Reinforced Feature Extraction and Multi-Resolution Learning for Driver Mobility Fingerprint Identification

Mahan Tabatabaie
University of Connecticut
mahan.tabatabaie@uconn.edu

Suining He
University of Connecticut
suining.he@uconn.edu

Xi Yang
University of Connecticut
xi.yang@uconn.edu

ABSTRACT

Taking into account the availability of the historical GPS trajectories of drivers, given a new GPS trajectory, *Driver mobility fingerprint* (DMF) identification aims at *(i)* determining whether a generated trajectory belongs to a potential driver, and *(ii)* detecting if a trajectory is likely anomalous based on a driver's historical data. Prior studies often consider hand-crafted feature engineering techniques to extract DMFs while contextual factors like weather and points-of-interest (POIs) are hardly accounted for, which might not achieve satisfactory identification results. To address above, we propose RM-Drive, a novel framework based on reinforced feature extraction and multi-resolution learning. Specifically, we first employ spatio-temporal inverse reinforcement learning (ST-IRL) to extract DMFs from historical trajectories. Then, we generate trajectory embeddings by fusing the extracted DMFs and the contextual factors using the multi-resolution trajectory embedding network (MTE-Net). Our proposed MTE-Net consists of multi-resolution convolutional neural network (MR-CNN), which enables the model to learn the multi-resolution features of the DMFs. Finally, we leverage the trajectory embeddings for the driver classification and anomaly detection. We have conducted extensive evaluation studies upon RM-Drive with two real-world datasets, and our results demonstrate the performance improvements from the state-of-the-art of driver classification and anomaly detection respectively by 21% and 11% on average based on several evaluation metrics, including accuracy, precision, and recall, etc.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**.

KEYWORDS

Driver Mobility Fingerprint, Driver Classification, Anomaly Detection, Multi-Resolution Feature Learning, Inverse Reinforcement Learning.

ACM Reference Format:
Mahan Tabatabaie, Suining He, and Xi Yang. 2021. Reinforced Feature Extraction and Multi-Resolution Learning for Driver Mobility Fingerprint Identification . In *29th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '21), November 2–5, 2021, Beijing, China.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3474717.3483911

1 INTRODUCTION

The Driver mobility fingerprint (DMF) identification problem refers to the task of learning a distinctive set of DMFs (*e.g.,* driving time, and frequencies of trajectories) from drivers' historical data (*i.e.,* GPS trajectories). As illustrated in Fig. 1, DMF can be used as *fingerprints* to identify the driver that generates the new mobility measurement (*i.e.,* driver classification), or to decide whether the measurement belongs to a specific driver (*i.e.,* anomaly detection).

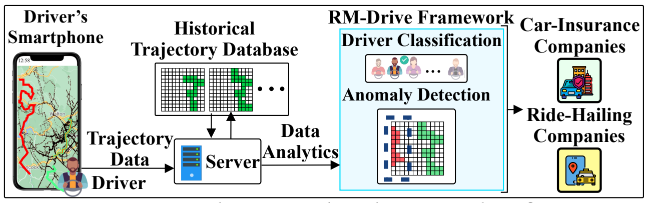


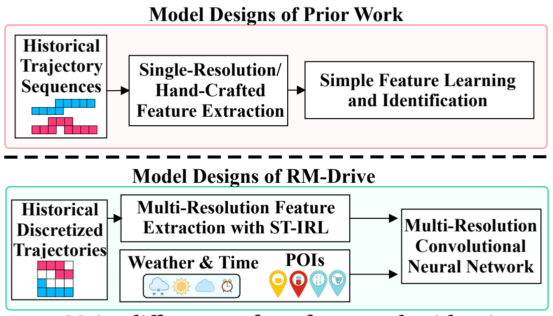
Figure 1: Potential outcomes based on DMF identification.

Realizing DMF identification can enable various practical applications, including prevention of unauthorized driving (say, unauthorized Uber ride sharing[1]) and enhancement of driving safety[2], which are particularly important for ride-hailing service providers (like Uber and Lyft) and the insurance companies (like State Farm and Progressive Car Insurance). Despite the prior approaches and recent advances [4, 8, 25], DMF identification remains challenging due to the following technical barriers:

• **Complex mobility features among drivers' trajectories and travel activities**: Due to complex spatio-temporal settings and drivers' decision-making behavior patterns, different drivers may potentially have very similar active regions. As a consequence, differentiating such drivers based on just simple features extracted from their historical GPS trajectories may not necessarily produce satisfactory identification results. Prior work investigated hand-crafted driver profile features [25], which, however, may not provide satisfactory identification accuracy under complex driver decision-making processes and dynamic environments.

• **Lack of spatio-temporal feature fusion with contextual factors**: Via our data analytics, we have observed that drivers' decision-making behavior patterns may take various latent effects from complex contextual factors such as points-of-interest (POIs) and weather conditions. Moreover, such complex spatio-temporal dependencies between these factors and the drivers' behaviors could not be effectively identified via simple hand-crafted feature-engineering

[1]https://www.cnet.com/news/uber-drivers-using-fake-identities-isnt-just-a-london-problem/
[2]https://help.uber.com/driving-and-delivering/article/can-i-share-my-account-with-friends—?nodeId=1d93388d-cf19-408f-9c41-743dbdd34d44


Model Designs of Prior Work
Historical Trajectory Sequences
Single-Resolution/ Hand-Crafted Feature Extraction
Simple Feature Learning and Identification
Model Designs of RM-Drive
Historical Discretized Trajectories
Multi-Resolution Feature Extraction with ST-IRL
Weather & Time
POIs
Multi-Resolution Convolutional Neural Network
Figure 2: Major differences of our framework with prior work.

techniques, making it highly challenging for the model designs to capture such complicated correlations.

• **Absence of multi-resolution driver mobility modeling**: Trajectory pre-processing often consists of discretization of the city map into grid-cells, converting a trajectory into a series of matched grid-cells for ease of computation. However, the information that is maintained during such spatial discretization depends on the resolution setting, *i.e.,* the width and height of the chosen grid-map. A single-resolution may discard the information of the DMFs that are essential for identification. Multi-resolution pre-processing and subsequent learning can benefit the DMF identification, which, nevertheless, remains challenging and largely unexplored.

In order to overcome the aforementioned challenges, we propose RM-Drive, a novel **R**einforced feature extraction and **M**ulti-resolution learning for **Drive**r mobility fingerprint identification. Specifically, RM-Drive integrates *(i) spatio-temporal inverse reinforcement learning* (ST-IRL) to extract multi-resolution spatio-temporal DMFs (*e.g.,* favorite geographical areas and driving-time preferences); *(ii)* a *multi-resolution trajectory embedding network* (MTE-Net) augmented with *multi-resolution convolutional neural network* (MR-CNN) to simultaneously capture various types of hidden features in multiple resolutions; *(iii)* a neural network based on MTE-Net for driver classification that receives the input DMF tensors of a trajectory and determines which the driver it belongs to, and *(iv)* a siamese network design using two identical MTE-Net modules for anomaly detection that receives sets of input DMF tensors from two trajectories and determines whether they are generated by a specific driver.

To summarize, we make the following major contributions in RM-Drive (Fig. 2):

(a) **Inverse Reinforced Spatio-Temporal Driver Mobility Feature Extraction** (Sec. 3): We design a novel spatio-temporal inverse reinforcement learning module (ST-IRL) that effectively extracts the hidden DMFs (*i.e.,* reward and policy) from the historical trajectories that represent the decision-making behavior patterns of the drivers. This way, RM-Drive adapts to the complex spatial and temporal contexts for DMF identification without hand-crafted feature engineering.

(b) **Multi-Resolution Deep Driver Mobility Fingerprint Learning** (Sec. 4): We have designed a novel fusion technique to integrate the complex spatio-temporal features and contextual factors (*i.e.,* POIs and weather conditions) with multiple resolutions. Such multi-resolution learning adaptively captures the most important characteristics of drivers' decision-making behavior patterns to enhance RM-Drive's overall accuracy and robustness, and subsequently reduces the efforts in the resolution fine-tuning.

(c) **Extensive Real-World Data Analytics and Experimental Studies** (Sec. 5): We have extensively analyzed and studied two different large-scale datasets collected in Porto, Portugal, and Rome, Italy, with more than 40K trajectories of 200 selected drivers combined to show the applicability of our method. We design and perform several tests on different subsets of the mentioned datasets to evaluate our framework compared to other baselines. Our experimental studies show that according to several evaluation metrics, including accuracy, top-5 accuracy, precision, and recall, RM-Drive has achieved respectively 21% and 11% higher scores on average compared to the state-of-the-art models for driver classification and anomaly detection.

Our proposed framework can potentially benefit existing or emerging urban computing applications/services, including car rental [26], ride hailing/sharing [1, 14–16], and related insurance companies to help identifying unauthorized drivers and mitigate potential violation of their management/insurance policies (*e.g.,* uninsured driving, shared driver accounts, and unnecessary detours[3,4]). Furthermore, these companies can use the information provided by our system as a reference to adjust insurance rates, avoid financial loss, and promote passenger safety while preserving the driver's privacy, *e.g.,* by reducing the need for frequent face recognition checks[5].

2 PROBLEM AND SYSTEM OVERVIEW

2.1 System Overview

The overall workflow of RM-Drive is illustrated in Fig. 3, which basically consists of the following three phases:

(1) **Data Pre-processing**: We use two driver trajectory datasets (see Appendix A and E), one from Porto, Portugal and the other from Rome, Italy, for our RM-Drive's development and experimental studies. In the pre-processing phase (Sec. 2.2), after removing the outliers (noisy GPS coordinate points), we first perform temporal discretization to divide each day into T time intervals with equal lengths. Next, based on the defined time intervals, given $\mathcal{H}_l$ and $\mathcal{W}_l$ as the height and width of the l-th spatial resolution, we perform spatial discretization to generate a grid-map for each time interval.

Given the raw GPS trajectories of a driver, we divide the trajectories into T groups based on their timestamps and convert them into series of grid-cells, *i.e.,* discretized trajectories. The information maintained from the trajectories depends on $\mathcal{H}_l$ and $\mathcal{W}_l$. Therefore, we define multiple spatial resolutions to realize adaptive feature extraction and mitigate spatial information loss. More specifically, the extracted features from each resolution contain distinctive and useful mobility information, and the combination of extracted features from all the resolutions would further assist the model to learn more distinctive DMFs.

[3]https://www.nbcboston.com/news/local/19-charged-in-ride-hailing-fake-driver-account-scheme/2375154/
[4]https://chicago.suntimes.com/news/2021/4/9/22375802/chicago-carjackings-ride-hailing-services-uber-rider-verification-feature-anonymous-payments
[5]https://www.earlynewspaper.com/uber-under-pressure-over-facial-recognition-checks-for-drivers/

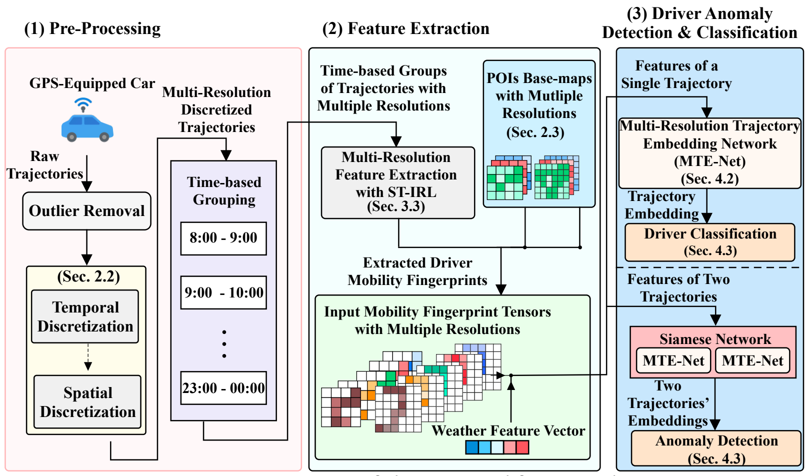

(1) Pre-Processing
GPS-Equipped Car
Multi-Resolution Discretized Trajectories
Raw Trajectories
Outlier Removal
(Sec. 2.2)
Temporal Discretization
Spatial Discretization
Time-based Grouping
8:00 - 9:00
9:00 - 10:00
23:00 - 00:00
(2) Feature Extraction
Time-based Groups of Trajectories with Multiple Resolutions
POIs Base-maps with Mutliple Resolutions (Sec. 2.3)
Multi-Resolution Feature Extraction with ST-IRL (Sec. 3.3)
Extracted Driver Mobility Fingerprints
Input Mobility Fingerprint Tensors with Multiple Resolutions
Weather Feature Vector
(3) Driver Anomaly Detection & Classification
Features of a Single Trajectory
Multi-Resolution Trajectory Embedding Network (MTE-Net) (Sec. 4.2)
Trajectory Embedding
Driver Classification (Sec. 4.3)
Features of Two Trajectories
Siamese Network
MTE-Net
MTE-Net
Two Trajectories' Embeddings
Anomaly Detection (Sec. 4.3)

Figure 3: Overview of the proposed framework.

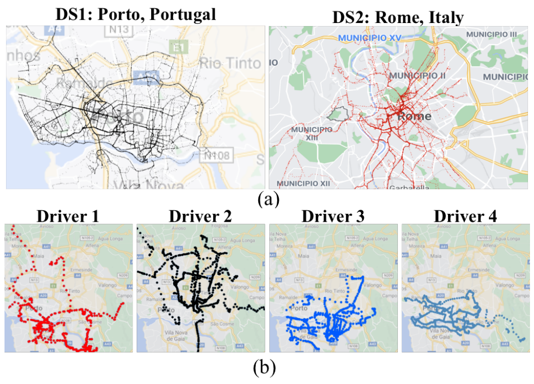


Figure 4: (a) Total trajectories of 100 drivers from each dataset over a month period. (b) Trajectories of four drivers from $DS1$ at 10:00am – 11:00am over a month.

(2) **Feature Extraction**: Then, we convert each group of trajectories into *state-action pairs* and formulate the ST-IRL problem (Sec. 3.2), and extract spatio-temporal DMFs based on the time interval and resolution of the given trajectories. Our ST-IRL module recovers the reward and policy *base-maps* per each group of trajectories that characterize the driver's decision-making behavior patterns in a specific time interval and a given resolution. Furthermore, we define several categories of POIs (see Appendix A) and calculate base-maps for each category with multiple resolutions, which basically records the density of each category of POIs on the map. We then use the set of the base-maps to create several heat-maps for each trajectory (*i.e.,* input mobility fingerprint tensor) with different resolutions. These heat-maps are fused with the weather feature vectors (see Appendix A) via the multi-resolution trajectory embedding network (MTE-Net) (Sec. 4.2).

(3) **Driver Classification & Anomaly Detection**: This stage consists of two tasks: *(i)* for *driver classification*, we feed the input mobility fingerprint tensors of the trajectories with the drivers' labels as the ground-truth values to the MTE-Net, which leverages multi-resolution convolutional neural network (MR-CNN) to identify the drivers; *(ii)* for *anomaly detection*, we leverage the siamese network [2] (Sec. 4.2). We use two MTE-Net modules with identical network structures and construct a binary classifier for driver anomaly detection.

2.2 Important Concepts

We then present the important concepts for our formulation.

• **Raw Driver Trajectory Data**: Taking into account that each vehicle is equipped with a GPS sensor, a driver d generates a series of GPS coordinate points (with longitudes and latitudes), each of which is given by {longitude, latitude, timestamp}. Thus, we define the set of trajectories generated by a driver d as

$$\mathcal{T}^d = \{\tau_1^d, \tau_2^d, ..., \tau_{N_d}^d\}, \tag{1}$$

where τ_j^d is the j-th raw trajectory of driver d, and N_d is the total number of trajectories of this driver.

Fig. 4(a) shows the aggregate trajectories of 100 drivers from each of the datasets. As mentioned earlier, we discretize a day (24 hours) into T time intervals, which we use to divide and aggregate the trajectories of each driver into T groups based on their timestamps. More specifically, each group contains the trajectories of a driver that occur at a specific time interval during a period of time (*e.g.,* all the trajectories that have happened at 8:00am to 9:00am over a month period). The reason behind this aggregation is to mitigate the negative effects of similar spatial activities of drivers, especially, in smaller cities, where drivers take more similar routes compared to large cities. Fig. 4(b) shows the trajectories of four random drivers that belong to a single time interval during a one-month period, which indicates that drivers have distinctive driving patterns during different times of the day and validate the effectiveness of the proposed grouping strategy. Details of the sample trajectories can be referred to Fig. 21 in Appendix D.

• **Temporal & Spatial Discretization**: We divide each day into T equal intervals $\{q_1, q_2, ..., q_T\}$. Thus, each point in a trajectory falls into one of the above-mentioned intervals based on its timestamp. Similarly, we spatially convert the city map into an $\mathcal{H}_l \times \mathcal{W}_l$ grid-map where $\mathcal{H}_l$ and $\mathcal{W}_l$ respectively represent the number of latitude-wise and longitude-wise discretizations. Each grid-cell is rectangular and of equal size in our settings. We let l be the l-th resolution (totally three resolutions in our studies). For each resolution, we build T grid-maps $\{G_l^1, G_l^2, G_l^3, \cdots, G_l^T\}$ for the city map (*i.e.,* one grid-map per each time interval), and each G_l^t represents an $\mathcal{H}_l \times \mathcal{W}_l$ matrix:

$$G_l^t = \begin{bmatrix} g_l^t[1,1] & g_l^t[1,2] & \cdots & g_l^t[1,\mathcal{W}_l] \\ \vdots & \vdots & \vdots & \vdots \\ g_l^t[\mathcal{H}_l,1] & g_l^t[\mathcal{H}_l,2] & \cdots & g_l^t[\mathcal{H}_l,\mathcal{W}_l] \end{bmatrix}, \tag{2}$$

where $g_l^t[h,w]$ is the grid-cell at the h-th row and the w-th column of grid-map G_l^t with the l-th resolution, and $t \in \{1, ..., T\}$.

• **Discretized Trajectory**: Based on the time and space discretization, we can discretize the j-th trajectory of driver d, τ_j^d, with multiple different resolutions. To discretize the raw trajectory τ_j^d with the l-th resolution, we map each point p in the trajectory, with {long, lat, ts} as its longitude, latitude, and timestamp respectively, towards a grid-cell $g_l^t[h,w]$. In other words, the coordinates $<$ long, lat $>$ fall in $g_l^t[h,w]$, and the timestamp ts $\in q_t$. We denote the series of such grid-cells of the j-th discretized trajectory of the driver d at the t-th time interval given the l-th resolution by $(\tau_{l,j}^{d,t})'$.

Thus, we define $(\mathcal{T}_l^{d,t})' = \{(\tau_{l,1}^{d,t})', \cdots, (\tau_{l,N_d}^{d,t})'\}$ as the set of all the discretized trajectories generated by driver d at time interval q_t, where each trajectory $(\tau_{l,j}^{d,t})'$ is a series of grid-cells, thus forming a *discretized trajectory*.

2.3 Problem Formulation & Definition

Based on above concepts, we further define different mobility features that are extracted from a driver's trajectory, and then present the problem formulation of RM-Drive.

• **Base-Map & Main Features**: We define a *base-map* as an $\mathcal{H}_l \times \mathcal{W}_l$ matrix with normalized values in $[0, 1]$ to represent different features of spatial inputs. We can use such base-maps to create trajectory heat-maps by calculating the intersection of the discretized trajectories and base-maps as illustrated in Fig. 5. For each disretized trajectory $(\tau_{l,j}^{d,t})'$, we create reward/policy heat-maps denoted by $\chi_{l,j}^{d,t}$ and $\psi_{l,j}^{d,t}$, which are generated from *reward* $\mathcal{R}_l^{d,t}$ and *policy* $\pi_l^{d,t}$ base-maps.

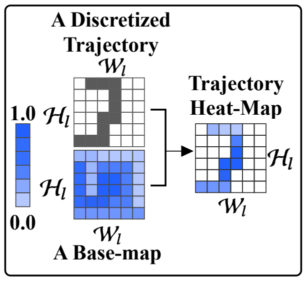


Figure 5: An example of creating a trajectory heat-map from a base-map.

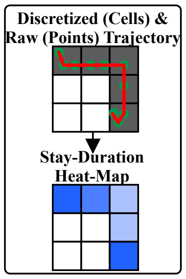


Figure 6: An example of creating the stay-duration heat-map.

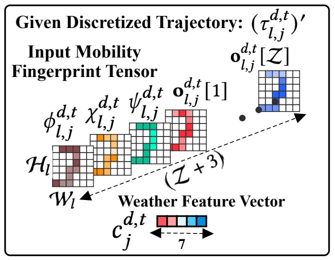

Given Discretized Trajectory: $(\tau_{l,j}^{d,t})'$
Input Mobility Fingerprint Tensor
$\mathbf{o}_{l,j}^{d,t}[\mathcal{Z}]$
$\phi_{l,j}^{d,t}$ $\chi_{l,j}^{d,t}$ $\psi_{l,j}^{d,t}$ $\mathbf{o}_{l,j}^{d,t}[1]$
$\mathcal{H}_l$
$(\mathcal{Z}+3)$
$\mathcal{W}_l$
Weather Feature Vector
$c_j^{d,t}$
7

Figure 7: Input mobility fingerprint tensor and weather feature vector of a discretized trajectory.

In addition, as illustrated in Fig. 6, from the discretized trajectory, $(\tau_{l,j}^{d,t})'$, we create the *stay-duration* heat-map denoted as $\phi_{l,j}^{d,t}$ where the value of each grid-cell represents the number of trajectory points in this grid. $\phi_{l,j}^{d,t}$, $\chi_{l,j}^{d,t}$ and $\psi_{l,j}^{d,t}$ together form the *main features*.

For each heat-map, the higher values (indicated by darker colors) of a grid-cell represent a higher reward/value learned by ST-IRL for reward/policy heat-maps, or more re-visitations or longer stay-duration by the driver for the stay-duration heat-map.

• **Contextual Factors**: We create $\mathcal{Z}$ categories of POIs, and generate a total of $\mathcal{Z}$ base-maps $\{\mathcal{P}_1, \cdots, \mathcal{P}_\mathcal{Z}\}$ based on their distributions upon the map. Therefore, we use each of these base-maps to create $\mathcal{Z}$ heat-maps $\mathbf{o}_{l,j}^{d,t}$ for a discretized trajectory $(\tau_{l,j}^{d,t})'$ in the same way as above. Besides, according to the time interval of each trajectory, we consider a set of 7 weather features, *e.g.*, temperature, rain, and snow, as another contextual factor. Thus, we define a vector $c_j^{d,t} \in \mathbb{R}^{1\times 7}$ for a discretized trajectory $\left(\tau_{l,j}^{d,t}\right)'$ as the additional weather feature vector.

• **Input Mobility Fingerprint Tensor**: Integrating all the main and contextual factors introduced above, we convert $(\tau_{l,j}^{d,t})'$, the j-th discretized trajectory of driver d from the t-th time interval, to an input mobility fingerprint tensor (Fig. 7) with the l-th resolution as follows:

$$f_{l,j}^{d,t} = \left(\phi_{l,j}^{d,t}, \chi_{l,j}^{d,t}, \psi_{l,j}^{d,t}, \mathbf{o}_{l,j}^{d,t}\right), \tag{3}$$

where $\phi_{l,j}^{d,t} \in \mathbb{R}^{\mathcal{H}_l\times\mathcal{W}_l}$ is the stay-duration heat-map, and $\chi_{l,j}^{d,t} \in \mathbb{R}^{\mathcal{H}_l\times\mathcal{W}_l}$ and $\psi_{l,j}^{d,t} \in \mathbb{R}^{\mathcal{H}_l\times\mathcal{W}_l}$ are heat-maps generated from reward and policy base-maps respectively (based on the time intervals of the trajectories) provided by the ST-IRL module.

Furthermore, we have the set of POI heat-maps created from $\mathcal{Z}$ POI categories,

$$\mathbf{o}_{l,j}^{d,t} = \{\mathbf{o}_{l,j}^{d,t}[1], \cdots, \mathbf{o}_{l,j}^{d,t}[\mathcal{Z}]\}, \tag{4}$$

where each $\mathbf{o}_{l,j}^{d,t}[z] \in \mathbb{R}^{\mathcal{H}_l\times\mathcal{W}_l}$, $z \in \{1, ..., \mathcal{Z}\}$ is a heat-map generated from one base-map of a POI category z. For each driver d at the t-th time interval with the l-th resolution, after obtaining the discretized trajectories $(\mathcal{T}_l^{d,t})' = \{(\tau_{l,1}^{d,t})', \cdots, (\tau_{l,N_d}^{d,t})'\}$, we correspondingly denote the set of all the input mobility fingerprint tensors and weather feature vectors as $\mathcal{F}_l^{d,t} = \{f_{l,1}^{d,t}, \cdots, f_{l,N_d}^{d,t}\}$, and $C^{d,t} = \{c_1^{d,t}, \cdots, c_{N_d}^{d,t}\}$, respectively.

• **Problem Statement of** RM-Drive: Given historical trajectories of $\mathcal{D}$ drivers $\{\mathcal{T}^1, \cdots, \mathcal{T}^\mathcal{D}\}$, we aim at using the multi-resolution input mobility fingerprint tensors, *i.e.*,

$$\{\{\mathcal{F}_l^{1,1}, \cdots, \mathcal{F}_l^{1,T}\}, \cdots, \{\mathcal{F}_l^{\mathcal{D},1}, \cdots, \mathcal{F}_l^{\mathcal{D},T}\}\}, \tag{5}$$

extracted from discretized trajectories, *i.e.*,

$$\{\{(\mathcal{T}_l^{1,1})', \cdots, (\mathcal{T}_l^{1,T})'\}, \cdots, \{(\mathcal{T}_l^{\mathcal{D},1})', \cdots, (\mathcal{T}_l^{\mathcal{D},T})'\}\}, \tag{6}$$

where $l \in \{1, 2, 3\}$, along with the weather feature vectors,

$$\{\{C^{1,1}, \cdots, C^{1,T}\}, \cdots, \{C^{\mathcal{D},1}, \cdots, C^{\mathcal{D},T}\}\}, \tag{7}$$

as inputs for RM-Drive, the DMF identification framework, to train the model for driver classification and anomaly detection. In particular, given a new driver trajectory, RM-Drive identifies if a target driver has generated the input trajectory (anomaly detection), as well as classifies which driver this trajectory likely belongs to (driver classification).

3 TRAJECTORY FEATURE EXTRACTION

3.1 Overview of Spatio-Temporal IRL

Using hand-crafted feature-engineering or simple feature extraction methods on GPS historical trajectories may not necessarily result in a highly accurate model for DMF identification. One may consider the drivers as the *agents*, and learn the drivers' decision-making behaviors via series of state-action pairs in the context of reinforcement learning (RL). The resulting *reward* and *policy* values may characterize the drivers' behaviors and form the DMFs. However, the reward and policy values in the conventional RL context are not available in our trajectories to reflect our drivers' decision-making behavior patterns and the latent driver identities.

To address this, we consider spatio-temporal inverse reinforcement learning (ST-IRL) to *recover* the reward and the policy, *i.e.*, DMFs; the recovered reward and the policy are formed as base-maps, and are used to create the reward $\chi_{l,j}^{d,t}$ and policy $\psi_{l,j}^{d,t}$ heat-maps.

To transform our problem into the inverse reinforcement learning context [31], we first convert the discretized trajectories of the drivers to series of *state-action* pairs (Sec. 3.2). Then, we feed series of such state-action pairs to ST-IRL to learn the reward and the policy base-maps (Sec. 3.3). These indicate the inherent mobility features and the drivers' decision-making behavior patterns and will serve as features to create distinctive DMFs for identification.

3.2 Formulation for ST-IRL in RM-Drive

• **States & Agents**: Considering the temporal and spatial discretization described in the previous sections, we define one state per each grid-cell of each grid-map. Consequently, the state-space consists of T state-map matrices $\{\mathcal{S}_l^1, \mathcal{S}_l^2, ..., \mathcal{S}_l^T\}$ with $\mathcal{H}_l \times \mathcal{W}_l$ dimensions. Each state-map $\mathcal{S}_l^t$ is defined as follows:

$$\mathcal{S}_l^t = \begin{bmatrix} s_l^t[1,1] & s_l^t[1,2] & \cdots & s_l^t[1,\mathcal{W}_l] \\ \vdots & \vdots & \vdots & \vdots \\ s_l^t[\mathcal{H}_l,1] & s_l^t[\mathcal{H}_l,2] & \cdots & s_l^t[\mathcal{H}_l,\mathcal{W}_l] \end{bmatrix}, \quad (8)$$

where $s_l^t[h,w]$ is the state at the h-th row and the w-th column of state-map $\mathcal{S}_l^t$ that corresponds to grid-cell $g_l^t[h,w]$ of grid-map G_l^t. In our study, we define each driver d as an agent, and we can use states to represent previous locations (*i.e.,* grid-cells) of the agent.

• **Actions** $\mathcal{A}$: For each state, we define the set of actions $\mathcal{A} = \{a_1, a_2, ..., a_9\}$, each of which, from a_1 to a_9, respectively represents east, west, south, north, south-east, north-west, south-west, north-east, and stay (*i.e.,* stay in the current state). We further illustrate the possible actions from a given state $s_l^t[h,w]$ to its neighborhood states in Fig. 8. We note that some of the actions are not available in certain states due to the grid-map boundary. For instance, for the state $s_l^t[0,0]$ in Fig. 8, there are only three possible actions (*i.e.,* east, south, and south-east) to neighborhood states. Furthermore, we confine the actions of transitions to only the potential states that are in the same time interval.

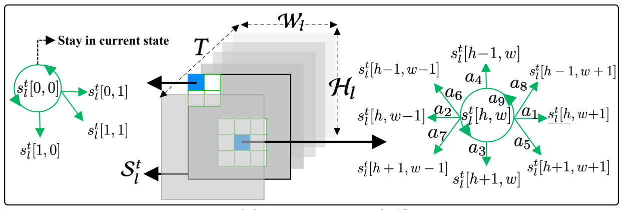


Figure 8: Possible actions in different states.

• **State-Action Trajectory**: Given the state and the action space defined ealier, we transform each discretized trajectory into a series of state-action pairs $(s_l^t[h,w], a_k)$, $s_l^t[h,w] \in \mathcal{S}_l^t$, $a_k \in \mathcal{A}$. Then, given the T time intervals (say, each is 1h in our case), we divide the state-action trajectories of driver d over a period of $\mathcal{L}_{\text{train}}$ consecutive days into T subsets $\{\tilde{\mathcal{T}}_l^{d,1}, \tilde{\mathcal{T}}_l^{d,2}, \cdots, \tilde{\mathcal{T}}_l^{d,T}\}$, where $\tilde{\mathcal{T}}_l^{d,t} = \{\tilde{\tau}_{l,1}^{d,t}, \cdots, \tilde{\tau}_{l,N_d}^{d,t}\}$ are the trajectories of driver d over a period of $\mathcal{L}_{\text{train}}$ days that have occurred at the t-th time interval q_t (such as 8:00 – 9:00am over 25 days).

• **Reward & Policy Base-Maps**: For each $\mathcal{S}_l^t$, given driver d, ST-IRL recovers the reward base-map $\mathcal{R}_l^{d,t} \in \mathbb{R}^{\mathcal{H}_l \times \mathcal{W}_l}$ using state-action trajectories of driver d at the t-th time interval, $\tilde{\mathcal{T}}_l^{d,t}$, over a period of $\mathcal{L}_{\text{train}}$ days. We define each reward base-map $\mathcal{R}_l^{d,t}$ as

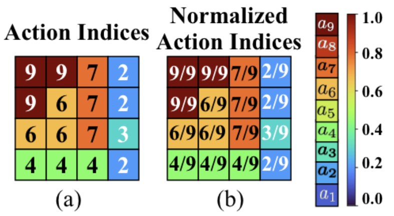


Figure 9: (a) An example of policy base-map; (b) an example of normalized policy base-map.

$$\mathcal{R}_l^{d,t} = \begin{bmatrix} r_l^{d,t}[1,1] & r_l^{d,t}[1,2] & \cdots & r_l^{d,t}[1,\mathcal{W}_l] \\ \vdots & \vdots & \vdots & \vdots \\ r_l^{d,t}[\mathcal{H}_l,1] & r_l^{d,t}[\mathcal{H}_l,2] & \cdots & r_l^{d,t}[\mathcal{H}_l,\mathcal{W}_l] \end{bmatrix}, \quad (9)$$

where $r_l^{d,t}[h,w]$ corresponds to the reward value of driver d at state $s_l^t[h,w]$. Besides, after recovering each reward base-map, ST-IRL returns the optimal policy base-map $\pi_l^{d,t} \in \mathbb{R}^{\mathcal{H}_l \times \mathcal{W}_l}$ corresponding to $\mathcal{R}_l^{d,t}$ with the approximate value-iteration algorithm [31]. We define each policy base-map $\pi_l^{d,t}$ as

$$\pi_l^{d,t} = \begin{bmatrix} v_l^{d,t}[1,1] & v_l^{d,t}[1,2] & \cdots & v_l^{d,t}[1,\mathcal{W}_l] \\ \vdots & \vdots & \vdots & \vdots \\ v_l^{d,t}[\mathcal{H}_l,1] & v_l^{d,t}[\mathcal{H}_l,2] & \cdots & v_l^{d,t}[\mathcal{H}_l,\mathcal{W}_l] \end{bmatrix}, \quad (10)$$

where $v_l^{d,t}[h,w]$ represents the index of the action that should be taken in state $s_l^t[h,w]$. Fig. 9(a) illustrates an example of a policy base-map, where each value indicates the index of the proposed action by the policy in that state. For instance, the elements with a value of 4 at the bottom of the policy base-map represent the action a_4=north for the corresponding states there. Furthermore, as depicted in Fig. 9(b), we normalize the policy base-map by the total number of actions (*i.e.,* 9).

• **Problem of** ST-IRL: Given historical state-action trajectories $\tilde{\mathcal{T}}_l^{d,t}$, ST-IRL in our RM-Drive aims to recover the reward $\mathcal{R}_l^{d,t}$ and policy $\pi_l^{d,t}$ base-maps.

As discussed in Sec. 3.1, RL is not applicable in our case because of unknown reward values. Therefore, we leverage ST-IRL to recover the reward and policy base-maps based on the given historical trajectories and output the features that are dependent upon the drivers' identities. According to [31], given state-action trajectories $\tilde{\mathcal{T}}_l^{d,t}$, the problem of recovering reward base-map $\mathcal{R}_l^{d,t}$ with model parameters θ is formulated into the following maximization problem in Bayesian statistics context, *i.e.,*

$$J(\theta) = \log P(\tilde{\mathcal{T}}_l^{d,t}, \theta | \mathcal{R}_l^{d,t}) = \log P(\tilde{\mathcal{T}}_l^{d,t} | \mathcal{R}_l^{d,t}) + \log P(\theta), \quad (11)$$

where the objective is to maximize the joint posterior distribution of the state-action trajectories $\tilde{\mathcal{T}}_l^{d,t}$ being generated given reward base-map $\mathcal{R}_l^{d,t}$. Let J_1 be the posterior probability term $\log P(\tilde{\mathcal{T}}_l^{d,t} | \mathcal{R}_l^{d,t})$, and J_2 be the model parameter regularization term $\log P(\theta)$ in Eq. (11). The gradient of $J(\theta)$ is then given by the sum of the gradients of J_1 and J_2 with respect to the model parameters θ *i.e.,*

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{\partial J_1}{\partial \theta} + \frac{\partial J_2}{\partial \theta}, \quad \text{where} \quad \frac{\partial J_1}{\partial \theta} = \frac{\partial J_1}{\partial \mathcal{R}_l^{d,t}} \times \frac{\partial \mathcal{R}_l^{d,t}}{\partial \theta}. \quad (12)$$

It is shown in [33] that the gradients of the maximum entropy cost function with respect to the reward base-map $\frac{\partial J_1}{\partial \mathcal{R}_l^{d,t}}$ equals to the difference in state-visitation counts of the state-action trajectories $\tilde{\mathcal{T}}_l^{d,t}$, and the expected visitation counts based on the derived policy, *i.e.,*

$$\frac{\partial J_1}{\partial \mathcal{R}_l^{d,t}} = \mu_{\tilde{\mathcal{T}}_l^{d,t}} - E[\mu]. \tag{13}$$

Note that we slightly adapt the gradients to be *only* applied to reward of states along the given state-action trajectories, *i.e.,* states visited by at least one trajectory in $\tilde{\mathcal{T}}_l^{d,t}$, and this minor change helps to provide a less noisy reward and policy base-maps in fewer iterations.

3.3 Policy & Reward Learning in ST-IRL

We further discuss how to use ST-IRL to recover reward and policy base-maps, *i.e.,* $\{\mathcal{R}_l^{d,1}, \cdots, \mathcal{R}_l^{d,T}\}$ and $\{\pi_l^{d,1}, \cdots, \pi_l^{d,T}\}$, for driver d within the past $\mathcal{L}_{\text{train}}$ days.

We detail the learning process of our ST-IRL in Algorithm (1). It takes in states features $f_{\mathcal{S}}$ (see Appendix C for details), and the state-action trajectories $\tilde{\mathcal{T}}_l^{d,t}$, and outputs the recovered reward $\mathcal{R}_l^{d,t}$ and policy $\pi_l^{d,t}$ base-maps. In particular, at each iteration, we update the reward base-map by feeding the states features $f_{\mathcal{S}}$ to the network (Line 6). Then, we use the updated reward base-map $\mathcal{R}_l^{d,t}$ to derive the policy base-map $\pi_l^{d,t}$ using the value-iteration algorithm (Line 7). Finally, we calculate the gradients according to Eq. (12) and update the network parameters (Lines 8 and 9).

Algorithm 1: Learning Reward and Policy with IRL.

```
1  Inputs: States Features $f_{\mathcal{S}}$, Disretized Trajectories $\tilde{\mathcal{T}}_l^{d,t}$.
2  Outputs: Reward $\mathcal{R}_l^{d,t}$, Policy $\pi_l^{d,t}$ Base-Maps.
3  $\mathcal{R}_l^{d,t} \leftarrow$ Initialize as an $\mathcal{H}_l \times \mathcal{W}_l$ zero matrix;
4  $\theta \leftarrow$ Initialize network parameters;
5  for $k \leftarrow 0$ to $N_{iterations}$ do
6      $\mathcal{R}_l^{d,t} \leftarrow$ Feed $f_{\mathcal{S}}$ to the network and predict the reward;
7      $\pi_l^{d,t} \leftarrow$ Calculate policy using approximate-value-iteration;
8      $\frac{\partial J(\theta)}{\partial \theta} \leftarrow$ Calculate gradients;
9      $\theta \leftarrow$ Update network parameters using the gradients;
10 return $\mathcal{R}_l^{d,t}, \pi_l^{d,t}$
```

We note that ST-IRL recovers the reward and policy base-maps based on the resolution of the trajectories. Therefore, we feed a group of trajectories to ST-IRL multiple times with different resolutions to produce the aforementioned base-maps, which we use to create multi-resolution input mobility fingerprint tensors that serve as the input for our driver classification and anomaly detection (Sec. 4) as illustrated in Fig. 11. As stated in Sec. 2.2, we use the recovered reward $\mathcal{R}_l^{d,t}$ and policy $\pi_l^{d,t}$ base-maps to create the reward $\chi_{l,j}^{d,t}$ and the policy $\psi_{l,j}^{d,t}$ trajectory heat-maps for given discretized trajectory $\left(\tau_{l,j}^{d,t}\right)'$. Fig. 10 further shows an example of an input mobility fingerprint tensor created for a discretized trajectory using reward, policy, and two POI category base-maps.

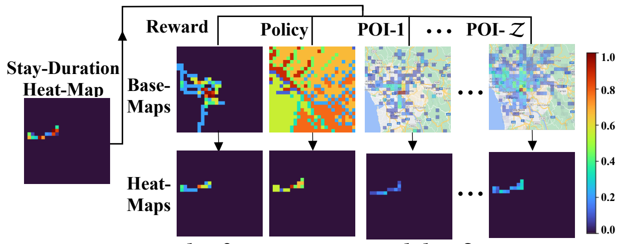


Figure 10: An example of creating input mobility fingerprint tensor for a discretized trajectory from dataset *DS*1.

4 MOBILITY FINGERPRINT IDENTIFICATION

4.1 Overview of Identification Designs

As stated earlier, single-resolution mobility features may not maintain all the important information that is critical to capture the complex decision-making behavior patterns of the drivers. Consequently, we consider creating drivers' input mobility fingerprint tensors with multiple resolutions to enhance the identification accuracy.

In particular, in this study, for each driver's trajectory, we generate three input mobility fingerprint tensors ($f_{l,j}^{d,t}$) ($l \in \{1,2,3\}$) with three resolutions, each of which consists of ($\mathcal{Z}+3$) heat-maps, including stay-duration heat-map, heat-maps created from reward, policy, and $\mathcal{Z} = 11$ categories of POI base-maps. Then, we employ multi-resolution trajectory embedding network (MTE-Net) (Sec. 4.2), which consists of three trajectory encoder networks as shown in Fig. 11, to process and generate embeddings for the three resolutions.

Each trajectory encoder network consists of two sub-modules, *(i)* reward-policy network (RP-Net) to learn and predict the reward and policy heat-maps; *(ii)* a multi-resolution convolutional neural network (MR-CNN) to further embed under multiple resolutions. Afterward, we concatenate the embeddings of the input mobility fingerprint tensors with the weather feature vector $\zeta(c_j^{d,t})$ generated by a fully connected layer (FC). This way, we obtain the final trajectory embedding $\omega_j^{d,t}$ for DMF identification. Specifically, we feed it to the softmax function along with the driver's label to perform driver classification. In terms of final identification tasks, we use the siamese network for anomaly detection; we feed the final embeddings, $\omega_j^{d_1,t}$ and $\omega_k^{d_2,t}$, of two discretized trajectories, $(\tau_{l,j}^{d_1,t})'$ and $(\tau_{l,k}^{d_2,t})'$, to the softmax classifier for anomaly detection.

4.2 Multi-Resolution Trajectory Embedding Network

• **Input**: We have shown in Sec. 3.3 how our ST-IRL recovers the reward and policy base-maps given a set of discretized trajectories, which completes the definition of the input mobility fingerprint tensor $f_{l,j}^{d,t} = \left(\phi_{l,j}^{d,t}, \chi_{l,j}^{d,t}, \psi_{l,j}^{d,t}, \mathbf{o}_{l,j}^{d,t}\right)$ ($l \in \{1,2,3\}$) for the discretized trajectory $(\tau_{l,j}^{d,t})'$. Furthermore, we defined the weather feature vector $c_j^{d,t}$ as an additional contextual factor for each trajectory in Sec. 2.2.

• **Trajectory Encoder Network**: For each input discretized trajectory $(\tau_{l,j}^{d,t})'$ with the l-th resolution, we use trajectory encoder network E_l to generate one embedding from the input mobility

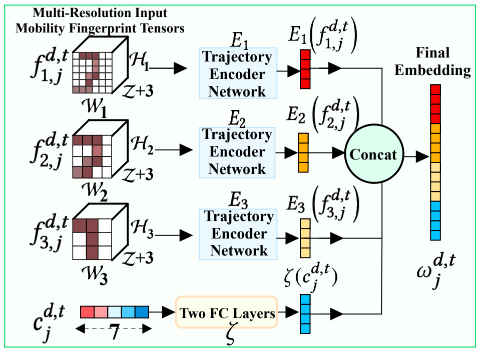

Multi-Resolution Input Mobility Fingerprint Tensors
Trajectory Encoder Network
Trajectory Encoder Network
Trajectory Encoder Network
Concat
Final Embedding
Two FC Layers

Figure 11: Multi-resolution trajectory embedding network architecture.

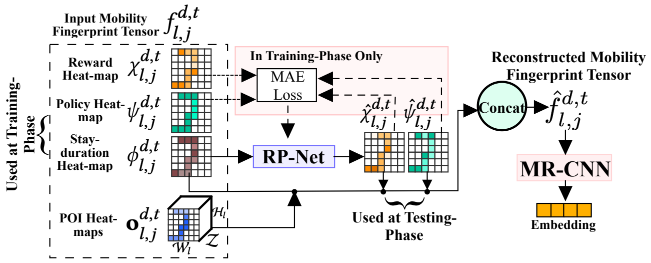


Figure 12: Structure of trajectory encoder network.

fingerprint tensor $f_{l,j}^{d,t}$. Fig. 12 illustrates the trajectory encoder network, which conducts the following three tasks. *(a)* The trajectory encoder network learns to reconstruct trajectory reward $\chi_{l,j}^{d,t}$ and policy $\psi_{l,j}^{d,t}$ heat-maps given the stay-duration heat-map $\phi_{l,j}^{d,t}$ using RP-Net at training-phase. *(b)* We concatenate the stay-duration, reconstructed reward $\hat{\chi}_{l,j}^{d,t}$ and policy $\hat{\psi}_{l,j}^{d,t}$ heat-maps, with the ones regarding POIs, $\mathbf{o}_{l,j}^{d,t}$, to get the reconstructed mobility fingerprint tensor $\hat{f}_{l,j}^{d,t} = \{\phi_{l,j}^{d,t}, \hat{\chi}_{l,j}^{d,t}, \hat{\psi}_{l,j}^{d,t}, \mathbf{o}_{l,j}^{d,t}\}$. *(c)* The trajectory encoder network generates the embedding $E_l\left(f_{l,j}^{d,t}\right)$ using MR-CNN for later anomaly detection and driver classification.

(i) RP − Net Designs: We first train RP-Net to reconstruct the reward $\chi_{l,j}^{d,t}$ and the policy $\psi_{l,j}^{d,t}$ heat-maps of the given discretized trajectory $\left(\tau_{l,j}^{d,t}\right)'$. Then, at testing-phase, we feed only the stay-duration heat-map $\phi_{l,j}^{d,t}$ to the RP-Net to get the reward and policy heat-maps. The network structure of RP-Net consists of *2D-CNN* and *2D-CNN-transpose* blocks given by

$$\begin{aligned} h'_i &= \mathtt{LReLU}(\mathtt{BatchNorm}(\mathtt{Conv2D}(x))), \\ \bar{h}_i &= \mathtt{LReLU}(\mathtt{BatchNorm}(\mathtt{Conv2DTranspose}(x))), \end{aligned} \tag{14}$$

where x is the input tensor before each layer, and LReLU is the leaky rectified linear unit activation function [12] that is used to add non-linearity. As illustrated in Fig. 13, to process the input, RP-Net uses three 2D-CNN blocks that are followed by three 2D-CNN-transpose blocks to adjust the output dimensions.

Given the reward $\chi_{l,j}^{d,t}$ and the policy $\psi_{l,j}^{d,t}$ heat-maps as ground-truth values, we calculate the mean absolute error (MAE) as the loss with tunable parameters α and β and use it to update the RP-Net parameters, *i.e.,*

$$J((\chi_{l,j}^{d,t}, \psi_{l,j}^{d,t}), (\hat{\chi}_{l,j}^{d,t}, \hat{\psi}_{l,j}^{d,t}) = \alpha \cdot (|\chi_{l,j}^{d,t} - \hat{\chi}_{l,j}^{d,t}|) + \beta \cdot (|\psi_{l,j}^{d,t} - \hat{\psi}_{l,j}^{d,t}|).$$

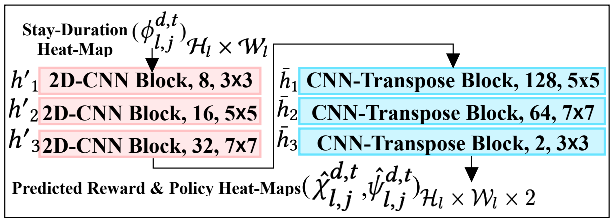

Stay-Duration Heat-Map $(\phi_{l,j}^{d,t})_{\mathcal{H}_l \times \mathcal{W}_l}$
h'_1 2D-CNN Block, 8, 3x3
h'_2 2D-CNN Block, 16, 5x5
h'_3 2D-CNN Block, 32, 7x7
$\bar{h}_1$ CNN-Transpose Block, 128, 5x5
$\bar{h}_2$ CNN-Transpose Block, 64, 7x7
$\bar{h}_3$ CNN-Transpose Block, 2, 3x3
Predicted Reward & Policy Heat-Maps $(\hat{\chi}_{l,j}^{d,t}, \hat{\psi}_{l,j}^{d,t})_{\mathcal{H}_l \times \mathcal{W}_l \times 2}$

Figure 13: The network structure of the RP-Net.

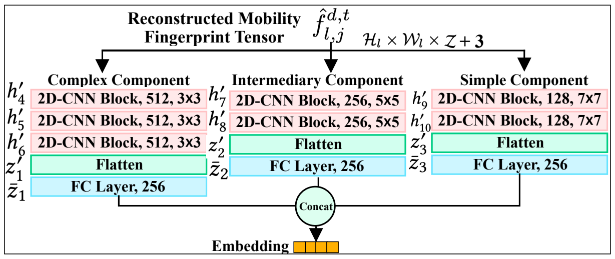

Reconstructed Mobility Fingerprint Tensor $\hat{f}_{l,j}^{d,t}$ $\mathcal{H}_l \times \mathcal{W}_l \times \mathcal{Z}+3$
Complex Component
h'_4 2D-CNN Block, 512, 3x3
h'_5 2D-CNN Block, 512, 3x3
h'_6 2D-CNN Block, 512, 3x3
z'_1 Flatten
$\bar{z}_1$ FC Layer, 256
Intermediary Component
h'_7 2D-CNN Block, 256, 5x5
h'_8 2D-CNN Block, 256, 5x5
z'_2 Flatten
$\bar{z}_2$ FC Layer, 256
Simple Component
h'_9 2D-CNN Block, 128, 7x7
h'_{10} 2D-CNN Block, 128, 7x7
z'_3 Flatten
$\bar{z}_3$ FC Layer, 256
Concat
Embedding

Figure 14: The network structure of the MR-CNN.

In testing-phase, the reward $\chi_{l,j}^{d,t}$ and the policy $\psi_{l,j}^{d,t}$ heat-maps are not available, and we use RP-Net to reconstruct these heat-maps given the stay-duration heat-map $\phi_{l,j}^{d,t}$.

(ii) MR − CNN Designs: After RP-Net, we further use MR-CNN to process the reconstructed mobility fingerprint tensor. In addition to the 2D-CNN block, MR-CNN consists of the flatten operation and fully connected (FC) layers given by

$$\begin{aligned} z'_i &= \mathtt{Dropout}(\mathtt{Flatten}(x)), \\ \bar{z}_i &= \mathtt{LReLU}(\mathtt{Dropout}(\mathtt{BatchNorm}(\mathtt{FC}(x)))), \end{aligned} \tag{15}$$

where FC represents fully connected layers, Flatten converts a tensor into a one dimensional vector, and Dropout adds regularization effects [12]. As illustrated in Fig. 14, MR-CNN processes the reconstructed mobility fingerprint tensor $\hat{f}_{l,j}^{d,t}$ with three components with different complexities without pooling layers (Appendix C), where the complex component uses more filters and smaller kernel compared to the intermediary and simple components. This way, the model can learn more distinctive DMFs with each component.

• **Output**: As shown in Sec. 4.2, the resulting embeddings $E_l\left(f_{l,j}^{d,t}\right)$ ($l \in \{1,2,3\}$) of all three resolutions, as well as the embedding of the weather feature vector $\zeta(c_j^{d,t})$ calculated by a fully connected layer are concatenated together and fed as the final embedding $\omega_j^{d,t}$ of a trajectory and the contextual factors, *i.e.,*

$$\omega_j^{d,t} = \mathtt{Concat}\left(E_1\left(f_{1,j}^{d,t}\right), E_2\left(f_{2,j}^{d,t}\right), E_3\left(f_{3,j}^{d,t}\right), \zeta(c_j^{d,t})\right). \tag{16}$$

4.3 Driver Classification & Anomaly Detection

• **Driver Classification**: For driver classification (Fig. 15(a)), we feed the final embedding vector $\omega_j^{d,t}$ to a fully connected layer with $\mathcal{D}$ (*i.e.,* number of the drivers or classes) neurons followed by a softmax function, *i.e.,*

$$\varphi_i = \mathtt{Softmax}(u)_i = \frac{\exp(u^i)}{\sum_j^{\mathcal{D}} \exp(u^j)}. \tag{17}$$

Finally, the cross entropy loss for the driver classification is then calculated as

$$\mathtt{CrossEntropy} = -\log(\varphi_p) = -\log\left(\frac{\exp(u^p)}{\sum_j^{\mathcal{D}} \exp(u^j)}\right), \tag{18}$$

where u^p is the probability score of the softmax function on the outputs of the final fully connected layer.

• **Anomaly Detection**: We leverage the siamese network (Fig. 15(b)) to detect whether a given trajectory is generated by a specific driver. In particular, let $(\tau_{l,j}^{d_1,t})'$ and $(\tau_{l,k}^{d_2,t})'$ be two discretized trajectories from drivers d_1 and d_2 respectively, where $l \in \{1, 2, 3\}$. We first use two identical MTE-Net modules to generate the embeddings $\omega_j^{d_1,t}$ and $\omega_k^{d_2,t}$. Then, we concatenate the generated embeddings Ω^{d_1,d_2} and feed them to a fully connected layer with two neurons, followed by the softmax function for binary classification (*i.e.,* anomaly or normal), *i.e.,*

$$\Omega^{d_1,d_2} = \mathtt{FC}(\mathtt{Concat}(\omega_j^{d_1,t}, \omega_k^{d_2,t})). \tag{19}$$

We use the cross entropy in Eq. (18) as the loss function to train our siamese network. We note that drivers d_1 and d_2 are either equal for a normal case, or different drivers in case of an anomaly.

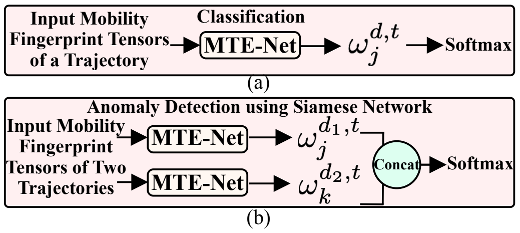

Input Mobility Fingerprint Tensors of a Trajectory
Classification
MTE-Net
$\omega_j^{d,t}$
Softmax
(a)
Anomaly Detection using Siamese Network
Input Mobility Fingerprint Tensors of Two Trajectories
MTE-Net
$\omega_j^{d_1,t}$
MTE-Net
$\omega_k^{d_2,t}$
Concat
Softmax
(b)

Figure 15: Driver classification & anomaly detection.

5 EXPERIMENTAL STUDIES

5.1 Experimental Settings

• **Baselines**: We compare RM-Drive with deep learning models based on residual network (ResNet) [13], convolutional neural network (CNN), long short-term memory (LSTM) [25], bidirectional LSTM (BLSTM), stacked recurrent neural network (SRNN) [8], fully connected (FC) networks, and three traditional machine learning models, *i.e.,* gradient boosting decision tree (GBDT) [28], random forest (RF) [5], and support vector machine (SVM) [24]. We present the details of the baselines in Appendix B. Our detailed settings, including important parameters and experimental evaluation designs, are presented in Appendix C.

• **Metrics**: To evaluate driver classification, we leverage top-5 accuracy (TP5), *i.e.,* if the ground-truth class of a sample is equal to one of the top-5 predictions of the model with the highest probabilities, we label the result as 1 (0 otherwise) and find the percentage of ones in all samples. Besides, we use *Accuracy* (*i.e.,* the number of correct predictions over the total number of samples) as another metric for driver classification. For anomaly detection, the class label is 1 for the anomalies (*i.e.,* two trajectories from two different drivers), and 0 for normal cases (*i.e.,* two trajectories from the same driver). Thus, *precision*, *recall*, and *F1* values of the anomaly class are also reported. We provide the details of the training and testing sets in Appendix C.

5.2 Experimental Results

• **Overall Performance**: Tab. 1 shows the performance of RM-Drive for the task of driver classification compared to the baselines. We can see that RM-Drive outperforms the most accurate baseline (CNN) in both *DS*1 and *DS*2 datasets. It is evident that the recurrent models do not perform as well as the CNN-based models, which shows that the input mobility fingerprint tensors are more effective than trajectory sequences in capturing the driver's decision-making behavior patterns. Traditional machine learning models cannot adapt to high dimensions of the features and hence achieve lower accuracy than ours. We also note that RM-Drive achieves robust results over two different datasets with the same system parameters including the resolutions and the network hyper-parameters, demonstrating the adaptivity and generality of the proposed framework.

Table 1: Driver classification results on *DS*1 & *DS*2 (%).

Model	TP5		Accuracy		Model	TP5		Accuracy	
	DS1	DS2	DS1	DS2		DS1	DS2	DS1	DS2
RM − Drive	**74.2**	**71.4**	**62.6**	**60.3**	LSTM	63.8	32.5	55.6	25.3
ResNet	59.6	54.4	51.2	48.8	FC	51.5	45.8	44.3	34.1
CNN	66.4	69.1	56.2	53.8	GBDT	57.2	32.6	46.3	21.1
BLSTM	64.4	32.7	52.3	24.8	RF	61.1	31.1	52.5	18.1
SRNN	59.1	31.1	52.5	23.2	SVM	32.5	23.3	17.1	10.3

Tab. 2 further demonstrates the anomaly detection task. We can see that RM-Drive performs this task with better performance compared to the baselines. As stated earlier, we set the class label for the anomaly case as 1. Therefore, the high precision achieved by our RM-Drive implies that the model has a low false-positive rate and does not frequently label a normal case as an anomaly.

Table 2: Anomaly detection results on *DS*1 & *DS*2 (%).

Model	Accuracy		Precision		Recall		F1	
	DS1	DS2	DS1	DS2	DS1	DS2	DS1	DS2
RM − Drive	**90.2**	**86.2**	**89.3**	**83.3**	**97.0**	**99.6**	**93.0**	**90.7**
ResNet	86.0	84.0	88.7	81.3	91.1	92.8	89.9	86.7
CNN	80.0	81.3	80.1	82.2	92.9	91.6	86.2	86.9
BLSTM	69.0	63.0	68.8	62.5	61.1	63.0	61.2	50.6
SRNN	68.8	66.2	70.1	65.2	68.8	66.3	69.2	65.6
LSTM	61.4	59.2	55.0	78.0	61.4	59.2	56.3	58.6
FC	58.0	54.2	70.4	77.2	63.6	48.5	66.9	59.6
GBDT	79.1	77.1	83.8	75.8	83.2	84.1	83.5	79.7
RF	66.0	59.8	75.6	75.2	71.2	61.4	73.3	67.6
SVM	68.0	63.1	74.2	80.0	78.3	59.8	76.2	68.4

• **Ablation Studies**: We further conduct ablation studies to demonstrate the importance of different components of RM-Drive and the features used. In particular, we consider the following variations: (1) the complete model with all the features, (2) without the weather feature vector, (3) without all the contextual factors, (4) without all the DMFs extracted with ST-IRL (*i.e.,* reward and policy heatmaps), (5) without both contextual factors and DMFs extracted with ST-IRL, (6) with single resolution CNN with a similar structure to the complex component of MR-CNN, and (7) with using only the highest resolution of the input mobility fingerprint tensor (*i.e.,* 50×50).

We illustrate the performance of the defined variations on *DS*1 in Fig. 5.2(a). Clearly, removing the DMFs extracted by ST-IRL causes two significant performance drops. Additionally, removing the contextual factors also causes an accuracy drop, which accounts for the importance of such factors to learn more distinctive DMFs. Besides, the performance drops caused by the removal of MR-CNN and the multi-resolution input mobility fingerprint tensors validate importance of our multi-resolution model and indicate that input DMF tensors with multiple resolutions are essential for a good accuracy. We note that as the driver classification on a large number of drivers is challenging, the removal of different components and features results in more significant performance drops.

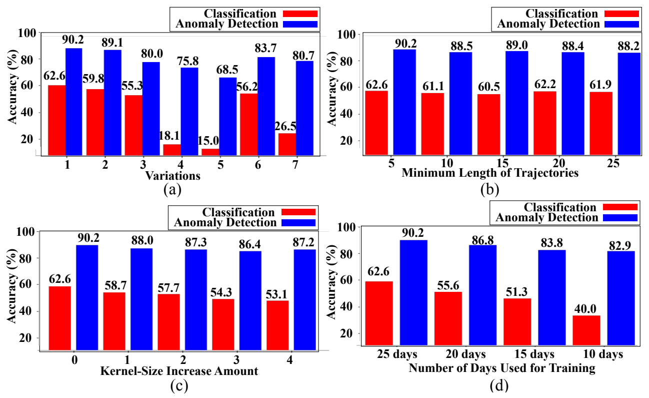


Figure 16: (a) Ablation studies results. Sensitivity studies of (b) minimum trajectory length, (c) kernel's size of MR-CNN, and (d) the number of days used in training-phase.

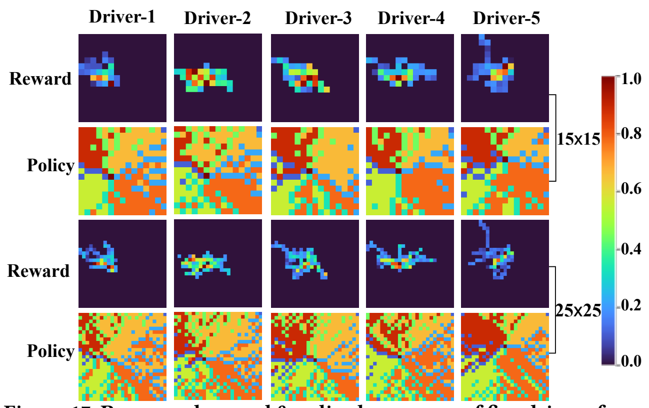


Figure 17: Recovered reward & policy base-maps of five drivers from $DS1$ with different resolutions at 18:00 – 19:00 time interval.

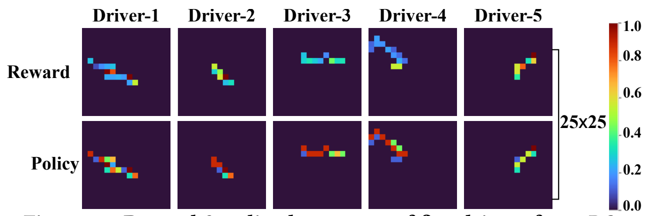


Figure 18: Reward & policy heat-maps of five drivers from $DS1$

• **Sensitivity Studies**: We have performed sensitivity studies regarding the following settings: *(i)* minimum length of the trajectories, *(ii)* the kernel size of the CNN layers used in MR-CNN, and *(iii)* the number of consecutive days over a one-month period that we use to aggregate historical trajectories for training-phase (*i.e.,* L_{train}). Fig. 5.2(b) shows the performance of RM-Drive when the minimum length of the trajectories ranges from 5 to 25. We can see that the maximum difference in the accuracy is at most 2% for both driver classification and anomaly detection, which indicates that our proposed model achieves robust accuracy given both short and long trajectories. Fig. 5.2(c) shows that the performance starts to degrade given larger kernels. We use 3×3, 5×5, and 7×7 kernels for the components of MR-CNN. To measure the sensitivity of these variables, we gradually increase the kernels' size in four steps as long as they are not limited by the input dimension (*e.g.,* the kernels are increased to 4×4, 6×6, and 8×8 in the first step). Fig. 5.2(d) shows that the performance drops when a smaller number of days is used. Therefore, to capture the complex DMFs, the data for more than two weeks should be available.

• **Visualization**: We further visualize the recovered reward and policy base-maps for five drivers in Fig. 17, which indicates that multiple levels of resolutions help differentiate the DMFs. We then show in Fig. 18 the random trajectories' reward and policy heat-maps from the stated drivers, in which we can observe that each heat-map has an important role to make the whole input mobility fingerprint tensor more distinctive. We also provide additional base-maps with a different resolution and more trajectory heat-maps in Figs. 20 and 21 in Appendix D.

6 RELATED WORKS

• **Driver Trajectory Data Mining**: With recent advances in vehicle computing (*i.e.,* traffic data analysis [21, 27], driver behavior studies [22], etc.), and mobility data collection and management [19], driver identification has attracted much attention. To extract features from drivers' trajectory data, various machine learning models have been studied, including auto-encoder [3, 4]. Chen *et al.* [4] use an auto-encoder to determine the sliding window size for statistical feature extraction. Likewise, Chen *et al.* [3] introduced a regularized auto-encoder architecture to learn the embeddings of taxi trips.

However, solely relying upon auto-encoder may likely ignore critical spatio-temporal features after feature compression. Furthermore, while focusing on reconstructing the original input, auto-encoder may not necessarily learn complex mobility features. Consequently, to overcome the limitations of the auto-encoders, other researchers have focused upon inverse reinforcement learning (IRL) to process trajectories [31]. Unlike auto-encoder, IRL does not necessarily require large amount of data to extract useful features and hence can be updated more efficiently given new data.

Despite these prior studies on general mobility feature learning (say, driving style [18], transportation-mode [32], and taxis' learning process and working preferences [23, 30]), few of these have specifically studied IRL for our target DMF identification. Besides, our work advances in the following two perspectives. *First*, our proposed ST-IRL design captures the spatio-temporal features of each driver independently (*i.e.,* via multiple reward and policy base-maps per driver) rather than learning a single reward base-map for all the drivers. This makes the learned features specifically representative of each driver's decision-making behavior patterns without taking effect from other drivers' data. *Second*, our design not only enables a faster and parallel driver mobility learning process but also eases the future feature updates for each driver given new trajectory data.

• **Trajectory-based Driver Identification**: Focusing on driver trajectory processing, we further review the driver identification in the following two categories.

(i) **Driver Classification**: Various learning models have been studied to support driver classification, which can be briefly divided into the following two groups.

– *Deep Learning Approaches*: Some works built their models based on recurrent neural networks (RNN), which processes sequences of trajectory data. ARNet [8] learns trip-level and segment-level classification using a supervised auto-encoder with Gated Recurrent Units (GRU). Fan *et al.* [10] used Bidirectional LSTM for human trajectory identification. To further extract the driver mobility patterns, 1D convolutional neural network (1D-CNN) has been adopted

(in combination with LSTM or Simple RNN) for driver classification, particularly based on the trajectory data [7, 9]. Such approaches, however, could not fully extract the differentiable features between drivers. Furthermore, many of the prior studies [20] have not taken into account the different resolutions of the drivers' data, leading to potential information loss and performance degradation.
– *Traditional Machine Learning Models*: Aside from deep learning models, there are several other papers that employed simpler machine learning models to build their classifiers, including Random Forest [5], SVM [24], and Gradient Boosting Decision Tree (GBDT) [28]. Compared with deep learning approaches, these models may not provide sufficient learnability and satisfactory results upon large-scale driver data analytics.
(ii) **Trajectory Anomaly Detection**: Different from driver classification, anomaly detection focuses on determining whether a new trajectory is likely anomalous considering the historical trajectories of a given driver. Dang *et al.* [6] proposed a siamese network architecture [2] based on two identical LSTM modules to detect anomalies in drivers' future activities. Furthermore, Ren *et al.* [25] proposed a similar architecture and improved the performance by considering the transition mode of the trajectories (*i.e.,* with or without passengers) and driver profile features that were calculated over time. Similarly, Wijnands *et al.* [29] presented an LSTM-based architecture to detect positive and negative changes in the drivers' behaviors. Different from the above works in *(i)* and *(ii)*, our `RM-Drive` advances from two major perspectives: *(a)* extracting complex spatio-temporal and contextual features from the drivers' trajectories based on a novel `ST-IRL` design; and *(b)* developing a multi-resolution model to capture the complex DMFs.

7 CONCLUSION

In this work, we propose `RM-Drive`, a novel framework that extracts DMFs using spatio-temporal inverse reinforcement learning (`ST-IRL`). We have designed multi-resolution trajectory embedding network (`MTE-Net`) that integrates multi-resolution convolutional neural network (`MR-CNN`) to learn more distinctive DMFs. In addition, our framework takes the correlation between contextual factors (*i.e.,* weather conditions and POIs) and driver's decision-making behavior patterns into account to enhance the prediction accuracy. Our extensive experimental studies on two large-scale real-world datasets confirm the accuracy, efficiency, and applicability of `RM-Drive` for DMF identification.

REFERENCES

[1] Mohammad Asghari and Cyrus Shahabi. 2017. An On-line Truthful and Individually Rational Pricing Mechanism for Ride-sharing. In *Proc. ACM SIGSPATIAL.* 1–10.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" Time Delay Neural Network. *Proc. NeurIPS* 6 (1993), 737–744.

[3] Chao Chen, Chengwu Liao, Xuefeng Xie, Yasha Wang, and Junfeng Zhao. 2019. Trip2Vec: a deep embedding approach for clustering and profiling taxi trip purposes. *Personal and Ubiquitous Computing* 23, 1 (2019), 53–66.

[4] Jie Chen, ZhongCheng Wu, and Jun Zhang. 2019. Driver identification based on hidden feature extraction by using adaptive nonnegativity-constrained autoencoder. *Applied Soft Computing* 74 (2019), 1–9.

[5] Chowdhury, Arijit and Chakravarty, Tapas and Ghose, Avik and Banerjee, Tanushree and Balamuralidhar, P. 2018. Investigations on Driver Unique Identification from Smartphone's GPS Data Alone. *Journal of Advanced Transportation* (2018).

[6] Hien Dang and Johannes Fürnkranz. 2019. Driver Information Embedding with Siamese LSTM networks. In *Proc. IEEE Symposium on Intelligent Vehicle.* 935–940.

[7] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. 2016. Characterizing Driving Styles with Deep Learning. *arXiv* (2016). arXiv:1607.03611

[8] Weishan Dong, Ting Yuan, Kai Yang, Changsheng Li, and Shilei Zhang. 2017. Autoencoder regularized network for driving style representation learning. *arXiv* (2017). arXiv:1701.01272

[9] Abdellah El Mekki, Afaf Bouhoute, and Ismail Berrada. 2019. Improving driver identification for the next-generation of in-vehicle software systems. *IEEE Transactions on Vehicular Technology* 68, 8 (2019), 7406–7415.

[10] Zipei Fan, Quanjun Chen, Renhe Jiang, Ryosuke Shibasaki, Xuan Song, and Kota Tsubouchi. 2019. Deep Multiple Instance Learning for Human Trajectory Identification. In *Proc. ACM SIGSPATIAL.* 512–515.

[11] Abenezer Girma, Xuyang Yan, and Abdollah Homaifar. 2019. Driver Identification Based on Vehicle Telematics Data using LSTM-Recurrent Neural Network. In *Proc. IEEE ICTAI.* 894–902.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning.* Vol. 1. MIT press Cambridge.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. IEEE CVPR.* 770–778.

[14] Suining He and Kang G. Shin. 2019. Spatio-Temporal Adaptive Pricing for Balancing Mobility-on-Demand Networks. *ACM TIST* 10, 4, Article 39 (July 2019), 28 pages.

[15] Suining He and Kang G. Shin. 2019. Spatio-Temporal Capsule-Based Reinforcement Learning for Mobility-on-Demand Network Coordination. In *Proc. WWW.* 2806–2813.

[16] Suining He and Kang G. Shin. 2020. Spatio-Temporal Capsule-based Reinforcement Learning for Mobility-on-Demand Coordination. *IEEE TKDE* (2020).

[17] Sasan Jafarnejad, German Castignani, and Thomas Engel. 2017. Towards a Real-Time Driver Identification Mechanism Based on Driving Sensing Data. In *Proc. IEEE ITSC.* 1–7.

[18] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. 2015. Learning Driving Styles for Autonomous Vehicles from Demonstration. In *Proc. IEEE ICRA.* 2641–2646.

[19] Ruiyuan Li, Sijie Ruan, Jie Bao, and Yu Zheng. 2017. A Cloud-Based Trajectory Data Management System. In *Proc. ACM SIGSPATIAL.* 1–4.

[20] Sobhan Moosavi, Pravar D Mahajan, Srinivasan Parthasarathy, Colleen Saunders-Chukwu, and Rajiv Ramnath. 2021. Driving Style Representation in Convolutional Recurrent Neural Network Model of Driver Identification. *arXiv* (2021). arXiv:2102.05843

[21] Sobhan Moosavi, Behrooz Omidvar-Tehrani, R Bruce Craig, Arnab Nandi, and Rajiv Ramnath. 2017. Characterizing Driving Context from Driver Behavior. In *Proc. ACM SIGSPATIAL.* 1–4.

[22] Sobhan Moosavi, Behrooz Omidvar-Tehrani, and Rajiv Ramnath. 2017. Trajectory Annotation by Discovering Driving Patterns. In *Proc. ACM SIGSPATIAL.* 1–4.

[23] Menghai Pan, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, and Jun Luo. 2019. Dissecting the Learning Curve of Taxi Drivers: A Data-Driven Approach. In *Proc. SIAM SDM.* 783–791.

[24] Mussadiq Abdul Rahim, Jiamou Liu, Zijian Zhang, Liehuang Zhu, Xin Li, and Salabat Khan. 2020. Who is Driving? Event-Driven Driver Identification and Impostor Detection Through Support Vector Machine. *IEEE Sensors Journal* 20, 12 (2020), 6552–6559.

[25] Huimin Ren, Menghai Pan, Yanhua Li, Xun Zhou, and Jun Luo. 2020. ST-SiameseNet: Spatio-Temporal Siamese Networks for Human Mobility Signature Identification. In *Proc. ACM SIGKDD.* 1306–1315.

[26] Jianhua Shao and Chris Greenhalgh. 2010. DC2S: a Dynamic Car Sharing System. In *Proc. ACM SIGSPATIAL.* 51–59.

[27] Youcheng Wang, Jian Xu, Ming Xu, Ning Zheng, Jinsheng Jiang, and Kaiwei Kong. 2016. A Feature-Based Method for Traffic Anomaly Detection. In *Proc. ACM SIGSPATIAL.* 1–8.

[28] Wang, Yan and Zhao, Tianming and Tahmasbi, Fatemeh and Cheng, Jerry and Chen, Yingying and Yu, Jiadi. 2020. Driver Identification Leveraging Single-turn Behaviors via Mobile Devices. In *Proc. IEEE ICCCN.* 1–9.

[29] Jasper S Wijnands, Jason Thompson, Gideon DPA Aschwanden, and Mark Stevenson. 2018. Identifying behavioural change among drivers usingLong Short-Term Memory recurrent neural networks. *Transportation Research Part F: Traffic Psychology and Behaviour* 53 (2018), 34–49.

[30] Guojun Wu, Yanhua Li, Shikai Luo, Ge Song, Qichao Wang, Jing He, Jieping Ye, Xiaohu Qie, and Hongtu Zhu. 2020. A Joint Inverse Reinforcement Learning and Deep Learning Model for Drivers' Behavioral Prediction. In *Proc. ACM CIKM.* 2805–2812.

[31] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. 2015. Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv* (2015). arXiv:1507.04888

[32] DaeYoung Yoon and Simon S Woo. 2020. Who is Delivering My Food? Detecting Food Delivery Abusers using Variational Reward Inference Networks. In *Proc. ACM CIKM.* 2917–2924.

[33] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning.. In *Proc. AAAI*, Vol. 8. 1433–1438.

APPENDIX

A DETAILS OF DATASETS

(a) Driver Trajectory Datasets: We use two datasets for our RM-Drive's development and experimental studies. In particular, we have:
• **Dataset 1 (*DS*1)**: The first dataset is collected in the city of Porto, Portugal, from 2014 to 2015 that contains 1,710,670 GPS trajectories of 442 taxis with a GPS sampling rate of 1/15 Hz. We select 100 drivers and study their total 29,353 trajectories (with an average of 29 GPS points per trajectory) over a one-month period[6].
• **Dataset 2 (*DS*2)**: The second dataset that we used for our experiments is collected in Rome, Italy in February 2014[7]. This dataset contains 58,589 GPS trajectories of 320 taxi cabs over a period of 30 days with a GPS sampling rate of 1/7 Hz. We perform data analytics on 26,278 trajectories (on average 161 GPS points per trajectory) of 100 selected drivers from this dataset.
(b) Contextual Factors: As a contextual factor, we can refer to the existence of various types and numbers of POIs at different locations. Moreover, some drivers might prefer to drive near shopping malls, while others might avoid it due to possible traffic congestion. To capture such decision-making behavior patterns, we create different categories of POIs, including but not limited to food (including any food-related buildings like cafes, restaurants, etc.), transportation (like bus stations, platforms, etc.), health (like hospital, pharmacy, clinic, etc.), and tourism (like hotel, museum, motel, etc.) in our studies.

Additionally, we take the weather conditions into account as an additional contextual factor. In particular, we consider 7 weather features based on the day that the trajectory has happened. The selected features are Rain, Shower, Drizzle, Snow, Fog, Wind, and Temperature.

B DETAILS OF BASELINES

We present the details of the baseline algorithms as follows.

(1) **ResNet**: We consider a smaller version of the ResNet architecture [13] consisted of the first two stages followed by the pooling and fully connected layers.
(2) **CNN**: We apply a CNN-based model with a network structure that is similar to the complex component of the MR-CNN module.
(3) **LSTM/BLSTM**: We leverage Long Short-term Memory (LSTM) or bidirectional LSTM (BLSTM [10]) network as the encoder module while the other modules remain the same as the ST-SiameseNet [25].
(4) **SRNN**: We adopt two stacked RNN layers [7] for driver classification and anomaly detection by taking the sequences of trajectories as inputs.
(5) **FC**: We leverage the CNN-based auto-encoder followed by fully connected layers (FC) for driver classification and anomaly detection.
(6) **GBDT**: Gradient Boosting Decision Tree (GBDT) [28] is a decision tree with gradient boosting technique that enhances classification and detection.
(7) **RF**: Random Forest (RF) [5] is another statistical learning method for driver classification and anomaly detection.
(8) **SVM**: We adopt support vector machine (SVM) [24], a supervised statistical learning method for driver classification and anomaly detection.

The input to the traditional machine learning models (*i.e.,* GBDT, RF, and SVM) is flattened input mobility fingerprint tensor that is not of the finest granularity (*i.e.,* 50×50) as these models could not handle the high dimensional data in a reasonable amount of time.

C DETAILED EXPERIMENTAL SETTINGS

For all the schemes including RM-Drive and baselines, we set T to be 24, which means that each day is divided into one-hour intervals. Also, a month of trajectory data is selected from each dataset, and the first 25 days of each month is used for training-phase ($\mathcal{L}_{\text{train}}$), and the rest are used for testing-phase ($\mathcal{L}_{\text{test}}$). Note that for future predictions beyond the $\mathcal{L}_{\text{test}}$, new historical data can be added to the previous data to easily update the features according to the new days.

We selected $\mathcal{D} = 100$ drivers from each dataset to perform our experiments. In particular, on each trial for anomaly detection, we use 9,000 and 1,650 pairs of trajectories as the testing set and the training set, respectively, where the number of anomalies and normal samples are equal. To create an anomaly sample, we randomly select two trajectories from two different drivers. On the other hand, to create a normal sample, we randomly select two trajectories from the same driver. We note that at testing-phase, each pair contains a new unseen trajectory and a randomly selected trajectory from the historical data that can belong to the same driver or a different one to create normal and anomaly test samples respectively. For driver classification, after removing the trajectories of *DS*1 with less than 10 points, it contains 10,953 and 4,779 trajectories in the L_{train} and the L_{test} days respectively. Similarly, *DS*2 contains 16,837 and 6,716 trajectories in the L_{train} and the L_{test} days respectively after preprocessing. Finally, to extract driver mobility features with ST-IRL, the training set for each driver is equal to her/his trajectories used for driver classification.

The parameter setting of RM-Drive is given as follows. As mentioned earlier, we define three resolutions to extract the features. We determine the highest resolution empirically, and the next two resolutions are then set accordingly. $\mathcal{H}_1 \times \mathcal{W}_1$, $\mathcal{H}_2 \times \mathcal{W}_2$, and $\mathcal{H}_3 \times \mathcal{W}_3$ are set to 50×50, 25×25, and 15×15, respectively. Furthermore, the kernels of the complex, intermediary, and simple components of MR-CNN are set to 3×3, 5×5, and 7×7 for all the resolutions. As stated earlier, MR-CNN does not use any pooling layers as we observed that these layers do not help improve the performance. It is mainly because the simple pooling approaches might remove some essential structural and latent features across different resolutions of the input mobility fingerprints, while our multi-resolution approach helps preserve these features and hence improves the performance.

Also, we set $\alpha = \beta = 1$ in Eq. (15). In the training-phase of all the models, we use the Adam Optimizer [12] with a learning rate of 0.0001. Furthermore, we set the batch-size and the dropout rate to 128 and 60%, respectively. For the neural network in the ST-IRL

[6]https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015
[7]https://crawdad.org/roma/taxi/20140717/

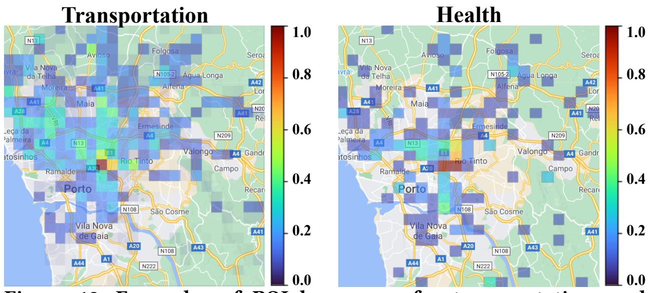


Figure 19: Examples of POI base-maps for transportation, and health categories in the city of Porto, Portugal (*DS*1).

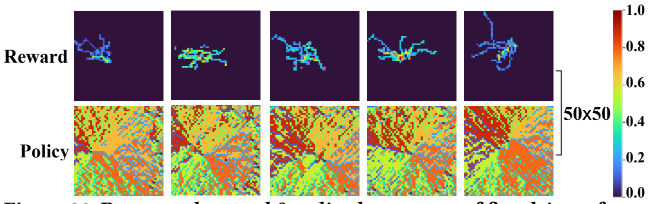


Figure 20: Recovered reward & policy base-maps of five drivers from *DS*1 with 50×50 resolution at 18:00 – 19:00 time interval.

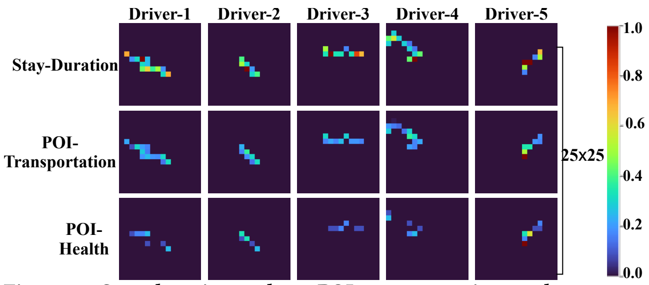


Figure 21: Stay-duration and two POI category trajectory heat-maps of five drivers from dataset *DS*1.

module, we set the learning rate and the number of iterations to 0.05 and 4, respectively.

With the above settings, it takes about 30 seconds to recover the reward and policy base maps for each driver per a time interval (on our machine with Core i7 CPU and RTX2060 GPU). After the feature extraction, the core model takes about 30 minutes per task (*i.e.*, classification and anomaly detection) to converge on the same computing device. Further efficiency enhancement can be achieved through parallel feature extraction for the drivers and adoption of high-performance computing infrastructures.

We set the parameters of the baselines as below. For FC, we build an auto-encoder network consisting of three consecutive CNN layers with 3×3, 5×5, and 7×7 kernels and 64 filters followed by three CNN-transpose layers with 7×7, 5×5, 3×3 kernels, and 64 filters except for the last layer, which has $(\mathcal{Z} + 3)$ filters. The output of the last CNN layer is then fed to a four-layer FC. For SVM, we use Radial Basis Function as the kernel. We set the maximum depth as 50 and the number of estimators as 100 in both RF and GBDT. For all LSTM, BLSTM, and SRNN, we use two layers with 256 neurons. Finally, for CNN, we use a network structure similar to the complex component of MR-CNN.

D VISUALIZATION

In this study, we created $\mathcal{Z} = 11$ categories of POIs, and further illustrate examples of two POI base-maps showing the density of each category in Fig. 19. We further show reward and policy base-maps with an additional resolution (*i.e.,* 50×50) from the same five drivers stated earlier in Fig. 20. Besides, Fig. 21 shows additional trajectory heat-maps of the stated drivers. Furthermore, if we compare the trajectories of the drivers, we will notice partial similarities regarding their shape and location, and if we solely rely on simple features, the model may not be able to classify them correctly. However, the difference in the reward, policy, and POI heat-maps of the trajectories implies that the extracted DMFs of the drivers are distinct and can further assist the model in identifying the drivers.

E DEPLOYMENT DISCUSSION

• *Extension to Other Driver Datasets*: Despite the current focus on historical driver trajectories, our RM-Drive framework is general enough to be integrated with other data sources or features, including the ones extracted from the Controller Area Network (CAN-bus) [11, 17] or from other sensors available upon advanced driver-assistance systems (ADAS), the drivers' smartphones, or wearable devices (*e.g.,* accelerometer and gyroscope) [28] for further accuracy enhancement, or more fine-grained anomaly detection (e.g., driving under the influence).

• *Data Privacy*: The datasets used in this research were anonymized, and no personal information is linked to the GPS records other than a random identifier for each driver. Moreover, the datasets are sanitized and made publicly available, which should address any concerns about the privacy of the drivers. In our research, we aimed to learn driver's data for classification and anomaly detection, and no attempt has been made to find the true identity of the drivers.