Exploiting Multi-source Data for Adversarial Driving Style Representation Learning

Zhidan Liu[1(✉)], Junhong Zheng[1], Zengyang Gong[2], Haodi Zhang[1], and Kaishun Wu[1]

[1] Shenzhen University, Shenzhen, China
{liuzhidan,hdzhang,wu}@szu.edu.cn,
zhengjun4hong2019@email.szu.edu.cn

[2] Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
zgongae@cse.ust.hk

Abstract. Characterizing human driver's driving behaviors from GPS trajectories is an important yet challenging trajectory mining task. Previous works heavily rely on high-quality GPS data to learn such driving style representations through deep neural networks. However, they have overlooked the driving contexts that greatly govern drivers' driving activities and the data sparsity issue of practical GPS trajectories collected at a low-sampling rate. To address the limitations of existing works, we present an adversarial driving style representation learning approach, named `Radar`. In addition to summarizing statistic features from raw GPS data, `Radar` also extracts contextual features from three aspects of road condition, geographic semantic, and traffic condition. We further exploit the advanced semi-supervised generative adversarial networks to construct our learning model. By jointly considering statistic features and contextual features, the trained model is able to efficiently learn driving style representations even from sparse trajectories. Experiments on two benchmark applications, *i.e.*, driver number estimation and driver identification, with a large real-world GPS trajectory dataset demonstrate that `Radar` can outperform the state-of-the-art approaches by learning more effective and accurate driving style representations.

Keywords: GPS trajectory · Multi-source data · Driving style representation · Generative adversarial networks

1 Introduction

The advances of GPS and wireless communication techniques have enhanced the ability of various systems in collecting the spatio-temporal vehicular trajectories. The massive GPS trajectories stimulate a number of trajectory mining tasks for better understanding human mobility patterns and behaviors [29], among which characterizing human driver's driving behaviors is an important yet challenging task. Similar as the bio-metrics, it is believed that each driver also has a distinguishable pattern of driving, which is referred as *driving style* [13]. Specifically, driving style reflects a driver's fine-grained behavioral habits of steering and speed control and their temporal combinations [2]. Learning drivers' driving style representations from their trajectories can benefit many intelligent applications, *e.g.*, driving assessment and assistance [25], driver-vehicle interaction [13], autonomous driving [12], and *etc*. In addition, auto insurance companies have been interested in utilizing the driving style information for risk assessments and personalized insurance pricing [9].

In the literature, some valuable efforts have been made to derive the driving style representations. Traditional approaches heavily rely on the data collected from automobile sensors (*e.g.*, controller area network buses) [6] or dedicated sensors (*e.g.*, high-definition cameras) [8] for driving style learning. However, it is difficult to retrieve data from automobile sensors while dedicated devices will incur installation costs. Recent studies [2,10,30] turn to leverage deep learning models to process GPS trajectories for learning the driving style representations. Compared to automobile and dedicated sensors, GPS sensor data are often easier to access and thus are more popular in large-scale study [2,29]. These works, however, require high-frequency rate of GPS data collections, which may be prohibited due to privacy and energy consumption [15]. Furthermore, these works merely focus on feature extractions from GPS data, but have overlooked the instant driving context information, such as road conditions and traffic conditions. As a result, they are inadequate to acquire accurate driving style representations.

Despite these research efforts, it is still non-trivial to efficiently learn driving style representation from GPS trajectories, mainly due to following challenges. First, practical GPS data are usually collected at a low-sampling rate, *e.g.*, 1 sample per 30 s [15], and are probably sparse, *i.e.*, there may be insufficient qualified data to train a deep learning model [10]. Second, driving is a complex activity and the resultant driving style is influenced by many factors. The GPS trajectory data themselves cannot capture the complete view of a driver's driving style, and hence the external context information should be taken into account. However, how to properly integrate the features from GPS data and context information into one model needs to be well designed and thus is challenging.

In this paper, we present an adve**r**s**a**rial **d**riving style represent**a**tion lea**r**ning approach, named `Radar`, which extracts comprehensive features from multi-source data and builds a semi-supervised generative adversarial networks (SGAN) based model to learn driving style representations from these extracted features. To better describe a driver's driving behaviors, `Radar` not only transforms raw GPS trajectory data to fine-grained statistic features about driver's habits of steering and speed control, but also additionally considers each GPS trajectory's contextual features, which are captured by three aspects of road condition, geographic semantic, and traffic condition. In particular, different from the specific GPS locations, geographic semantic encodes high-level geographic features of a trajectory by mapping it to the whole city area. These driving contexts greatly govern a driver's driving activity, and thus are important complements for learning driving styles. To tackle data sparsity issue, `Radar` makes use of SGAN to construct the learning model, which equally treats statistic features and contextual features as the input to learn the driving style representations. Our learning model consists of three different components: generator, discriminator, and classifier, which work together to not only classify drivers from inputted trajectories but also generate fake samples close to the training data. As a result, `Radar`' learning model can achieve better generalization ability through both data augmentation and the competition between generator and discriminator.

In summary, the contributions of our work are as follows:

- To the best of our knowledge, we are the first to consider the problem of context-aware driving style representation learning from sparse trajectories, which improves existing works by considering the driving contexts.
- We propose an adversarial driving style representation learning approach – `Radar`, which exploits multi-source data and a SGAN based learning model to efficiently learn driving style representations from practical trajectories.
- We conduct extensive experiments with two benchmark applications, namely *driver number estimation* and *driver identification*, using a large real-world trajectory dataset. Experimental results demonstrate `Radar` outperforms state-of-the-art approaches, *e.g.*, on average improving the accuracy of driver number estimation and driver identification by 9.6% and 5.6%, respectively.

The remainder of the paper is organized as follows. We review related works in Sect. 2. The problem statement is presented in Sect. 3. We elaborate and evaluate our proposed approach in Sect. 4 and Sect. 5, respectively. Finally, Sect. 6 concludes this paper.

2 Related Work

The related works can be grouped into two categories: driving behavior analysis and trajectory mining. We review and discuss these works as follows.

Driving Behavior Analysis. Extensive studies have been conducted on the driving behavior analysis. Previous works primarily rely on the data collected from automobile sensors, *e.g.*, on-board diagnostic systems [9], controller area network buses [6], and digital cameras [8], to analyze drivers' driving behaviors. For example, Ezzini *et al.* utilize the measurements taken from various in-vehicle sensors to realize driver identification and fingerprinting [3]. However, it is relatively difficult to collect data from these automobile sensors, while dedicated devices like cameras bring installation costs. These constraints greatly limit their usability. Some recent works [1] resort to collect driving data using the internal sensors in smartphones and analyze the sensing data for monitoring drivers' behaviors. These works concern about driving safety, rather than driving styles.

Compared to automobile sensors, GPS sensor data are much easier to collect and GPS trajectory based driving behavior analysis has attracted many research efforts [2,10,25,27] in recent years. For example, Yang *et al.* analyze GPS traces of peer vehicles to proactively alter drivers of the vehicles with dangerous behaviors nearby [27]. By jointly modeling the peer and temporal dependencies of driving trajectories, Wang *et al.* enable the applications of driving score prediction and risk area detection [25]. The two works, however, mainly concern the identification of dangerous driving behaviors, rather than capturing a driver's latent driving styles. Instead, Dong *et al.* propose an autoencoder regularized deep neural network and a trip encoding framework to learn drivers' driving styles directly from GPS trajectories [2]. Tung *et al.* propose a trajectory-to-image representation framework that encodes both geographic features and driving behaviors of trajectories into multi-channel images [10]. Although the two works can achieve remarkable performances, they are still not sufficiently efficient and practical. First, they require high-quality GPS trajectories that are collected at a high-sampling rate such 1 Hz, while most practical GPS trajectories are collected at a low frequency, *e.g.*, 1 sample per 30 s, due to concerns of energy consumption and privacy [15]. Second, they merely extract features from GPS data while overlooking the driving contexts, within which a trajectory has been generated. Our approach overcomes these limitations by utilizing multi-source data to fully describe driving behaviors and the advanced SGAN modeling, and thus can learn more effective and accurate driving style representations.

Trajectory Mining. The wide availability of GPS trajectories has inspired a wide range of applications [29], *e.g.*, urban traffic estimation [17] and prediction [16], personalized recommender systems [28], and ridesharing [14]. To enable such applications, various trajectory mining tasks have been widely studied, *e.g.*, trajectory pattern mining [11], trajectory-user linking [30], and *etc.*. In particular, trajectory-user linking, which links trajectories to users who produce them, is quite relevant to our work. Existing works on this problem mainly analyze mobility trajectories by exploiting various deep learning models to learn the semantic trajectory representations [4,19,22,30]. For example, Feng *et al.* present a deep learning framework to link heterogeneous mobility data, which are collected from different online services, to the users [4]. Ren *et al.* build a spatio-temporal Siamese network model to predict whether an income set of trajectories belong to a certain agent based on historical trajectory data [22]. In addition, Miao *et al.* utilize recurrent networks with attention mechanism to solve the trajectory-user linking problem [19]. Different from these works, we aim to learn drivers' driving style representations from practical GPS trajectories, which involves more complex human behaviors and thus is more challenging.

3 Problem Statement

3.1 Definitions and Notations

The GPS trajectory data are collected when a set of drivers $\mathbb{U} = \{u_1, \cdots, u_{|\mathbb{U}|}\}$ drive their vehicles, which have been equipped with GPS sensors, on a road network. The GPS trajectory set $\mathbb{T}_{u_i}$ generated by driver u_i implicitly encodes u_i's driving style. Accurately learning the driving style representation can benefit many potential applications, such as driving assessment and assistance [25], driver-vehicle interaction [13], autonomous driving [12], and so on.

Definition 1 ***(GPS trajectory).*** *Let $\mathcal{T}_j^i \in \mathbb{T}_{u_i}$ denotes the j-th trajectory generated by driver u_i. Specifically, $\mathcal{T}_j^i = \{g_1, \cdots, g_{|\mathcal{T}_j^i|}\}$ is a time-ordered sequence of GPS records, where each record is denoted as a tuple $< ts, lat, lng, v, dir >$, indicating that u_i's vehicle located at latitude lat and longitude lng at time ts, with instant travel speed v in direction dir.*

Due to GPS localization errors, we have to map raw GPS locations to their actual locations on the roads through map matching techniques [20]. Therefore, a trajectory $\mathcal{T}_j$[1] could be mapped to a travel route $\mathcal{R}_j$ on the road network $\mathbb{G}$.

Definition 2 ***(Road network).*** *A road network is modelled as graph $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$, where $\mathbb{V}$ represents the set of road intersections and $\mathbb{E}$ represents the set of road segments in a city. In addition, each road segment has following attributes: ID of road segment, road type, number of lanes, and one-way indicator.*

Definition 3 ***(Travel route).*** *The travel route $\mathcal{R}_j$ for a GPS trajectory $\mathcal{T}_j$ is denoted by a sequence of road segments, i.e., $\mathcal{R}_j = \{e_1, e_2, \cdots, e_{|\mathcal{R}_j|}\}$, on road network $\mathbb{G}$, where $e_i \in \mathbb{E}$ is a road segment in route $\mathcal{R}_j$ and $|\mathcal{R}_j|$ is the number of all traveled road segments. Note that end point of e_i is the start point of e_{i+1}.*

3.2 Problem Statement

Definition 4 ***(Context-aware driving style representation learning).*** *Given a set of GPS trajectories generated by drivers in $\mathbb{U}$, we aim to learn driving style representations for drivers in $\mathbb{U}$ by exploiting necessary context information, so as to support applications like driver number estimation and driver identification.*

Different from previous works [2,10] that heavily rely on high-quality GPS trajectories, we should devise an approach that works well for practical trajectories and incorporates contextual information for much better driving style representation learning. To that end, we have to address the following challenges.

[1] We omit the upper-script if the context is clear.

(1) *Data sparsity.* This challenge is arised from two aspects. On one hand, in practice GPS data are usually collected at a low-sampling rate, *e.g.*, 0.1Hz. On the other hand, trajectories are of different lengths and may contain deficient driving behavior information, resulting in insufficient qualified trajectories. These factors will lead to low-quality data for training the deep learning models and thus impair their performances.

(2) *Balanced integration of features from GPS data and contextual information.* Although driving contexts would benefit driving style representation learning, how to efficiently encode these contextual information and further gracefully integrate features extracted from raw GPS data and driving contexts should be wisely designed. The driving contexts involve various information, and the resultant feature vectors may be of different dimensions.

4 The Design

4.1 Overview

Figure 1 illustrates the architecture of our approach, which consists of three major modules: *GPS data transformation*, *driving context representation*, and *learning model*. At high-level, Radar takes raw GPS trajectories and road map as the input, and exploits the modules of *GPS data transformation* and *driving context representation* to extract features from a GPS trajectory and the corresponding contexts. The integrated feature tensors are fed into *learning model* to compute driving style representations, which can support many trajectory mining applications, *e.g.*, driver number estimation and driver identification.

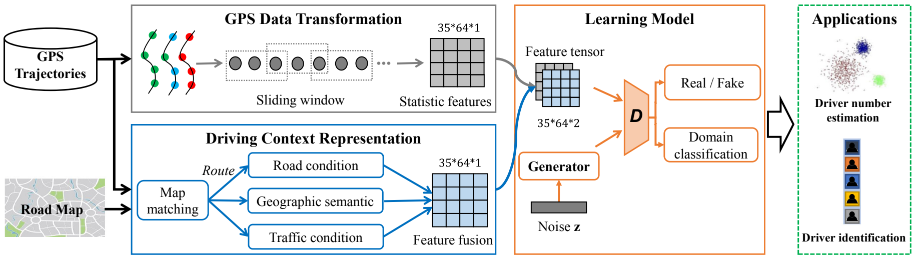

Fig. 1. The architecture of Radar.

Specifically, *GPS data transformation* utilizes a sliding window to calculate various statistics of the GPS data, which finally form the statistic feature matrix. For the *driving context representation* modules, it firstly applies map matching technique to transform each GPS trajectory to an actual travel route. With this route, Radar derives context information about road conditions, geographic semantic (*i.e.*, geographical distribution of the route over the whole city area), and traffic conditions. These context information are fused to form a contextual feature matrix. Lastly, both statistic feature matrix and contextual feature matrix are integrated as the input for the *learning model*. In particular, we adopt the emerging semi-supervised generative adversarial network architecture [21] for building our model to learn effective and accurate driving style representations.

4.2 GPS Data Transformation

Instead of inputting raw GPS data to deep learning models, we will transform each GPS trajectory into more stable statistic features. Similar as previous work [2], we divide a GPS trajectory into segments of a fixed length L_s, with a shift of $\frac{L_s}{2}$ to avoid much information loss between any two adjacent segments. We employ five basic features to capture the instantaneous vehicular movement features, namely *speed norm*, *difference of speed norm*, *acceleration norm*, *difference of acceleration norm*, and *angular speed*. To reduce the possible impact of outliers, we further divide a segment into frames of a fixed size L_f, with a shift $\frac{L_f}{2}$. For each frame, we calculate seven statistics for each basic feature, including mean, minimum, maximum, 25%, 50% and 75% quartiles, and standard deviation. For each trajectory $\mathcal{T}_j$ consisting of a sequence of time-ordered GPS records in the form of $g_i =< ts, lat, lng, v, dir >$, we can easily calculate speed statistics using travel speed v, acceleration statistics with location (lat, lng), and angular statistics with travel direction dir, respectively. As a result, we can derive a set of statistic feature matrices, each of which consists of $5 \times 7 = 35$ rows and $2 \times \lfloor \frac{L_s}{L_f} \rfloor$ columns. A statistic feature matrix encodes the driving behavior information of a trajectory segment, and serves as partial input to the learning model with its class label (*i.e.*, the driver identifier) as the original GPS trajectory $\mathcal{T}_j$.

In our implementation, we set $L_s = 195$ and $L_f = 6$ for the best performance. Therefore, we obtain a set of statistic feature matrices of size 35×64 for each trajectory. In particular, if a trajectory segment is shorter than L_s, we will pad zeros into the matrix, so as to unify the size of all statistic feature matrices. In principle, long trajectories contain more information about the driving behaviors, and thus are more preferable for the model training.

4.3 Driving Context Representation

Since driving activities will be implicitly governed by the surrounding driving environment, thus `Radar` also takes driving context information into consideration to let machines deeply "understand" drivers' behaviors especially under certain circumstances. In the design of `Radar`, we particularly consider the three contexts of road conditions, geographic semantic, and traffic conditions.

Figure 2 illustrates how `Radar` processes each raw GPS trajectory to generate the contextual features. For each GPS trajectory $\mathcal{T}_j$, we firstly recover the travel route $\mathcal{R}_j$ through map matching techniques [20]. Since GPS data transformation module outputs one statistic feature matrix for each trajectory segment, thus the driving context representation module operates on trajectory segment and

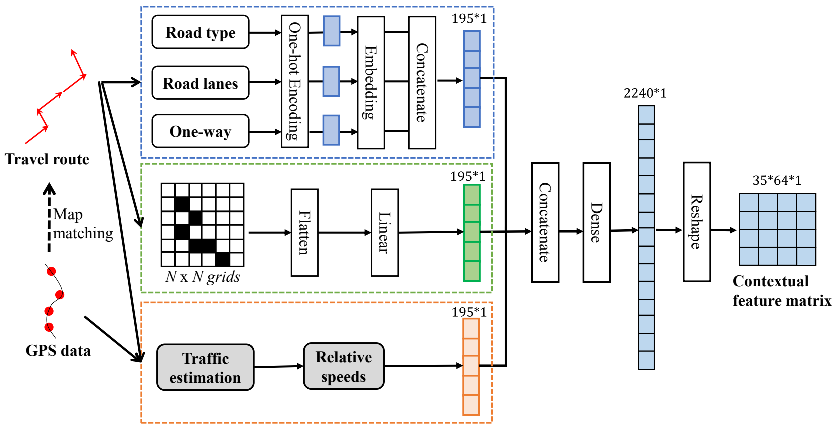

Fig. 2. The framework of driving context representation module.

its associated travel route segment as well, and accordingly produces one contextual feature matrix. Based on road network $\mathbb{G}$, the s-th trajectory segment $\mathcal{T}_{js}$ and its travel route segment $\mathcal{R}_{js}$, we derive each context representation as follows.

Road Condition. We utilize static road attributes of road type, number of lanes, and one-way indicator to characterize road conditions. Let n_t, n_ℓ, and n_o to represent the numbers of possible values in the three types of categorical attributes, we thus employ three attribute vectors of length n_t, n_ℓ, and n_o, respectively, to encode the attributes of each road segment, respectively. Specifically, one-hot encoding is adopted to generate the attribute vectors. Given a travel route $\mathcal{R}_{js}$, we derive road type vectors of road segments covered by $\mathcal{R}_{js}$, and sequentially connect them into one vector, which describes the road types a vehicle had traveled when generating trajectory $\mathcal{T}_{js}$. In addition, we adopt an embedding layer to reduce the dimensionality of the sparse attribute vector. Similarly, we apply the same operations to the attributes of road lanes and one-way, and derive their attribute vectors for route $\mathcal{R}_{js}$, respectively. Finally, we concatenate the three embedding vectors into one vector of size 195×1.

Geographic Semantic. The GPS data only reflect the instantaneous driving statuses, but not capture the high-level geographic semantic of a trajectory, *e.g.*, origin, destination, and traveled regions. Thus, `Radar` maps each GPS trajectory segment to the whole city area to derive its geographic semantic representation, which is formally defined as follows.

Definition 5 *(Geographic semantic representation). We partition the city area into $N \times N$ grids. For each trajectory $\mathcal{T}_j$, we compute a geographic semantic* *representation matrix* $\mathbf{M}_j$*, where we set* $\mathbf{M}_j[a,b] = 1$ *if the travel route* $\mathcal{R}_j$ *of* $\mathcal{T}_j$ *intersects with the grid* $[a,b]$*; otherwise* $\mathbf{M}_j[a,b] = 0$*.*

As shown in Fig. 2, we further flatten the matrix $\mathbf{M}_j$ as a vector, which is fed into a linear layer for reducing the dimensionality. In our design, we set the final geographic semantic vector of size 195×1. It is worthy noting that we generate such a vector for each trajectory segment $\mathcal{T}_{js}$ as well.

Traffic Condition. In addition to road conditions, another factor that has great impact on driving activities is real-time traffic conditions. Considering both vehicle's instantaneous movements and surrounding traffic conditions could better define a driver's driving behaviors. Therefore, we use the *relative speed*, which is calculated as the ratio between vehicle's travel speed and average travel speed of the vehicle's locating road segment, to represent traffic condition context.

To that end, we make use of all available GPS data to estimate the real-time traffic conditions. For each road segment, its traffic condition can be approximated as the average travel speed of all vehicles passing by within a time slot Δt. Therefore, we classify all GPS records to road segments according to their map matching results. For a given road segment, we calculate its average travel speed of a specific time slot using the GPS records falling into that time slot. Due to data sparsity, we may not derive a complete traffic conditions of the whole road network $\mathbb{G}$ over all time slots. For simplicity, we directly apply temporal-spatial interpolations to infer the traffic conditions of uncovered road segments by leveraging the inherent traffic correlations among roads. In fact, some advanced traffic estimation methods [15] can be adopted to compute the complete real-time traffic conditions. Once we obtain the traffic conditions of all road segments, we calculate the relative speeds for the road segments covered by travel route $\mathcal{R}_{js}$ of a trajectory segment $\mathcal{T}_{js}$. These relative speeds then form $\mathcal{T}_{js}$'s traffic condition representation.

As shown in Fig. 2, when the three representations of driving contexts are ready, `Radar` concatenates them into one vector, which is then fed into a dense layer to derive a vector of size 2240×1. To be compatible with the statistic feature matrix, we reshape it into a contextual feature matrix of size $35 \times 64 \times 1$.

4.4 Learning Model

To tackle the poor data quality issue, we employ generative adversarial networks (GAN) [7] to construct the learning model. Essentially, GAN operates by training two neural networks that play a *min-max* game: a *discriminator* is trained to discriminate real samples from fake ones, while a *generator* tries to generate fake training data to fool the discriminator. Therefore, GAN is able to generate samples very similar to real trajectories for training data augmentation and as a result improves the generalization ability of the derived model.

In particular, we adopt the emerging semi-supervised GAN (*SGAN*) architecture [21] to build our learning model, which mainly consists of a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$, as shown in Fig. 3. In SGAN, discriminator $\mathcal{D}$ can also

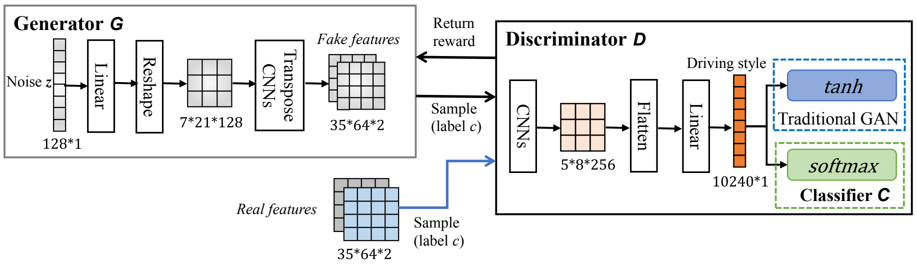

Fig. 3. The framework of learning model.

act as a classifier $\mathcal{C}$ to classify each input sample into one of the predefined $(k+1)$ classes, where k is the number of classes and the additional class label is added for a new "*fake*" class. The competition and interaction (via *reward*) between generator and discriminator will improve the quality of resultant driving style representations. Therefore, our model can not only classify drivers according to the learned driving styles, but also for a given class c generates corresponding fake driving style features, which are similar to training samples belonging to class c. To achieve this goal, the model training will involve both traditional unsupervised GAN task and supervised classification task simultaneously. Training in unsupervised mode allows our model to learn useful feature extraction capabilities from unlabeled samples, whereas training in supervised mode allows the model to use the extracted features and apply classifications.

Discriminator $\mathcal{D}$ (and classifier $\mathcal{C}$). As shown in Fig. 3, discriminator takes either real samples, generated from GPS data and context information, or fake samples, produced by generator $\mathcal{G}$, as the input, which are further processed by a neural network to derive driving style representations. Discriminator $\mathcal{D}$ is trained in both unsupervised mode and supervised mode.

- *Unsupervised mode.* In this mode, discriminator $\mathcal{D}$, with parameter θ_d, predicts whether a sample is true (sampled from real trajectory data) or fake (generated by the generator $\mathcal{G}$) by calculating the probability score $\mathcal{D}(x|\theta_d)$ that the sample x is true. We train our learning model like traditional GANs by maximizing the score for real samples and minimizing it for fake ones. We achieve this objective by minimizing $\mathcal{L}^{(\mathcal{D})}$, which is defined as follows.

$$\mathcal{L}^{(\mathcal{D})} = -[\mathbb{E}_{x\sim p_r(x)} \log \mathcal{D}(x|\theta_d) + \mathbb{E}_{x\sim \mathcal{G}} \log (1 - \mathcal{D}(x|\theta_d))], \quad (1)$$

  where $p_r(x)$ represents the distribution of real samples from trajectory data.
- *Supervised mode.* In this mode, discriminator $\mathcal{D}$ acts as classifier $\mathcal{C}$ to complete a multi-class classification problem. For each sample, classifier $\mathcal{C}$, with parameter θ_c, predicts if the sample belongs to one of the predefined $(k+1)$ classes. Because the label of driving style features generated by the generator $\mathcal{G}$ is known, classifier $\mathcal{C}$ can also utilize the labels of fake samples for training. Thus the generalization ability of the model could be improved. In addition,

classifier $\mathcal{C}$'s classification on both real and fake samples can be used as feedback (via reward) to improve generator $\mathcal{G}$, *i.e.*, higher classification accuracy will bring more returns. To train the classifier $\mathcal{C}$, we aim to minimize the classifier loss $\mathcal{L}^{(\mathcal{C})}$, *i.e.*, the cross entropy loss on true labeled samples that is computed using the overall classifier score.

$$\mathcal{L}^{(\mathcal{C})} = -\mathbb{E}_{p(x_c,c)}[\log \mathcal{C}(c|x_c, \theta_c)], \tag{2}$$

where x is a sample of class c, and $\mathcal{C}$ should correctly classify it as class c.

We implement above two modes in one unified framework, as shown in Fig. 3. Discriminator $\mathcal{D}$ and classifier $\mathcal{C}$ share the same feature extraction layers, but have different output layers. Specifically, we use a stack of convolution layers with LeakyReLu to process each input sample. After a series of convolutions, we get a feature tensor that is flatten and inputted to a dense layer to derive the driving style representation vector. For traditional GAN task, the vector is fed into `tanh` to discriminate real samples and fake ones. For classifier $\mathcal{C}$, the vector is fed into `softmax` to obtain classification probabilities of the $(k+1)$ classes.

Generator $\mathcal{G}$. Given the distribution $p_r(x)$ of real samples and k class labels from real training data, generator $\mathcal{G}$ aims to find the parameterized conditional distribution $\mathcal{G}(\mathbf{z}, c, \theta_g)$ that is close to the real distribution $p_r(x)$. The generated fake samples are conditioned on the network parameters θ_g, noise vector $\mathbf{z}$, and class label c, which are sampled from prior distribution p_z and p_c, respectively. Label c of a fake sample y can be known when the generator $\mathcal{G}$ generates y, so that the actual classification label of each generated sample is retained for training classifier $\mathcal{C}$. Following the feature matching technique proposed to addresses the instability of GANs [23], we train $\mathcal{G}$ by minimizing loss $\mathcal{L}^{(\mathcal{G})}$ expressed as:

$$\mathcal{L}^{(\mathcal{G})} = ||\mathbb{E}_{x \sim p_r(x)} \mathbf{f}(x, \theta_f) - \mathbb{E}_{\mathbf{z} \sim p_z} \mathbf{f}(\mathcal{G}(\mathbf{z}, c, \theta_g), \theta_f)||_2^2, \tag{3}$$

where $\mathbf{f}(\cdot)$ denotes activation on an intermediate layer (*e.g.*, the stack of convolution layers) of discriminator $\mathcal{D}$, θ_f is the parameter subset of θ_d corresponding to the intermediate layer of discriminator $\mathcal{D}$, and c is the class label of real sample x. The objective of generator training is thus to minimize the discrepancy between the real and generated data distributions in feature space.

As shown in Fig. 3, generator $\mathcal{G}$ is implemented with four deconvolution layers, which transform noise vector $\mathbf{z}$ into fake driving style features. In particular, each deconvolution layer is followed by a nonlinear activation based on batch normalization and rectified linear unit (ReLU). $\mathbf{z}$ is a 128 dimensional vector sampled from a uniform distribution p_z, and it is processed by dense and reshape layers before inputting to the deconvolution layers. Finally, generator $\mathcal{G}$ outputs a $35 \times 64 \times 2$ feature tensor as the same size of real feature tensors. The values of tensor items are shape squashed within $[-1, 1]$ through `tanh` function.

5 Performance Evaluation

5.1 Experimental Setup

Dataset. We use a large real-wold anonymized GPS trajectory dataset for the experiments. This dataset contains 1.3 billions GPS records that are collected from 10000 drivers in Shanghai city, China, during a six-month period in 2015. The GPS records are collected at a low-sampling rate as 0.1 Hz (*i.e.*, one sample per ten seconds). Each GPS record includes the driver identifier, a timestamp, location with longitude and latitude, travel speed, and travel direction. Furthermore, we download the road network of the city area covered by GPS records from OpenStreetMap (OSM)[2], and model the road network as a graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$, which has 159386 vertices and 30336 edges (*i.e.*, road segments) in total. In addition, we obtain the attributes of each road segment from OSM as well. After map matching, we have 430 trajectories for each driver on average.

Baseline Approaches. We compare `Radar` with following baseline approaches, which can also learn driving style representations from GPS trajectories.

- *ARNet* is one of the state-of-the-art approaches. *ARNet* proposes an autoencoder regularized neural network for driving style representation learning, merely from raw GPS data [2].
- *T2INET* is one of the state-of-the-art approaches as well. *T2INET* represents a GPS trajectory as the multi-channel images that capture both geographic and driving behavior features using a sequence of convolution layers [10].
- `Radar-C` serves as one variant of our approach `Radar` by disabling the driving context representation module. As a result, `Radar-C` only takes the statistic features extracted from GPS records as input for the learning model to compute driving style representations.

Implementation. We implement `Radar` and all baseline approaches in Python 3.7.3 with Keras[3] 2.3.1 and TensorFlow[4] 2.2.0 for building various machine/deep learning models. We set `Radar`'s parameters as follows. We set $L_s = 195$ and $L_f = 6$ for GPS data transformation. The city area is partitioned into 80×80 grids for geographic semantic representation. In graph $\mathbb{G}$, we have $n_t = 5$ road types, maximum number of lanes $n_\ell = 6$, and $n_o = 2$ for indicating one-way or not. We estimate traffic conditions with time slot $\Delta t = 30$ min. For the learning model, we use Adadelta as the optimizer, and set learning rates for generator $\mathcal{G}$ and discriminator $\mathcal{D}$ as 0.0001 and 0.0004, respectively. We set batch size as 128 and the epochs as 5000. Besides, we directly adopt the implementations of *ARNet* [2] and *T2INET* [10], which are provided by the authors respectively, and tune their parameters with our data to achieve their best performances.

We evaluate these approaches with two benchmark applications, *i.e.*, *driver number estimation* and *driver identification*, on a server, which is equipped with

[2] OpenStreetMap: https://www.openstreetmap.org/.
[3] Keras: https://keras.io/.
[4] TensorFlow: https://www.tensorflow.org/.

Intel Core i9-9900K CPU@3.60 GHz, NAVIDA GeForce RTX 2080 Ti GPU, and 32 GB memory. We repeat each experiment setting 10 times, and only the average results are reported in this section.

5.2 Driver Number Estimation

This application aims to estimate the number of drivers from a set of anonymous trajectories. To solve this problem, we train the driving style representation learning models with a set of labeled trajectories (*i.e.*, with known driver identifiers), and exploit trained models to represent each testing trajectory as a driving style representation vector. Then, we employ the affinity propagation [5] clustering algorithm to classify all representation vectors into clusters. In theory, a desired model should effectively learn drivers' driving styles, and would classify the testing trajectories generated by a specific driver into the same cluster. Finally, the number of clusters is regarded as the number of drivers.

Training and Testing. We randomly select 10 drivers from the driver set $\mathbb{U}$ and take their labeled trajectories as the training data. In addition, we randomly select κ drivers from the remaining drivers, who are absent in the training data, to form a group, denoted by *Group* κ. We vary κ from 1 to 10. For each group, we randomly sample 50 trajectories from the κ drivers, and use these trajectories as the testing data. We repeat 10 runs for each κ value and report average results.

Performance Metrics. We compare different approaches on the following two performance metrics: (1) the mean absolute error (*MAE*), which is the difference between the ground truth of driver number and the estimation; (2) the adjusted mutual information score (*AMI*) [24] that measures the clustering quality. The AMI values fall in the range of [0, 1], and larger AMI values are preferable.

Table 1. Performance comparisons on MAE for driver number estimation.

Group κ	*ARNet*	*T2INET*	Radar-C	Radar
1	**0.64** ± 0.60	0.70 ± 0.68	0.80 ± 0.64	0.78 ± 0.65
2	**0.82** ± 0.80	0.88 ± 0.74	0.92 ± 1.20	0.84 ± 0.97
3	1.08 ± 1.26	1.22 ± 1.48	1.02 ± 1.24	**0.98** ± 1.24
4	1.18 ± 1.40	1.04 ± 1.46	**0.92** ± 1.02	1.02 ± 1.07
5	0.98 ± 1.24	**0.88** ± 1.56	1.20 ± 0.90	1.02 ± 0.88
6	1.24 ± 0.98	1.24 ± 0.96	**1.04** ± 1.24	**1.04** ± 1.06
7	1.60 ± 1.24	1.42 ± 1.64	1.42 ± 1.44	**1.24** ± 1.12
8	1.48 ± 1.46	1.46 ± 1.45	1.46 ± 1.50	**1.38** ± 1.24
9	1.74 ± 1.48	1.82 ± 1.46	1.62 ± 1.42	**1.56** ± 1.48
10	2.32 ± 1.50	2.10 ± 1.68	1.94 ± 1.54	**1.82** ± 1.46
Average	1.308	1.276	1.234	**1.168**

Experimental Results. Table 1 shows the MAE results and deviations, where the best result of each group is marked in bold. When κ increases, the driver number estimation problem becomes harder, and thus the MAE is larger. Among all the experiments, we see our approach (`Radar` and `Radar-C`) wins 7 best results (*i.e.*, the smallest MAE) out of ten tests. For the three lost cases, our approach falls behind with marginal differences, *e.g.*, 0.14 at most. As shown by the average experiment results in the last row of Table 1, `Radar-C` achieves slightly better performance than the state-of-the-art approaches, *i.e.*, *ARNet* and *T2INET*. It implies that our learning model is more effective on capturing driving style features from raw GPS data. By incorporating the driving context information, `Radar` further improves `Radar-C` by reducing average MAE from 1.234 to 1.168. Overall, our approach `Radar` can improve *ARNet* and *T2INET* on the performance metric of MAE by 10.7% and 8.5%, respectively.

Table 2 presents the AMI results and deviations, where we also mark the best AMI of each group in bold. Similarly, we find that `Radar` outperforms other two baselines in most cases, with six wins out of ten tests. The results in Table 2 are in accordance with the results in Table 1. In general, a better clustering quality (*i.e.*, a larger AMI) potentially leads to a better estimation of driver number (*i.e.*, a smaller MAE). The average AMI values of the four approaches are 0.212, 0.234, 0.225, and 0.239, respectively. The results in both Table 1 and Table 2 demonstrate that `Radar` is capable of learning more effective and accurate driving style representations, which thus well support the application of driver number estimation, with smaller MAE and larger AMI.

Table 2. Performance comparisons on AMI for driver number estimation.

Group κ	*ARNet*	*T2INET*	`Radar-C`	`Radar`
1	**0.34** ± 0.06	0.32 ± 0.12	0.27 ± 0.06	0.25 ± 0.09
2	**0.37** ± 0.08	0.36 ± 0.07	0.25 ± 0.08	0.28 ± 0.03
3	0.21 ± 0.04	0.21 ± 0.08	0.26 ± 0.08	**0.27** ± 0.03
4	0.16 ± 0.08	0.24 ± 0.05	0.22 ± 0.05	**0.25** ± 0.04
5	0.19 ± 0.06	**0.23** ± 0.08	0.19 ± 0.08	0.18 ± 0.06
6	0.18 ± 0.05	0.22 ± 0.04	**0.26** ± 0.06	**0.26** ± 0.07
7	0.17 ± 0.07	0.17 ± 0.05	0.20 ± 0.08	**0.23** ± 0.02
8	**0.19** ± 0.06	**0.19** ± 0.06	0.14 ± 0.05	0.18 ± 0.04
9	0.15 ± 0.08	0.14 ± 0.08	0.20 ± 0.04	**0.22** ± 0.08
10	0.16 ± 0.07	0.26 ± 0.04	0.26 ± 0.05	**0.27** ± 0.04
Average	0.212	0.234	0.225	**0.239**

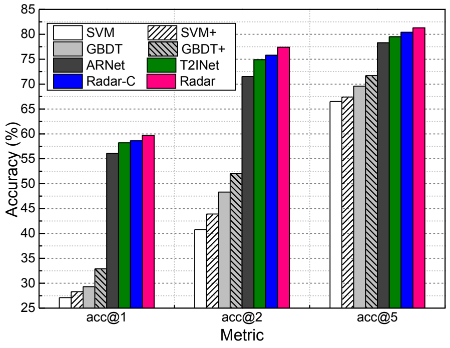

Fig. 4. Performance comparisons on Top-n accuracy with the long trajectories.

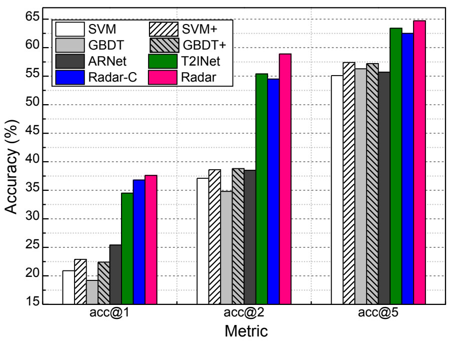

Fig. 5. Performance comparisons on Top-n accuracy with the short trajectories.

5.3 Driver Identification

The driver identification problem aims to identify the driver of a given unlabeled trajectory, which belongs to the supervised multi-class classification problem.

Training and Testing. In each experiment, we randomly select 10 drivers and use their GPS trajectories for model training and testing. Specifically, 70% of the trajectory data are used for training, 10% for validation, and the remaining 20% for testing. The models of all approaches are trained with labeled trajectories, and for a testing trajectory the models should predict its driver identifier.

Performance Metric. We employ the top-n accuracy (denoted by $acc@n$) to evaluate the prediction performances of all approaches. In particular, $acc@n$ is calculated as the percentage of testing trajectories for which the ground truth drivers are in the top n predictions. For a testing trajectory, we rank the predicted driver identifiers in the descending order of probability values.

Experimental Results. In addition to the aforementioned three baselines, we further include two typical supervised learning models, *i.e.*, support vector machines (SVM) [26] and gradient boosting decision trees (GBDT) [18], for performance comparisons. More specifically, SVM and GBDT take the statistic features produced by `Radar` as input for the predictions, while SVM+ and GBDT+ make use of both statistic and contextual features generated by `Radar` for the predictions. Furthermore, we partition drivers' trajectories into two sets: long trajectories (with duration more than 1950 s) and short trajectories (with duration less than 1950 s). We conduct experiments on each set of trajectories separately, and present the results in Fig. 4 and Fig. 5, respectively.

As shown in Fig. 4, when n increases, the top-n accuracy of each approach becomes higher. Our approach achieves the highest $acc@5$ accuracy as 81.3%. These deep learning models, *i.e.*, *ARNet*, *T2INET*, `Radar-C` and `Radar`, always have better predictions than traditional supervised learning models, *i.e.*, SVM and GBDT and their variants, with the largest performance gap as 36.6% on *acc*@2. It implies that deep learning models are indeed powerful at representation learning, and thus can support various applications better. On the other hand, by comparing the performances of traditional models, we find that SVM+/GBDT+ outperform SVM/GBDT, *e.g.*, with *acc*@1 accuracy improvement by 1.2% and 3.6%, respectively. Hence, it is necessary to include contextual features for better modeling. Compared to state-of-the-art *ARNet* and *T2INET*, our approach `Radar` has more accurate predictions, *e.g.*, on average improving them by 2.6%, 4.2%, and 2.4% for *acc*@1, *acc*@2, and *acc*@5, respectively.

The prediction results on short trajectory set are plotted in Fig. 5. Since short trajectories contain less information, and thus the prediction performances of all approaches have been seriously deteriorated. However, we find that the performance gap between *ARNet*/*T2INET* and our approach becomes even larger, *i.e.*, on average `Radar` improves the two advanced approaches by 7.1%, 12.0%, and 5.2% for *acc*@1, *acc*@2, and *acc*@5, respectively. These comparisons reflect that `Radar` is able to extract more useful and accurate features from low-quality trajectory data, and thus can still achieve reasonably high prediction accuracy.

6 Conclusion

In this paper, we present an adversarial driving style representation learning approach – `Radar`. Different from previous works, `Radar` not only extracts statistic features from raw GPS data, but also builds contextual features by jointly considering road conditions, geographic semantics, and traffic conditions. We further exploit an advanced semi-supervised GAN architecture to construct the learning model to compute more effective and accurate driving style representations. Experiment results from a large GPS trajectory dataset demonstrate that `Radar` outperforms state-of-the-art approaches on two benchmark applications.

Acknowledgments. This work was supported in part by the National Science Foundation of China (NSFC) under Grant Nos. 61802261, 61806132, 61872248, and U2001207, the grant of Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515011502), Tencent Rhino-Bird Young Faculty Open Fund, Guangdong NSF No. 2017A030312008, Shenzhen Science and Technology Foundation (No. ZDSYS20190902092853047), the Project of DEGP (No. 2019KCXTD005),the Guangdong "Pearl River Talent Recruitment Program" under Grant No. 2019ZT08X603, Guangdong Science and Technology Foundation (Nos. 2019B111103001, 2019B020209001).

References

1. Castignani, G., Derrmann, T., Frank, R., Engel, T.: Driver behavior profiling using smartphones: a low-cost platform for driver monitoring. IEEE Intell. Transp. Syst. Mag. **7**(1), 91–102 (2015)
2. Dong, W., Yuan, T., Yang, K., Li, C., Zhang, S.: Autoencoder regularized network for driving style representation learning. In: IJCAI (2017)

3. Ezzini, S., Berrada, I., Ghogho, M.: Who is behind the wheel? Driver identification and fingerprinting. J. Big Data **5**(1), 9 (2018)
4. Feng, J., et al.: User identity linkage via co-attentional neural network from heterogeneous mobility data. IEEE Trans. Knowl. Data Eng. **1**, 1–15 (2020)
5. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
6. Fugiglando, U., et al.: Driving behavior analysis through CAN bus data in an uncontrolled environment. IEEE Trans. Intell. Transp. Syst. **20**(2), 737–748 (2018)
7. Goodfellow, I., et al.: Generative adversarial nets. In: NeurIPS (2014)
8. Guangyu Li, M., et al.: DBUS: human driving behavior understanding system. In: IEEE ICCV Workshops (2019)
9. He, B., Zhang, D., Liu, S., Liu, H., Han, D., Ni, L.M.: Profiling driver behavior for personalized insurance pricing and maximal profit. In: IEEE Big Data (2018)
10. Kieu, T., Yang, B., Guo, C., Jensen, C.S.: Distinguishing trajectories from different drivers using incompletely labeled trajectories. In: ACM CIKM (2018)
11. Kim, Y., Han, J., Yuan, C.: TOPTRAC: topical trajectory pattern mining. In: ACM SIGKDD (2015)
12. Kuderer, M., Gulati, S., Burgard, W.: Learning driving styles for autonomous vehicles from demonstration. In: IEEE ICRA (2015)
13. Lin, N., Zong, C., Tomizuka, M., Song, P., Zhang, Z., Li, G.: An overview on study of identification of driver behavior characteristics for automotive control. Math. Probl. Eng. **2014**, 1–15 (2014)
14. Liu, Z., Gong, Z., Li, J., Wu, K.: Mobility-aware dynamic taxi ridesharing. In: IEEE ICDE (2020)
15. Liu, Z., Li, Z., Li, M., Xing, W., Lu, D.: Mining road network correlation for traffic estimation via compressive sensing. IEEE Trans. Intell. Transp. Syst. **17**(7), 1880–1893 (2016)
16. Liu, Z., Li, Z., Wu, K., Li, M.: Urban traffic prediction from mobility data using deep learning. IEEE Network **32**(4), 40–46 (2018)
17. Liu, Z., Zhou, P., Li, Z., Li, M.: Think like a graph: real-time traffic estimation at city-scale. IEEE Trans. Mob. Comput. **18**(10), 2446–2459 (2018)
18. Mason, L., Baxter, J., Bartlett, P.L., Frean, M.R.: Boosting algorithms as gradient descent. In: NeurIPS (2000)
19. Miao, C., Wang, J., Yu, H., Zhang, W., Qi, Y.: Trajectory-user linking with attentive recurrent network. In: ACM AAMAS (2020)
20. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: ACM SIGSPATIAL (2009)
21. Odena, A.: Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 (2016)
22. Ren, H., Pan, M., Li, Y., Zhou, X., Luo, J.: ST-SiameseNet: spatio-temporal siamese networks for human mobility signature identification. In: ACM SIGKDD (2020)
23. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: NeurIPS (2016)
24. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010)
25. Wang, P., Fu, Y., Zhang, J., Wang, P., Zheng, Y., Aggarwal, C.: You are how you drive: peer and temporal-aware representation learning for driving behavior analysis. In: ACM SIGKDD (2018)

26. Wu, T.F., Lin, C.J., Weng, R.C.: Probability estimates for multi-class classification by pairwise coupling. J. Mach. Learn. Res. **5**(Aug), 975–1005 (2004)
27. Yang, S., Wang, C., Zhu, H., Jiang, C.: APP: augmented proactive perception for driving hazards with sparse GPS trace. In: ACM MobiHoc (2019)
28. Zhao, K., et al.: Discovering subsequence patterns for next POI recommendation. In: AAAI (2020)
29. Zheng, Y.: Trajectory data mining: an overview. ACM Trans. Intell. Syst. Technol. **6**(3), 1–41 (2015)
30. Zhou, F., Gao, Q., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F.: Trajectory-user linking via variational AutoEncoder. In: IJCAI (2018)