tsinghua University. He has authored or coauthored more than 210 peer-reviewed international journal articles and seven monographs in the areas of fault diagnosis, fault-tolerant control, and operational safety evaluation. He is a fellow of CAA and IET, a member of IFAC TC on SAFEPROCESS, the Vice Chairman of the Chinese Association of Automation (CAA), and the TC Chair of the SAFEPROCESS Committee, CAA. He was the NOC Chair of the 6th IFAC Symposium on SAFEPROCESS 2006. He is an Associate Editor of *Journal of Process Control*.

Kailun Wu (Member, IEEE) received the B.E. degree from Tsinghua University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of Intelligent Technology and Systems, Department of Automation. His research interests include deep learning and sparse coding.

Jieping Ye (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Minnesota, Twin Cities, MN, USA, in 2005. He is currently the VP of Didi Chuxing and a Didi Fellow. He is also an Associate Professor with the University of Michigan, Ann Arbor, MI, USA. His research interests include big data, machine learning, and data mining, with applications in transportation and biomedicine. He was a recipient of the NSF CAREER Award in 2010. His papers have been selected for the Outstanding Student Paper at ICML in 2004, the KDD Best Research Paper Runner Up in 2013, and the KDD Best Student Paper Award in 2014. He has served as the Senior Program Committee/Area Chair/Program Committee Vice-Chair for many conferences, including NIPS, ICML, KDD, IJCAI, ICDM, and SDM. He has served as an Associate Editor for *Data Mining and Knowledge Discovery*, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.

Changshui Zhang (Fellow, IEEE) received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, in 1989 and 1992, respectively. In 1992, he joined the Department of Automation, Tsinghua University, where he is currently a Professor. He has authored more than 200 articles. His current research interests include pattern recognition and machine learning. He is an Associate Editor of *Pattern Recognition* and the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.