Making Sense of Human-generated Spatial-temporal Data from Urban Environment

A Dissertation

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

by

Menghai Pan

April 22, 2021

APPROVED:

Professor Yanhua Li Worcester Polytechnic Institute Advisor	Professor Elke A. Rundensteiner Worcester Polytechnic Institute Committee Member
Professor Mohamed Y. Eltabakh Worcester Polytechnic Institute Committee Member	Professor Xun Zhou The University of Iowa External Committee Member
Professor Craig Wills Worcester Polytechnic Institute Head of Department	

Abstract

With the fast pace of global urbanization and the wide use of smart devices and GPS sets, there are massive *human-generated spatial-temporal data* in the urban environment, e.g., the trajectory data of the human-operated vehicles in the ride-hailing service or traditional taxi service. *Human-generated spatial-temporal data* can represent human behavior intrinsically, which enable us to understand human behavior in a data-driven fashion. Understanding human behavior can benefit people in many aspects. For example, understanding the decision-making process of taxi drivers can help them improve their operation efficiency, making sense of the behavior of the urban commuters can help urban planners better design the urban transportation system. In this dissertation, we propose and develop several novel machine learning techniques to help people make sense of the massive *human-generated spatial-temporal data* from urban environment.

The ultimate goal of this dissertation is to help bridge the gap between real-world applications and laboratory researches. In particular, we try to deliver appropriate machine learning and statistical analysis solution frameworks for making sense of *human-generated spatial-temporal data* from urban environment in the following four aspects.

1. Human Learning Curve Dissection. Understanding human learning curve can reward people in different ways, e.g., help the new learners improve their performance faster. The temporal dynamics of human behavior can reflect the

learning curve of human beings, which makes it possible for us to understand human learning curve from their behavior data. In this topic, we propose data-driven approaches to understand *what* and *how* human agents learn over time.

2. Human Behavior Explanation. Recent research demonstrates successes in learning human decision-making strategies from their behavior data using deep neural networks (DNNs). Such DNN-based models are "black box" models in nature, making it hard to explain *what* knowledge the models have learned from human. To solve this problem, in this topic of the dissertation, we propose an explainable imitation learning framework.

3. Human Mobility Signature Identification. Identifying human agents from their behaviors is a significant task, which is helpful in many real-world applications, e.g., identifying drivers in ride-hailing service. In this topic, we propose the human mobility signature identification solution to identify human agents from their mobility data.

4. Smart Cloud Commuting System. The significant achievements on developing autonomous vehicles stimulate us to envision the future transportation system with shared autonomous vehicles. In this topic, we employ the real-world demand and service data in current taxi system to study the feasibility of the future smart cloud commuting system.

Acknowledgements

I sincerely thank my adviser, Prof. Yanhua Li, for multiple fruitful and inspiring discussions, his wise comments to numerous preliminary versions of my publications, his energy and sleepless nights. I thank him for pushing on me and encouraging me to complete my PhD dissertation. He played a very important role in every step of my PhD studies from being accepted at WPI in 2016, through every milestone, publication, internship, etc. He was both supportive and encouraging while supervising my research from a vague idea to a polished publication at a top-level venue. Without his help I would not be where I am now in my professional and personal development.

I would like to thank my committee members, Prof. Elke A. Rundensteiner, Prof. Mohamed Y. Eltabakh, and Prof. Xun Zhou for careful reading of my publications and this manuscript and improving them by their numerous suggestions. I also thank all members of DSRG group for listening to my preliminary and final talks and enriching them by bright ideas and critical comments.

I thank Prof. Craig E Wills and Computer Science department of Worcester Polytechnic Institute, for supporting me as a teaching assistant.

I would like to thank my collaborators, Prof. Xun Zhou, Prof. Rui Song, Prof. Zhenming Liu, Prof. Zhi-Li Zhang, Prof. Hui Lu, Dr. Jun Luo, Dr. Jie Bao, Dr. Yu Zheng, Huimin Ren, for their brilliant ideas, critical comments and

productive collaborations. They helped me with elaborating and polishing explanation of my publications. It was a pleasure to work with them, and I am looking forward to future collaborations.

I also thank members of our research group, especially Guojun Wu, Huimin Ren, Xin Zhang, Yingxue Zhang, Xin Dai, Dr. John Boaz Lee, Biao Yin, Dr. Zhongfang Zhuang, Dr. Xiao Qin, Dr. Xinyue Liu, Ermal Toto, Tom Hartvigsen for helping me improve my research work during our disccusions and DSRG group meetings.

I am endlessly grateful to my friends, Yun Qin, Dr. Yangyang Fan, Tianhao Guo, Han Liu, Rongan Jin, Dr. Mi Feng, Yingnan Liu, Jianjun Luo, for their always supports during my Ph.D. studying.

Last but not least, I would like to thank my family, Jianjun Pan, Yanhua Pan, Mengting Pan, Yuelian Zhou, Dr. Weijun Gui for their love and supports all the way. This dissertation is devoted to them.

Publications

Publications Contributing to this Dissertation

In this section, I briefly summarize the publications pertinent to the topics covered in this dissertation.

Topic 1: Human Learning Curve Dissection

Topic 1 of this dissertation deliberates on dissecting the learning curve of human agent via data-driven approaches. The publications in this topic are listes below.

1. [SDM'19] (**Best Applied Data Science Paper award.**) **Menghai Pan**, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, and Jun Luo. *Dissecting the Learning Curve of Taxi Drivers: A Data-Driven Approach.* SIAM International Conference on Data Mining (SDM19), Hyatt Regency Calgary — Calgary, Alberta, Canada, May 2 - 4, 2019.

2. [TIST] **Menghai Pan**, Weixiao Huang, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, Zhihong Tian, and Jun Luo. *DHPA: Dynamic Human Preference Analytics Framework - A Case Study on Taxi Drivers' Learning Curve Analysis*. ACM Transactions on Intelligent Systems and Technology (TIST).

3. [SIGSPATIAL'20] **Menghai Pan**, Weixiao Huang, Yanhua Li, Xun Zhou, Zhenming Liu, Jie Bao, Yu Zheng, and Jun Luo. *Is Reinforcement Learning the Choice of Human Learners? A Case Study of Taxi Drivers.* 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Virtual Conference, Tuesday November 3 - Friday November 6, 2020, (22.1% Acceptance Ratio=33/149).

Topic 2: Human Behavior Explanation

Topic 2 of this dissertation focuses on explaining human behavior from non-linear model.

4. [KDD'20] **Menghai Pan**, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. *xGAIL: Explainable Generative Adversarial Imitation Learning for Explainable Human Decision Analysis.* the 26th SIGKDD conference on Knowledge Discovery and Data Mining, San Diego, CA, August 23 - 27, 2020, (16.8% Acceptance Ratio=216/1279).

Topic 3: Human Mobility Signature Identification

Topic 3 of this dissertation tareting on solving the problem of human mobility signation identification.

5. [KDD'20] Huimin Ren, **Menghai Pan (co-first author)**, Yanhua Li, Xun Zhou, and Jun Luo. *ST-SiameseNet: Spatio-Temporal Siamese Networks for Human Mobility Signature Identification.* the 26th SIGKDD conference on Knowledge Discovery and Data Mining, San Diego, CA, August 23 - 27, 2020, (16.8% Acceptance Ratio=216/1279).

Topic 4: Smart Cloud Commuting System

Topic 4 of this dissertation envisions and studies the smart cloud commuting system.

6. [TBD] **Menghai Pan**, Yanhua Li, Zhi-Li Zhang, and Jun Luo. *SCCS: Smart Cloud Commuting System with Shared Autonomous Vehicles*. IEEE Transactions on Big Data (TBD).

7. [Urbcomp'18] **Menghai Pan**, Taihui Li, Yanhua Li, Zhi-Li Zhang, and Jun Luo. *Smart Cloud Commuting with Shared Autonomous Vehicles: A First Feasibility Study.* Urbcomp'18: The 7th International Workshop on Urban Computing, August 20, 2018, London, UK.

8. [SmartCity'17] **Menghai Pan**, Guanxiong Liu, Yanhua Li, Zhi-Li Zhang, and Jun Luo. *Modeling Urban Trip Demands in Cloud-Commuting System: A Holistic Approach.* SmartCity'17: The 3rd IEEE INFOCOM Workshop on Smart Cities and Urban Computing, May 1-4 2017, Atlanta, GA, USA.

Contents

List of Figures

List of Tables

1

Overview

1.1 Introduction

According to a report[1] published by the United Nations in 2018, 55% of the world's population are urban in 2018. It is said that the proportion is expected to increase to 68% by 2050[1]. Also, the smart devices and GPS sets are widely used in the urban area, for example, smart phones, smart watches and the traditional GPS sets. Stimulated by the large urban population and the wide use of the devices, there exist massive human-generated spatial-temporal data in the urban environment. For example, the trajectory data of the vehicles in ride-sharing services, food delivery services, taxi services, etc. Given the large amount of human-generated spatial-temporal data, a question people ask is what we can extract from it? In this dissertation, we try to answer this question from the following 4 different perspectives.

1) Human Learning Curve Dissection. Many real world human behaviors can be modeled and characterized as sequential decision making processes, such as taxi driver's choices of working regions and times. Each driver possesses unique preferences on the sequential choices over time and improves their working efficiency. Understanding the

dynamics of such preferences helps accelerate the learning process of taxi drivers. Prior works on taxi operation management mostly focus on finding optimal driving strategies or routes, lacking in-depth analysis on what the drivers learned during the process and how they affect the performance of the driver. In this topic, we make the first attempt to establish Dynamic Human Preference Analytics (DHPA) [2, 3]. We inversely learn the taxi drivers' preferences from data and characterize the dynamics of such preferences over time. We extract two types of features, i.e., profile features and habit features, to model the decision space of drivers. Then through inverse reinforcement learning we learn the preferences of drivers with respect to these features. The results illustrate that self-improving drivers tend to keep adjusting their preferences to habit features to increase their earning efficiency, while keeping the preferences to profile features invariant. On the other hand, experienced drivers have stable preferences over time.

Going beyond figuring out what human agents learn over time, we also study how human beings learn. Learning to make optimal decisions is a common yet complicated task. While computer agents can learn to make decisions by running reinforcement learning (RL), it remains unclear how human beings learn. In another work in this topic, we perform the first data-driven case study on taxi drivers to validate whether humans mimic RL to learn[4]. We categorize drivers into three groups based on their performance trends and analyze the correlations between human drivers and agents trained using RL. We discover that drivers that become more efficient at earning over time exhibit similar learning patterns to those of agents, whereas drivers that become less efficient tend to do the opposite. Our study (1) provides evidence that some human drivers do adapt RL when learning, (2) enhances the deep understanding of taxi drivers' learning strategies, (3) offers a guideline for taxi drivers to improve their earnings, and (4) develops a generic analytical framework to study and validate human learning strategies.

2) Human Behavior Explanation. To make daily decisions, human agents devise

their own "strategies" governing their mobility dynamics (e.g., taxi drivers have preferred working regions and times, and urban commuters have preferred routes and transit modes). Recent research such as generative adversarial imitation learning (GAIL) demonstrates successes in learning human decision-making strategies from their behavior data using deep neural networks (DNNs), which can accurately mimic how humans behave in various scenarios, e.g., playing video games, etc. However, such DNN-based models are "black box" models in nature, making it hard to explain *what* knowledge the models have learned from human, and *how* the models make such decisions, which was not addressed in the literature of imitation learning. In this topic of the dissertation, we address this research gap by proposing xGAIL, the first explainable generative adversarial imitation learning framework [5]. The proposed xGAIL framework consists of two novel components, including Spatial Activation Maximization (SpatialAM) and Spatial Randomized Input Sampling Explanation (SpatialRISE), to extract both global and local knowledge from a well-trained GAIL model that explains how a human agent makes decisions. Especially, we take taxi drivers' passenger-seeking strategy as an example to validate the effectiveness of the proposed xGAIL framework. Our analysis on a large-scale real-world taxi trajectory data shows promising results from two aspects: i) global explainable knowledge of what nearby traffic condition impels a taxi driver to choose a particular direction to find the next passenger, and ii) local explainable knowledge of what key (sometimes hidden) factors a taxi driver considers when making a particular decision.

3) Human Mobility Signature Identification. Given the historical movement trajectories of a set of individual human agents (e.g., pedestrians, taxi drivers) and a set of new trajectories claimed to be generated by a specific agent, the Human Mobility Signature Identification (HuMID) task aims at validating if the incoming trajectories were indeed generated by the claimed agent. This problem is important in many real-world applications such as driver verification in ride-sharing services, risk analysis for auto insurance

companies, and criminal identification. Prior work on identifying human mobility behaviors requires additional data from other sources besides the trajectories, e.g., sensor readings in the vehicle for driving behavior identification. However, these data might not be universally available and is costly to obtain. To deal with this challenge, in this topic of the dissertation, we make the first attempt to match identities of human agents only from the observed location trajectory data by proposing a novel and efficient framework named Spatio-temporal Siamese Networks (ST-SiameseNet) [6]. For each human agent, we extract a set of profile and online features from his/her trajectories. We train ST-SiameseNet to predict the mobility signature similarity between each pair of agents, where each agent is represented by his/her trajectories and the extracted features. Experimental results on a real-world taxi trajectory dataset show that our proposed ST-SiamesNet significantly outperforms the state-of-the-art techniques.

4) Smart Cloud Commuting System. Emergence of autonomous vehicles (AVs) offers the potential to fundamentally transform the way how urban transport systems be designed and deployed, and alter the way we view private car ownership. In this topic, we advocate a forward-looking, ambitious and disruptive *smart cloud commuting system* (SCCS) for future smart cities based on shared AVs. Employing giant pools of AVs of varying sizes, SCCS seeks to supplant and integrate various modes of transport – most of personal vehicles, low ridership public buses, and taxis used in today's private and public transport systems – in a unified, on-demand fashion, and provides passengers with a fast, convenient, and low cost transport service for their *daily commuting* needs. To explore feasibility and efficiency gains of the proposed SCCS, we model SCCS as a queueing system with passengers' trip demands (as jobs) being served by the AVs (as servers). Using a 1-year real taxi trip dataset from Shenzhen China, we quantify (i) how design choices, such as the numbers of depots and AVs, affect the passenger waiting time and vehicle utilization; and (ii) how much efficiency gains (i.e., reducing the number of service vehicles,

and improving the vehicle utilization) can be obtained by SCCS comparing to the current taxi system. Our results demonstrate that the proposed SCCS framework can serve the trip demands with 22% fewer vehicles and 37% more vehicle utilization, which shed lights on the design feasibility of future smart transportation systems.

1.2 Dissertation Organization

The remainder of this dissertation is organized as follows. In **Chapter 2**, we introduce our work about Human Learning Curve Dissection, containing the projects of studying what human agents learn [2, 3], and how they learn [4] over time, via data-driven approaches. And in **Chapter 3**, our proposed explainable framework for Human Behavior Explanationis presented [5]. Following in **Chapter 4**, a Spatial-temporal Siamese network is designed for Human Mobility Signature Identification[6]. In **Chapter 5**, we envision the Smart Cloud Commuting System for future urban transportation with shared autonomous vehicles, and study the feasibility with the real-world taxi demands data [7, 8, 9]. A conclusion of this dissertation is drawn in **Chapter 6**.

For consistency and to allow the reader to easily jump from one topic to another, we present the detailed research for each topic in a largely self-contained set of sections following this particular pattern:

- Overview: Includes the introduction, motivation and problem definition of the project;
- Related work: Reviews the state-of-the-art related works, and clarifies the innovation of our proposed framework.
- Methodology: Introduces the details of the proposed method, including the mathematical foundation, machine learning model, and specific design for each particular

problem.

- Evaluation: Presents the experimental evaluation of the proposed method, containing the comparison with the state-of-the-art approaches, and the ground-truth (if available).

2

Human Learning Curve Dissection

2.1 Dynamic Human Preference Analytics Framework

2.1.1 Overview

2.1.1.1 Introduction

Taxi service is a vital part of the transportation systems in large cities. Improving taxi operation efficiency is a crucial urban management problem, as it helps improve the transportation efficiency of the city and at the same time improves the income of taxi drivers. In the same city, taxi operation efficiency might differ significantly. Fig. 2.1 shows the earning efficiency (total amount earned normalized by total working time) of different taxi drivers in Shenzhen, China. The top drivers earn 3 to 4 times more money than the bottom drivers.

A major cause of such difference is the difference in working experiences. Fig. 2.2 shows the growth of earning efficiency of new drivers over years[1]. From March 2014 to December 2016, the new drivers became more experienced and had much higher earning

[1]The dataset we have contains the records in March and November of 2014 and July to December of 2016.

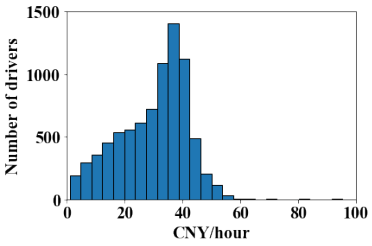


Figure 2.1: The distribution of earning efficiency in July 2016

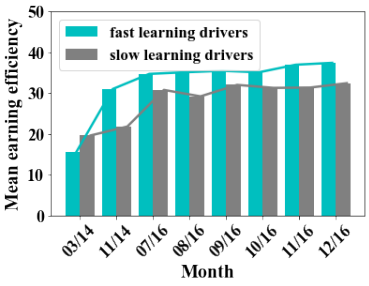


Figure 2.2: The mean earning efficiency of new drivers over months

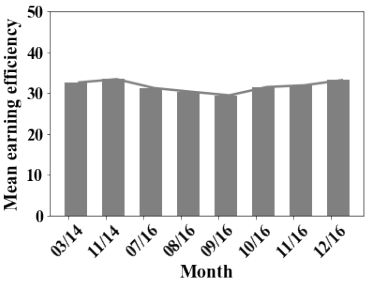


Figure 2.3: The mean earning efficiency of all drivers over months

efficiency. During the same time as shown in Fig. 2.3, there is no obvious change to the local economy or market, since the average earning efficiency of all the drivers are pretty stable. This shows that drivers are trying to improve their own strategies of looking for passengers based on their increasing knowledge of the city.

However, each driver might learn different knowledge during the learning process, which in turn developed different preferences, when making decisions. For instance, some drivers tend to look for passengers around regions near their homes, and some others might prefer to take passengers from city hubs, e.g., train stations, airport. These preferences might be unique to individual drivers and ultimately lead to differences in earning efficiency. Fig. 2.2 shows that the ”smart” drivers (in blue) improve their earning efficiency faster than “average” drivers and reach a higher level of earning efficiency eventually. Finding what adaptation strategies these “smart” drivers carry could help us understand the learning process of successful drivers and therefore help new drivers to become more successful.

The passenger-seeking behavior of taxi drivers can be modeled as a Markov Decision Process (MDP). Prior work on taxi operation management focused on recommending the optimal policy or routes to maximize the chance of finding passengers or making profit [10, 11, 12, 13]. However, these works only studied how to find the “best” strategies based on data, rather than fundamentally understanding how the drivers learned these

strategies over time.

In this work, we make the first attempt to establish Dynamic Human Preference Analytics (DHPA). We inversely learn the taxi drivers' decision-making preferences, which lead to their choices while looking for passengers. We also study how these preferences evolve over time and how they help improve the earning efficiency. The results shed lights on "how" the successful drivers became successful, and suggests "smarter" actionable strategies to improve taxi drivers' performances. Our **main contributions** are as follows:

(1) We are the first to employ Inverse Reinforcement Learning to infer the taxi drivers' preferences based on a Markov Decision Process model.

(2) We extract various kinds of interpretable features to represent the potential factors that affect the decisions of taxi drivers.

(3) We infer and analyze the preference dynamics of 3 groups of taxi drivers, i.e, the self-improving drivers, the stabilized drivers and the exploring drivers.

(4) We analyze the preference trend of different groups of taxi drivers.

(5) We conduct experiments with taxi trajectories from more than $17k$ drivers over different time spans. The results verify that each driver has unique preferences to various profile and habit features. The preferences to profile features tend to be stable over time, and the preferences on habit features change over time, which leads to higher earning efficiency.

2.1.1.2 Motivation

It is a common perception that new drivers gradually learn how to make smart choices as time goes and can improve their working efficiency over time. We verify this perception through data analysis. In Fig. 2.2, the average earning efficiency of new drivers who

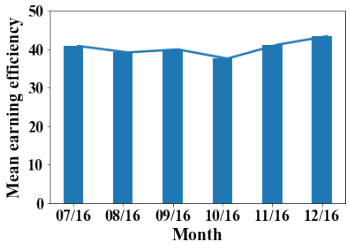


Figure 2.4: The mean earning efficiency of experienced drivers over months in 2016

joined in March 2014 increased by up to 100% in 2 years, while in Fig. 2.4, the same measure of experienced drivers in 2016 did not change much. This can be explained by the fact that experienced drivers have learned enough knowledge to make nearly-optimal decisions.

We further noticed that drivers have very different learning curves, which affects ultimately how much earning improvements they can achieve. As previously mentioned, in Fig. 2.2, the two colors represent two sub-groups of new drivers who joined in March 2014. One group (in blue) are those who became "top" drivers after 2 years with higher earning efficiency, and the other (gray) are the rest of the drivers. Apparently the former had learned more useful knowledge that contributed to their earning improvement.

Little is known about what specific knowledge the drivers learned, and which pieces are contributing the most to the earning improvement. Answering these questions would potentially guide and train new drivers to become a quick learner.

We consider such "knowledge" as a series of preferences of a driver when making each decision, such as "how frequent to visit the train station", "how far away from home to go when seeking passengers". Specifically, we extract features from the data to represent such decisions a taxi driver might face while working. To achieve the aforementioned goal, in this study we aim to answer two questions: (1) how to recover the preferences of taxi drivers when making these choices, and (2) how these preferences change over time for different groups of drivers.

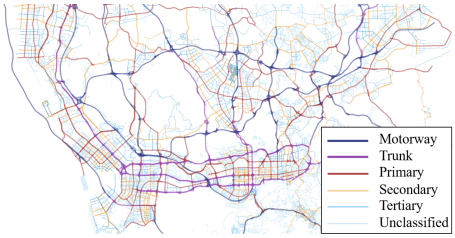


Figure 2.5: Shenzhen road map

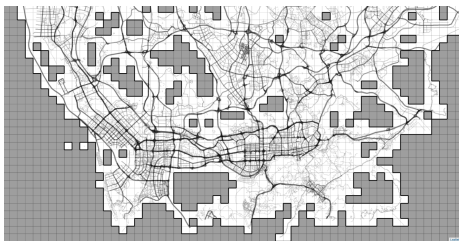


Figure 2.6: Map gridding

Problem Definition. In a time interval T_0 i.e., 1 month, given a taxi driver's trajectory data $\tilde{\mathcal{T}}$, and k environmental features $[f_0, f_1, ..., f_k]$, that influence drivers' decision-making process over time, we aim to learn the driver's preference $\theta = [\theta_0, \theta_1, ..., \theta_k]$, i.e., weights to features when the driver makes decisions. Secondly, for a long time horizon, with multiple time intervals $[T_0, T_1, ..., T_m]$, we analyze the evolution pattern of the driver's preferences over time.

2.1.1.3 Data Description

Our analytical framework takes two urban data sources as input, including (1) taxi trajectory data and (2) road map data. For consistency, both datasets are collected in Shenzhen, China in 2014 and 2016.

The **taxi trajectory data** contain GPS records collected from taxis in Shenzhen, China during March and November in 2014, and July to December in 2016. There were in total 17, 877 taxis equipped with GPS sets, where each GPS set generates a GPS point every 40 seconds on average. Overall, a total of 51,485,760 GPS records are collected on each day, and each record contains five key data fields, including taxi ID, time stamp, passenger indicator, latitude and longitude. The passenger indicator field is a binary value, indicating if a passenger is aboard or not.

The **Road map data** of Shenzhen covers the area defined between $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. The data is from OpenStreetMap [14] and

has 21,000 roads of six levels. Fig.2.5 shows the road map in Shenzhen.

2.1.2 Related Work

Taxi operating strategies (e.g., dispatching, passenger seeking), and driver behavior analysis have been extensively studied in recent years due to the emergence of the ride-sharing business model and urban intelligence. However, to the best of our knowledge, *we make the first attempt to employ inverse reinforcement learning to analyze the preference dynamics of taxi drivers*. Related works to our study are summarized below.

Urban Computing, transportation and geo-informatics are general research areas which integrate urban sensing, data management and data analytic together as a unified process to explore, analyze and solve crucial problems related to people's everyday life [13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. In particular, a number of work study taxi operation management, such as dispatching [27, 28] and passenger seeking [10, 11, 29], aiming at finding an optimal actionable solution to improve the performance/revenue of individual taxi drivers or the entire fleet.

[12] solved the passenger seeking problem by giving direction recommendations to drivers. However, all of these works focus on finding "what" are the best driving strategies (as an optimization problem), rather than finding "why" and "how" good drivers make these decisions. By contrast, our work focuses on analyzing the evolving preferences of good drivers, that helped them to make better and more profitable decisions.

Inverse Reinforcement Learning(IRL) aims to recover the reward function under which the expert's policy is optimal from the observed trajectories of an expert. There are various of IRL methods, for example, [30] found that there are a class of reward functions that can lead to the same optimal policy, and it proposed a strategy to select a reward function. However, this method is not proper to analyze human behaviors because it uses the deterministic policy in the Markov Decision Process while human decisions tend to

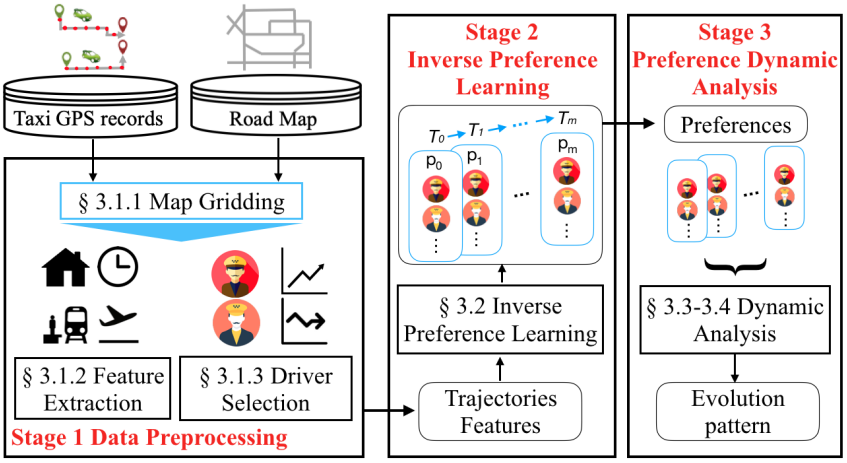


Figure 2.7: DHPA Framework

be non-deterministic. And [31] proposed a IRL method by maximizing the entropy of the distribution on state-actions under the learned policy. Although this method can employ stochastic policy, the computation efficiency is not friendly to large scale state space, and it requires the information of the model. In this work, we employ Relative Entropy IRL [32] which is model-free and employs softmax policy. Our work, compared to the above related work, is the first to apply IRL to study the evolving driving preferences of taxi drivers.

2.1.3 Methodology

Fig. 3.2 outlines our Dynamics Human Preference Analytics (DHPA) framework, which takes two sources of urban data as inputs and contains three key analytical stages: (1) data preprocessing, (2) inverse preference learning and (3) preference dynamic analysis.

2.1.3.1 Data Preprocessing

Map and Time Quantization. We use a standard quantization trick to reduce the size of the location space. Specifically, we divide the study area into equally-sized grid cells with a given side-length s in latitude and longitude. Our method has two advantages: (i) we

have the flexibility to adjust the side-length to achieve different granularities, and (ii) it is easy to implement and highly scalable in practice [15, 33]. Fig. 2.6 shows the actual grid in Shenzhen, China with a side-length $l = 0.01^\circ$ in latitude and longitude. Eliminating cells in the ocean, those unreachable from the city, and other irrelevant cells gives a total of 1158 valid cells.

We divide each day into five-minute intervals for a total of 288 intervals per day. A spatio-temporal region r is a pair of a grid cell s and a time interval t . The trajectories of drivers then can be mapped to sequences of spatio-temporal regions.

Feature Extraction. Taxi drivers make hundreds of decisions throughout their work shifts (e.g., where to find the next passenger, and when to start and finish working in a day). When making a decision, they instinctively evaluate multiple factors (i.e., features) related to their current states and the environment (i.e., the current *spatio-temporal region*). For example, after dropping off a passenger, a driver may choose to go back to an area that she is more familiar with, or a nearby transport station, e.g., airport, train station. Here, we extract key features the drivers use to make their decisions.

Note in our framework, each feature is defined as a numeric characteristic of a specific spatio-temporal region, which may or may not change from driver to driver. For example, let f_r represent the average number of taxi pickups in history in location s during time slot t. Apparently the value of feature f_r is the same for every driver. However, another feature g_r at r could be the distance from s to the home of the driver. The value of this feature varies from driver to driver, depending on their home locations. However, it does not change over time. The features we extract can be roughly categorized by *profile features* and *habit features*, as detailed below.

Profile Features. Each driver has unique personal (or profile) characteristics, such as home location, daily working schedule (time duration), and preferred geographic area. For each spatio-temporal region, we build the profile features. Here, we extract 4 profile

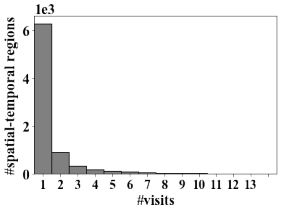


Figure 2.8: *P1:* Number of visits

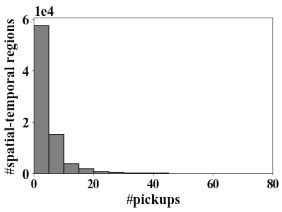


Figure 2.9: *H1:* Number of pickups

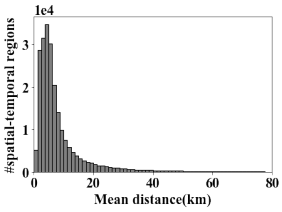


Figure 2.10: *H2:* Mean trip distance

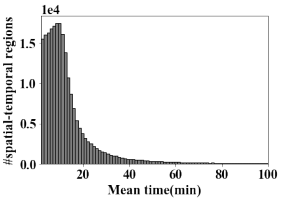


Figure 2.11: *H3:* Mean trip time

features:

P1: Visitation Frequency. This group of features represents the numbers of daily visits to different regions of a driver as extracted from the historical data. Fig. 2.8 shows the distribution of visitation frequency to different regions of an arbitrarily chosen driver. Here, visitation frequencies vary significantly across regions.

P2: Distance to Home. Each taxi driver has a home location, which can be extracted from their GPS records. This feature characterizes the distance (in miles on the road network) from the current location to the driver's home location. Different drivers may have different preferences in working close to their homes or not.

P3 & P4: Time from Start & Time to Finish. Taxi drivers typically work according to consistent starting and finishing times. We construct two features to characterize the differences of the current time from the regular starting and finishing time.

Habit Features. These represent the habits of the drivers, which are typically governed by experience (e.g., remaining near the train station instead of traveling around to find passengers). We extract 6 habit features. *H1: Number of pickups.* This feature characterizes the demands in a cell during a time interval, and is extracted and estimated using the historical trajectories from all drivers. The distribution on the numbers of pickups is shown in Fig. 2.9.

H2 & H3: Average Trip Distance & Time. These features represent the average distance and the travel time of passenger trips starting from a particular spatio-temporal region.

Different spatio-temporal regions can have different expected trip distances and travel time. For example, the passengers picked up near the airport probably have longer trip distances than the passengers picked up near the train station since the airport is farther away from the downtown area than the train station. A driver's preferences to these features characterize to what extent the driver prefers long versus short distance trips and how well the driver learns the knowledge on the lengths of trips over the spatio-temporal regions.

The distribution of these features across spatio-temporal regions are showed in Fig. 2.10 and Fig. 2.11, respectively.

H4: Traffic Condition. This feature captures the average traffic condition based on the time spent by a driver in each spatio-temporal region. A long travel time implies traffic congestion. The preference of drivers over this feature represents how much drivers would like avoid the traffic.

H5 & H6: Distance to Train Station & Airport. These features reflect the distances from the current cell to Shenzhen train station and airport, respectively.

Driver Selection. Different drivers have different earning efficiencies as shown in Fig. 2.1. Below, we describe the criteria we use to select drivers.

We estimate the earning efficiency of each driver in different time periods from their historical data. The estimated earnings E of a driver in the whole sampling span (e.g., per month) is calculated from the distance d_o that the taxi is occupied with passenger. The factors we take into consideration include the taxi fare in Shenzhen in 2014 and 2016 and the expense for the gas. And the taxi fare is 11 CNY for the first 2 km, and the charges for each additional kilometer is 2.4 CNY. The estimated expense for gas is 0.5 CNY per

kilometer. The calculation of E is as following.

$$E = \begin{cases} 11 - 0.5 * d_o & if\ d_o < 2 \\ 11 + (d_o - 2) * 2.4 - 0.5 * d_o & else. \end{cases} \tag{2.1}$$

Note that our model is easy to extend to other definitions of "earnings". Given the data we have, and without losing much accuracy of calculated earnings in terms of representing the driver's profits, we employ Equation 2.1 to estimate the earning of each driver.

The earning efficiency r_e is defined as the average per hour income (i.e., in eq.2.17).

$$r_e = \frac{E}{t_w}, \tag{2.2}$$

where E is the income in the whole sampling time span, span (e.g., per month), and t_w represents the driver's working time.

Driver selection criterion: We select drivers with the highest earnings, because the preference learning algorithms require the input data to be generated by the converged policy (see more details in Sec 2.1.3.2). We note that drivers with high earning efficiencies are likely the most experienced (i.e., they use converged policies to make decisions).

2.1.3.2 Inverse Preference Learning

This section explains our inverse learning algorithm for extracting drivers' decision-making process. We use a Markov Decision Process (MDP) to model drivers' sequential decision-making and relative entropy inverse reinforcement learning (REIRL) to learn their decision-making preferences.

Markov Decision Process.

A Markov Decision Process(MDP) [34] is defined by a 5-tuple $\langle S, A, T, \gamma, \mu_0, R \rangle$ so that

- S is a finite set of states and A is a finite set of actions,
- T is the probabilistic transition function with $T(s'|s, a)$ as the probability of arriving at state s' by executing action a at state s,
- $\gamma \in (0, 1]$ is the discount factor[1],
- $\mu_0 : S \to [0, 1]$ is the initial distribution, and
- $R : S \times A \to$ is the reward function.

A randomized, memoryless policy is a function that specifies a probability distribution on the action to be executed in each state, defined as $\pi : S \times A \to [0, 1]$.

We use $\tau = [(s_0, a_0), (s_1, a_1), \ldots, (s_L, a_L)]$ to denote a trajectory generated by MDP. Here L is the length of trajectory. We model the decision-making process of taxi drivers with MDP as follow:

- State: a spatio-temporal region, specified by a geographical cell and a time slot.
- Action: traveling from the current cell to one of the eight neighboring cells, or staying in the same cell.
- Reward: the inner product of the preference function (as a vector) θ and the feature vector $\mathbf{f}$ on each state-action pair.

Note that in the MDP settings, the reward of each state-action pair is the inner product of the preference function and the feature vector. The preferences are the weights of each feature. The driver aims to maximize the accumulated reward when making decisions. To interpret each of the preferences recovered, we can consider two factors, i.e., the sign and the magnitude of the preference (weights of each feature). A positive preference means

[1]Without loss of generality, we assume $\gamma = 1$ in this work, and it is straightforward to generalize our results to $\gamma \neq 1$.

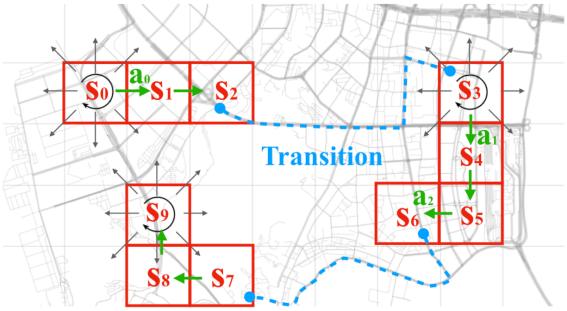


Figure 2.12: MDP of taxi driver's decision making process

that the driver prefers to go to the regions where the corresponding feature value is higher than other locations, and larger magnitude can imply that the driver pays more attention to the corresponding feature, and vise versa for negative sign and smaller magnitude. And in this work, we design two categories of features, i.e., the profile features and the habit features. The intuition is that the preferences to some features can be time-invariant because these features are closely related to the drivers' profiles, e.g., home location, working schedule, etc., which usually do not change. And the preferences to some other features can be time-variant because these features are related to the habits of drivers, e.g., number of pickups, distance to train stations, etc.. The preferences to the habit features are considered as the habits of the drivers. The drivers can learn to change their habits, i.e., preferences to habit features, to improve their earning efficiencies.

Fig. 2.12 shows an example of trajectory in the MDP: a driver starts in state s_0 with the taxi idle, and takes the action a_0 to travel to the neighboring cell S_1. After two steps, the driver reaches state S_2, where she meets a passenger. The destination of the new trip is cell S_3. The trip with the passenger is a transition in the MDP from S_2 to S_3.

Inverse Preference Learning. Given the observed trajectory set $\tilde{\mathfrak{T}}$ of a driver and the features extracted on each state-action pair (s, a), the inverse preference learning stage aims to recover a reward function (i.e., preference vector θ) under which the observed trajectories have the highest likelihood to be generated [30]. Various inverse reinforcement

learning approaches, e.g., Apprenticeship learning [35], Maximum Entropy IRL [31], Bayesian IRL [36] and Relative Entropy IRL [32], have been proposed in the literature.

Our problem possesses two salient characteristics: (i) the state space is large. We have 1158 cells and 288 time intervals. Therefore, the total number of states is $1158 \times 288 \approx$ 330k, and (ii) the transition probability is hard to measure because in part of the large state space issue.

Therefore, we adopt a model-free IRL approach, namely, relative entropy IRL [32] that does not require estimating transition probabilities and is more scalable than other alternatives.

The optimization problem. Let $\mathcal{T}$ denote the set of all possible trajectories of the driver decision-making MDP. For any $\tau \in \mathcal{T}$, denote $P(\tau)$ as the trajectory distribution induced by the taxi driver's ground-truth policy, and $Q(\tau)$ as the trajectory distribution induced by a base policy. The Relative Entropy between $P(\tau)$ and $Q(\tau)$ (in eq.2.3) characterizes as the distribution difference between $P(\tau)$ and $Q(\tau)$.

$$H(P\|Q) = \sum_{\tau \in \mathcal{T}} P(\tau) \ln \frac{P(\tau)}{Q(\tau)}. \tag{2.3}$$

The driver's trajectory distribution is governed by the driver's preference θ, thus is a function of θ, i.e., $P(\tau|\theta)$. The relative entropy IRL aims to find a reward function θ, that minimizes the relative entropy in eq.2.3 and matches the trajectory distribution to the

observed trajectory data. **P1: Relative Entropy IRL Problem:**

$$\min_{\theta}: \quad H(P(\theta)\|Q) = \sum_{\tau\in\mathcal{T}} P(\tau|\theta)\ln\frac{P(\tau|\theta)}{Q(\tau)}, \tag{2.4}$$

$$\text{s.t.:} \quad |\sum_{\tau\in\mathcal{T}} P(\tau|\theta)f_i^\tau - \hat{f}_i| \le \epsilon_i, \forall i \in \{1,...,k\}, \tag{2.5}$$

$$\sum_{\tau\in\mathcal{T}} P(\tau|\theta) = 1, \tag{2.6}$$

$$P(\tau|\theta) \ge 0, \qquad \forall\tau\in\mathcal{T}, \tag{2.7}$$

where i is the feature index, and f_i^τ is the i's feature count in trajectory τ, and $\hat{f}_i = \sum_{\tau\in\tilde{\mathcal{T}}} f_i^\tau/|\tilde{\mathcal{T}}|$ is the feature expectation over all observed trajectories in $\tilde{\mathcal{T}}$. ϵ_i is a confidence interval parameter, which can be determined by the sample complexity (the number of trajectories) via applying a Hoeffding's bound. The constraint eq.2.5 ensures that the recovered policy matches the observed data. The constraints eq.2.6–eq.2.7 guarantees the $P(\tau|\theta)$'s are non-negative probabilities, thus sum up to one.

Solving P1. The function $Q(\tau)$ and $P(\tau|\theta)$ can be decomposed as

$$Q(\tau) = T(\tau)U(\tau) \text{ and } P(\tau|\theta) = T(\tau)V(\tau|\theta),$$

where $T(\tau) = \mu_0(s_0)\prod_{t=1}^{K} T(s_t|s_{t-1},a_{t-1})$ is the joint probability of the state transitions in τ, for $\tau = [(s_0,a_0),(s_1,a_1),\cdots,(s_K,a_K)]$, with $\mu_0(s_0)$ as the initial state distribution. $U(\tau)$ (resp. $V(\tau|\theta)$) is the joint probability of the actions conditioned on the states in τ under driver's policy π_θ (resp. a base policy π_q). As a result, eq.2.4 can be written as follows.

$$H(P(\theta)\|Q) = \sum_{\tau\in\mathcal{T}} P(\tau|\theta)\ln\frac{V(\tau|\theta)}{U(\tau)}. \tag{2.8}$$

Moreover, when $\pi_q(a|s)$ at each state s is uniform distribution, e.g., $\pi_q(a|s) = 1/|A_s|$, with A_s as the set of actions at state s, the problem **P1** is equivalent to maximizing the causal entropy of $P(\tau|\theta)$, i.e., $\sum_{\tau \in \mathcal{T}} P(\tau|\theta) \ln V(\tau|\theta)$, while matching $P(\tau|\theta)$ to the observed data [37]. Following the similar process outlined in [32], **P1** can be solved by a gradient descent approach, with the step-wise updating gradient as follows.

$$\nabla g(\theta) = \hat{f}_i - \frac{\sum_{\tau \in \mathcal{T}^\pi} \frac{U(\tau)}{\pi(\tau)} \exp(\sum_{j=1}^{k} \theta_i f_j^\tau)}{\sum_{\tau \in \mathcal{T}^\pi} \frac{U(\tau)}{\pi(\tau)} \exp(\sum_{j=1}^{k} \theta_i)} - \alpha_i \epsilon_i, \tag{2.9}$$

where $\alpha_i = 1$ if $\theta_i \leq 0$ and $\alpha_i = -1$ otherwise. $\mathcal{T}^\pi$ is a set of trajectories sampled from $\tilde{\mathcal{T}}$ by an executing a given policy π. $U(\tau)$ is the joint probability of taking actions conditioned on the states in a observed trajectory τ, induced by uniform policy $\pi_q(a|s) = 1/|A_s|$.

See Algorithm 1 for the IRL algorithm.

Input: Demonstrated trajectories $\tilde{\mathcal{T}}$, feature matrix F, threshold vector ϵ, learning rate α, and executing policy π.
Output: Preference vector θ.
1: Randomly initialize preference vector θ.
2: Sample a set of trajectories. $\mathcal{T}^\pi$ using π.
3: Calculate feature expectation vector $\hat{f}$.
4: **repeat**
5: Calculate each feature count f_i^τ.
6: Calculate gradient $\nabla g(\theta)$ using Eq 2.9.
7: Update $\theta \leftarrow \theta + \alpha \nabla g(\theta)$.
8: **until** $\nabla g(\theta) < \epsilon$.

Algorithm 1: Relative Entropy IRL

2.1.3.3 Preference Dynamic Analysis.

Using Algorithm 1, we can inversely learn the preference θ for each driver, during each time interval (e.g., a month) over time, and obtain a sequence of preference vectors $\{\theta_1, \cdots, \theta_N\}$. For each driver, we can conduct hypothesis testing to examine if the change

of the preference vectors over months is significant or not. We denote the preference vector learned for taxi driver p in period T_i as θ_i^p, and that in period T_j as θ_j^p. Then, we can obtain two preference vector sample sets in i-th and j-th months as S_i and S_j over a group of n drivers as follow:

$$S_i = \{\theta_i^1, \theta_i^2, ..., \theta_i^n\}, \tag{2.10}$$

$$S_j = \{\theta_j^1, \theta_j^2, ..., \theta_j^n\}. \tag{2.11}$$

With S_i and S_j, we will examine if the entries in preference vectors changed significantly or not from the i-th to j-th month, using paired sample t-test [38]. For each feature f_m, the null hypothesis is that the difference between the m-th entry of each θ_i^p in S_i and θ_j^p in S_j equals 0, which means drivers' preference to feature f_m does not change significantly from the i-th month to the j-th month. Otherwise, the alternative hypothesis indicates a significant change. Taking the difference between S_i and S_j as $\Delta S_{ij} = \{\Delta\theta_{ij}^1, \Delta\theta_{ij}^2, ..., \Delta\theta_{ij}^n\} = \{\theta_i^1 - \theta_j^1, \theta_i^2 - \theta_j^2, ..., \theta_i^n - \theta_j^n\}$.

The t-test statistics of the m-th entry is as follow.

$$t_{ij}(m) = \frac{Z}{s} = \frac{\Delta\overline{\theta}_{ij}(m) - \mu}{\delta/\sqrt{n}}. \tag{2.12}$$

where μ is the sample mean, n is the sample size and δ is the sample square error. The t-distribution for the test can be determined given the degree of freedom $n - 1$. Given a significance value $0 < \alpha < 1$, we can get a threshold of the t value t_α in the t-distribution. Then if $t_{ij}(k) > t_\alpha$, the null hypothesis should be rejected with significance α, otherwise, we can accept the null hypothesis with significance α. Usually, we set $\alpha = 0.05$, which also means the confidence of the test is $1 - \alpha = 0.95$.

2.1.3.4 Preference Trend Analysis

In Section 2.1.3.3, we employ hypothesis test to examine if the change of the preference over months for each driver is significant or not. In this section, we investigate what the trends of the preferences are for different groups of drivers regarding the significantly changed preferences. For significantly changed preference to feature f_s, we can obtain the preference of driver k in the i-th and j-th ($i < j$) month as $\theta_i^k(s)$ and $\theta_j^k(s)$. Then the set of preference to feature f_s over a group of n drivers in the i-th month can be denoted as:

$$S_i(s) = \{\theta_i^1(s), \theta_i^2(s), ..., \theta_i^n(s)\}, \tag{2.13}$$

$$S_j(s) = \{\theta_j^1(s), \theta_j^2(s), ..., \theta_j^n(s)\}. \tag{2.14}$$

We want to investigate in detail how the preference change, i.e., some preferences trend up over time, while some others trend down. Here, we define the positive trend rate r_p in Eq(2.15) to characterize the increasing trend of the each preference for each group of drivers, and the negative trend rate r_n to characterize the decreasing trend, which equals to $1 - r_p$.

$$r_p = \frac{\sum_{k=1}^{n} I_{ij}^s(k)}{n}, \tag{2.15}$$

where n is the number of drivers in the group, and:

$$I_{ij}^s(k) = \begin{cases} 1 & if\ \theta_i^k(s) < \theta_j^k(s) \\ 0 & if\ \theta_i^k(s) > \theta_j^k(s). \end{cases} \tag{2.16}$$

2.1.4 Evaluation

In this section, we conduct experiments with real world taxi trajectory data to learn the preferences of different groups of taxi drivers, and analyze the preference evolution pat-

terns for each group.

2.1.4.1 Experiment Settings

When analyzing the temporal dynamics of the drivers' decision-making preferences, the null hypothesis is that the difference between the preferences in two time periods is not significant. The alternative hypothesis is the temporal preference difference is significant. We choose the t-test significance value $\alpha = 0.05$.

Driver Group Selection. We aim to analyze how taxi drivers' decision making preferences evolve over time. For each month, we select 3000 drivers with the highest earning efficiency. The reason why we select these drivers is that they are likely more experienced drivers, thus with near-optimal policies, which is required by the maximum entropy principle [31] to ensure a precise preferences recovered by IRL from the demonstrations. To evaluate the preference change across two months, i.e., the i-th and j-th months, we find those drivers from those experience drivers, who also show up in both months for our study. For example, in 07/2016 and 12/2016, there are 2151 experienced drivers in common.

Then, we calculate the difference of earning efficiency of each driver in the two months. Fig. 2.13 shows the gap distribution in 07/2016 and 12/2016. We will choose three groups of drivers for preference dynamics analytics based on the drivers' earning efficiency gaps.

- Group #1 (Self-improving Drivers): 200 drivers whose earning efficiencies increase the most.
- Group #2 (Stabilized Drivers): 200 drivers whose earning efficiency gaps are small, i.e., close to 0.
- Group #3 (Exploring Drivers): 200 drivers whose earning efficiencies decrease the

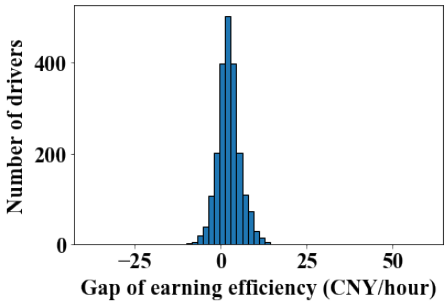


Figure 2.13: Earning efficiency gap distribution

most.

The self-improving drivers are more likely to have learned a lot during the time span from July 2016 to December 2016. By analyzing their preference dynamics, we can get a sense of how they learned overtime. The stabilized drivers are those whose earning efficiencies did not change much from July 2016 to December 2016, and we want to validate if their preferences were also stable during the time span to cross-validate how taxi drivers learn knowledge overtime. The exploring drivers are those whose earning efficiencies decreases the most from July to December, and we want to figure out why this happened to these drivers by analyzing their learning curve via our DHPA framework. And as the first attempt to analyze the learning curve of taxi drivers, in this work, we do not explore deeper to individual taxi drivers. And in our future work, we will explore the learning curve of individual taxi drivers.

Experiment Plan. We use 12 months trajectory data across three years of time span for our study, i.e., 07/2014–12/2014, and 07/2016–12/2016. We evaluate the preference dynamics across months pairs. First, we setup the month 07/2016 as the base month, and compare the preferences of drivers in Group #1 and Group #2, with that of 5 subsequent months (08/16, 09/16, 10/16, 11/16, and 12/16), respectively. To examine the dynamics of potential habits' preferences in a short period and a long period. We define the short period including 07/16 to 08/16 and 07/16 to 09/16, and the long period including 07/16

to 11/16 and 07/16 to 12/16. We apply family-wise t-tests with Bonferroni correction to avoid an inflation of false positives.

2.1.4.2 Preference Dynamics Analysis

Now we present the results of the analysis of the preference dynamics of two driver groups over time.

Results for Group #1. The table in Fig. 2.14 shows the t-values obtained for comparing preferences (with respect to each feature) in 07/16 to that of 08/16, 09/16, 10/16, 11/16, and 12/16, respectively. For these self-improving drivers, The boxes of failed tests are marked with red color, and the corresponding t values. Note that these tests are conducted individually without comparisons among them because we want to examine whether there exists a significant preference change for individual feature in a specific month compared with July. First, with a time span of less than three months, the preferences do not show any significant change. However, when the time space is larger than three months, preferences to some habit features changed significantly if viewed as individual tests, including *H1: Number of pickups*, *H3: average trip time* and *H4: traffic condition*. This makes sense, since over time, the self-improving drivers tend to learn the knowledge of where the demands, low traffic, long trip orders are. On other hand, the preferences to all four profile features and other habit features stay unchanged over the half a year.

According to the results of the individual tests above, we notice that the preferences to three habit features (H1, H3, H5) might change significantly in a long period, i.e., after 3 months. To validate these preference dynamics in a long period, we conduct family-wise hypothesis tests to examine if the preferences to these features change significantly after a long period. We consider July to August and July to September as the short period, and July to November and July to December as the long period. Since only the preferences

Group #1 (Self-improving Drivers)

	P1	P2	P3	P4	H1	H2	H3	H4	H5	H6
Aug	-1.09	0.82	-0.27	0.61	0.15	-0.44	-0.03	0.93	1.23	-1.33
Sep	0.70	0.79	0.26	-0.36	-1.88	1.11	0.66	0.70	-0.33	-0.43
Oct	-0.18	0.08	-0.48	0.51	0.02	-1.66	0.89	-1.30	0.19	1.12
Nov	1.75	-0.96	-0.10	-1.63	-2.20	0.58	1.39	2.80	1.30	-0.34
Dec	-0.43	0.43	-0.32	-0.10	-2.51	-0.28	2.22	2.11	0.01	-0.34

Figure 2.14: Preference dynamics between July and each of the following 5 months of Group #1 drivers

Group #2 (Stabilized Drivers)

	P1	P2	P3	P4	H1	H2	H3	H4	H5	H6
Aug	-1.05	-1.23	1.30	-1.28	0.71	-0.66	-0.28	-1.94	-0.47	1.21
Sep	-0.08	-0.23	1.44	-0.85	0.09	0.83	-0.99	-0.98	-0.63	0.24
Oct	0.04	-1.74	1.87	-1.20	0.84	0.25	-0.63	-1.48	-0.19	-0.40
Nov	-0.62	0.77	-0.11	-0.95	1.05	-0.25	-1.50	-1.06	0.44	0.05
Dec	0.88	-1.13	1.89	0.36	-0.21	-0.36	0.57	-0.76	-1.27	-0.11

Figure 2.15: Preference dynamics between July and each of the following 5 months of Group #2 drivers

to the three habit features potentially change significantly, we have 6 tests in total. After Bonferroni correction, the results are presented in Fig. 2.16. We observe that, after a long period, the preferences to *H1: Number of pickups* and *H4: traffic condition* change significantly, while preference to *H3: average trip time* does not show a significant change. And preferences to these three habit features do not change after a short period. **Results for Group #2.** The table in Fig. 2.15 shows the t-values obtained for preference comparison of drivers in Group #2. Clearly, the preferences to all profile and habit features stay unchanged over the half a year, which means that these stabilized drivers have kept the same strategy of finding passengers in the half a year. This is consistent with their unchanged earning efficiencies over time. The reason why the stabilized drivers maintained a stable preferences either because they were very experienced and already obtained the optimal strategies given the profiles they had, which means they had reached the upper-bound of earning efficiency for the drivers who have similar profiles, or they had found a near optimal strategy and still have potential space for improvement, but they were not

	H1	H3	H4
08 & 09	-1.18	0.45	1.20
11 & 12	-2.83	2.09	3.19

Figure 2.16: Validation on the long-term preference dynamics

Group #3 (Exploring Drivers)

	P1	P2	P3	P4	H1	H2	H3	H4	H5	H6
Aug	0.94	-0.35	0.76	-0.68	0.35	-1.23	1.64	-1.33	0.45	-0.96
Sep	-2.32	-1.22	1.43	2.43	2.45	-3.22	-2.09	1.16	-1.35	-2.33
Oct	-2.58	1.78	-3.43	-4.24	-3.35	-4.92	4.11	-3.43	3.22	-5.25
Nov	-3.54	-2.45	2.54	2.23	2.97	3.21	-2.25	-2.63	2.23	2.44
Dec	3.11	-2.31	2.37	2.55	-3.18	-2.54	3.77	-2.75	-3.47	-2.87

Figure 2.17: Preference dynamics between July and each of the following 5 months of Group #3 drivers

Group #3 (Exploring Drivers)

	P1	P2	P3	P4	H1	H2	H3	H4	H5	H6
Aug	0.51	0.47	0.42	0.43	0.49	0.44	0.52	**0.63**	0.44	0.45
Sep	0.49	0.47	0.50	**0.38**	**0.39**	0.55	**0.37**	0.47	0.53	**0.39**
Oct	0.55	**0.60**	0.46	0.45	**0.38**	**0.62**	**0.63**	0.51	**0.63**	0.53
Nov	**0.64**	0.45	**0.39**	0.48	0.44	**0.37**	0.55	0.44	0.55	**0.65**
Dec	0.52	0.51	0.59	0.50	**0.61**	0.45	**0.36**	0.57	**0.37**	0.47

Figure 2.18: Positive rates of each preference in Group #3

motivated to find a better one, or some other potential reasons. As the first attempt to analyze the preference dynamics in this work, we do not dig that deep to figure out the exact reasons why these drivers maintained a stabilized preferences, but we will investigate this problem in the future work.

Results for Group #3. The table in Fig. 2.17 shows the t-values obtained for preference comparison of drivers in Group #3. The preference of this group of drivers changed a lot over months, most habit features changed, and the preferences to 1 or 2 profile features changed in November and December. Group #3 drivers are the exploring drivers, whose earning efficiencies drop over these months, and the results can tell that they try changing their preferences significantly over time to explore new strategies, but the attempts do not work for the growth of their earning efficiencies.

2.1.4.3 Preference Trend Analysis

In this section, we present the results of the preference trend analysis over three driver groups.

	P1	P2	P3	P4	H1	H2	H3	H4	H5	H6
Group #1	0.48	0.49	0.47	0.51	**0.38**	0.46	**0.63**	**0.65**	0.43	0.57
Group #2	0.57	0.47	0.58	0.50	0.46	0.45	0.49	0.51	0.48	0.51

Figure 2.19: Positive rate of each preference in Group #1 & #2

Results for Group #1. The results of the preference trend analysis for Group #1 are shown in Fig. 2.19. The values in the table are r_p's for each preference between July and December. The values in red indicate most drivers have a positive trend on the preference, which is consistent with the result of the preference dynamics analysis, since the most significantly changed preferences are the same, i.e., the preferences to *H1: Number of pickups*, *H3: average trip time* and *H4: traffic condition*. For example, the preference to feature H1 has an $r_n = 0.62$, which is prominently larger than the $r_p = 0.38$. This indicates that most of the self-improving drivers tend to reduce their preference to feature H1, i.e., the number of pickups. This is reasonable because the regions with large number of pickups is usually crowded, e.g., downtown area. To complete a service trip is time-consuming, which can damage the earning efficiency of taxi drivers. As for the significantly changed preferences to features *H3: average trip time* and *H4: traffic condition*, the r_p's are prominently larger than the r_n's respectively, which indicates the number of self-improving drivers who increase their preferences on feature H3 and H4 is prominently greater than those who decrease.

Results for Group #2. The results of the preference trend analysis for Group #2 are shown in Fig. 2.19. We notice that the r_p's are close to 0.50 regarding the preference to each of the feature, which is consistent with the results of the preference dynamics analysis in Section 2.1.4.2.

Results for Group #3. The results of the preference trend analysis for Group #3 are shown in Fig. 2.18. The values in the table are the r_p's calculated between July and each of the following months. The values in red indicate most drivers have a positive trend on the preference comparing with July, and the blue ones indicate most drivers have a negative trend on the preference comparing with July. We find that the exploring drivers change their preferences randomly. Taking the preference to feature *H5: Distance to train station* as an example, significant positive trend is found in August and October, while in

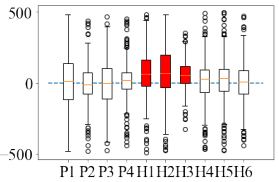


Figure 2.20: All drivers

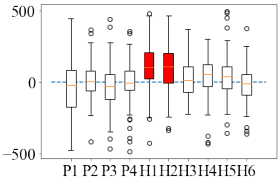


Figure 2.21: Group 1: Self-improving drivers

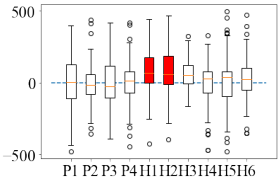


Figure 2.22: Group 2: Stabilized drivers

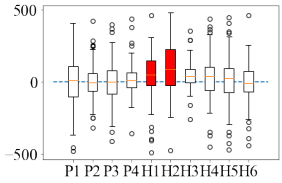


Figure 2.23: Groups 3: Exploring drivers

December, it switches to a negative trend. Similar patterns can be found in the preferences to feature H1, H3, and P3.

2.1.4.4 Preference Distribution Analysis

To explore what different features that different groups of drivers pay attention to, we analyze the distribution of the preferences in December for self-improving (Fig. 2.21), stabilized (Fig. 2.22), and exploring (Fig. 2.23) groups as well as the entire taxi driver population (Fig. 2.20). We find that in Fig. 2.20 the preference median of profile features are all close to 0, and the preference median to *H1: Number of pickups*, *H2: average trip distance* and *H3: average trip time* are relatively higher than other three habit features, which implies that overall taxi drivers prefer higher number of pickups and longer trips. From Fig.2.21-2.23, we find that the self-improving drivers and the stabilized drivers have higher preferences to *H1: Number of pickups*, which indicates that they learned sufficient knowledge on which regions have high demands. In contrast, the exploring drivers have lower preferences to *H1: Number of pickups*. It reflects that they are still learning the distribution pattern of travel demands. Moreover, the preference to *H2: average trip distance* of the exploring drivers is higher than that of other groups, which implies that the exploring drivers paid excessive attention to the long distance trips. This may be one of the negative effects leading to a decreasing earning efficiency trend since the longer the trip distance is, the more time it costs.

2.1.4.5 Case Study.

Case of Preference Dynamics. To further understand the preference dynamics, we look into individual drivers to showcase how the preference and working behaviors evolve over time. Here we show one randomly selected driver from Group #1. Let us call him "John". John's earning efficiency grew from 41.84 CNY/hour to 52.24 CNY/hour from 07/16 to 12/16. His preferences in both months are listed in Fig. 2.24 -2.25. Clearly, the preferences to the profile features remain unchanged, while the preferences to some habit features, such as *H1 Number of pickups*, *H5&H6 Distance to Train Station & Airport* changed. When we look into John's driving behaviors, it matches the preference change perfectly.

Preference change to H1. The preference change to feature H1 indicates that John increased his preference to areas with high volume of pickup demands. Fig. 2.26-2.27 shows the distribution of trajectories when the taxi was idle in the morning rush hours in 07/16 and 12/18, respectively. Fig. 2.28-2.29 shows the all taxi pickup demand distributions in the morning rush hours in 07/16 and 12/16, respectively. The city-wide demand distribution does not change. However, during the morning rush hours, John changed his strategy from 07/16 to 12/16, i.e., to look for passengers from the high demand areas. This is consistent to the preference change to feature H1 (number of pickups).

Preference change to H5. The preference change to feature H5, i.e., distance to train station, is also significant. The negative preference indicates John prefers to be closer to the train station to look for passengers. Over time this preference became stronger. To explain this phenomenon, we highlighted the train station in Fig. 2.26-2.27. The statistics we obtained from John's trajectory data showed that the percentage of order received near the train station increased from 11.93% in 07/16 to 14.21% in 12/16, which is consistent with the preference change.

The results of preference dynamics analysis in Section 4.2.1 shows the overall pattern

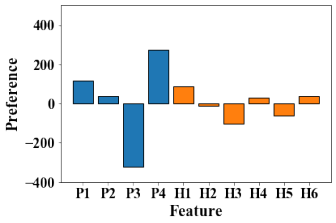


Figure 2.24: John's Preference in 07/16

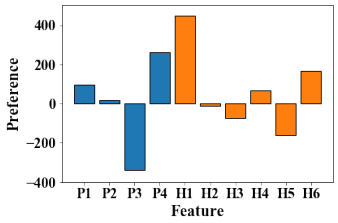


Figure 2.25: John's Preference in 12/16

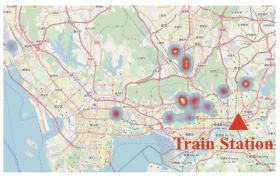


Figure 2.26: John's Trajectory in 07/16

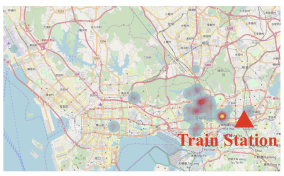


Figure 2.27: John's Trajectory in 12/16

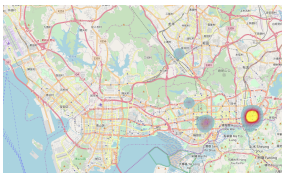


Figure 2.28: Overall Pickups in 07/16

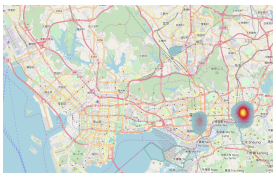


Figure 2.29: Overall Pickups in 12/16

of Group #1, which illustrate that the self-improving drivers showed significant dynamics on their preferences to features *H1: Number of pickups* and *H4: traffic condition*, and the preference to H1 tends to decrease in the group. While in the case study, "John" is an individual driver randomly selected from the group, who was happened to have a reversed trend regarding the preference to feature H1. This is not a contradiction since the results in Section 4.2.1 are for the overall group, while in the case study, the randomly selected driver is an individual in the group, who might not maintain the same preference dynamics as the whole group.

Case of Preference Trend. To further understand the preference trend, we also look into an individual driver's show case to investigate how exactly the preference changed

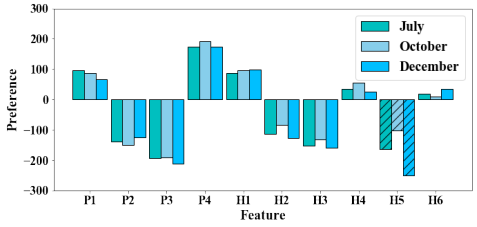


Figure 2.30: The decision-making preferences of Mike in July, October and December

over time from July to December. We randomly select a driver from Group 3. Let's call him "Mike". Mike's earning efficiency dropped from 46.15 CNY/hour to 41.34 CNY/hour from 07/16 to 12/16. His preferences in 07/16, 10/16 and 12/16 are shown in Fig. 2.30. The preferences to some features changed randomly, e.g., H5. H5 is the habit feature: distance to the train station. The negative values indicate that Mike prefers to be closer to the train station. Comparing with the preference to H5 in July, the preference goes weaker in October and goes stronger in December. To explain this, we visualize the distributions of the trajectories of Mike in Fig. 2.31-2.33. The distribution near the train station becomes more scattered in October and more concentrated in December comparing with that in July.

2.1.4.6 Takeaways and Discussions.

From our studies on a large amount of taxi trajectory data spanning for 3 years, *we made the first ever report on how real world taxi drivers make decisions when looking for passengers, and how their preferences evolve over time.* Overall, three key takeaways are summarized as follows.

1. Each driver has its unique preferences to their profile features, which tend to be stable over time.

2. Drivers while learning the environments, may change their preferences to habit features.

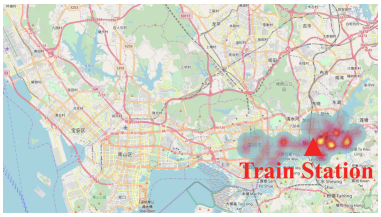


Figure 2.31: July, 2016

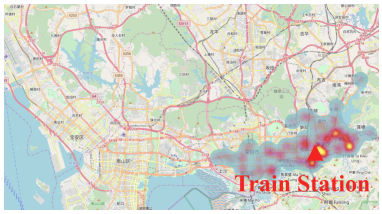


Figure 2.32: October, 2016

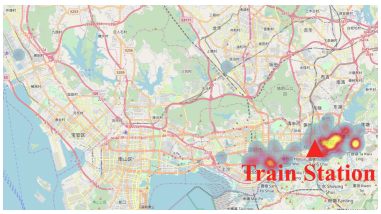


Figure 2.33: December, 2016

3. Drivers while exploring the environments, may change their preferences to profile and habit features randomly.

Our findings can be potentially utilized to assist and guide taxi drivers to improving their earning efficiencies. For example, for those slow learning drivers, by learning their preferences, especially, the preferences to habit features, we can diagnose which knowledge in terms of the features they are lacking, e.g., not familiar with the high demand regions. As a result, some guiding messages, may be sent directly to the drivers about such information, to assist the drivers to improve a better policy faster. In addition, our proposed DHPA framework can easily adapt to different time interval analyses, e.g. over months, over days, over time in a day, etc. One only needs to change the trajectory extraction in Stage 1 according to the different settings of time interval.

2.2 Human Learning and Reinforcement Learning

2.2.1 Overview

2.2.1.1 Introduction

Learning to make decisions is ubiquitous for human beings. For example, a Go player learns to imitate other players to devise better game strategies. A physician learns to determine doses of drugs through extensive case studies and sometimes ad-hoc experimentation. A professional driver learns to cruise through repeated practice to effectively find the next passenger. While a learning process is often complex, recent advances in machine learning have enabled computer agents to automate some learning tasks. For example, reinforcement learning (RL) is used to train AlphaGo [39] to beat the human champion and build systems to recommend medical treatments [40]. Optimizing taxi operation strategies has also been extensively studied in the literature [2, 12, 41, 42, 43].

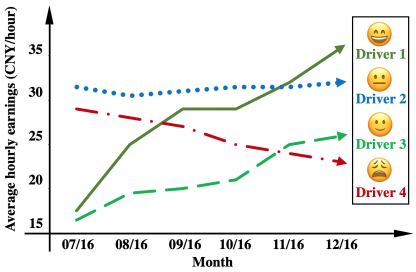


Figure 2.34: Diverse patterns of drivers' per-hour income dynamics in Shenzhen, China.

Many recent solutions also rely on RL techniques.

While progress was made to design RL algorithms for computer agents to learn, it remains unclear how the human counterpart learns. Do human learning processes exhibit similar patterns to the one driven by RL algorithms, or they deviate from any known learning strategies? Answering this problem is important for three reasons: *(i)* many decision-making problems remain challenging for machines and still require "human learning", so it becomes important to distill decision strategies from humans; *(ii)* effective humans learning strategies can be used to train beginners such as new Go players and new taxi drivers; and *(iii)* it also advances cognitive and social science research by taking an algorithmic lens at human learners' behaviors.

This work examines how traditional taxi drivers learn to cruise for seeking their next passengers. Here, traditional drivers refer to those that do not rely on mobile-based platforms such as Uber, Lyft, or DiDi. These drivers represent a significant portion of personnel in taxi service, despite recent growth of online platforms. Prior studies [2] on this problem assumed drivers are rational and use inverse reinforcement learning to characterize drivers' behaviors. Although these works offer useful insights, not all drivers are rational. Some drivers learn faster than others. Some drivers' performance even deteriorates over time. For example, Fig. 2.34 shows the dynamics of per-hour incomes from four typical taxi drivers in 2016 in Shenzhen, China. Driver 1 (dark green line) started at a relatively low-income level, but then rapidly doubled the income level by the end of

the year. Driver 2 (light green line) started at a similar (low) income level as Driver 1, but had a much slower increasing trend. Driver 3 (blue line) had a stable income level over time. Moreover, the income level of Driver 4 (red line) went down roughly by 30% in six months.

This example suggests that different drivers use different strategies to learn. Thus, our work focuses on investigating i) *what learning strategies they are following, especially for those "quick learners"?;* ii) *how do these strategies compare to what a computer agent would follow in reinforcement learning?*

Specifically, we investigate and validate human learning strategies through a data-driven case study on taxi drivers. To the best of our knowledge, this is the first attempt of its kind in the context of taxi operations. Specifically, we extract trips of taxi drivers from a large-scale dataset spanning 6 months with over 17,000 taxis. We categorize drivers into different groups based on their hourly earning dynamics. For each group of drivers, we build estimation procedures to construct the time series of a driver's policy and advantage functions and examine whether their patterns are consistent with those of an agent in a RL algorithm. In addition, we validate under what scenarios the drivers are following the paradigm of RL, if not always.

Our major finding is that a taxi driver's improvement in earning efficiency is positively correlated with how well he/she follows the process of RL algorithm. In addition, human drivers usually do not completely follow RL when learning. They tend to follow RL first for those scenarios (e.g., certain urban areas) that lead to higher earning improvement. *Our contributions* are summarized as follows: **(1)** We propose a three-stage analytical framework to rigorously validate whether human agents (e.g., taxi drivers) follow RL paradigms to improve their earning efficiencies.

(2) It is evident from the analytical results on a large-scale taxi trajectory dataset that successful drivers are likely those who follow the RL paradigm better. Moreover, they

tend to follow RL first for those scenarios (e.g., certain urban areas) that lead to higher earning improvement.

2.2.1.2 Problem Definition

Problem Definition: Given real trajectory data of taxi drivers $\tilde{\mathfrak{T}}$ in a sequence of time intervals $T_0, T_1, ..., T_n$, we aim to validate or reject the following hypotheses: (1) Drivers that are successful in learning passenger-seeking experiences (i.e., with increasing earning efficiency), employ learning strategies that are closer to reinforcement learning (RL) paradigms; (2) The RL paradigm is followed by human drivers only at certain scenarios (e.g., locations, times) rather than all circumstances. We also aim to identify what these scenarios are.

We also use two data sources in this project: *(i)* taxi trajectory data and *(ii)* road map data, both collected in Shenzhen, China in 2016. See Sec. 2.1.1.3 for details. **Data Preprocessing:** We preprocess the datasets by map gridding and time discretization.

(1) Map gridding. The urban road network forms a continuous space. We use the gridding-based method to simply partition the road map into equally sized grids [15, 33]. This method is easy to implement and make adjustment. It allows us to adjust the size of the grids, and examine the impact of the grid size. We let s be the side-length of each cell. Cells adjacent to each other are considered reachable if there is at least one road across their boundary. Fig. 2.6 visualizes of our gridding results with side-length of $s = 0.01°$ in latitude and longitude. By removing grid cells in those unreachable regions in the city (e.g., in the center of a part), we have a total of $n = 1,018$ valid cells (highlighted in light colors in Fig. 2.6) covered by the road network.

(2) Time Discretization. We divide each day (24 hours) into three time intervals, i.e., 00:00 – 06:00, 06:00 – 16:00, and 16:00 – 24:00, based on the common schedules

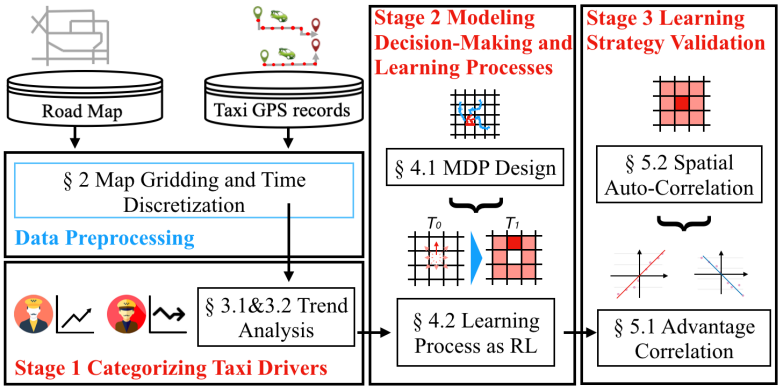


Figure 2.35: Solution framework

of taxi drivers. In Shenzhen, each taxi is usually operated by two drivers. One driver operates in day time and the other operates at nights. Thus, taxi trajectories in different time intervals are considered from different drivers. Two drivers usually switch at around 6AM and 4PM everyday. Finally, because there are exceedingly small numbers of taxi trips between mid-night and early morning, we focus on only two time intervals, i.e., 06:00 - 16:00 and 16:00 - 24:00.

Solution Framework: Our proposed solution framework is outlined in Fig. 2.35, which takes two sources of urban data as inputs and contains three analytical stages: (1) categorizing taxi drivers in section 2.2.3.1, (2) modeling decision-making and learning process in section 2.2.3.2, (3) learning strategy validation in section 2.2.3.3.

2.2.2 Related Work

Taxi operating strategies (e.g., dispatching, passenger seeking), and driver behavior analysis have been extensively studied in recent years due to the emergence of the ride-sharing business model and urban intelligence. The related works are summarized below.

Urban Computing integrates urban sensing, data management, and data analytic as a unified process to explore, analyze, and solve problems related to people's everyday life [13, 15, 16, 17, 18, 19, 21, 44, 45, 46, 47, 48, 49]. In particular, a group of works

have studied the topic of taxi operation [27, 28, 50, 51, 52, 53, 54, 55], such as vehicle dispatching with reinforcement learning [52, 56, 57, 58, 59, 60, 61, 62, 63, 64], and passenger-seeking strategies [3, 10, 11, 29, 65, 66]. They aim to find optimal solutions to improve the revenue of individual taxi drivers as well as the entire fleet. For instance, [12] solved the passenger-seeking problem by giving direction recommendations to drivers. However, few studies investigate the relation between the machine learned strategies and human drivers' strategies. Some studies directly assume that human drivers follow reinforcement learning [2, 41, 67] without validation through real cases. To the best of our knowledge, *our study makes the first attempt to validate if taxi drivers follow the paradigm of reinforcement learning when earning their driving experiences*.

Human Learning is a process of interacting between a person and the external environment, which leads human to change capacity permanently not due to biological maturation [68]. To characterize how the process works, research in Cognitive Neuroscience, Psychological Sciences, and Behavioural Sciences has studied over five decades [69]. [70] investigated the role of brain's modular structures and found that flexibility measured by the allegiance of nodes to modules in a past session could predict the relative amount of learning in a future session. [71] contended the essential factors that can lead to progress in learning mathematics from the perspective of psychology. [72] introduced a structured learning tool and teaching process to translate the learning principles into practice for learning clinical skills regarding behavioral sciences. Compared with previous works, *we deliver an innovative insight of leveraging the understanding of human learning to engineer the learning process through machine learning*.

2.2.3 Methodology

2.2.3.1 Stage I: Categorizing Taxi Drivers

This section introduces the definition of taxi drivers' earning efficiencies, the earning efficiency dynamics of taxi drivers, and classification of taxi drivers based on the trends of their earning efficiencies.

Quantifying Taxi Drivers' Earning Efficiencies. To quantify the earning efficiencies of taxi drivers, we need to address two issues: *1. Effective working hours. 2. Changes in earning efficiencies.* Drivers' earning efficiencies evolve over time. Thus, we re-estimate drivers' earning efficiencies every week.

Let r_e^i be the earning efficiency of driver e in week i ($1 \leq i \leq 27$). We let

$$r_e^i = \frac{E_e^i}{t_e^i}, \tag{2.17}$$

where E_e^i is his/her total income in week i and t_e^i is the total working hours. Here, the total working hours are the time when the driver is seeking for passengers or serving passengers. We eliminate the time when the driver takes a break (the taxi stays still for 30 minutes or more).

Earning Efficiency Trend Analysis. We aim to detect the following patterns in drivers' earning efficiencies changes:

- *Monotonic increase/decrease*. The increase or decrease occurs constantly over the entire time series.
- *Abrupt increase/decrease*. At a certain time point, an abrupt increase or decrease occurs, differing the statistics of time series before and after that significantly.

When the efficiency of a driver does not exhibit any of the above changes, we define the driver as a *stabilized* driver. Fig. 2.36-2.38 show examples of different learner groups.

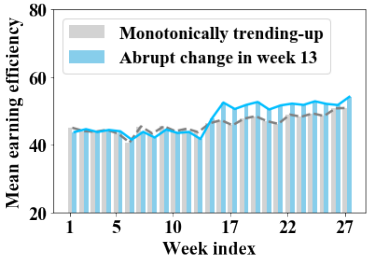


Figure 2.36: Trending-up drivers

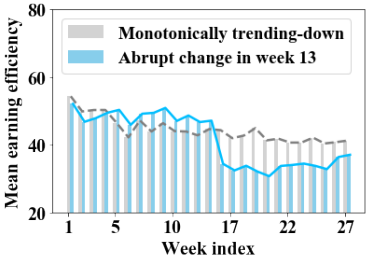


Figure 2.37: Trending-down drivers

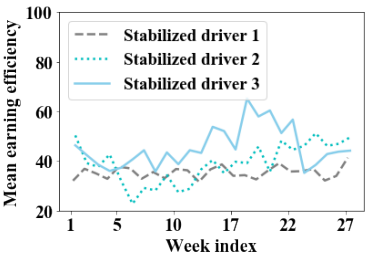


Figure 2.38: Stabilized drivers

Next, we devise two statistical tools to detect the aforementioned patterns.

Mann-Kendall (MK) Trend Test [73] is a hypothesis test method for monotonic trend in time series data, which indicates whether a trend exists and whether the trend is positive or negative. The null hypothesis H_0 is **no monotonic trend**, while the alternative hypothesis H_1 is **monotonic trend is present**.

The statistic of Mann-Kendall test can be calculated as follows,

$$Z_{MK} = \begin{cases} \frac{S-1}{\sqrt{VAR(S)}} & if\ S > 0, \\ 0 & if\ S = 0, \\ \frac{S+1}{\sqrt{VAR(S)}} & if\ S < 0, \end{cases} \tag{2.18}$$

$$S = \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} \text{sgn}(r_e^j - r_e^k), \tag{2.19}$$

$$\text{sgn}(r_e^j - r_e^k) = \begin{cases} 1 & if\ r_e^j - r_e^k > 0, \\ 0 & if\ r_e^j - r_e^k = 0, \\ -1 & if\ r_e^j - r_e^k < 0, \end{cases} \tag{2.20}$$

$$VAR(S) = \frac{1}{18}[n(n-1)(2n+5)]. \tag{2.21}$$

Given a confidence α, the null hypothesis is rejected if $|Z_{MK}| > Z_{1-\alpha}$, where $Z_{1-\alpha}$

is the $(100(1-\alpha))^{th}$ percentile of the standard normal distribution.

Pettitt's Test [74] is to detect change points in time series data. A change point in a time series $r_e^1, r_e^2, r_e^3, ..., r_e^t, ..., r_e^n$ refers to a time index t such that $\{r_e^{t_1}\}_{t_1 \leq t}$ and $\{r_e^{t_2}\}_{t_2 > t}$ follow two distributions [75]. The null hypothesis H_0 is **no abrupt change points exist**, while the alternative hypothesis H_1 is **an abrupt change point exists**.

Pettitt's test uses a non-parametric test statistics U_t defined as

$$U_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} \text{sgn}(r_e^i - r_e^j). \tag{2.22}$$

Then we can calculate:

$$K = \max_{1 \leq t \leq n} U_t. \tag{2.23}$$

The change-point of the series is located at time K, provided that the statistic is significant. The significance probability of K is approximated for $p \leq 0.5$ with:

$$p \approx 2 \exp \frac{-6K^2}{n^3 + n^2}. \tag{2.24}$$

Results on Trend Analysis. We next describe our result. Our dataset contains 2,403 taxis in the 6am to 4pm interval and 2,790 taxis in the 4pm to 12am interval. We categorize taxi drivers into three groups: **(1) Trending-up:** if at least one of the tests (MK and Pettitt) show significant increasing trend, **(2) Trending-down:** if at least one of the tests show significant decreasing trend, and **(3) Stabilized** if none of the tests is significant. The two tests do not produce any inconsistent conclusions among drivers we examine (i.e., one test shows it trends up whereas the other shows it trends down).

Fig. 2.39 presents the results. Note Week #14 and #15 are excluded from the dataset because they have much smaller trip numbers due to the national holiday. This is to avoid biased results.

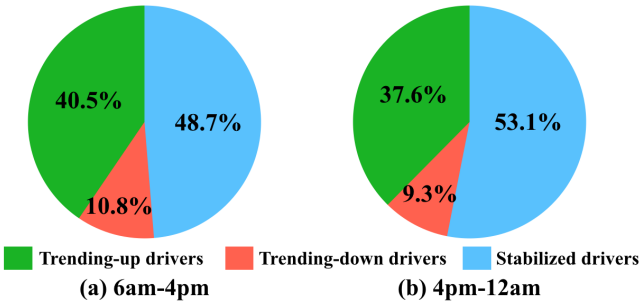


Figure 2.39: Driver groups

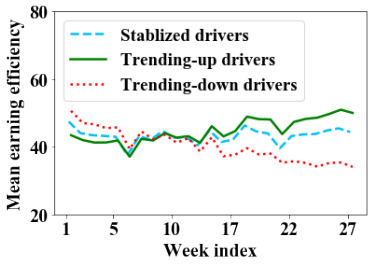


Figure 2.40: Mean earning efficiency of each group

We can see that around half of the drivers are stabilized drivers, and the number of trending-up drivers is larger than the number of trending-down drivers in both intervals. Fig. 2.40 shows the average earning efficiencies for each group of drivers over 25 weeks. The trends exhibit here are consistent with the test results.

2.2.3.2 Stage II: Modelling Decision-Making and Learning Processes

We next model the drivers' behaviors. We need to model: *(i) how drivers make decisions* (i.e., how they look for and serve passengers). This is modeled by a Markov Decision Process. *(ii) how drivers learn to make decisions* (i.e., how they use their past experience to update their decision policies over time). Based on our hypothesis, we use reinforcement learning (RL) to model this process.

Decision-Making Process as an MDP. A taxi driver needs to determine the travel direction when the taxi is idle and this decision impacts his/her chance to find a new passenger. We model this decision-making process as a Markov Decision Process (MDP) [34].

Review of MDP. An MDP is represented as a 5-tuple $\langle S, A, T, \gamma, \mu_0, R \rangle$.

- S is a finite set of states;
- A is a finite set of actions;
- T is the probabilistic transition function with $T(s'|s, a)$ as the probability of arriving at state s' by executing action a at state s;

- $\gamma \in (0, 1]$ is the discount factor[1];
- $\mu_0 : S \to [0, 1]$ is the initial state distribution;
- $R : S \times A \to \mathbb{R}$ is the reward function.

A randomized, memoryless policy is a function that specifies a probability distribution on the action to be executed in each state, defined as $\pi : S \times A \to [0, 1]$. We use $\tau = [(s_0, a_0), (s_1, a_1), \ldots, (s_L, a_L)]$ to denote a trajectory generated by MDP. Here L is the length of trajectory.

Applying MDP to model drivers. We model the decision-making process of taxi drivers with MDP as follow:

- State: a spatial region, specified by a geographical grid cell, created with map gridding in data preprocessing phase;
- Action: traveling from the current cell to one of the eight neighboring cells.

Fig. 2.12 shows an example of taxi trajectory as an MDP: a driver starts in state s_0 with the taxi idle, and takes the action a_0 to travel to the neighboring cell S_1 on the right. After two decisions, the driver traverses S_1 and reaches state S_2, where a passenger is found at S_2. Then, a passenger trip corresponds to a transition in the MDP from starting state S_2 to ending state S_3. Each decision made at a certain state would lead to a reward as the expected monetary income of finding and serving a passenger. The policy π employed by a driver is a probability distribution of choosing each action at each state.

Learning Process as Reinforcement Learning.

Hypothesis. When one starts working as a taxi driver, he/she may not have knowledge about where to find the next passenger, and may choose a simple initial policy π_0. Actor

[1]Without loss of generality, we assume $\gamma = 1$ in this study, and it is straightforward to generalize our results to $\gamma \neq 1$.

Table 2.1: Typical methods of RL

	Typical Method	Update function
Value-based Methods	Q-learning	$Q(s,a) \leftarrow Q(s,a)+$ $\alpha[r(s,a)+\gamma \max_{a'} Q(s',a')-Q(s,a)]$ [76]
	SARSA	$Q(s,a) \leftarrow Q(s,a)+$ $\alpha[r(s,a)+\gamma Q(s',\pi_Q(s'))-Q(s,a)]$ [76]
Actor-Critic Methods	Actor-Critic	$\nabla \overline{R_\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(Q^{\pi_\theta}(s_t^n,a_t^n)-$ $V^{\pi_\theta}(s_t^n))\nabla \log p_\theta(a_t^n\|s_t^n)$ [77]
	Advantage Actor-Critic(A2C)	$\nabla \overline{R_\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(r_t^n+V^{\pi_\theta}(s_{t+1}^n)-$ $V^{\pi_\theta}(s_t^n))\nabla \log p_\theta(a_t^n\|s_t^n)$ [77]
Policy-based Methods	Policy gradient	$\theta \leftarrow \theta+\alpha\nabla\overline{R_\theta}$, $\nabla \overline{R_\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}$ $(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n-b)\nabla \log p_\theta(a_t^n\|s_t^n)$ [78]

The driver repeats this process so his/her policy evolves continuously (see also [12, 41, 42]).

Types of reinforcement learning. Reinforcement learning (RL) algorithms can be classified into three major categories including value-based RL [76], policy-based RL [78], Actor-Critic based approach [77]. We briefly outline the key ideas of the three types of RL algorithms below. A key similarity of all these algorithms is that they optimize the policy functions by "taking the gradient" with respect to the advantage function, which is defined as the additional reward gained from the current policy comparing to the one in the previous iteration.

• *Value-based RL* [76] does not learn the optimal policy directly. It learns the so-called Q value (or V value) instead, which is defined on each state-action pair (s,a), namely, $Q(s,a)$ (or on each state s, namely, $V(s)$). Specifically, $Q(s,a)$ refers to the expected future reward, after taking an action a at a state s, while $V(s)$ refers to the expected reward after leaving a state s. Once Q-functions are well learned, the optimal policy π^* can be recovered from the optimal value function of each state-action pair (e.g., $Q(s,a)$). The Q-learning [76] and State-Action-Reward-State-Action (SARSA) methods [76] are the state-of-the-art value-based RL algorithms.

• *Policy-based RL* [78] learns an optimal policy directly. Usually, policy π is represented

by a (deep) neural network with parameter set θ. A well known policy-based method is policy gradient [78]. The objective of policy gradient is to maximize the expected future reward over trajectories:

$$\max_{\theta} \overline{R_\theta} = \max_{\theta} \{\mathbf{E}(R_\theta)\} = \max_{\theta} \{\sum_{\tau} R(\tau) p_\theta(\tau)\}. \tag{2.25}$$

Where $R(\tau)$ is the accumulated reward in trajectory τ and $p_\theta(\tau)$ denotes the probability of generating trajectory τ under the policy with parameter θ. Then, we can apply gradient ascent to find the optimal θ. The gradient of the objective function with respect to θ is:

$$\nabla \overline{R_\theta} \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b) \nabla \log p_\theta(a_t^n | s_t^n), \tag{2.26}$$

where N is the number of trajectories, T_n is the length of trajectory n, t and t' are the time steps. b is the baseline, i.e., average reward received.

• *Actor-Critic based RL* [77] combines both value-based and policy based methods, $\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$ is evaluated using $Q^{\pi_\theta}(s_t^n, a_t^n)$, and we can use V^{π_θ} to be the baseline b. Moreover, $Q^{\pi_\theta}(s_t^n, a_t^n) - V^{\pi_\theta}$ is denoted by $A^\theta(s_t, a_t)$ which is called the *advantage function*. If the expected reward after taking a state-action pair is higher than the average expected reward after exiting the state, i.e., the advantage is positive, the agent will increase the probability of taking this action in this state. The advantage function is used to update the gradient, which in turn updates the parameter of the policy network.

Similarities of three RL paradigms. The gradient update functions of the three typical methods of RL are listed in Table 2.1. They all try to maximize the expected accumulated reward in each state or state-action pair, which is related to $Q(s, a)$ and $V(s)$. In other words, all these RL algorithms are equivalent, in a sense that a larger advantage of an state-action pair results in a increased probability of choosing such pair in the future policy.

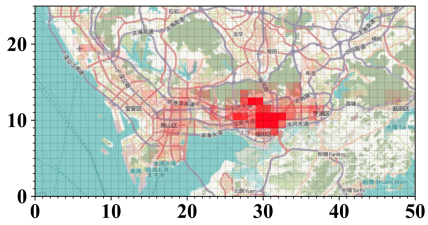


Figure 2.41: Heatmap of a driver's $D(s)$

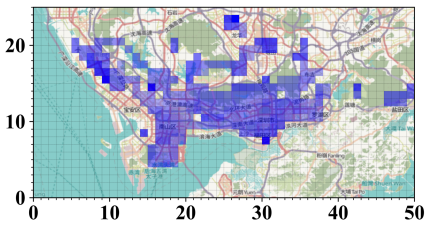


Figure 2.42: Heatmap of a driver's $V(s)$

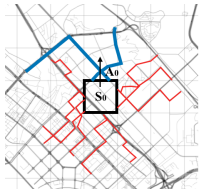


Figure 2.43: $Q(S_0, A_0)$, $V(S_0)$

0.10	0.15	0.10
0.15		0.15
0.10	0.15	0.10

Figure 2.44: Weight matrix

Empirical estimates. Our main goal is to validate whether the real-world learning process of the drivers is consistent with the policy gradient method. Here, we describe how the key variables/functions are estimated through data.

• *Estimation of advantage functions.* Recall that the advantage function captures the additional reward gained from the change of one's policy. We estimate the advantage function value of each state-action pair for each driver. In time span T_0, the advantage of a driver in each state-action pair can be estimated by the empirical Q value and empirical V value. The empirical Q value is the average earning efficiency of the driver within a certain range of time after exiting each state via each action, whereas the empirical V value is the average earning efficiency of the driver with a certain range of time after exiting each state. The difference between Q value and V value is that V value characterizes the expected reward after leaving each state s, while Q value characterizes the expected reward after taking each state-action pair (s, a) As shown in Fig. 2.43, $V(S_0)$ is calculated using all red trajectories and blue trajectories, which are the service trips exiting S_0, whereas $Q(S_0, A_0)$ is calculated using only the blue trajectories, which are the service trips exiting

S_0 through action A_0.

$$A(s,a) = Q(s,a) - V(s). \tag{2.27}$$

• *Estimation of policy functions and their differences.* We also need to estimate the difference of policies between two consecutive time spans, T_0 and T_1. The empirical policy $\pi(s,a)$ of each state-action pair in each time span can be estimated via the visitation frequencies,

$$\pi(s,a) = \frac{D(s,a)}{D(s)}, \tag{2.28}$$

where $D(s,a)$ and $D(s)$ denote the visitation frequency of the state-action pair (s,a) and state (s) respectively. Validating if taxi drivers follow RL is equivalent to examine if there exists significant correlation between the difference of policy $\Delta\pi(s,a)$ and the advantage $A(s,a)$. Next section continues the discussion of the validation process.

2.2.3.3 Stage III: Learning Strategy Validation

This section describes our validation process. This consists of *(i)* identifying the correlation between the policy difference and the advantage, and *(ii)* correcting spatial bias of the empirical policy difference and the advantage by analyzing the spatial auto-correlation.

Advantage Correlation. To validate if there exists a correlation between the policy difference $\Delta\pi(s,a)$ and the advantage $A(s,a)$, a correlation coefficient should be used. A common one is Pearson's correlation coefficient[79], but it has the assumption of independent and identical distribution of data. The Spearman's rank correlation coefficient [80] works for non-parametric data measuring a statistical relationship between two variables, which is more applicable in our ordinal data. Therefore, we employ **Spearman's rank correlation coefficient** in addition to the Pearson's correlation coefficient to evaluate the correlation coefficient and test its significance. The statistic of Spearman's rank correla-

tion coefficient can be calculated by the formula below:

$$\rho = \frac{\sum_{i=1}^{n}(rank(A_i) - \overline{rank(A)})(rank(\Delta\pi_i) - \overline{rank(\Delta\pi)})}{\sqrt{\sum_{i=1}^{n}(rank(A_i) - \overline{rank(A)})^2 \sum_{i=1}^{n}(rank(\Delta\pi_i) - \overline{rank(\Delta\pi)})^2}}, \tag{2.29}$$

where A_i is the advantage of the $i-th$ sample, and $\Delta\pi_i$ is the policy difference of the $i-th$ sample. $rank$ denotes the ordinary rank of the corresponding value, and n is the sample size.

ρ ranges from -1 to 1, and the sign of ρ indicates the direction of the association between the advantage and the policy difference, e.g., if the sign is positive, the policy difference tends to decrease with the increase of the advantage.

We can also determine the significance of the ρ. We calculate the t value according to the formula below:

$$t = \rho\sqrt{\frac{n-2}{1-\rho^2}}. \tag{2.30}$$

Then we check the p value by calculating the t value according to the Student's t distribution.

Incorporating Spatial Auto-Correlation. Intuitively, nearby grids may cover the same urban functional zone in a city and share similar demand patterns. This can be observed from the real world data. Fig. 2.41 & 2.42 show the heatmaps of the $D(s)$ and $V(s)$ of a driver in July 2016, where we can observe that similar values are clustered. Therefore, it's reasonable to incorporate spatial auto-correlation when estimating $D(s)$ and $V(s)$.

(1) Quantifying spatial auto-correlation in $D(s)$ and $V(s)$. Given a grid cell, we consider the eight neighboring grid cells as its spatial neighbors (i.e., the Queen neighborhood). A weight matrix is used to define the strength of correlation between pairs of locations, based on the inverse Manhattan distance between each pair of grid cells, i.e.,

the original weight w_{ij} between grid i and grid j ($i \neq j$) is:

$$w_{ij} = \begin{cases} \frac{1}{Manh\_dist(i,j)+1} & if\ neighbor(i,j) = True, \\ 0 & if\ neighbor(i,j) = False, \end{cases} \tag{2.31}$$

where $Manh\_dist(i,j)$ returns the Manhattan distance between grid i and grid j, and $neighbor(i,j)$ returns $True$ if grid i and j are neighboring and vise versa. Then the weights for each grid are normalized among its neighbors. Fig. 2.44 shows an example of the weights between the neighboring grids and the red grid.

Moran's I [81] is a measure of spatial auto-correlation. The statistic of Moran's I test can be calculated in Eq. 2.32

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij}(x_i - \overline{x})(x_j - \overline{x})}{\sum_i (x_i - \overline{x})^2}, \tag{2.32}$$

where x is the value of interest in each location, N is the number of spatial units, i, j are the indexes of two spatial locations, w_{ij} is the weight between location i and location j, W is the sum of all w_{ij}. Value of I ranges from -1 to 1, and values significantly below $\frac{-1}{N-1}$ indicate negative spatial autocorrelation and values significantly above $\frac{-1}{N-1}$ indicate positive spatial autocorrelation [81]. To verify if there is a significant spatial auto-correlation in the data, a hypothesis test is conducted, the null hypothesis H_0 is the values are spatially independent and assigned at random among the regions, while the alternative hypothesis H_1 is the values are spatially correlated. The null hypothesis is rejected if the statistical significance (p-value) of a Moran's I score is below a given threshold. It can be calculated through estimating the distribution of z-score of I.

The average I score of $V(s)$ and $D(s)$ among the drivers is 0.5241 and 0.5551. Given the confidence level 0.95 ($p < 0.05$), for $V(s)$, the values from 99.25% drivers reject the null hypothesis, which means $V(s)$ has spatial correlation; and for $D(s)$, the values from

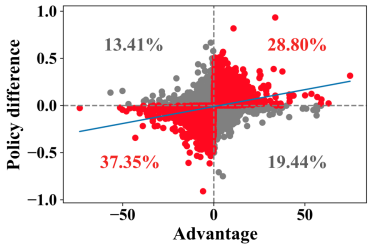


Figure 2.45: Policy difference VS. Advantage of Trending-up drivers

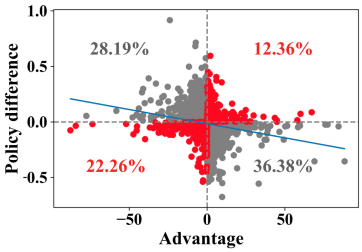


Figure 2.46: Policy difference VS. Advantage of Trending-down drivers

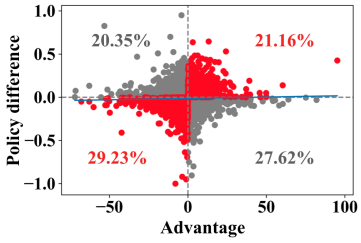


Figure 2.47: Policy difference VS. Advantage of Stabilized drivers

99.07% drivers reject the null hypothesis, which indicates $D(s)$ also has strong spatial correlation.

(2) Integrating spatial auto-correlation in advantage correlation analysis. From the results above, it is safe to conclude that both $D(s)$ and $V(s)$ exhibit spatial auto-correlations under the weight matrix designed. Thus we should take the spatial auto-correlation into account to reduce the bias. The spatial normalized value $SN(x)$ can be calculated by Eq. 2.33

$$SN(x_i) = \alpha x_i + (1-\alpha)\sum_{j \neq i} w_{ij} x_j, \tag{2.33}$$

where x is either $D(s)$ or $V(s)$. $SN(x_i)$ is a convex combination of x_i and weighted sum of that from its neighboring cells, with combination parameter $\alpha \in [0, 1]$. In this study, we employ $\alpha = 0.5$.

2.2.4 Evaluation

In this section, we apply the proposed analysis on the aforementioned real world taxi trajectory data from Shenzhen, China, to validate the established hypotheses of this work. We quantitatively evaluate the correlations between the advantage and the policy difference among the different groups of drivers and present a case study to show that how typical drivers learn experiences in a paradigm which is similar to reinforcement learning

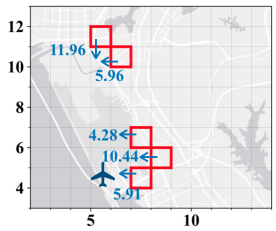


Figure 2.48: Mike's state-action pairs with greatest advantages in July

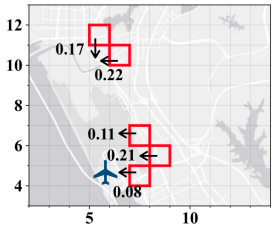


Figure 2.49: Mike's greatest policy differences between July and August

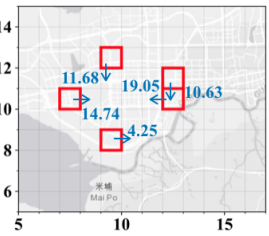


Figure 2.50: Jacob's state-action pairs with greatest advantages in July

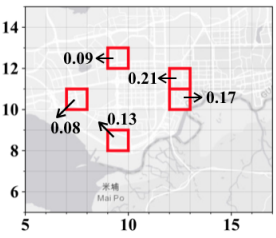


Figure 2.51: Jacob's greatest policy differences between July and August

(RL).

2.2.4.1 Experiment Settings

Following the steps discussed in Section 2.2.1.2 and extracting trips of taxi drivers, we use 6 months trajectory data in 2016, i.e., 07/2016-12/2016, with an average of around $600k$ trips per day. We conduct the experiments in two different time intervals respectively: 6am-4pm (day-time driver working hours) and 4pm-12am (night-time driver working hours). After eliminating those taxis whose records are not complete during these 6 months, there are $2,760$ valid taxis found in 6am-4pm time interval, while $2,403$ found in 4pm-12am time interval.

We apply Pearson's correlation coefficient and Spearman's rank correlation coefficient for the correlation analysis between policy difference and the advantage, and evaluate the statistical significance of the correlations to test our hypotheses.

Table 2.2: Results of correlation analysis

	Trending-up drivers	Trending-down drivers	Stabilized Drivers
Pearson's Corr	0.26	-0.21	0.023
Pearson's p-value	$6.59e^{-}12$	$2.30e^{-}25$	0.11
Spearman's Rank Corr	**0.39**	**-0.32**	0.029
Spearman's p-value	$1.17e^{-26}$	$6.85e^{-65}$	0.05

2.2.4.2 Correlation analysis

In this section, we present the correlation results between the policy difference and the advantage for each of the three groups of taxi drivers. To reduce bias in the analysis, we only consider grids (i.e., states) with sufficient visits in the data. Here we set 20 as the minimum visit count threshold, and exclude grids with fewer visits. As discussed in Section 2.2.3.3 for each driver we calculate the advantage over each state-action pair in a time slot T_0 and the policy difference of the same state-action pair in the next time slot T_1 over T_0 to understand how they adjust their strategies based on historical experiences. Here we use 3 weeks as the length of each time slot since it may take certain time for the adjustments to be observed.

Analysis Results. Fig. 2.45-2.47 shows the results of the three groups drivers, respectively. Each point in the plot represents the policy difference and advantage of a state-action pair of one driver. The x-axis is the advantage in the first time span T_0, and the y-axis is the preference difference between T_0 and T_1. The blue line is the linear regression line of the points.

Fig. 2.45 shows the results for the **trending-up drivers**. There is a positive correlation between the policy difference and the advantage, which imply the state-action pairs with larger advantages tend to have larger policy difference. In other words, the drivers are leaning towards increasing the relative visitation frequency of an state-action pair if she found that the advantage of the state-action pair was large in the previous time slot, and vise versa.

Fig. 2.46 shows the results of the **trending-down** drivers. The linear regression line of the points (blue) has a negative slope. It shows that the trending-down drivers have a negative correlation between the policy difference and the advantage, which states these drivers increase the relative visitation frequency of those state-action with smaller advantages, and vise versa, which is a counter-act with the learning process of policy-gradient

RL.

Fig. 2.47 shows the results of the **stabilized drivers**. The slope of the linear regression line is close to 0. The stabilized drivers reflect little correlation between the policy difference and the advantage. We consider that these drivers have finished the learning process and reached a stable status.

Table 2.2 provides the quantitative results of three groups of taxi drivers. For **trending-up drivers**, the Spearman's rank correlation coefficient is 0.39 with a p-value of $1.17e^{-26}$, which means that the correlation between the policy difference and the advantage is significantly positive. A similar conclusion is drawn based on the result of the Pearson's correlation coefficient. Although the Pearson's correlation coefficient is smaller, it still suggests a significant positive correlation. For the **trending-down** drivers, the Spearman's rank correlation coefficient is -0.32 with a p-value of $6.85e^{-65}$. It implies that the correlation between the policy difference and the advantage is significantly negative. The Pearson's correlation analysis results suggest the same conclusion. **Stabilized drivers** have little correlation between the policy difference and the advantage with the Spearman's rank correlation coefficient of 0.029 and a p-value of 0.05.

Correlation Analysis Summary: The trending-up drivers who improved earning efficiencies over time show a similar learning process as that of the agent in an policy gradient RL algorithm, while the trending-down drivers who worsened earning efficiencies show an opposite learning process to the learning process of the agent in a policy gradient RL algorithm. This in turn proves that (1) the trending-up taxi drivers are following the paradigm of RL effectively when learning strategies, and (2) drivers tend to be more successful in terms of their increasing earning efficiency if they better follow the learning process of RL.

The result of stabilized drivers implies that these taxi drivers may have found strategies that they believe to be "optimal". They are loyal to the strategies and not temporally

affected by the advantages. They are similar as agents in RL that have already reached the optimal status.

2.2.4.3 Case Study

In this section, we provide two concrete examples from two real drivers to help illustrate our findings in details.

(1) *A trending-up driver.* We select a driver, Mike, from the group of trending-up. Mike's earning efficiency shows a monotonic increasing trend from the first week in 07/16 to the last week in 12/16. We extracted the top 5 grids with the highest visitation frequency of Mike during July and August, as shown in Fig. 2.48. We can see that Mike likes working near the Airport. We calculated the advantage of the state-action pairs of these 5 grids. The state-action pair with the highest advantage value among the state-action pairs of each state is marked with a blue arrow in Fig. 2.48. These blue arrows show that the driver tends to get closer to the airport to get better earnings. Then, we extract the policy difference of these state-action pairs from July to August, and the state-action pair with the largest policy difference among the state-action pairs in each state are marked with black arrows in Fig. 2.49. Comparing Fig. 2.49 with Fig. 2.48, we can find that from July to August Mike increased the probability of taking those exact actions which he learned to have the highest advantage based on experiences from July. Mike maintained a similar strategy as the agent in RL, which helped him improve his earning efficiency from July to August.

(2) *A trending-down driver.* We select another driver from the group of trending-down, namely, "Jacob". Jacob's earning efficiency shows a monotonic decreasing trend from the first week in 07/16 to the last week in 12/16. We extracted the top 5 grids with the highest visitation frequency of Jacob during July and August, as shown in Fig. 2.50. Jacob likes working near the downtown area. Similarly, we calculate the advantages and

the policy difference for each state-action pair in these grids. The results are marked in Fig. 2.50 & 2.51. Comparing the results in these two figures, we can find that from July to August Jacob increased the probability of taking those actions that didn't give him high advantages in July. It is the opposite to what an agent in RL would do. Thus, the earning efficiency of Jacob was lowered from July to August.

2.2.4.4 Takeaways and Discussions

Based on our study upon a large real-world taxi trajectory dataset, we acquired promising findings about whether a taxi driver follows the learning process of RL and why different groups of taxi drivers have different earning efficiency trends over time. The takeaways are summarized:

(1) Taxi drivers, especially the ones with improving earning efficiencies, indeed follow the learning process of RL. Drivers with the different trends of earning efficiency result from the different extents to follow the paradigm of RL.

(2) Even the best drivers cannot completely follow the RL paradigm in all the scenarios. The possible reasons are that human drivers have limited memories and they do not precisely calculate the advantage over all the state-action pairs. Trending-up drivers tend to better follow the RL paradigm for those state-action pairs with low-to-medium expected rewards. The reason could be that their strategies are already (near) optimal for those high-reward state-action pairs. The improvement primarily comes from the low-to-medium reward scenarios.

Our findings establish the foundation for future research related to behavior analysis of taxi drivers. It can be used for strategy recommendations. For example, for slow-growing drivers, one can focus on helping them keep better track of their advantages so that they better follow RL and their earning efficiencies grow faster. Also, one can expect drivers to learn the best strategies in the most profitable areas quickly. Learning is efficient if drivers focus more on improving their decisions in low-to-medium reward areas.

3

Human Behavior Explanation

3.1 Explainable Generative Adversarial Imitation Learning

3.1.1 Overview

3.1.1.1 Introduction

Humans make daily decisions, based on their own "strategies" (such as taxi drivers' passenger-seeking processes and commuters' transit mode choices). It is crucial to understand what factors humans think about when making decisions, which can greatly facilitate many applications. As three examples shown in Fig. 4.1, understanding the decision-making strategies from taxi drivers, personal vehicle drivers, and urban commuters can facilitate the service providers (e.g., taxi/ride-hailing companies) to better serve the passengers, and enable the urban planners to design better road networks and transit routes to meet the needs of urban travelers.

Many real-world humans' decision-making processes (e.g., taxi passenger-seeking and transit mode choices) can be modeled as Markov Decision Processes (MDPs) [2,

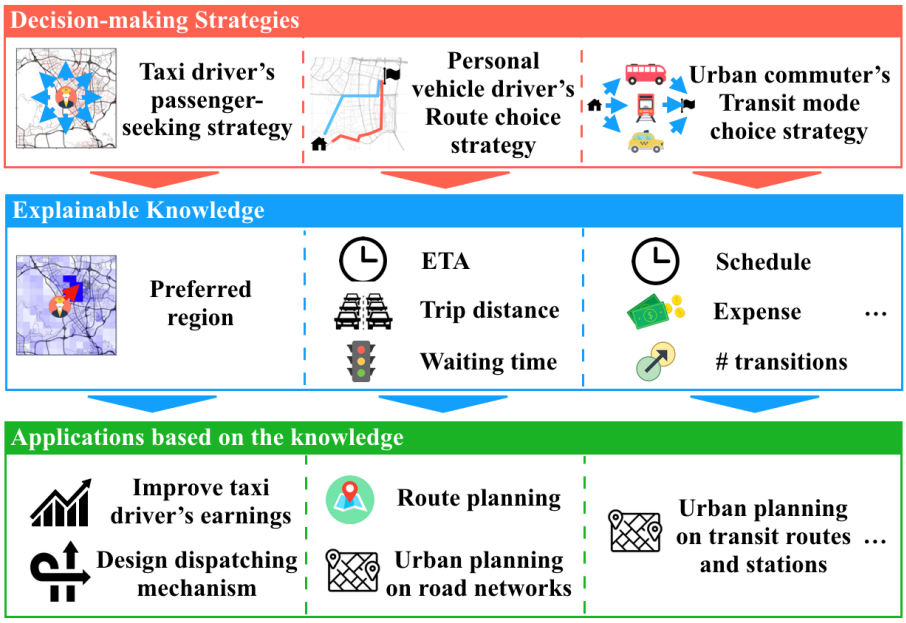


Figure 3.1: Applications with Explainable Knowledge from Human Decision-Making Strategies

3, 12, 41, 42, 43, 67, 82, 83]. In the MDP model, the human agents' decision-making strategies (e.g., the passenger-seeking strategies) can be captured by sequences of human decisions, which aims to maximize his/her total "rewards". In the literature, inverse reinforcement learning (IRL) and imitation learning (IL) techniques have been applied to recover such reward functions to learn how humans make decisions. For example, Pan et al. propose to use relative entropy based IRL to recover linear reward functions and to dissect drivers' preference dynamics over time [2]. Zhang et al. extend Generative Adversarial Imitation Learning (GAIL) [84] to conditional GAIL (cGAIL) to unveil taxi drivers' policies by transferring knowledge across taxi drivers and locations [82]. A GAIL model [84] consists of two deep neural networks (DNNs), i.e., a policy net (learning a non-linear policy function) and a reward net (learning a non-linear reward function).

However, there are significant limitations in these solutions. The IRL approaches [31, 32, 37] manually extract features to represent the linear reward function. It is likely to neglect some counter-intuitive while effective features [84]. On the other hand, although

a GAIL model [84] is able to integrate high dimensional rich feature sets and better imitate a human agent's strategy, it is hard to explain what knowledge the model has learned from human. This is due to the "black-box" nature of deep neural networks. Therefore, both types of approaches are not much helpful in understanding human agents' strategies. In recent years, a number of approaches have been proposed to interpret machine learning models, such as classifier interpretations [85, 86, 87]. However, none of them focuses on the explanation of knowledge learned by imitation learning models (e.g., GAIL) from *human-generated spatial-temporal* data, such as vehicle trajectories.

In this work, we make the first attempt to address the above challenges by proposing xGAIL, a novel explainable Generative Adversarial Imitation Learning model for learning both i) human decision-making strategies (as deep neural networks) to mimic how a human behaves, and ii) human-understandable knowledge to explain how a human (and the learned model) makes decisions. The proposed xGAIL framework consists of two novel components, Spatial Activation Maximization (SpatialAM) and Spatial Randomized Input Sampling Explanation(SpatialRISE), which extracts both global and local knowledge from a pre-trained GAIL model that has learned a human agent's decision-making strategy. Especially, we take taxi drivers' passenger-seeking strategies as an example to validate the effectiveness of our proposed xGAIL framework. Our main contributions are summarized as follows:

- We formulate human agents' decision-making processes (using taxi drivers' passenger-seeking processes as an example) as Markov Decision Processes (MDPs), and to inversely learn each agent's decision-making strategy by a Generative Adversarial Imitation Learning (GAIL) model.
- We propose an explanation framework with both global and local interpretation mechanisms, i.e., Spatial Activation Maximization (SpatialAM) and Spatial Ran-

domized Input Sampling Explanation (SpatialRISE), to explain what knowledge a GAIL model learns so as to generate a specific decision-making strategy.

- We conduct a case study using real-world taxi driver's trajectory data to validate our framework. Our analysis shows interesting results from two facets: i) global explainable knowledge of what nearby traffic condition impels a taxi driver to choose a particular direction to find the next passenger, and ii) local explainable knowledge of what key (sometimes hidden) factors a taxi driver considers when making a particular decision.

3.1.1.2 Sequential Human Decision-Making Processes as MDPs

Markov decision processes (MDPs) [88] provide a mathematical framework for modeling decision-making processes. An MDP includes an agent as the decision maker and an environment that interacts with the agent. An MDP is defined as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ represents the probability $P(s_{t+1}|s_t, a_t)$ of transiting to state s_{t+1} from s_t after taking action a_t, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function of each state-action pair, and $\gamma \in (0, 1]$ is the discount factor. An agent at a state $s \in \mathcal{S}$ makes a decision of taking an action $a \in \mathcal{A}$ following a memoryless policy π. The memoryless policy π is a function that specifies a probability distribution on the action to be executed in each state, defined as $\pi : S \times A \mapsto [0, 1]$. Taking a passenger-seeking process as an example, a taxi driver makes a sequence of decisions about which directions (as actions) to go based on his/her own decision-making strategy. The MDP components of a passenger-seeking process are highlighted as follows.

- *State* $s \in S$: A state of a taxi driver can be uniquely defined by the spatial location and time stamp.

- *Action* $a \in A$: There are 9 possible actions that a taxi driver can choose at a state s, including traveling to 8 neighboring directions, and staying at the current location.
- *Reward* $R(s, a)$: The reward that a taxi driver receives follows an inherent function $R(s, a)$ to evaluate an action a taken at a state s.
- *Policy* $\pi(a|s)$: A policy $\pi(a|s)$ of a taxi driver is a mapping from a state s to an action a, i.e., the probability distribution of choosing an action a given a state s.

As a result, a human agent's (e.g., taxi driver's) decision-making strategy can be characterized by two functions: *policy function* $\pi(a|s)$ controlling how the agent chooses an action, *reward function* $R(s, a)$ governing how the agent evaluates states and actions.

Decision-making Strategy Learning with Generative Adversarial Imitation Learning (GAIL). Given a large amount of trajectory data from a human agent (e.g., a taxi driver), each **trajectory** is defined as a sequence of decisions, namely, state-action pairs, $\tau = [(s_0, a_0), (s_1, a_1), ..., (s_L, a_L)]$, with L as the trajectory length. Generative adversarial imitation learning (GAIL) [37, 84] was proposed to inversely learn both the policy function $\pi(a|s)$ and reward function $R(s, a)$ employed by the agent. As defined in [84], the strategy learning problem can be modeled as the following constrained optimization problem, namely, finding the policy π with maximum causal entropy (eq. (3.1)), and finding the reward function R such that the expected reward of a trajectory under π matches that under the empirical policy π_E from observed data (enforcing eq.(3.2)).

MaximumCausalEntropy Inverse Reinforcement Learning

$$\max_{R} \min_{\pi} : \ -H(\pi), \tag{3.1}$$

$$\text{s.t.} : \mathbb{E}_{\pi}[R(s,a)] = \mathbb{E}_{\pi_E}[R(s,a)], \tag{3.2}$$

$$\sum_{a \in A} \pi(a|s) = 1, \forall s \in S, \tag{3.3}$$

where $H(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t(-\log \pi(a_t|s_t))]$ is the γ-discounted causal entropy π, $\mathbb{E}_{\pi}[R(s,a)] = \mathbb{E}_{\pi}\big[\sum_{t=0}^{T} \gamma^t R(s_t,a_t)\big]$ represents the expected reward of a trajectory under the policy π, and π_E (empirical policy) represents the policy observed from the collected data. GAIL [84] proves that the above maximum causal entropy inverse reinforcement learning problem is equivalent to solving a minimax problem (eq. (3.4)) with the objective as a Jensen-Shannon (JS) divergence as follows.

$$\max_{R} \min_{\pi \in \Pi} -\lambda H(\pi) + \mathbb{E}_{\pi}[\log(R(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - R(s,a))], \tag{3.4}$$

with Π as the policy probability simplex space, guaranteeing constraint eq. (3.3), λ as the Lagrangian multiplayer. As a result, a generative adversarial networks (GANs) framework [89] is naturally employed to solve the strategy learning problem with a generator network G (equivalent to the policy function π) and a discriminator network D (equivalent to the reward function R). However, the policy and reward functions are learned as two deep neural networks, thus it is hard to explain what knowledge and aggregated features the two networks have learned from human agents' trajectory data, i.e., depending on what complex factors, human agents make decisions. Below, we formally define the strategy explanation problem and outline our solution framework.

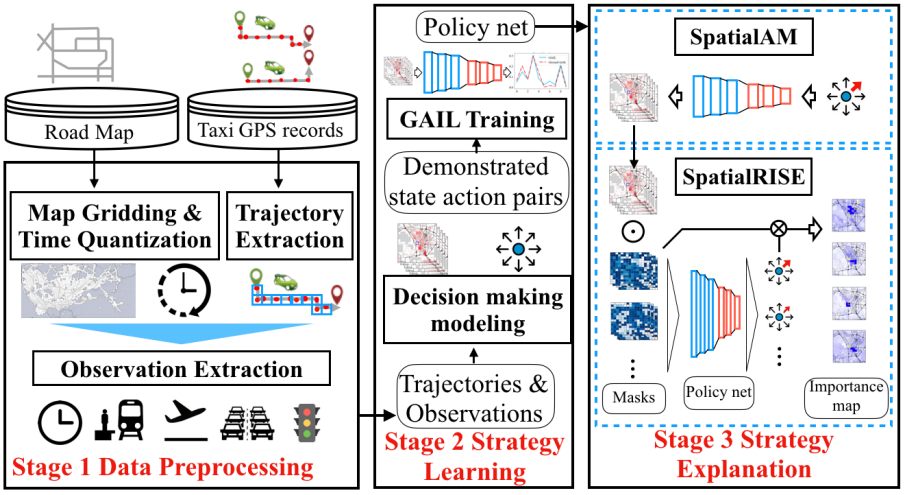


Figure 3.2: xGAIL Solution Framework

3.1.1.3 Strategy Explanation Problem and Solution

Problem Definition. We aim to extract human understandable knowledge from the learned policy (π) and reward (R) nets in the GAIL model to understand why and how a human agent (e.g., a taxi driver) makes a certain decision a at a state s.

Solution Framework. The proposed strategy explanation problem is challenging, because the policy function π learned from GAIL is a deep neural network (DNN), which as a blackbox model is hard to explain. Fig. 3.2 outlines our proposed explainable generative adversarial imitation learning (xGAIL) framework (using taxi driver passenger-seeking strategy as an example). xGAIL takes two sources of data as inputs and consists of three stages, including (1) data preprocessing, (2) GAIL, and (3) Strategy Explanation Module, which will be detailed below from Sec 3.1.3.2 to Sec 3.1.3.3.

3.1.2 Related Work

In this section, we summarize the literature from two related areas, imitation learning (IL) and explainable artificial intelligence (XAI).

Imitation learning (IL), also known as learning from demonstrations, inverse rein-

forcement learning (IRL), inversely recovers the agent's policy and reward functions from the collected demonstrations. IL approaches [31, 32, 37] have been proposed based on different principles, including maximum entropy, maximum causal entropy, and relative entropy principles [31, 32, 37]. All the approaches assume that the underlying reward function is a linear function and features have to be manually extracted. Generative adversarial imitation learning (GAIL) [84], and its extension works cGAIL [82] and adversarial IRL [90] learn the non-linear policy and reward functions as two deep neural networks (DNNs), with theoretical connections to generator and discriminator in generative adversarial networks (GANs) structure. These works either rely on manually extracted features or learn policies through black-box models (i.e., DNNs) which make the processes hard to explain the key features human agents are considering. In this work, we make the first attempt to tackle this challenge.

Explainable artificial intelligence (XAI) as an emerging topic has been extensively studied in recent years [91, 92, 93], which all aim to provide explanations of what DNNs capture. In the category of post-hoc global explanation, Activation Maximization (AM) aims to generate an input that maximizes the activation of a neuron in a network [94, 95, 96]. Karpathy et al. provide an analysis of Long Short-Term Memory (LSTM)'s representations, predictions, and error types through character-level language models [97]. Kádár et al.'s word level interpretation approach estimates the amount of contribution of individual tokens in the input to the final prediction [98]. Augasta et al. introduce a new neural network rule extraction algorithm *RxREN* to overcome the lack of explanation capability of neural network models [99]. The algorithm prunes the insignificant input neurons and constructs the classification rules only with significant input neurons based on reverse engineering technique [99]. In addition, other research focuses on local explanation. Ribeiro et al. identify an interpretable model, *LIME*, over the interpretable representation of a binary vector indicating the presence or the absence that is locally faithful to

the classifier [85]. Petsiuk et al. establish *RISE* which can generate the importance maps through tons of pixel masks [86]. Ribeiro et al. put forward high-precision rules representing local "sufficient" conditions for predictions [100]. Differing from these works, we focus on a framework to explain what GAIL model learned from human-generated spatial-temporal data.

3.1.3 Methodology

3.1.3.1 Stage 1: Data Preprocessing

In this section, we take a taxi driver passenger-seeking process as an example to illustrate the data preprocessing mechanism. The novelty of the data preprocessing is our design of the state observation. Note that the data preprocessing approaches can be easily applied to other scenarios, such as commuter transit mode choice, personal vehicle route choice, etc.

Data Description. We employ two datasets, the GPS dataset and the road map dataset, in this study.

GPS dataset. Taxi trajectory dataset records were collected from July to September in 2016 in Shenzhen, China. The dataset recorded the traces from $17,877$ unique taxis. For each taxi, a GPS point was collected every 30 seconds on average. There were a total of $51,485,760$ GPS points generated in a day. Each GPS point contains five attributes, a unique taxi ID, a timestamp, a latitude, a longitude, and a passenger indicator. The passenger indicator is a binary value with 1, indicating the taxi was occupied, and 0, indicating it was vacant.

Road map dataset. The road map data were collected from OpenStreetMap [14] for the region of Shenzhen in China, ranging from $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. There are $455,944$ road segments collected in this region.

Map and Time Quantization. The human agents (i.e., taxi drivers) traverse the spa-

tial and temporal spaces when seeking and serving passengers. We define those states that a driver can visit by i) dividing and discretizing Shenzhen city into equal side-length *(spatial)* grid cells with a given side-length $l = 0.01^\circ$ in latitude and longitude, ii) a day into 288 five-minute *(temporal)* intervals. By eliminating grid cells in the ocean and unreachable regions in the city, there are a total of $1,934$ remaining cells that are well-connected by the road network. Each cell is represented as $\ell = (x, y)$, where x and y are longitudinal and latitudinal cell indexes, respectively. A spatial-temporal state s is then uniquely defined by a spatial gird cell ℓ, a time interval t, and the day of the week d, i.e., $s = (x, y, t, d)$.

State Observation of A Human Agent. Each human agent makes a sequence of decisions to traverse geographical locations over time when seeking for passengers. Each decision (e.g., which direction to go) made by the driver is based on various features (such as traffic) in the surrounding urban environment of the nearby area, referred to as the state observation of the driver. Given a spatial-temporal state s, we model a taxi driver's observations as the state observation $\mathbf{O}_s = [\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_4, \mathcal{O}_5]$, with five statistics for the surrounding 15×15 grid cells of s, including $\mathcal{O}_1$ the number of pickups, $\mathcal{O}_2$ the traffic volume, $\mathcal{O}_3$ traffic speed, $\mathcal{O}_4$ the waiting time; and $\mathcal{O}_5$ the distances to points of interests (POIs), such as train stations, airports, shopping malls, ports, and hospitals in the city. For example, Fig. 3.4 shows an example of the traffic volume observation map $\mathcal{O}_2$ of a state (the blue box at the center).

3.1.3.2 Stage 2: Strategy learning with GAIL

Now, we introduce the structure of the generative adversarial imitation learning (GAIL) model [84] for leaning a human agent's decision-making strategy (from his/her generated data).

A GAIL model trains a generator network for the policy function $\pi(a|s)$, and a dis-

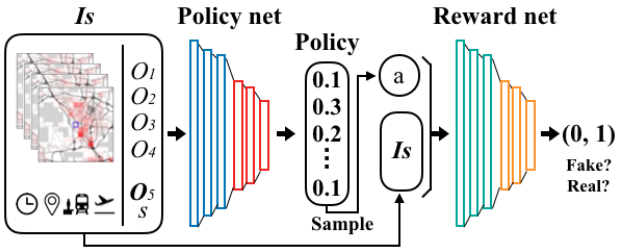


Figure 3.3: GAIL model with a policy net π and a reward net R trained as a GAN.

criminator network for the reward function $R(s, a)$ (see Fig. 3.3).

The generator network (i.e., policy) takes the state observation $\mathfrak{O}_s$, the high dimensional feature maps, as the input, and outputs the decision-making policy $\pi(a|s)$. Based on the learned policy, an action (namely, a direction to go to find the next passenger) is then randomly chosen.

The discriminator network (i.e., reward) takes both the state observation $\mathfrak{O}_s$ of state s, and the sampled action a as input, and outputs the reward signal which indicates to what degree the generated state-action pair matches the demonstrated trajectories.

When implementing GAIL, we employ convolutional neural networks [101]. For the policy and reward nets, they both consist of three convolutional layers with ReLU activation functions. Given the input state observation with the size of $5 \times 15 \times 15$ (5 channels are the number of pickups, traffic volume, speed, waiting time, and distances to POIs), we use a kernel size of 3×3 for the convolutional layers with padding of size 1. The sampling process for actions makes the entire network no longer differentiable, so that it is not trainable by backpropagation [102]. We, thus, use the Reinforcement Learning (RL) based approach [88] to train the network, i.e., using the output of reward net as signals to update the policy net.

3.1.3.3 Stage 3: Strategy Explanation with xGAIL

The trained GAIL model recovers the taxi driver's strategy. Given a state and its state

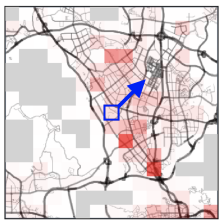


Figure 3.4: Real state
$\pi(a_1) = 0.30$

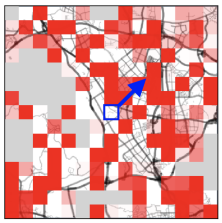


Figure 3.5: Original AM
$\pi(a_1) = 0.98$

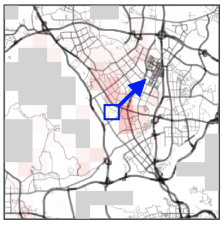


Figure 3.6: L_2 AM
$\pi(a_1) = 0.42$

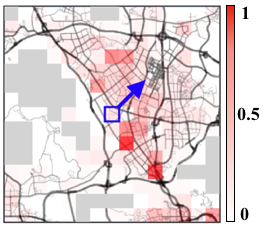


Figure 3.7: SpatialAM
$\pi(a_1) = 0.41$

observation, the policy net predicts an action just as the driver does. However, the policy and reward nets both are "black boxes". The inscrutable internal processes cause considerable difficulty in explaining why and how the nets generate that specific strategy and make that specific action. In other words, what "knowledge" the nets learned remains unknown. To solve this problem, in this section, we formally propose Explainable Generative Adversarial Imitation Learning (xGAIL), a strategy explanation framework to extract human understandable knowledge from the trained nets. Our xGAIL framework is designed to provide both global and local explanations for the learned policy and reward nets. We use the policy net as an example to illustrate xGAIL. The xGAIL framework consists of the global explanation and the local explanation. The global explanation aims to reveal the state observation $\mathfrak{O}_s^*(a)$ which leads to the highest probability of choosing a target action a, and the local explanation extracts the most effective local features.

1) SpatialAM: Global Explanation Method for GAIL.

Design Goal. Given a policy net π, the goal of the global explanation is to extract the state observation $\mathfrak{O}_s^*(a)$ that maximizes the probability of a target action a in policy $\pi(a|s)$ among all the actions. It thus can be formulated as below,

$$\mathfrak{O}_s^*(a) = \arg\max_{\mathfrak{O}_s} \pi(a|\mathfrak{O}_s). \tag{3.5}$$

Limitations of state-of-the-art works. This objective function has been extensively studied

in the literature as an activation maximization (AM) problem [94, 95, 96]. For example, the AM model from [94] aims to find the image that maximizes the likelihood of being classified as a goose, which introduces an L_2 regularization term to guarantee the obtained image is close to a real image, without overfitting. The optimal input can be obtained by gradient ascent via back propagation.

However, for our policy net explanation problem, the input traffic state observations possess intrinsic geographic characteristics, i.e., spatial auto-correlations across grids. As a result, the activation maximization model with L_1 and L_2 norm regularization cannot preserve these spatial auto-correlations in the obtained $\mathfrak{O}_{\boldsymbol{s}}^*(a)$. Fig. 3.5 3.6 show $\mathfrak{O}_{\boldsymbol{s}}^*(a)$ (in traffic volume distribution) obtained using AM without regularization and with L_2 norm regularization. Comparing to the real state observation in Fig. 3.4, neither of them captures the real traffic volume distribution. To tackle this problem, we propose Spatial Activation Maximization (SpatialAM).

Spatial Activation Maximization (SpatialAM). To enable activation maximization to output $\mathfrak{O}_{\boldsymbol{s}}^*(a)$ that preserves the spatial auto-correlation pattern presented in the real world observations, we introduce a new spatial regularization term into the AM problem (to capture the realness of a state observation) as

$$Realness(\mathcal{O}_s(a)) = -Dist(\mathfrak{O}_{\boldsymbol{s}}(a), \bar{\mathfrak{O}}_{\boldsymbol{s}}(a)), \tag{3.6}$$

where $Dist(\mathfrak{O}_{\boldsymbol{s}}(a), \bar{\mathfrak{O}}_{\boldsymbol{s}}(a))$ is the mean square distance of $\mathfrak{O}_{\boldsymbol{s}}(a)$ from $\bar{\mathfrak{O}}_{\boldsymbol{s}}(a)$, which is the mean state observation from the demonstration data, and captures what a real state observation looks like. And the objective function of SpatialAM is

$$\mathfrak{O}_{\boldsymbol{s}}^*(a) = \arg\max_{\mathfrak{O}_{\boldsymbol{s}}} \{\pi(a|\mathfrak{O}_{\boldsymbol{s}}) + \lambda \cdot Realness(\mathcal{O}_s)\}, \tag{3.7}$$

where λ is the weight of the regularization term. The introduced spatial regularization

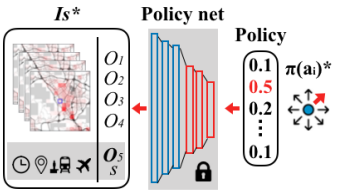


Figure 3.8: SpatialAM

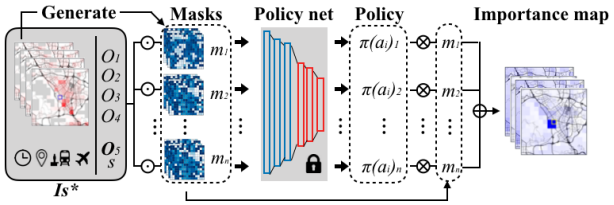


Figure 3.9: SpatialRISE

term guides the activation maximization problem to find a state observation that maximizes the probability of a, and minimizes the difference to the mean state observation $\bar{\mathcal{O}}_{\boldsymbol{s}}(a)$. Fig. 3.7 shows $\mathcal{O}^*_{\boldsymbol{s}}(a)$ obtained by SpatialAM, is clearly closer to the real state observation (and with a higher probability of 0.41 for the target action a). An illustration of SpatialAM is shown in Fig. 3.8.

2) SpatialRISE: Local Explanation Method for GAIL.

Post-hoc local interpretation approaches help us learn the local explanations of neural networks. We aim to answer the question "which areas of the input layer play important roles in producing the policies in the learned policy and reward nets". One of the basic ideas behind the local explanation is to generate an *importance map*, which can show how important each entry of the input is to the prediction of the model.

Spatial randomized input sampling explanation. Randomized Input Sampling Explanation (RISE) [86], as a local interpretation approach, can discover the importance map of the input by probing the model with randomly masked versions of the input image and obtaining the corresponding outputs. The masks are then aggregated into the importance map according to the corresponding outputs. However, when being applied to the GAIL model that learns spatial features, RISE has two limitations. First, it does not segment the input observation map based on intrinsic geographic characteristics. As a result, the high importance areas identified by RISE are large and do not align meaningfully with the functional region of the city. Second, RISE employs a bi-linear interpolation method to generate the mask values, which ignores the spatial auto-correlation of the features.

This may lead to drastically different importance values being assigned to highly similar locations in the same functional region of the city. To deal with these challenges, we propose a novel spatial importance discovery model named **SpatialRISE** to discover the importance of geographic regions with respect to a specified output in the learned GAIL model. It consists of three steps: map segmentation, mask generation, and importance map generation.

Map segmentation. As LIME [85] tries to discover the importance of the meaningful super-pixels in the image, we want to discover the importance of the functional regions of the city. To do this, we first segment the map into functional regions. Within each functional region, the observation values are expected to have strong spatial auto-correlation.

We measure the strength of spatial auto-correlation at each location using a Local Getis-Ord Gi^* statistic[103]. The local Gi^* statistic can be calculated via eq.(3.8),

$$Gi^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{n\sum_{j=1}^{n} w_{i,j}^2 - (\sum_{j=1}^{n} w_{i,j})^2}{n-1}}}, \tag{3.8}$$

where x_j is the observation value at location j, $w_{i,j}$ is the spatial neighborhood indicator between location i and j, and n is the total number of locations, $\bar{X} = \sum_{j=1}^{n} x_j/n$, and $S = \sqrt{\sum_{j=1}^{n} x_j^2/n - (\bar{X})^2}$. In this work, $w_{i,j} = 1$ if i and j are geographically neighboring to each other, otherwise $w_{i,j} = 0$. A large positive local Gi^* score indicates a hotspot (high observation values clustered), and a small negative local Gi^* score indicates a coldspot (low observation values are clustered). Thus, we can segment the map into clusters according to the local Gi^* scores of the grids given cut-off threshold. The cluster algorithm is shown in Algorithm 2.

Mask generation. With the segmented observation maps, we are able to take the spatial auto-correlation into consideration to generate masks. The mask values within each clus-

Input: observation map $\mathcal{O}$, threshold Gi^t of $|Gi^*|$;
Output: Clusters C for all grids;
$C = \{\}$, $k = 0$;
Calculate Gi^* for each grid g_i based on $\mathcal{O}$;
for Each grid g_i in the observation map **do**
    if $|Gi^*| > Gi^t$ **then**
        if g_i is neighboring to any grid in an existing cluster $c_j (0 < j \leq k)$ and have the same sign of Gi^* **then**
            Add g_i to the cluster c_j;
        else
            $k = k + 1$;
            Create a new cluster with g_i as the first grid in the cluster c_k;
        end if
    else
        $k = k + 1$;
        Consider g_i itself as an independent cluster c_k;
    end if
end for
Return the clusters $C = \{c_1, ..., c_k\}$ for all grids.

Algorithm 2: Observation map segmentation

ter have strong spatial auto-correlation. To generate mask values for the grids in each cluster, we first randomly sample an overall trend $tr \in \{1, 0\}$ for each cluster, i.e., preserving or covering the original observation values, with the covering probability p. Then, inside each cluster, we assign mask value, $m(i)$, to grid i according to eq.(3.9),

$$m(i) = \begin{cases} 1 - \alpha * random(0, 1), \text{if } tr = 1; \\ \alpha * random(0, 1), \text{if } tr = 0, \end{cases} \tag{3.9}$$

where $\alpha \in (0, 1)$ is the weight of the randomness. The mask generation method can adapt to all kinds of random distributions. In this work, we employ uniform distribution to generate random values, which makes sure that the mask values of the grids in the same cluster are within an expected range.

Importance map generation. Once we generate a set of masks, we can estimate the im-

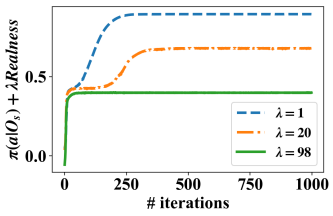


Figure 3.10: Learning curves with different λ's

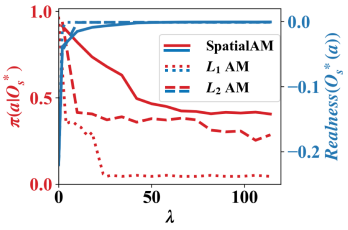


Figure 3.11: Impact of λ in SpatialAM

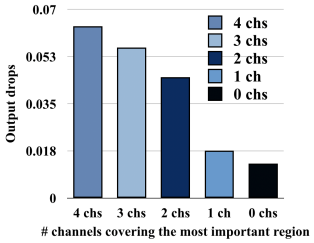


Figure 3.12: Importance map evaluation

portance map for each observation map. Since we introduce randomness in each cluster, the importance map produced by our proposed SpatialRISE can tell the pixel-wise importance.

The framework of SpatialRISE is shown in Fig. 3.9. The input of the policy net is I_s, and $\pi(a|I_s)$ is the output of the policy net regarding action a. Let $m : \Lambda \rightarrow [0, 1]$ be a random mask, and M be the population of all possible masks following distribution $\mathcal{D}$. $I_s \odot m$ is the masked input. Then the importance map $Ipt_{I_s,\pi,a}$ of I_s regarding the output of action a in policy net π can be calculated by the weighted sum of the masks with the model output $\pi(a|I_s \odot m)$ as the weight for each mask m:

$$Ipt_{I_s,\pi,a} = \frac{1}{\mathbb{E}(M)} \sum_{m \in M} \pi(a|I_s \odot m) \cdot m \cdot P[M = m]. \tag{3.10}$$

The intuition is that $\pi(a|I_s \odot m)$ is high if entries of I_s preserved by mask m are important. Empirically, we can estimate $Ipt_{I_s,\pi,a}$ by sampling a set of N masks M' according to $\mathcal{D}$:

$$Ipt_{I_s,\pi,a} \approx \frac{1}{\mathbb{E}(M') \cdot N} \sum_{m \in M'} \pi(a|I_s \odot m) \cdot m. \tag{3.11}$$

3.1.4 Evaluation

In this section, we evaluate our xGAIL framework on a pre-trained GAIL model to interpret what knowledge the policy net (learned from GAIL) has learned.

3.1.4.1 Model Performance Evaluation

We conduct experiments to evaluate the effectiveness and efficiency of our proposed SpatialAM and SpatialRISE.

SpatialAM model evaluation. We evaluate SpatialAM algorithm by comparing it with baselines and examining the impact of the choice of regularization weight λ.

Comparison with Baselines. We first compare our SpatialAM model with two baseline models, including Activation Maximization with L_1-norm and L_2-norm regularization terms [104]. Fig. 3.11 shows the comparison results about the *learned policy* and *the realness of state observation* (measured by the realness regularization term $Realness(O_s^*(a))$) with different regularization weight λ ranging from 1 to 100. It clearly indicates that when increasing λ, the policy probability $\pi(a|O_s^*)$ decreases, and the realness of state observation increases for all methods, which makes sense because λ controls how much to maximize the policy vs. maximize realness of the solution state observation. However, note that when λ is sufficiently large (i.e., $\lambda \geq 98$), all three approaches tend to the similar high realness, but SpatialAM can always find state observations with higher policy probability than the baselines. The comparisons show that SpatialAM can generate like real state observations with higher policy probabilities, thus, it provides a better view of the global explanation of what an ideal state observation looks like for the human agent to choose a target action a.

Impact and Choice of λ. Fig. 3.10 shows an example of the learning curve of SpatialAM with $\lambda = 1, 20$, and 98, respectively, the y-axis is the output of the objective function defined in eq.(3.7), the x-axis is the number of iterations. The results illustrate that the

Table 3.1: Gap between the maximum policy from real states and the policy from SpatialAM

Mean gap	Max gap	Min gap
0.2882	0.3885	0.1113

learning process of our proposed SpatialAM converges to a state observation $\mathfrak{O}_s^*$ with monotonically increased objective function by gradient ascent.

λ is designed to balance the trade-off between maintaining the spatial auto-correlation in the generated state observation, and obtaining maximum output policy. Fig. 3.11 shows the optimal output with different settings of λ, which illustrates that, with the increase of λ, the maximum policy $\pi(a|\mathfrak{O}_s^*)$ obtained by SpatialAM decreases, and the realness regularization term $Realness(\mathfrak{O}_s^*(a))$ increases. Fig. 3.11 shows that when λ is sufficiently large, i.e., $\lambda \geq 98$, the spatial regularization term converges to a large realness regularization term $Realness(\mathfrak{O}_s^*(a)) \geq -5e^{-3}$, i.e., small distance from true state observation. On the other hand, the policy $\pi(a|\mathfrak{O}_s^*)$ converges to 0.41. As a result, we select $\lambda \geq 98$.

To better illustrate the ability of SpatialAM in generating like-real observations and maximizing the policy probability of a target action, Table 3.1 summarize the statistical results, by comparing the maximum policy obtained by SpatialAM vs that from the dataset. Overall different target actions, SpatialAM can generate observations with on average 0.2882 (i.e., the mean gap) more policy probabilities than that from the dataset.

SpatialRISE model evaluation. We first quantitatively evaluate the importance map generated by SpatialRISE, then we compare spatialRISE with RISE, PixelRISE, and SolidRISE, respectively.

Importance map evaluation. The question we aim to answer in this evaluation is, "*Is the important region found by SpatialRISE really important to the maximum policy?*" The more important a region is, the greater its impact on the output policy should be. We propose a measurement that the impact of a region on the output policy can be quantified by the drop amount of the output policy when covering the region in a channel, i.e., a state

observation obtained by SpatialAM. Therefore, we make comparisons in the output policy drops between covering the most important regions found by SpatialRISE and covering other regions. Fig. 3.12 shows the results of the output drops when covering different regions. The x-axis is the number of channels, i.e., state observations, where the most important regions found by SpatialRISE are covered. For example, "3 chs" means that in 3 out of 4 channels the most important regions found by SpatialRISE are covered, while in the remaining channel a region other than the most important one is covered. Since there are multiple regions other than the most important one, we just show the maximum output drops as the y-axis in Fig. 3.12. The results prove that covering the most important regions in all 4 channels leads to the most significant output policy drop. In other words, the impact of the SpatialRISE detected regions is much greater than the impacts of other regions. Thus, the important region found by SpatialRISE is the key to the maximum policy.

Comparison experiments. We compare our proposed SpatialRISE with the following baselines:

- **RISE** [86]: masks are generated with bilinear interpolation;
- **RISE with pixel-independent masks(PixelRISE)**: The mask value in each grid is independent with each other;
- **RISE with solid cluster masks(SolidRISE)**: We first partition the underlying spatial region into clusters using Algorithm 2. The masks are generated with respect to the clusters, such that all grids in the same cluster are assigned with the same random number.

Comparison Results. For all baselines, we set the zero mask value probability $p = 0.3$. Taking the observation channel of "the number of pickups" as an example, Fig. 3.13 shows the importance map generated from the original RISE. Although it provides the

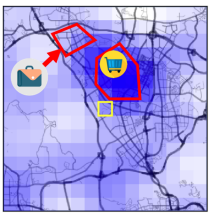


Figure 3.13: RISE

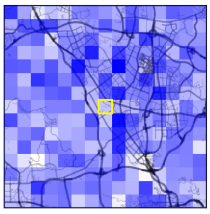


Figure 3.14: PixelRISE

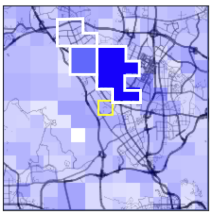


Figure 3.15: SolidRISE

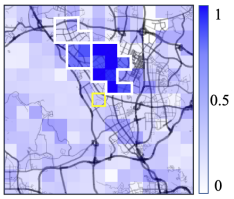


Figure 3.16: SpatialRISE

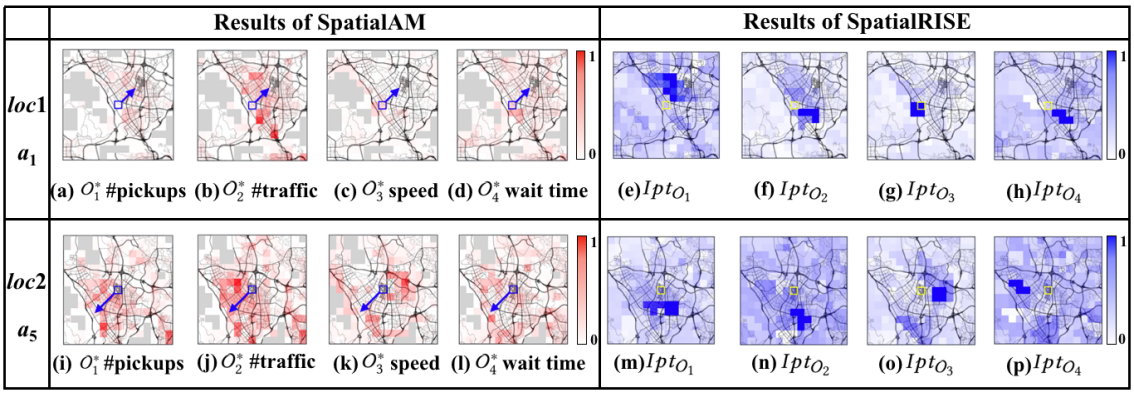


Figure 3.17: Results of SpatialAM and SpatialRISE

importance of grids, it does not take the underlying geographic information into consideration. Thus, the city functional regions, such as Dalang business center and Longhua Market marked by the red boxes in Fig. 3.13, cannot be detected. Fig. 3.14 is the importance map generated by RISE with pixel-independent masks, which scatters noise with no reliable information of importance. The reason is that the pre-trained policy net is not sensitive to the change of individual pixels with the pixel-independent masks. Fig. 3.15 represents the importance map generated via RISE with solid cluster masks. The clusters extracted by Algorithm 2 (the white boxes in Fig. 3.15) identify the two nearby functional regions, as highlighted in Fig. 3.13. However, since the mask values in the same cluster are the same, the results can only provide cluster-level importance, rather than pixel-wise importance interpretation in finer granularity. Fig. 3.16 is the importance map using our proposed spatialRISE. It is able to distinguish the geographic functional regions, as well as provide the pixel-wise importance, i.e., the importance score of each pixel integrates

both region-level and pixel-level importance information.

3.1.4.2 Explainable knowledge Learned from Passenger-Seeking Strategies

To interpret how the input observations affect the taxi driver's passenger-seeking policy, we generate optimal state observations $\mathcal{O}_1^*, \mathcal{O}_2^*, \mathcal{O}_3^*, \mathcal{O}_4^*$ in different locations, which maximize the policy on a target action via SpatialAM, and use SpatialRISE to generate importance maps for state observations. By examining the results using SpatialAM and SpatialRISE for different locations, we observe and present three interesting findings which explain how human taxi drivers make decisions for seeking passengers.

Experimental results of SpatialAM. Fig. 3.17a-d & Fig.3.17i-l present the generated observation maps maximizing the policy at location $loc1$ on action a_1 (northeast direction) and $loc2$ on action a_5 (southwest direction) respectively. Taking Fig. 3.17a as an example, Fig. 3.17a-d are four observation maps of $\mathcal{O}_1^*(a_1)$ (number of pickups), $\mathcal{O}_2^*(a_1)$ (traffic volume), $\mathcal{O}_3^*(a_1)$ (traffic speed), and $\mathcal{O}_4^*(a_1)$ (waiting time), respectively. Except for the unreachable grey area, the color map spanning from white to red corresponds to the small to large observation values. For example, a grid cell in Fig. 3.17a with white color means that in this grid cell the number of pickups is close to 0, and a grid cell with red color means the number of pickups in it is close the the maximum of the map. These plots show the global observations of the four input features under which the driver's likelihood to go northeast at $loc1$ and southwest at $loc2$ is the highest.

Experimental results of SpatialRISE. The importance maps produced by SpatialRISE for the observations generated by SpatialAM at location $loc1$ on action a_1 and $loc2$ on action a_5 are shown in Fig. 3.17e-h & Fig.3.17m-p. Except the unreachable grey area, the color map spanning from white to blue indicates the importance value ranging from 0 to 1. A grid cell in Fig. 3.17e, for example, with dark blue color (value close to 1) means that the value of $\mathcal{O}_1^*(a_1)$ at this grid cell is quite important for obtaining the maximum pol-

icy on action a_1, namely, the taxi driver considers the number of pickups in this particular grid cell heavily, when making decisions.

Knowledge learned from SpatialAM and SpatialRISE results. Integrating the results of SpatialAM and SpatialRISE, we observe the following interesting findings which explain what human agents (i.e., taxi drivers) think about when making decisions:

Finding 1: **Taxi drivers prefer the regions with large numbers of pickups.** Take the case of location $loc1$ and action a_1 as an example, Fig. 3.17a shows that there are many pickups in the grids in the direction of the northeast. Fig. 3.17e indicates that the taxi driver pays her attention to the grids in the direction of the northeast where there is Longhua Market. Recall that both Fig. 3.17a and Fig. 3.17e are based on the maximized policy of the action towards northeast, and generated time slot for the state observation in Fig. 3.17a is 6:50 pm- 6:55 pm which is the evening rush hours. Thus, the taxi driver prefers northeast direction primarily because of a large number of pickups near Longhua Market in the evening. A similar observation can be found in the case of $loc2$ and action a_5 from Fig. 3.17i & Fig. 3.17m.

Finding 2: **Taxi drivers prefer to avoid visiting regions with high traffic volume and long waiting time.** For the case of location $loc1$ and action a_1, Fig. 3.17b and Fig. 3.17d suggest that the traffic volume and waiting time in the direction of the southeast are high. Fig. 3.17f and Fig. 3.17h show that the driver cares much about the grids in the southeast direction, where there is Shenzhen North Railway Station. As a result, the high traffic volume and waiting time near the railway station propel the driver choosing to go another direction (northeast in this case). The possible reason is that the high traffic volume and long waiting time indicate traffic jams near the railway station. The driver wants to avoid approaching these areas. A similar finding can be interpreted in the case of location $loc2$ and action a_5 from Fig. 3.17j & l and Fig. 3.17n & p.

Finding 3: **Taxi drivers do not prefer regions with high traffic speeds.** From the case

of location $loc1$ and action a_1, Fig. 3.17c and Fig. 3.17g indicate that the high traffic speed in the southwest direction leads the driver to go to another direction, i.e., northeast. This is somehow counter-intuitive because a high traffic speed usually means a good traffic condition, which taxi drivers should prefer. However, in fact a high traffic speed probably also imply that the path is for vehicles only, such as highway and expressway. Therefore, there are few pedestrians. Taxi drivers know that they are unlikely to find passengers.

4

Human Mobility Signature Identification

4.1 Spatio-Temporal Siamese Networks for Human Mobility Signature Identification

4.1.1 Overview

4.1.1.1 Introduction

Given the historical movement trajectories of a set of individual human agents (e.g., pedestrians, taxi drivers) and a set of new trajectories claimed to be generated by a specific agent, the Human Mobility Signature Identification (HuMID) problem aims at validating if the incoming trajectory was indeed generated by the claimed agent. The HuMID problem has many real-world applications. Fig. 4.1 shows a few such examples. One of the major applications is automatic driver identification for taxi and rider-sharing services. According to the New York City Taxi and Limousine Commission (TLC) released statistics, there were on average $850,000$ trips taken by taxis and ride-sharing services per day

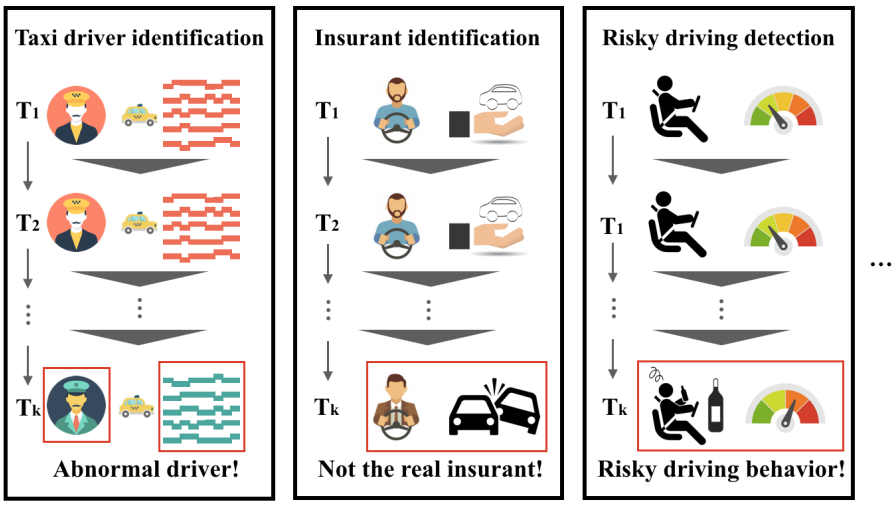


Figure 4.1: Applications of HuMID problem

in New York City in 2018 [105]. Meanwhile, the safety concerns have been raised by people recently. For example, some unauthorised drivers are reported to have taken the place of authorised drivers, and behave offensively towards passengers. Companies like Uber have taken actions to ensure the safety of passengers by enabling on-trip reporting from the APP [106, 107]. HuMID can help identify the above illegal driver substitutions as early as possible and help improve the safety of the passengers. Another example is insurer identification in the auto insurance industry. Insurance companies need to make sure that a vehicle was driven by the insured driver rather than others when the insurer filed a claim. All of these examples are downstream applications of and can benefit from solving HuMID problems.

Many prior works pay attention to the driving behavior identification problem, an instance of the HuMID problem. Hallac et al. [108] identified driver using automobile sensor data from a single turn. They monitored 12 sensors installed inside and outside the vehicle and implemented a hand-crafted rule-based classifier, which classifies up to 5 drivers. Chowdhury et al. [109] extracted 137 statistical features from smartphone sensors and used a random forest classifier to classify trajectories into small groups of

4 to 5 drivers. Kieu et al. [110] presented a multi-task learning model which captured geographic features and driving behavior features of trajectories in 3D images as input to perform trajectory clustering and driver identification. Oh and Iyengar [111] used inverse reinforcement learning in sequential anomaly detection problem. They estimated reward function for each driver and evaluated 10 individuals from GeoLife-GPS dataset and aggregated normal behaviors of taxi drivers from Taxi Service Trajectory dataset.

Nonetheless, there exist significant limitations when implementing these methods in real-world applications. First, some of these works require additional data rather than the GPS records by installing sensors on the vehicles, for example, 12 sensors in Hallac et al.'s work [108]. However, few vehicles is equipped with these additional sensors, and it will be costly to install sensors to the vehicles. Second, most existing works can only deal with a small group of drivers because they employ classification or clustering approaches. For example, Chowdhury et al. [109] employed random forest to classify the trajectories of 4 to 5 drivers, and Oh et al. [111] estimated 10 reward functions for 10 drivers by using inverse reinforcement learning. In real-world cases, the pool of drivers is large. Thus, these methods are hard to be implemented. Third, a group of previous works require that all the drivers be known in advance [108, 109, 111]. Those methods are unsuitable for applications where only a subset of the drivers is known at training.

To address these limitations, in this work we propose a Spatio-temporal Siamese networks (ST-SiameseNet) framework to identify the behavior of a large group of human agents (e.g., drivers) by using only their movement trajectory data. Since GPS devices are widely equipped on vehicles and smart phones nowadays, the data ST-SiameseNet requires can be easily collected. Also, ST-SiameseNet can deal with large groups of human agents in a single model and be used on new agents who are previously-unseen from training pool. To be more specific, we first extract different transit modes of the agents from the trajectory data. For example, there are **two** transit modes in taxi driving, i.e.,

the seeking trajectory where the vehicle has no passengers on-board, and the driving trajectory where the vehicle has passengers on-board. Besides, we extract different profile features and online features from the historical trajectories of each agent to augment the performance of ST-SiameseNet. Then, we input the trajectories together with the profile features to ST-SiameseNet pair-wisely and train the ST-SiameseNet to identify the similarity of each pair of inputs. Experiments on a real-world taxi trajectory dataset show that ST-SiameseNet outperforms all baselines in identification performances. *Our main contributions* are summarized as follows:

- We formulate the Human Mobility Signature Identification (HuMID) problem as a predictive analysis problem and, for the first time, employ the idea of the Siamese network to identify agents by their "mobility signatures" from solely their trajectory data.
- We design a novel ST-SiameseNet framework that can handle multimodal trajectory data. We also utilize both profile features and online features extracted from the agents' trajectory data to train ST-SiameseNet.
- We conduct substantial experiments using a real-world taxi trajectory dataset to evaluate the performance of our proposed ST-SiameseNet.

4.1.1.2 Problem Definition

In this section, we introduce some important definitions and formally define the problem.

Definition 4.1.1. Human-generated spatio-temporal trajectory tr. With the wide use of GPS sets, people can generate massive spatio-temporal data while they are using the devices equipped with GPS sets, e.g., the GPS records of vehicles, smartphones, smart watches, etc. Each GPS point p consists of a location in latitude lat and longitude lng, and a time stamp t, i.e. $p = \langle lat, lng, t \rangle$. A trajectory tr is a sequence of GPS points with

a label of the agent a who generated the data, denoted as $tr = \{a, \langle p_1, p_2, ..., p_n \rangle\}$, where the set of trajectories is $\mathcal{T}$.

Definition 4.1.2. Transit mode. Transit modes are defined as a set of categories of trajectories, where each category is generated under a different mobility pattern. For example, taxi driving trajectories can be categorized into two modes, i.e., with and without any passenger on-board. Private car trajectories can be grouped into commute and recreational driving trajectories, etc. In this work, we use taxi driving as the application. The seeking trajectory $\mathcal{T}_s$ is the sequence of GPS records while the vehicle is without any passengers on-board, and the driver is seeking for passengers to serve. The driving trajectory $\mathcal{T}_d$ is the sequence of GPS records while the vehicle is with passengers on-board, and the driver is taking the passengers to the destination.

Definition 4.1.3. Profile feature $\mathbf{f}_p$. Each agent has unique personal (or profile) characteristics which can be extracted from his/ her trajectory data, such as frequent start/end locations, average trip time duration, and preferred geographic area, working as different dimensions $f_{p,i}$ of the profile features, where i is the i-th dimension of these features. The profile features of each agent can be extracted in different time period. Here we denote time period as T, where T can be one hour, one day or one week, etc.

Definition 4.1.4. Online feature $\mathbf{f}_o$. Online features represent agents' mobility patterns resulting from the agent's personal judgment, experience and skills, such as speed, acceleration, turning left, turning right of each grid cell, working as different dimensions $f_{o,i}$ of the online features, where i is the i-th dimension of these features. For each trajectory, we build the online features.

Problem definition. Given a set of historical trajectories $\mathcal{T}$ collected from a group of agents A in time periods $T_0, T_1, ..., T_t$, we aim to develop a framework to verify if the

incoming trajectories $\mathcal{T}^{t+1}$ which are claimed being collected from an agent a's vehicle in T_{t+1} are indeed matching the agent a's behavior.

4.1.1.3 Data Description

The purpose of our framework is to recognize human mobility signatures with GPS records. In this work, we **use taxi driving scenario as an example** to demonstrate our techniques. However, the proposed solution can be easily generalized to other types of agents and trajectories. Our analytical framework takes two urban data sources as input, including (1) taxi trajectory data and (2) road map data. For consistency, both datasets are collected in Shenzhen, China in July 2016.

Taxi trajectory data contains GPS records collected from taxis in Shenzhen, China during July 2016. There were in total $17,877$ taxis equipped with GPS sets, where each GPS set generates a GPS point every 40 seconds on average. Overall, a total of $51,485,760$ GPS records are collected on each day, and each record contains five key data fields, including taxi ID, time stamp, passenger indicator, latitude and longitude. The passenger indicator field is a binary value, indicating if a passenger is aboard or not.

Road map data of Shenzhen covers the area defined between $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. The data is from OpenStreetMap [14] and has $21,000$ roads of six levels.

4.1.1.4 Solution Framework

Our proposed solution framework is outlined in Fig. 4.2, which takes two sources of urban data as inputs and contains two stages: (1) extracting trajectories and profile features, (2) identifying driving behavior. c

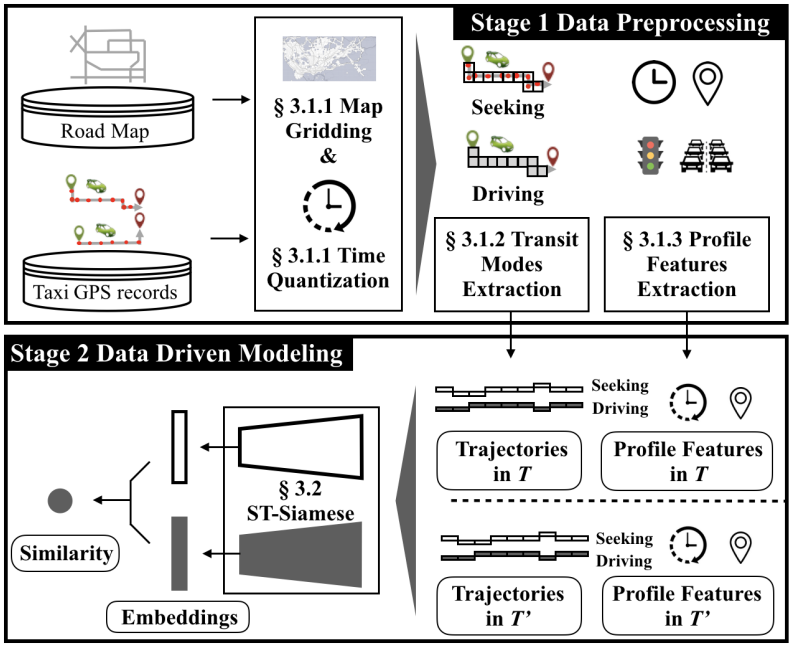


Figure 4.2: HuMID Solution framework

4.1.2 Related Work

Human mobility signature identification has been extensively studied in recent years due to the emergence of the ride-sharing business model and urban intelligence[15, 17, 21, 112]. However, to the best of our knowledge, *we make the first attempt to employ siamese network to verify human mobility signature identification.* Related work are summarized below.

Urban computing. Urban computing is a general research area which integrates urban sensing, data management and data analytic together [13, 16, 18, 19, 20]. In particular, a group of work focus on taxi operation management, such as dispatching [27, 28] and passenger seeking [10, 11, 29]. They aim at finding an optimal actionable solution to improve the performance/revenue of individual taxi drivers or the entire fleet. Rong et al.[12] solved the passenger seeking problem by giving direction recommendations to drivers. However, all of these works focus on finding "what" are the best driving strategies (as an optimization problem), rather than considering the benefits of passengers. By

contrast, our work focuses on driver identification, which can enhance the safety of passengers.

Driver behavior learning. Most existing literature on human driving behavior rely on human-defined driving style feature set. These handcrafted vehicle movement features derived from sensor data or constructed from real-world GPS data [108, 109, 112, 113, 114]. They used supervised classification, unsupervised clustering or reinforcement learning to solve problems as such driver identification, sequential anomaly detection,etc [110, 111, 112, 113, 115]. Ezzini et al. [114] addressed the driver identification problem using real driving datasets consisting of measurements taken from in-vehicle sensors, such as driver camera, smartphone are placed within the car and the driver is connected to electrodes and skin conductance response. However, such existing work require expensive sensor installed in the vehicle or excessively rely on human-defined features. Dong et al. [115] proposed a deep-learning framework to driving behavior analysis based on GPS data. They used CNN and RNN respectively to predict driver identity among 50 and 1000 drivers for a given trajectory. Such frameworks are generally require all the categories be known in advance as well as the training examples be available for all the categories, as opposed to our objective which only a subset of the categories is known at the time of training.

Siamese network. The siamese network [116] is an architecture for similarity learning of inputs, which has been widely used in multiple applications, namely but a few, vision area, unsupervised acoustic modelling, natural language processing [117, 118, 119, 120, 121]. Chopra et al.[117] learned complex similarity metrics of face verification by introducing convolutional networks to siamese networks. Hoffer and Ailon [118] proposed a variant of siamese networks, triplet networks to learn an image similarity. Hu et al.[119] applied siamese networks with convolutional layers to match two sentences. However, to our best knowledge, we are the first one to employ siamese network to human-generated

spatio-temporal data.

4.1.3 Methodology

4.1.3.1 Data Preprocessing

In this stage, we employ the GPS trajectory data and the road map data to extract the seeking and driving trajectories and online features, together with the profile features of each human agent in each time period.

Map Gridding and Time Quantization. We use a standard quantization trick to reduce the size of the location space. Specifically, we divide the study area into equally-sized grid cells with a given side-length s in latitude and longitude. Our method has two advantages: (i) we have the flexibility to adjust the side-length to achieve different granularity, and (ii) it is easy to implement and highly scalable in practice [2, 15, 33]. Fig. 2.6 shows the actual grid in Shenzhen, China with a side-length $l = 0.01^\circ$ in latitude and longitude. Eliminating cells in the ocean, those unreachable from the city, and other irrelevant cells gives a total of 1,934 valid cells. We denote each grid cell as g_i, with $1 \leq g_i \leq 1,934$, and the complete grid cell set as $\mathcal{G} = \{g_i\}$. We divide each day into five-minute intervals for a total of 288 intervals per day, denoted as $\mathcal{I} = \{\tilde{t}_j\}$, with $1 \leq j \leq 288$. A spatio-temporal region r is a pair of a grid cell s and a time interval $\tilde{t}$. Each GPS record $p = \langle lat, lng, t \rangle$ and be represented as an aggregated state $s = \langle g, \tilde{t} \rangle$, where the location $(lat, lng) \in g$, the time stamp $t \in \tilde{t}$. A trajectory of agent a then can be mapped to sequences of spatio-temporal regions, $tr = \{a, \langle s_1, s_2, ..., s_n \rangle\}$.

Transit Modes Extraction. Different transit modes can show different patterns of driving behavior. In the taxi driving scenario, seeking and driving trajectories reflect different characteristics for each human agent taxi driver. Thus, we split the trajectories into seeking $\mathcal{T}_s$ and driving trajectories $\mathcal{T}_d$ based on the status of the vehicle whether

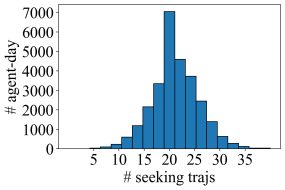


Figure 4.3: No. of seeking trajectories

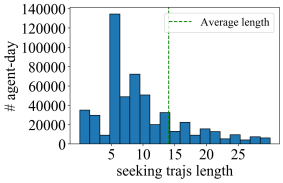


Figure 4.4: Length of seeking trajectories

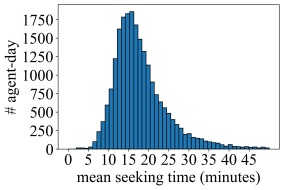


Figure 4.5: Mean seek time

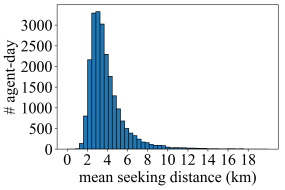


Figure 4.6: Mean seek distance

there are passengers on board. Fig. 4.4 and 4.3 illustrate the distribution of the number of driving trajectories and the length of each driving trajectory for each agent in each day, respectively. Here, $T = 1$ day. The distributions suggest that most agents have 20 seeking trajectories every day, and the average length of each seeking trajectory is around 14.03 km. The ratio between driving and seeking trips per day is approximately 1:1.

Features Extraction. Each agent has unique personal (or profile) characteristics, such as the location with the longest stay (possibly home location), daily working schedule (time duration), preferred geographic area, etc. These characteristics can be the statistical values extracted from their trajectory data. In this work, to augment the performance of driving behavior identification, we extracted the following 11 profile features for each agent in each time period of analysis. Moreover, we extract one online feature, i.e., speed, over time for each trajectory.

$f_{p,1}$ & $f_{p,2}$*: The coordinates (in longitude and latitude direction) of the longest-staying grid.* Each agent can have his/her own preference on **where** to take a break during work, thus we extract $f_{p,1}$ & $f_{p,2}$*: longest-staying grid* to represent the place where an agent takes a break. The longest staying grid is the grid where the GPS records remain unchanged for the longest time.

$f_{p,3}$ & $f_{p,4}$*: Break start & end time.* Similarly, each agent can have his/her own preference on **when** to take a break during work, thus we extract $f_{p,3}$ & $f_{p,4}$*: Break start & end time* to capture the schedule when an agent takes a break.

$f_{p,5}$ & $f_{p,6}$*: The coordinates of the most frequently visited grid.* Each agent has his/her own favorite region to go, which can help identify the agent. Thus, we extract $f_{p,5}$ & $f_{p,6}$*: most frequently visited grid* to capture the region that an agent visits the most frequently in *T*.

$f_{p,7}$ & $f_{p,8}$*: Average seeking trip distance & time.* Each agent has his/her own efficiency on looking for passengers. The experienced agents can find passengers quickly after serving a trip, while the new agents may take longer time and distance to find a new passenger. Thus, we extract $f_{p,7}$ & $f_{p,8}$*: Average seeking trip time & distance* to capture their efficiency on finding new passengers. The distribution of these two features are shown in Fig. 4.5 and 4.6, respectively, where the x-axis the average seeking distance (in km) and the average seeking time (in min) for an agent in a day, and the y-axis is the number of driver-day's. Here $T = 1$ day. From the figures, we can see that averagely agents spend 15 minutes to seek passengers within 3 km from his/her current location.

$f_{p,9}$ & $f_{p,10}$*: Average driving trip time & distance.* Each of the agent can have his/her own preference on the length of trips that he/ she serves. For example, some agents prefer to serve long trips, because they think they can earn much more at a time, and they may look for passengers near the airport or train station where the passengers have higher probabilities of asking for long trips. Some other agents prefer to look for short trips because they think they can earn money more efficiently by serving short trips. Thus, we calculate the average driving trip time and distance to capture each agent's unique preference on the length of driving trips.

$f_{p,11}$*: Number of trips served.* Each agent has his/ her own strategy on looking for passengers. The experienced agents may serve more trips in *T* than the new agents. Thus, we count the number of trips served of each agent in *T* to capture each agent's level of experience. This feature is just the number of driving trajectories in *T*.

$f_{o,1}$*: Speed.* Given a trajectory $tr = \{a, \langle(g_1, \tilde{t}_1)...(g_n, \tilde{t}_n)\rangle\}$, we also extract an online feature by calculating the speed information, denoted as v, for each data point in each

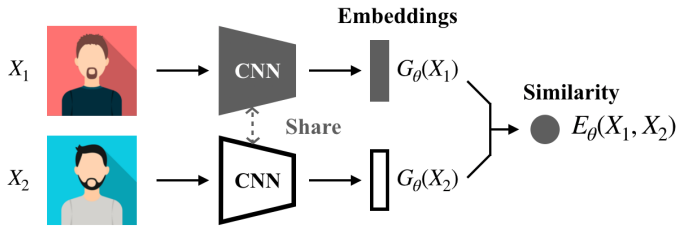


Figure 4.7: Siamese Network

trajectory to extract more information about driving behavior. The updated trajectory would be $\tau = \langle (g_1, \tilde{t}_1, v_1) ... (g_n, \tilde{t}_n, v_n) \rangle$, where the set of trajectories is $\tilde{\mathcal{T}}$.

4.1.3.2 Data Driven Modeling

The increasingly pervasiveness of GPS sensors has accumulated large scale driving behavior data, which makes it possible to identify human mobility signature from trajectories. However, two challenges arise in achieving this goal. First, the pool of agents is large but the number of trajectories per agent is limited and a large number of new agents rise up every day, thus the data is sparse and maybe only subset of the data can be seen during training. Second, as a type of sequential data, trajectories has temporal dependencies which needs to be learned. We outline how we tackle these two main challenges next.

Siamese networks. To address the first challenge, we employ the siamese networks[117, 122]. Siamese networks train a metric to measure the similarity (or dissimilarity) from data, where the number of categories is very large or even not known during training, and where the size of training samples for a single category is very small. The key idea of the siamese networks is to find a function that maps the input patterns X into a lower-dimensional target space E_θ to approximate the "semantic" distance in the input space, where similar inputs are closer and dissimilar inputs are separated by a margin. Learning the dissimilarity metric is done by training a network, which consists of two identical sub-networks with shared weights. Fig. 4.7 shows an illustration of this structure. In

particular, the end-to-end dissimilarity metric learning is replicated twice (one for each input) and the representations $G_\theta(X_1)$, $G_\theta(X_2)$ are used to predict whether the two inputs belong to the same category. A commonly used optimization function for siamese networks training[117] is :

$$\begin{aligned} \min_{\theta} & -((1-Y)L_s(E_\theta(X_1,X_2)^i) + Y L_d(E_\theta(X_1,X_2)^i)) \\ & \text{s.t. } E_\theta(X_1,X_2) = \|G_\theta(X_1) - G_\theta(X_2)\|, \end{aligned} \tag{4.1}$$

where θ is the weights of the neural network, $Y = 0$ if the inputs X_1 and X_2 belong to the same category and $Y = 1$ otherwise, $E_\theta(X_1, X_2)^i$ is the i-th sample, which consists of a pair of inputs and a label (similar or dissimilar), L_s is the partial loss function for a similar pair, L_d is the partial loss function for an dissimilar pair.

Long short-term memory (LSTM) networks. The second challenge is that trajectory data has temporal dependencies. Therefore, we employ LSTM networks[123], which are capable of learning long-term dependencies for sequential data $(x_1, x_2, ..., x_T)$. LSTM sequentially updates a hidden-state representation by introducing a memory state C_t and input gate i_t, output gate o_t and forget gate f_t to control the flow of information through the time steps. At each time step $t \in \{1, 2, ..., T\}$, the hidden-state vector h_t as:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t, \end{aligned} \tag{4.2}$$

where W_x represents the weights for the respective gate(x) neurons and b_x is the bias

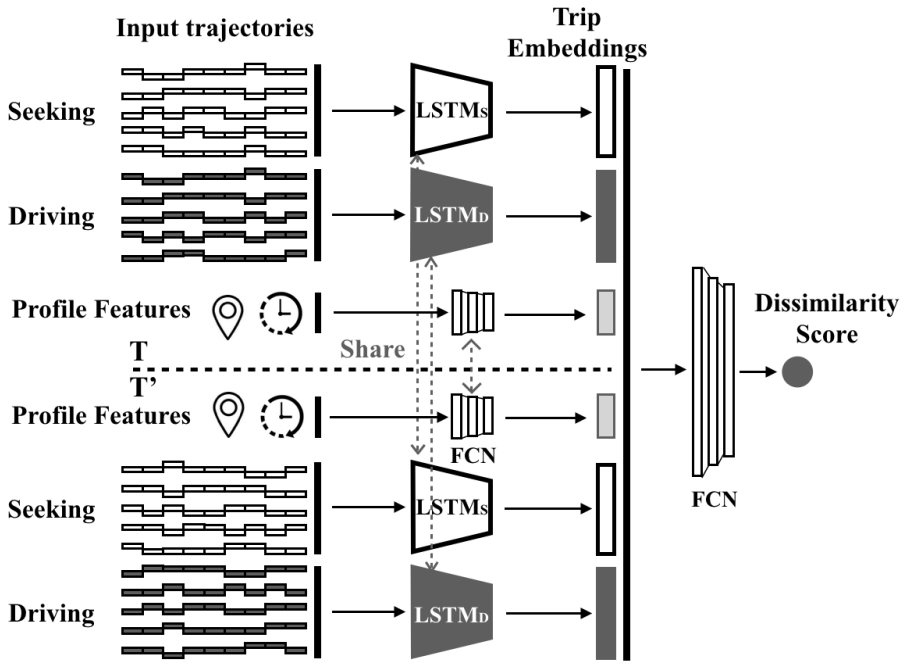


Figure 4.8: ST-SiameseNet framework

for the respective gate(x). Since the trajectory data is a sequence of $(g, \tilde{t}, v)$, we employ LSTM to learn the embeddings of trajectories in the framework of siamese networks.

ST-SiameseNet. Given those two challenges, we present a spatio-temporal representation learning method to verify human mobility signature identity by implementing siamese networks with LSTM, named as ST-SiameseNet. The purpose of this work is to learn the representation of trajectory with limited data and to use representation of trajectory data to compare or match new samples from previously-unseen categories (e.g. trajectories from agents not seen during training). To represent the feature of trajectory and learn the dissimilarity of driving behavior, we incorporate LSTM and fully-connected networks (FCN) into the middle layer of ST-SiameseNet, which is briefly depicted in Fig. 4.8.

We first extract profile features for each agent and the online feature for each data point in the trajectory. And then randomly select a pair of seeking trajectories $\tilde{\mathcal{T}}_{s,1}$ and $\tilde{\mathcal{T}}_{s,2}$, a pair of driving trajectories $\tilde{\mathcal{T}}_{d,1}$ and $\tilde{\mathcal{T}}_{d,2}$ (all the trajectories contain the online feature) and

a pair of profile features $\mathbf{f}_{p,1}$ and $\mathbf{f}_{p,2}$ in each time period. In the original siamese networks, there are two sub-networks with identical weights. To a further step, we introduce six sub-networks where each two identical sub-networks share the set of weights since we have three types of inputs, i.e., $\tilde{\mathcal{T}}_{s,1}$ and $\tilde{\mathcal{T}}_{s,2}$ would share the weights of $LSTM_S$, while $\tilde{\mathcal{T}}_{d,1}$ and $\tilde{\mathcal{T}}_{d,2}$ use the same $LSTM_D$ to learn the representation. Since each agent has a vectorized profile features in each time period, here we implement fully-connected layers as a *profile-learner* to project the profile features. ST-SiameseNet learns the driving behavior from seeking, driving trajectories and profile features respectively and aggregates the embedding layers with a sequence of fully-connected layers, i.e. *dissimilarity-learner* as the dissimilarity metric. Differing from previous works [117] that use the L_1 norm to approximate the "semantic" distance, we utilize neural networks (as a more powerful function) to learn the dissimilarity.

The learning process minimizes the binary cross entropy loss that drives the dissimilarity metric to be small for pairs of trajectories from the same agent, and large for those from different agents. To achieve this property, we pose the following ST-SiameseNet optimization problem:

$$\begin{aligned} \min_{\theta} & -(y \log(D_\theta(X_1, X_2)) + (1-y)\log(1 - D_\theta(X_1, X_2))), \\ & \text{s.t. } X_1 = (\tilde{\mathcal{T}}_{s,1}, \tilde{\mathcal{T}}_{d,1}, \mathbf{f}_{p,1}), X_2 = (\tilde{\mathcal{T}}_{s,2}, \tilde{\mathcal{T}}_{d,2}, \mathbf{f}_{p,2}), \end{aligned} \tag{4.3}$$

where $y = 0$ if the trajectories belong to the same agent and $y = 1$ if the trajectories come from two different agents, $D_\theta(X_1, X_2)$ is the prediction probability of how likely the trajectories are from two different agents.

Algorithm 3 shows the training process of the ST-SiameseNet model. During the training process, we apply the gradient descent approach to update parameters $\boldsymbol{\theta}$, with learning rate α and a predefined i_{max}, (i.e. the total number of iterations). We first extract profile features $\mathbf{f}_p$ from trajectories $\mathcal{T}$ for each agent. And then compute the online feature

$f_{o,1}$, i.e. speed information in each grid cell and update trajectories with the online feature from $\mathcal{T}$ to $\tilde{\mathcal{T}}$. Moreover, we split trajectories into seeking trajectories $\tilde{\mathcal{T}}_s$ and driving trajectories $\tilde{\mathcal{T}}_d$ for each agent. Since ST-SiameseNet pair-wisely trains the data, we randomly select a pair of trajectories, either from the same agent or from different agents in two time periods, with equal probability in each iteration. Next, we update ST-SiameseNet parameters θ by using Eq 4.3, with α as the step size (Line 7).

Input: Trajectories $\mathcal{T}$, initialized parameters $\boldsymbol{\theta}$, learning rate α, max iteration i_{max}.
Output: A well trained ST-SiameseNet with parameters $\boldsymbol{\theta}$.
1: Extract profile feature expectation vector $\mathbf{f}_p$.
2: Calculate the online feature $f_{o,1}$ and update trajectories from $\mathcal{T}$ to $\tilde{\mathcal{T}}$.
3: Split seeking $\tilde{\mathcal{T}}_s$ and driving $\tilde{\mathcal{T}}_d$ trajectories.
4: Sample a pair of trajectories with the online feature $\tilde{\mathcal{T}}_{s,i}$, $\tilde{\mathcal{T}}_{d,i}$ and a pair of profile features $\mathbf{f}_{p,i}$.
5: **while** iter $< i_{max}$ **do**
6: Calculate gradient $\nabla g(\boldsymbol{\theta})$ using Eq 4.3.
7: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla g(\boldsymbol{\theta})$.
8: **end while**

Algorithm 3: ST-SiameseNet Training

4.1.4 Evaluation

In this section, we demonstrate the effectiveness of our proposed method by utilizing GPS records collected 10 workdays from 2197 taxis in Shenzhen, China in July 2016. We compare our model with other baseline methods, analyze the generalization of our approach and evaluate the importance of transit modes and profile features for each agent.

4.1.4.1 Evaluation Metrics

To evaluate the performance of our proposed model and baseline methods, we measure accuracy, precision, recall and F_1 score against the ground truth among labels. In our implementation, the dissimilarity score threshold is set to 0.5. If it is less than 0.5, we

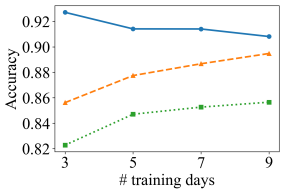


Figure 4.9: Models accuracy across days

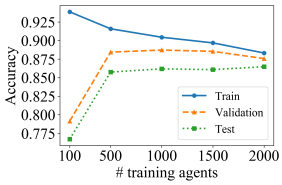


Figure 4.10: Models accuracy across agents

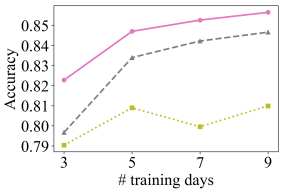


Figure 4.11: Transit modes across days

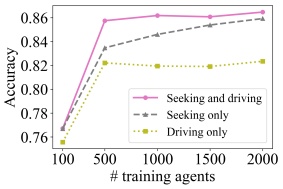


Figure 4.12: Transit modes across agents

consider that the trajectories belong to the same agent. Otherwise they are from different agents. Note the threshold can be tuned on different datasets. In particular, precision is intuitively the ability of the classifier not to confuse different agents. Recall shows the ability of the classifier not to miss pairs of different drivers. The F_1 score is a weighted average of the precision and recall.

4.1.4.2 Baseline Algorithms

We compare the performances of our method against the following baseline algorithms.

- **Support Vector Machine (SVM).** Taigman et al. [122] tested the similarity between faces using a linear SVM. Here we utilize the set of profile features as input, described in section 4.1.3.1. We conduct absolute difference between profile feature vectors of two agents and then employ SVM to classify whether these two agents are same.

- **Fully-connected Neural Network (FNN).** Fully-connected neural network is a basic classification or regression model in deep learning. Here we concatenate all the trajectories in two time periods from two agents together as the inputs of the neural network. In addition, we compare the model accuracy with and without features.

- **Naive Siamese Network.** Chopra et al. [117] used a Siamese architecture for face verification. Here we train a network which consists of two identical fully

connected networks that share the same set of weights. We concatenate all the trajectories in two time periods from each agent and evaluate the baseline with and without features.

- **ST-SiameseNet-L_1.** To evaluate the advantages of using FCN than a predefined function in learning dissimilarity, we replace FCN with the L_1-norm distance to approximate the "semantic" distance.

4.1.4.3 Results

Comparison results. We compare our ST-SiameseNet with the baseline models in terms of precision, recall and F_1 score. All the models train with trajectories from 500 agents in 5 days, validate with trajectories from the same agents as training set but in another 5 days and test with trajectories from 197 new agents in the latter 5 days. Similar to the training dataset, we uniformly sample two sets of trajectories from the same agent or different agents in two time periods during validation and testing. Table 4.1 shows the evaluation metrics from all the methods. It is clear that our approach achieves the best performance. SVM outperforms other baseline models using profile features but is worse than our model. This is because SVM is not able to model sequential inputs and the aggregation will lose information of driving behavior. With profile and basic features added, both FNN and Siamese FNN work better, indicating that features can provide useful information in both models. However, all of the deep learning and machine learning models perform worse than our model, since ST-SiameseNet has a more effective ability to capture the information of sequential inputs by using LSTM. In addition, ST-SiameseNet-L_1 performs worse than ST-SiameseNet with FCN to learn the dissimilarity, showing that L_1 norm has limited ability to learn the dissimilarity between two identities. In particular, the F_1 score of ST-SiameseNet is over 0.85, which is significantly higher than all baselines.

Table 4.1: Average F_1, recall and precision on real-world dataset and comparison with baselines

Methods	Precision	Recall	F_1 score
ST-SiameseNet	**0.8710**	**0.8317**	**0.8508**
SVM	0.8100	0.7661	0.7874
FNN (with features)	0.6112	0.6298	0.6195
FNN (without features)	0.5266	0.5470	0.5365
Naive Siamese (with features)	0.6137	0.6707	0.6407
Naive Siamese (without features)	0.5580	0.5657	0.5617
ST-SiameseNet-L_1	0.8052	0.7775	0.7910

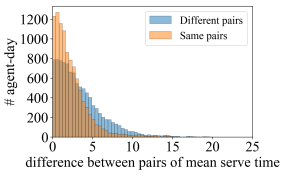


Figure 4.13: Mean serving time

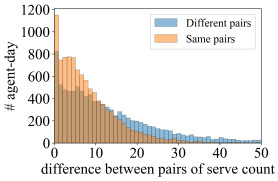


Figure 4.14: No. of serves per day

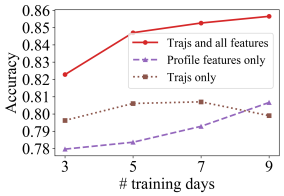


Figure 4.15: Features across days

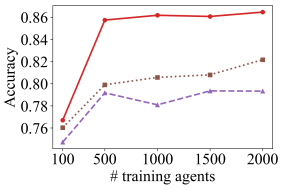


Figure 4.16: Features across agents

Model generalization. We evaluate different design choices of our model on classification accuracy. Similar to the previous experiments, we use the trajectories of the first 500 agents in 10 consecutive days as training and validation sets and vary the split ratio.

Impact of different number of days. First, we vary the number of days in the training set of the 500 agents to $N_{day} = 3, 5, 7$ and 9 respectively. We train trajectories of 500 agents from Day 1 to Day N_{day}, validate trajectories of the same 500 agents from Day $(N_{day}+1)$ to Day 10, test trajectories of the new 197 agents from Day $(N_{day}+1)$ to Day 10. Fig. 4.9 depicts the training, validation and test accuracy across different days. As more days are added to the training dataset, the training accuracy decreases slightly, while validation and test accuracy gradually increase, indicating that larger datasets can help with overfitting problem. In addition, when the number of days extending from 3 to 5, both the validation and test accuracy have a dramatically increase, while the validation and test accuracy have a small increase after adding more days, indicating that trajectories of 500

agents from 5 days contain enough information to learn the similarity of agents.

Impact of different number of agents. we also vary the training dataset size by using a subset of the agents. Subsets of sizes N_{agents} = 100, 500, 1000, 1500 and 2000 agents in 5 days are used. We train trajectories of N_{agents} agents from Day 1 to Day 5, validate trajectories of the same N_{agents} agents from Day 6 to Day 10, test trajectories of new 197 agents from Day 6 to Day 10. Fig. 4.10 shows the training, validation and test accuracy across different number of agents. With more agents added to the training dataset, the neural networks have better generalizability and can have a better performance when seeing new data. There is an enormous increase of validation and test accuracy when the number of training agents growing from 100 to 500, indicating that the neural networks benefits from the increase of diversity of agents. However, similar to the impact of different number of days, the validation and test accuracy are flattening when adding more agents to the training pool, showing that trajectories of 500 agents from 5 days are sufficient to learn the mobility signatures of agents.

Importance of transit modes. Transit modes can show different driving habits among different agents. Seeking and driving trajectories are two typical transit modes, defined based on the situation of the vehicle whether any passenger on-board. Different agents would have different strategies to seek passengers [2]. In this section, we would test if each of the transit mode can contribute to identify the drivers' behavior. As can be seen from Fig. 4.11 and 4.12, both seeking and driving trajectories have the ability of discriminating the same and different agents. All the accuracy of seeking trajectories are higher than driving trajectories, which is consistent with human intuition that different agents have different strategies when seeking passengers, while agents do not have the choice of destination when passengers on board. With both seeking and driving trajectories included, ST-SiameseNet performs the best by integrating the information of both seeking and driving trajectories. Fig. 4.11 and 4.12 also show the same trend as Fig. 4.9 and 4.10

that models of seeking trajectories only and driving trajectories only benefit from larger dataset.

Importance of features. In this section, we evaluate the importance of features by comparing our model with and without features. Each agent has unique personal characteristics which can be extracted from his/her trajectory data over a time period [2]. In addition, Speed information of each trajectory are able to capture agents' driving behavior [115]. Fig.4.13 and 4.14 describe the distribution of two profile features, mean serving time and number of service trips. The orange bar depicts the absolute difference of profile features between same agents, while the blue bar otherwise. There is an obvious difference between same agents and different agents, indicating the profile features are able to identify the similarity of driving behavior.

From Fig. 4.15 and 4.16, we can see that our ST-SiameseNet with trajectories and features works the best comparing with the model with profile features only as well as with trajectories only in different number of training days and different number of training agents. In particular, the model with profile features only gets the worst performance, indicating that the aggregation may lose information of driving behavior. Besides, if we only use trajectories as inputs i.e. $tr = \langle s_1, s_2 ... s_n \rangle$ ($s = \langle g, \tilde{t} \rangle$), all the test accuracy across days and agents are also lower than our ST-SiameseNet with both trajectories and features included, probably because some statistical information, such as mean seeking distance, mean seeking time, number of serves, cannot be captured by LSTM. In addition, Fig. 4.15 and 4.16 shows the same trend as Fig. 4.9 and 4.10 that the neural networks benefit from larger datasets. Overall, with raw trajectory data and extracted features working together, we can learn the driving patterns from trajectory data more effectively and verify the agent more accurately. Note that we only extract 11 profile features and 1 online feature, which requires little human work of feature engineering comparing with [109], which extracted 137 statistical human-defined features.

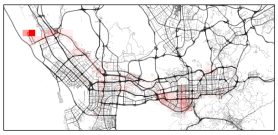


Figure 4.17: Driver 1 in Case 1

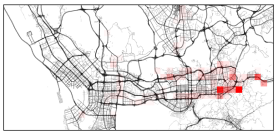


Figure 4.18: Driver 2 in Case 1

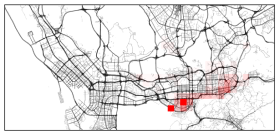


Figure 4.19: Driver 3 in Case 2

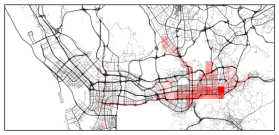


Figure 4.20: Driver 4 in Case 2

4.1.4.4 Case Studies

To further understand how ST-SiameseNet identifies agents' behaviors, we investigate individual agents' cases to show what factors ST-SiameseNet considers when identifying the agents. Four case studies are presented.

Cases 1: identifying different drivers. First, we show an example of two randomly selected human agent taxi driver, driver 1 and driver 2. We extract their trajectories and profile features on July 4^{th}, 2016, then, our proposed ST-SiameseNet consumes the trajectories and features and produces a dissimilarity score of **0.99**, which means ST-SiameseNet identifies that this pair of inputs is from two different agents. To figure out what factors that ST-SiameseNet consider to identify them, we visualize the heat map of their visitation frequency on that day to each grid of the city in Fig.4.17&4.18. The darker red color in the grid indicates higher visitation frequency. Fig.4.17&4.18 illustrate that driver 1 and driver 2 have significantly different active regions. Driver 1 likes working in the west part of the city, especially near the airport, while driver 2 prefers to work in the east part near the downtown area. The difference in the active regions of driver 1 and driver 2 helps ST-SiameseNet identify them. Fig.4.21 shows the comparison of the profile features of driver 1 and driver 2, which illustrates that they have significantly different feature values on $f_{p,2}$*: the longest staying grid id in latitude direction* and $f_{p,6}$*: the most frequently visited grid id in latitude direction*. This is consistent with the finding from the heat map, i.e., the difference in their active regions.

Cases 2: identifying different drivers with similar active regions. ST-SiameseNet

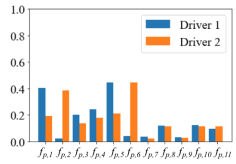


Figure 4.21: Driver 1 & driver 2 Profile Features

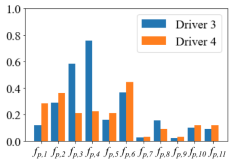


Figure 4.22: Driver 3 & driver 4 Profile Features

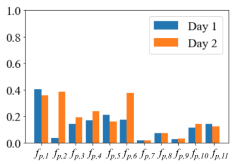


Figure 4.23: Case 3 profile features

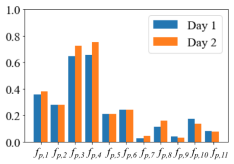


Figure 4.24: Case 4 profile features

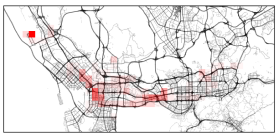


Figure 4.25: Day 1 in Case 3

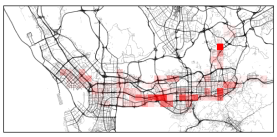


Figure 4.26: Day 2 in Case 3

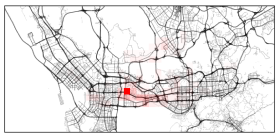


Figure 4.27: Day 1 in Case 4

Figure 4.28: Day 2 in Case 4

can also identify different drivers even if their active regions are similar to each other. We select another case of identifying different drivers from our data randomly. Fig.4.19&4.20 show the heat map of the visitation frequency of driver 3 and driver 4, which indicate that driver 3 has similar active region as driver 4. They both like working near the downtown area. And our proposed ST-SiameseNet successfully identifies them with a dissimilarity score of **0.88**, which means they are significantly different. Fig.4.19&4.20 show that driver 3 and driver 4 have similar active region near the downtown area, and the difference on $f_{p,2}$*: the longest staying grid id in longitude direction* and $f_{p,6}$*: the most frequently visited grid id in longitude direction* is small as shown in Fig.4.22. However, they have significantly different profile feature values on $f_{p,3}$ *&* $f_{p,4}$*: Break start & end time.*, and this information has been discovered by ST-SiameseNet, thus driver 3 and driver 4 can be identified by ST-SiameseNet.

Case 3: identifying *abnormal* behavior of one driver ID. Apart from the case of identifying different drivers, ST-SiameseNet can deal with the case of identifying abnormal driving behavior of ”one” driver. Our dataset contains only the licence plate of each vehicle. Therefore, abnormal behaviors of the same vehicle can might suggest a change

of driver. Here, we study a driver's behavior in 2 days from July 5^{th} to July 6^{th} 2016. Let's call this driver "John". Our ST-SiameseNet produces a dissimilarity score of **0.84** for the trajectories in these 2 days, which indicates that John's behavior changes significantly from day 1 to day 2. To figure out how John's behavior changed significantly, we plot the heat map of his visitation frequency in Fig.4.25 & 4.26, from which, we find that his active region changes from the west part of the city in day 1 to the east part in day 2. Also, Fig.4.23 shows the profile features in these 2 days. Most of the profile features change significantly, e.g., $f_{p,2}$*: the longest staying grid id in longitude direction*, $f_{p,6}$*: the most frequently visited grid id in longitude direction*, $f_{p,3}$ *&* $f_{p,4}$*: Break start & end time.* We also study a few more days after day2, the behaviors are similar to that in day2, thus, this abnormal behavior after day 1 appears to be the result of a new driver operating the vehicle after day 1.

Case 4: identifying *normal* behavior of one driver ID. For the normal behavior of a driver, ST-SiameseNet can identify it correctly. Here, we study a driver's behavior in 2 days from July 5^{th} to July 6^{th} 2016. Let's call this driver "Mike". Our ST-SiameseNet produces a dissimilarity score of **0.03**, which indicates that Mike's behavior remains consistent from day 1 to day 2. The heat maps of his visitation frequency in Fig.4.27 & 4.28 illustrate his active region does not change. Also, his profile features in these 2 days as shown in Fig.4.24 remain stable.

5

Smart Cloud Commuting System

5.1 Feasibility Study of Smart Cloud Commuting System

5.1.1 Overview

5.1.1.1 Introduction

In most urban cities today, there are two primary modes of transit: i) *Public* transit services such as buses, subways which run along fixed routes with fixed timetables, and have limited coverage areas. These limitations mean that one cannot take public transport between any two arbitrary points in a city. ii) *private* transit services such as taxis, shared-van shuttles, (mobile app-based) ride-hailing services (e.g., Uber or Lyft) are largely "on-demand" – although their service may not be immediate or "real-time". However, taxi and ride-hailing services can be expensive, limiting them mostly for *ad hoc* use, namely, occasional short trips.

It is optimistic for us to imagine that, in the near future, there will be autonomous vehicles [1] (AVs) on the road networks either for commerce or private use. So far there

[1] Colloquially known as "self-driving cars" – however in our study we will use the term AVs to refer to not only passenger cars, but also "self-driving" shuttles, vans or busses; namely, AVs of *varying* sizes.

are some exciting news about the application of AVs, for example, Voyage Auto, Optimus Ride and Waymo One have deployed robo-taxi systems in Florida, California and Arizona. Voyage Auto employs AV taxis to shuttle residents in a large retirement community in Florida. TuSimple, Kodiak, Ike Robotics, and Pronto. AI are developing long-haul autonomous driving system for trucks. Nuro.AI, Starship Technologies, Refraction AI and others are developing smaller, slower speed vehicle systems designed for last mile delivery (from a local warehouse to customer's home or business) of groceries and packages[124]. The emergence of autonomous vehicles although will offer new potentials to address the challenges facing the current urban transit systems, and challenge and transform how we view and design public and private transport systems in future smart cities. For instance, with their autonomy, would it still make sense to take "self-driving" cars to work, but have them spend most time parked, when in fact they can go somewhere by themselves? We envisage a forward-looking, ambitious and disruptive cloud commuting based transport system – *smart cloud commuting system* (SCCS) – for *future* smart cities based on *shared* AVs. Employing giant pools of AVs of varying sizes, SCCS seeks to supplant and integrate various modes of transport – most of personal vehicles, taxis, and low ridership public buses used in today's private and public transport systems – in a *unified*, *on-demand fashion*, and provides passengers with a *fast, convenient*, and *low cost* transport service for their *daily commuting* needs.

We postulate the four key aspects of *system efficiency gains* that could potentially be achieved in a smart cloud commuting system with shared AVs. This work constitutes a first attempt at exploring the feasibility and efficiency gains of the proposed SCCS; due to space limitation, we focus primarily on the *temporal multiplexing gain through time-sharing of AVs*. To this end, we model SCCS as a queueing system with passengers' trip demands (as jobs) being served by the AVs (as servers). Using a 1-year real trip dataset from Shenzhen China, we quantify (i) how various design choices – such as the number

of shared AVs and number and locations of *depots* (where idle AVs are stationed) – affect the passenger waiting time and vehicle utilization; and (ii) how much system efficiency gain (e.g., in terms of number of AVs and vehicle utilization) can be attained through SCCS.

- Utilizing a large-scale taxi trip dataset, we develop generative models to capture the arrival and service patterns of urban taxi trip demands over different time periods of the day.
- By modeling SCCS as an $M/G/k$ queuing system, we propose an theoretical framework to estimate the average waiting time of all passengers, given the total number of AVs and the number/locations of depots.
- We investigate the impacts of different design choices, e.g., number of AVs and number/locations of depots, on passenger waiting time and vehicle utilizations.
- We quantify the temporal multiplexing efficiency gain of time-sharing AVs achieved via SCCS, and compare that with the current urban taxi system. The evaluation results obtained using the 1-year taxi trip dataset demonstrate that the proposed SCCS can serve the trip demands with 22% less vehicles and 37% more vehicle utilization.

5.1.1.2 Motivation and Problem Definition

In this section we first motivate the proposed SCCS. We then lay out a general queuing system model for SCCS for studying its feasibility and quantifying its potential efficiency gains.

Smart Cloud Commuting System (SCCS). As alluded in the introduction, today's urban transit systems suffer many well-known shortcomings. Taking taxis as an example, Fig. 5.1 shows the number of on-road taxis in 3 days from $03/04/2014 - 03/06/2014$ for each 5-minutes time interval in Shenzhen, which indicates on average more than 60% of

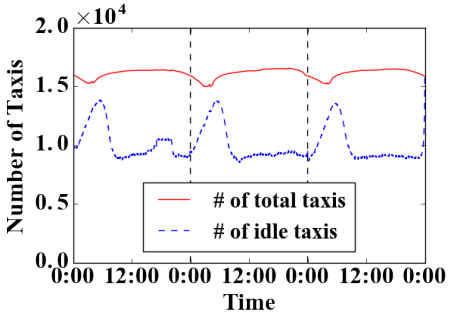


Figure 5.1: Idling taxis

taxis are idle over time. Now imagine a (perhaps not-so-distant) future where we live in a smart city with autonomous vehicles or "self-driving" cars. How would the transport systems, both public and private, be designed in such a smart city? What transport services would be needed or plausible? Our envisaged SCCS is a bold attempt to re-imagine and re-design transport for future smart cities by fusing information technologies with AVs to offer a new kind of *mobility-as-a-service* that targets more specifically *daily commuting needs* for most (if not all) users in cities and metro areas (urban and suburban). As shown in Fig.5.2, in SCCS, each AV is controlled by (centralized) dispatch servers residing in the cloud. Once a passenger requests a trip, the cloud servers will arrange an AV to pick up and send the passenger to the destination. When a trip demand is completed, the vehicle can be re-used for other passengers. In this work, we introduce the SCCS implemented with centralized servers, and in our future work, we will study the implementation of SCCS with decentralized cloud computing system. Our proposed SCCS can also benefit public transportation system with high-capacity AVs, e.g., autonomous buses. To integrate public transportation in SCCS, the high-capacity AVs can be assigned to pick up a group of passengers who can share the trips along the way. Employing giant pools of shared AVs of varying sizes, SCCS aims to provide users with a fast, convenient, and low cost transport service to meet their daily commuting needs. The *scale* and the resulting *abilities to maximize system efficiencies* via shared AVs differentiate our envisaged SCCS

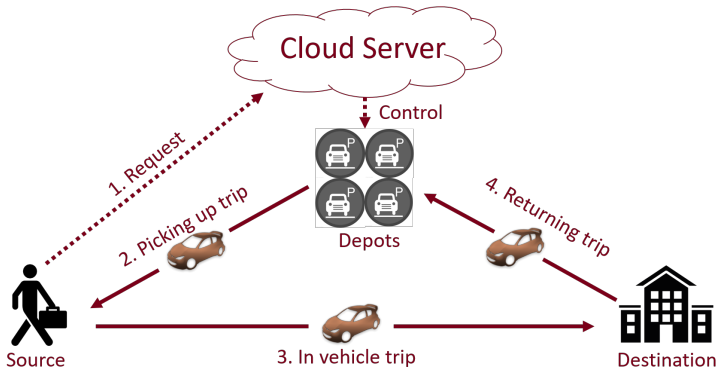


Figure 5.2: Framework of SCCS

from today's ride-hailing services, which are designed primarily to serve *ad hoc* trips.

We postulate the following *four key aspects* of system efficiency gains that could potentially be achieved in a smart cloud commuting system with shared AVs. (i) *Temporal multiplexing gain through time-sharing of AVs:* by leveraging "bursty" travel demands and sharing of AVs over time, the number of AVs needed would be significantly less than what would be if every user had his or her personal AV. This is analogous to the statistical multiplexing gain attained by a packet-switched data network. (ii) *Payload multiplexing gain through ride-sharing among users*: By utilizing AVs of varying sizes to enable ride-sharing among users (similar to today's car-pooling, shared shuttle or transit services, but leveraging the autonomy of AVs), the number of AVs needed can be further reduced. (iii) *Elastic demand gain through smart trip scheduling*: Many travel demands are elastic in nature (a trip to a store for grocery shopping now may not be crucial and thus can be delayed, say, for 30 minutes). Even for peak hour travel demands, as long as a user can reach her destination within a desired time window, the trip can be scheduled dynamically to leverage such elasticity to achieve additional system efficiency gain. (iv) *Road network efficiency gain through intelligent control of AVs:* With fewer vehicles on the road through shared AVs, road congestion can be alleviated or avoided, thus shortening trip times. Road network efficiency gain can be further increased by packing more AVs

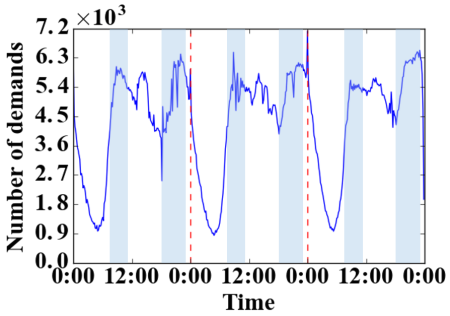


Figure 5.3: Request pattern

during peak demands (e.g., by reducing inter-car spacing) without creating safety issues, and by intelligent routing of AVs through less congested roads.

As a *first* attempt at studying the feasibility of the envisaged SCCS, in this work we focus primarily on the first aspect of the system efficiencies, namely, *temporal multiplexing gain through time-sharing of AVs*, that can be potentially achieved through SCCS. In particular, by modeling SCCS as a queueing system, we investigate how various design choices – such as the numbers of vehicles and the number/locations of depots – affect the quality of services (QoS) of passengers (e.g., waiting time) and the overall system performance (e.g., vehicle utilization). For this study, we utilize a real-world, taxi trip dataset from Shenzhen, China over a period of one year. One interesting and important feature of this dataset lies in that due to the limited area coverage (and the fact that the public transit capacity cannot meet the demands during the *peak hours*), many residents in the city rely on taxis for daily commuting needs (see Fig. 5.3). This feature enables us to study the feasibility of the proposed SCCS to meet daily commuting needs and compare its system performance with that of the existing taxi system.

Modeling SCCS as a Queuing system. SCCS can be viewed as a queuing system. Passengers request for commute services from SCCS. Their requests will be placed in a queue, if the servers (i.e. AVs) are busy. Fig.5.4 shows the queuing model of SCCS, an arrival event is a request received from a passenger, and a service event is the process of

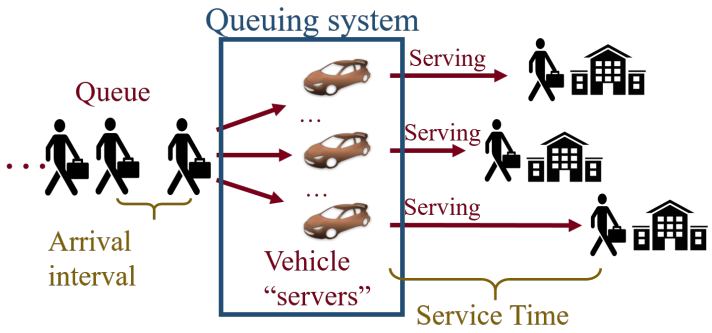


Figure 5.4: Queuing system

an AV taking the passengers to the destination. As a queueing system, there are three components charactering the system performances, including the *arrival pattern*, *service pattern* and *number of servers*.

Arrival pattern is the distribution of the arrival events coming into the queuing system. We can use arrival rate and arrival interval to capture the arrival pattern of a queuing system. *Service pattern* captures the distribution of the service time.

Definition 5.1.1 (**Arrival interval** A)**.** The arrival interval is the time period between each two successive trip requests.

Definition 5.1.2 (**Arrival rate** λ)**.** The arrival rate is the number of trip requests arriving the system within a unit time slot.

Definition 5.1.3 (**Service time** S)**.** The service time is the time period when a self-driving vehicle is dispatched to serve a passenger.

If the passengers' requests arrive the queue while all of the AVs are busy, the requests will be placed in a queue to wait for the next available AV. The waiting time indicates how long a passenger waits in a queue, which characterizes the quality of experience of the passenger in SCCS.

Definition 5.1.4 (**Waiting time** W)**.** The waiting time is the time period from the arrival of a passenger request to an AV being dispatched to the passenger.

Problem Definition. Thanks to the fast development of location sensing technologies, the increasing prevalence of embedded sensors inside mobile devices, vehicles has led to an explosive increase of the scale of urban mobility datasets, including the trip demands data of passengers in urban areas.

Definition 5.1.5 (**Trip demand**). A trip demand of a passenger indicates the intent of a passenger to travel from a source location src to a destination location dst from a given starting time t_s with an expected trip duration Δt, which can be represented as a 4-tuple $\langle src, dst, t_s, \Delta t \rangle$.

Fig. 5.3 shows the temporal distribution of urban taxi trip demands for each 10-minute time interval in Shenzhen from $03/04/2014 - 03/06/2014$, which exhibits a clear diurnal pattern. Such pattern is driven by the daily commuting needs between residential and working locations. Given such strong diurnal pattern, we divide each day into a few time intervals, and focus on the daily dynamics of trip demands over intervals.

Problem definition. Given the total number of available self-driving vehicles k and the number of depots d, we aim to (1) estimate the impact of design choices (in k and d) on passenger waiting time and vehicle utilization; and (ii) evaluate the efficiency gains of SCCS comparing to the current taxi system, in terms of numbers of vehicles needed and the vehicle utilization.

5.1.2 Related Work

To the best of our knowledge, we are the first to propose a Smart Cloud Commuting System (SCCS) for future smart cities with AVs, and quantify its feasibility and efficiency gains. In this section, we introduce two research areas that are related to our work, including (1) mobility-on-demand system, and (2) urban computing.

Mobility-on-demand system (MoD). MoD ([125, 126, 127, 128, 129, 130, 131])

is an emerging concept in solving urban transportation problems, such as unbalanced supply-demand rates and traffic congestion. MoD aims to provide transit supplies, such as shuttle/taxi services according to dynamic urban trip demands. In [125], authors design a simulation platform to explore the performance of autonomous vehicle based MoD system under various vehicle dispatching models. In another work [126], a general mathematical model is proposed, which could make real-time assignment decision in high-capacity ride-sharing system. This model is designed to handle a large number of passenger demands and dynamically generate optimal assignment solution to urban trip demands. In [127] and [128], authors propose two spatial queueing-theoretical models, that capture salient dynamic and stochastic features of customer demand, for Autonomous mobility-on-demand system which has autonomous vehicles in it. In [132, 133, 134, 135], the authors envisioned mobility systems with AVs, and analyzed the performance of the system via simulation. [136] provides a review of recent studies on investigating the impacts of AVs on travel behaviour and land use. Differing from these works with focus on the (ride-sharing) dispatching algorithms for load balancing of vehicles, we employ real world data (rather than simulation) to analyze the underlying trip demand patterns with queuing theory and evaluate design trade-offs and efficiency gains under a unifying SCCS framework.

Urban Computing is a thriving research area which integrates urban sensing, data management and data analytic together as a unified process to explore, analyze and solve crucial problems related to people's everyday life [15, 16, 17, 18, 137, 138, 139, 140, 141, 142, 143]. For examples, [15] presents a data-driven optimization framework to deploy charging stations and charging points with the goal of minimizing the seeking and waiting time of electric vehicle drivers. [137] develops novel models to predict future crowd flow traffic in subway stations. [141] introduces a method to estimate the travel time in a road segment using sparse trajectories data. [138] proposes a model to discover urban

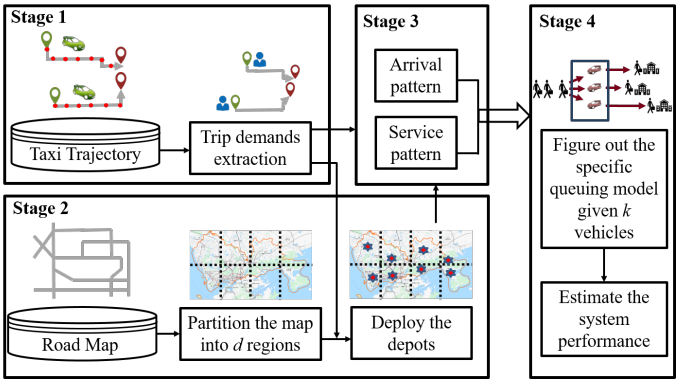


Figure 5.5: Framework of Feasibility Study of SCCS

function zones by exploring latent activity trajectory data. In [142], the authors propose a method to diagnose the noises environment in New York city by extracting ubiquitous data over the city. Differing from these works, in this work, we propose a future smart cloud commuting system (SCCS) with shared autonomous vehicles, and quantitatively evaluate the feasibility and efficiency gains of SCCS.

5.1.3 Methodology

In this section, we introduce our design model of SCCS given the total number of vehicles k and the number of depots d, and provides an analytical framework for analyzing the system performances and passenger quality of experience.

5.1.3.1 Framework

Fig. 5.5 illustrates our solution framework, that takes two sources of urban data as inputs and contains four key analytical stages: (1) trip demands extraction, (2) depots deployment, (3) arrival and service pattern extraction (4) system performance evaluation.

- **Stage 1 (Trip demands extraction)** This stage aims to extract the passengers' trip demands from the collected taxi GPS data. In our datasets, each taxi trajectory consists of a sequence of time-stamped GPS points, where a GPS point is collected

every 40 seconds on average. A GPS data point includes the time stamp, latitude, longitude, and binary indicator (indicating if a passenger is aboard). Moreover, the raw trajectory data are noisy, with spatial errors from the groud-truth locations, due to the accuracy limit of the GPS devices. By cleaning the taxi GPS data, we can extract the passenger taxi trips, indicated by four key elements: (1) starting location src, (2) ending location dst, (3) starting time t_s, (4) trip duration Δt. As a result, each trip represents a passenger demand.

- **Stage 2 (Depots deployment)** Given the number of depots d and the number of AVs k, this stage aims to identify the depot locations and assign AVs to depots. First, the urban area is divided into d grids with equal sizes. Second, the trip demands extracted in stage 1 can be aggregated into each grid based on the source locations. Then, for each grid with trip demands, we will deploy a AV depot. To reduce the dispatching distance, the depot location is obtained by the average geo-location of all trip source locations inside the grid. If the location is not exactly on a road segment, the depot location will be shifted to the nearest road network.

- **Stage 3 (Arrival/Service pattern extraction)** With a particular SCCS design (from stage 2), this stage will examine the arrival and service patterns. The trip requests arrive in a sequence of time stamps, i.e., $\{t_{s_1}, t_{s_2}, ..., t_{s_m}\}$. We will quantify the arrival pattern of such time sequence. Moreover, with all trip durations (as system service times), we will characterize the service pattern.

- **Stage 4 (System performance estimation)** With generative models for arrival and service patterns of the urban trip demands, we can naturally view the taxi service system as a queuing system, with trip demands as the customers and taxis as the servers. In Stage 4, by modeling the SCCS as an $M/G/k$ queueing system, we will quantify the average waiting time of passengers and vehicle utilization.

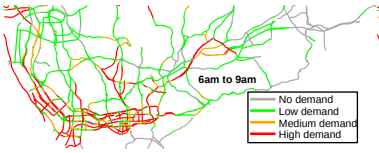


Figure 5.6: Heat map of starting location

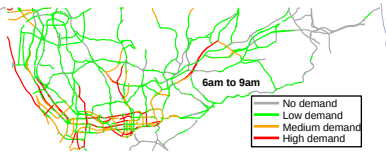


Figure 5.7: Heat map of ending location

Figure 5.8: Shenzhen road map

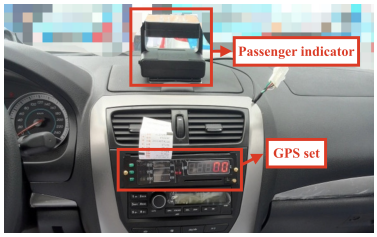


Figure 5.9: GPS set and passenger indicator on taxis

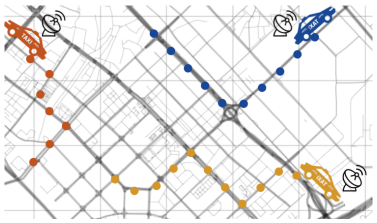


Figure 5.10: GPS data generated by taxis

5.1.3.2 Data Description

Our analytical framework takes two urban data sources as input, including (1) taxi trajectory data and (2) road map data. For consistency, both datasets are collected in Shenzhen, China in 2014. We introduce the details of these datasets below.

Taxi trajectory data are GPS records collected from taxis in Shenzhen, China during 2014. There were in total $17,877$ taxis equipped equipped with GPS sets and passenger indicators as shown in Fig.5.9, where each GPS set generates a GPS point every 40 seconds on average. The passenger indicator will be pulled down if there is a passenger aboard, and it sends binary values indicating if a passenger is aboard or not to the GPS set. Overall, a total of $51,485,760$ GPS records are collected on each day, and each record contains five key data fields, including taxi ID, time stamp, passenger indicator, latitude and longitude. The passenger indicator field is a binary value, indicating if a passenger is aboard or not. Note that, in this work, the results are from the data collected in 2014, and similar results can be obtained with new data collected in 2016.

Road map data. In our study, we use Google GeoCoding [144] to retrieve a bound-

ing box of Shenzhen, which is defined between $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. The covered area covers a total of $1,300km^2$. Within such a bounding region, we crawl road map data in Shenzhen from OpenStreetMap [14]. The road map data contain six levels of road segments, which are detailed in Table 5.1 and visualized with different colors in Fig.5.8.

Table 5.1: Road Map Data in Shenzhen

Type	Counts	Type	Counts
Motorway	563	Secondary	868
Trunk	258	Tertiary	1,393
Primary	745	Unclassified	16,829

5.1.3.3 Stage 1: Demands Extraction

In stage 1, we clean and extract the urban trip demands from the raw trajectory data.

Trajectory data cleaning. The trajectory data are noisy in nature. First of all, the GPS locations are with errors of around 15 meters. Secondly, there are GPS points outside the bounding box of Shenzhen. We conduct two steps to clean the noisy trajectory data, including map-matching and spatial filtering. *Map-matching* is a process that project the noisy GPS locations back to the road segments, which has been extensively studied in the literature We apply the map-matching technique [145] to our dataset. Secondly, we apply a simple *spatial filtering* step to remove GPS records that are outside the bounding region of Shenzhen.

Trip demand extraction. The passenger indicator field in the taxi trajectory data is the key enabler to extract the taxi trip demands. A taxi trip can be represented as a sequence of taxi GPS points with the passenger indicator as 1. The first and last GPS locations of the taxi trip capture the source/destination locations (src, dst) of a trip demand, and the corresponding time stamps characterize the trip starting/ending time t_s/t_e. The trip

duration can be obtained as the elapsed time from t_s to t_d, i.e., $\Delta t = t_e - t_s$. Once we have all trip demand tuples $\langle src, dst, t_s, \Delta t \rangle$, we observe that there are a small number of trip demands with extremely short or long trip durations. From the size of the bounding region of Shenzhen and the road map, any trip could be done within 2 hours (including the rush hours with traffic congestion). Moreover, people would not take a taxi trip shorter than 2 minutes in general. Thus, we simply filter out those noisy taxi trips longer than 2 hours or shorter than 2 minutes, which may be due to the issues with hardware or data collection processes. Note that in this work, each demand can be from either one individual passenger or a group of passengers sharing the entire trip. Without loss of generality, we assume each demand is from one passenger.

After the two steps, we obtain a total of $595,501$ daily trip demands from our trajectory data. Fig.5.6 and Fig.5.7 show the geo-distributions of source and destination locations in Shenzhen during the morning rush hours 6–9AM on March 6th, 2014.

5.1.3.4 Stage 2: Depots deployment

Given the number of depots d and total number of available vehicles k, our system deployment model works as follows: (1) road map partitioning, (2) depot placement, (3) vehicles assignment.

Step 1: Road map partitioning. We first get the boundary of Shenzhen from OpenStreetMap, which is defined between $22.44°$ to $22.87°$ in latitude and $113.75°$ to $114.63°$ in longitude. Then, we partition the area of the city into d grids with the sizes.

Step 2: Depot placement. After the regions are divided, we try to deploy one depot in each region, and totally d depots will be deployed. First, we aggregate the trip demands extracted in stage 1 into each grid. In SCCS, the request in a grid will be served by the depot in that region. We allocate those demands into grids based on their source locations. Then, to reduce the dispatching distances, in each grid, the center location of

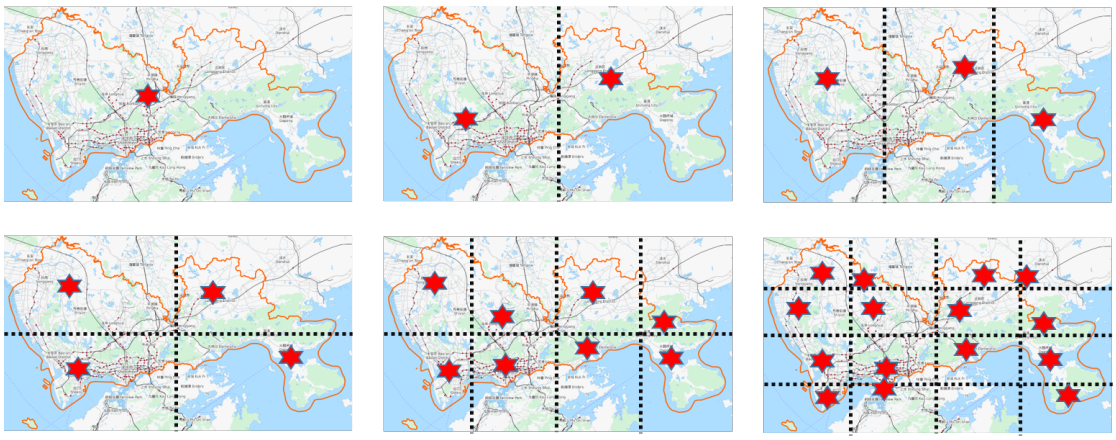


Figure 5.11: Depot placement in Shenzhen

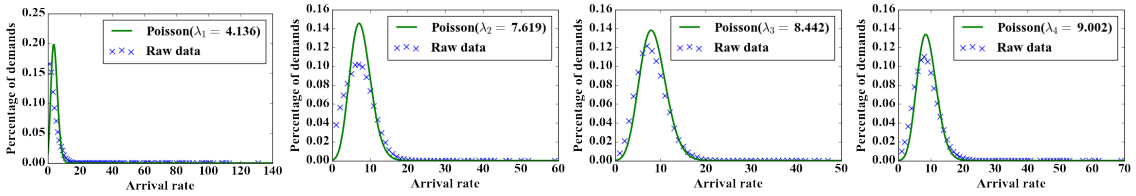
all the source demand locations are calculated to place the depot. Moreover, if the center source locations is not on the road network, it will be shifted to the nearest road segment. Fig.5.11 shows the result of road map partition and depot deployment. Note that one region is in the ocean, and we do not deploy a depot in that region.

Step 3: Vehicle assignment. After deploying the depots, the vehicles are assigned to each depot according to the portion of demands in the region. Let N be the total demands in the urban area, N_i be the number of demands in region i. The total number of vehicles assigned to region i is thus $k_i = k \cdot N_i/N$.

5.1.3.5 Stage 3: Arrival/Service pattern

SCCS can be viewed as a *queuing system*. Each trip demand and the corresponding trip represent a customer arrival event and a service event, respectively. Self-driving vehicles are the servers in the system. Now we characterize the arrival pattern and service pattern from the trips.

Arrival pattern analysis. We chose the time unit as one second, and count the number of arrived trip demands over each second in demand data we obtained from Stage 1. Fig.5.12 shows the distributions of the arrival rate in four different intervals of a day. The



12am-6am 6am-12pm 12pm-6pm 6pm-12am

Figure 5.12: Arrival rate (#requests/s)

Table 5.2: Parameters of arrival rate distributions

Time slot	12am-6am	6am-12pm	12pm-6pm	6pm-12am
λ	4.1375	7.6189	8.4415	9.0023

x-axis represents the arriving rates and the y-axis is the percentage of demands. The blue dots are obtained from original demands data, which nicely fit Poisson distributions. The green curves are the best fitting curves with Poisson distribution. The parameters λ's of Poisson distributions are the mean arrival rates, which are listed in Table 5.2 for different time intervals in a day.

Service pattern analysis. As shown in Fig.5.2, the service time of an AV include three time intervals. The first part is *pickup time*, namely, the passenger sends a request to the cloud servers to request a trip service. The cloud servers arrange a vehicle to pick the passenger up, if there is an available vehicle in the depot, otherwise, the passenger would wait in the queue. After the vehicle picked up the passenger, it will take the customer to the destination, during which the passenger experiences *in-vehicle time*. When the trip is completed, the vehicle returns to the nearest depot to the passenger dropoff location, which is the *return time*.

Note that a complete service time include all three time intervals, i.e., pickup, in-vehicle, and return times. Though passenger does not experience the return time, it is counted, because the vehicle is still "reserved" and cannot serve other passengers (on the trip back to the depot)[1].

[1]Note that the system can be further designed to allow vehicles to direct pick up the next passengers

Table 5.3: Average service time

# Depots	1	2	3	4	8	16
$\overline{S}$**(min)**	58.45	51.67	49.80	41.31	31.60	29.38

Since each request will be served by a vehicle from the depot in the source region, and the destination of the demand may be in a different region, a vehicle balancing approach is required. We adopt a simple schedule-based approach for vehicle rebalancing: Every 12 hours, the vehicles will be rebalanced to the initial numbers of vehicles. Moreover, the on-road travel time can be estimated by OSRM API [146] from one place to another. Thus, the picking up time and the returning time of each demand can be estimated by the API.

To extract the service time pattern from the demand data, we choose the unit time as minute. Taking $k = 12000$ as an example, Fig.5.13 show the distributions of service time given different number of depots: 1,2,3,4,8,16 depots, in the 12pm-6pm time slot on March 5th in 2014. The x-axis represents the service time and the y-axis is the percentage of demands. The black dots are from the raw demand data, which cannot be fitted by a simple distribution. Hence, the service pattern follows a general distribution, denoted as G in queueing theory. The average service times with different number of depots are listed in Table 5.3.

5.1.3.6 Stage 4: Estimating the system performance

Now, we are in a position to introduce our queuing theory based approach to estimate the average waiting time in SCCS, given the number of available vehicles k.

We have shown that the trip demands arrival rate follows a Poisson distribution, but the service pattern is general. When k vehicles are available in SCCS, we can denote

without going back to depot, which require more complex system design model. To simplify our feasibility and performance gain analysis, we adopt this simple model, and leave it for our future work to evaluate more complex system design.

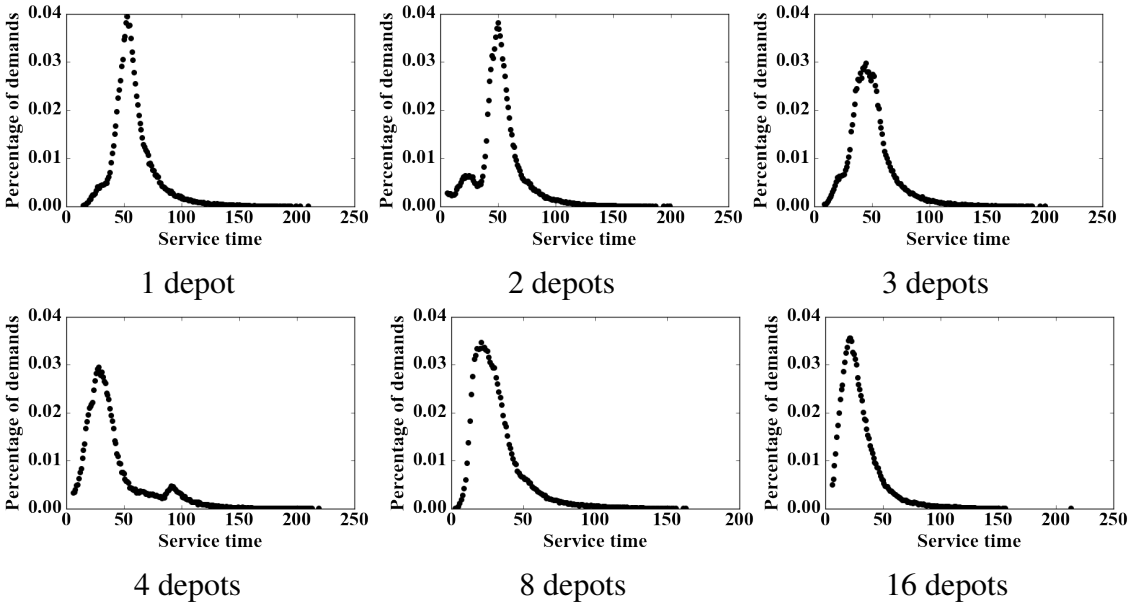


Figure 5.13: Service time($k = 12000$)

this queuing system as an $M/G/k$ queue. It is still an open question to exactly quantify the features of such a queue, such as waiting time [147]. We employ the approximation algorithm [148] to estimate the average waiting time in $M/G/k$ queue by adjusting the mean waiting time in a corresponding $M/M/k$ queue. Equation (5.1) shows the approximation function of the average waiting time in $M/G/k$ queue. where $E[W^{M/G/k}]$ and $E[W^{M/M/k}]$ are the expected waiting times of the $M/G/k$ and $M/M/k$ queues, respectively. The $M/M/k$ queue has the same mean service time as the $M/G/k$ queue.

$$E[W^{M/G/k}] = \frac{C^2 + 1}{2} E[W^{M/M/k}] \tag{5.1}$$

where C is the coefficient of variation of the service time distribution in $M/G/k$ queue. In $M/M/k$ queue, the average waiting time can be calculated in Eq (5.2).

$$E[W^{M/M/k}] = \frac{Er_c(k, \rho)\overline{S}}{k - \rho}, k > \rho \tag{5.2}$$

where ρ is the utilization in a queuing system, which equals to $\lambda\overline{S}$, and $Er_c(k,\rho)$ is the Erlang C formula(Eq (5.3)), which indicates the probability that an arriving customer has to wait, which is also the proportion of time that all k servers are busy. $k > \rho$ ensures the system can reach the steady state.

$$Er_c(k,\rho) = \frac{\frac{k\rho^k}{(k-\rho)k!}}{\sum_{k=0}^{k-1}\frac{\rho^n}{n!} + \frac{k\rho^k}{(k-\rho)k!}} \tag{5.3}$$

Finally, we can approximate the average waiting time in $M/G/k$ queue. Taking one depot deployment as an example, the arrival rate in $12pm - 6pm$ slot is 5.0594, and the average service time of the system is 3536.45249, so the utilization $\rho = 17876.4137$, and the coefficient of variation of the service time distribution $C = 0.5563$. Given the number of vehicles $k = 18000$, we can first get $Er_c(18000, 17876) = 0.2547$, which means that 25.47% of the time when all of the servers are busy. Finally the approximate average waiting time is 4.0134 seconds.

5.1.4 Evaluation

In this section, we use real taxi trip data to conduct experiments to evaluate (1) the performance of the design choices of number of available vehicles k and the number depots d. (2) the efficiency gain in SCCS comparing with current taxi system.

5.1.4.1 Evaluation settings

Time intervals in a day. We observe that the trip demand arrival and service patterns change dramatically over time intervals in a day. In our evaluations, we divide a day into 4 time intervals, we have the cutting-off times as $[12am, 6am, 12pm, 6pm]$. and evaluate how the granularities affect the performances of our proposes models.

Baselines. We compare the performances of our SCCS (in different design choices)

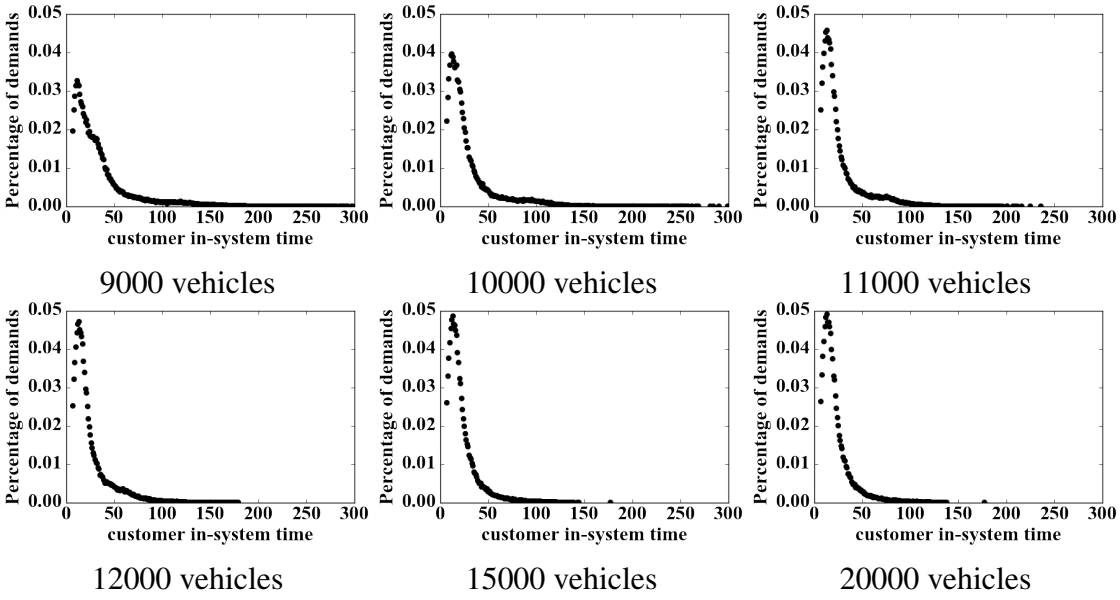


Figure 5.14: Impact of total number of taxis

with the current taxi system. To evaluate how our SCCS performs when serving the same set of trip demands in our taxi data, we employ a data-driven simulation approach as follows: The real world trip demands arrive by the order of their starting times. If there are available vehicles in its regional depot, the waiting time of this demand will be 0. Otherwise, the waiting time is the time interval from the starting time to the moment when a vehicle returns to that depot. The results introduced below show that our SCCS can achieve several efficiency gains comparing with the current transit system in vehicle utilization and number of vehicles needed.

Metrics. For the design choices, we use the customer in system time and vehicle idle rate to evaluate the performance of the system. The efficiency gain is evaluated by the number of vehicles needed, and the utilization of the vehicles while serving the same amount of demands in our system and current urban taxi transit system.

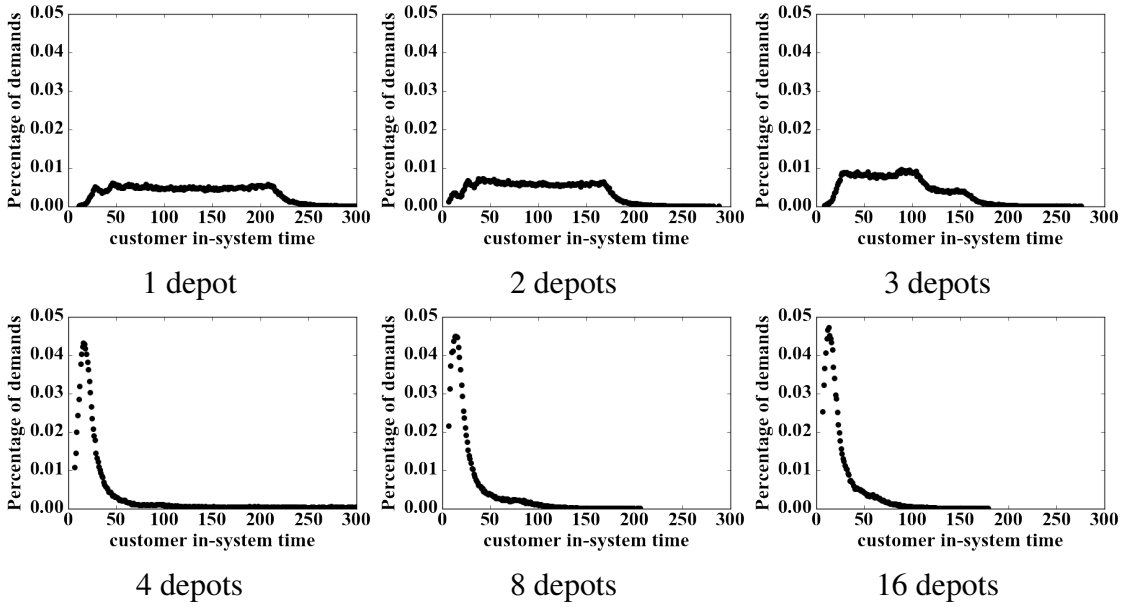


Figure 5.15: The impact of number of depots (k=12,000)

5.1.4.2 Design choices

Impact of k. From the passengers' perspectives, the service process consists of two parts: passenger waiting time and in-vehicle time. The passenger waiting time includes the system waiting time W^1 and the picking up time. We denote the total service time passenger experienced as the in-system time, namely, the total of waiting time, pickup time, and in-vehicle time. The in-system time is what passenger actually experiences, and is considered as the quality of service the passenger received.

Taking 16 depots as an example, given the number of vehicles 9000, 10000, 11000, 12000, 15000, 20000, we can simulate the whole service in our SCCS, and get the passenger in-system time, which is shown in Fig.5.14. We can observe that as we increase the number of vehicles, the passenger in-system time decreases.

Moreover, Fig. 5.20 shows the average in-system time and the idle rate for different numbers of AVs. With the increase of the total number of vehicles, the in-system time

[1]Note that the system waiting time is different from the passenger waiting time, where the former is the time from the request arrival to the time a vehicle is dispatched, and the latter includes both the system waiting time and pickup time.

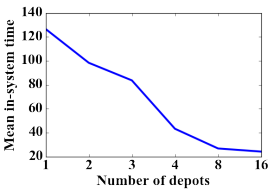


Figure 5.16: Average in-system time

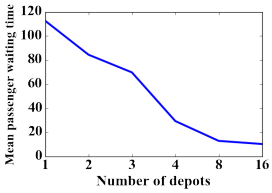


Figure 5.17: Average passenger waiting time

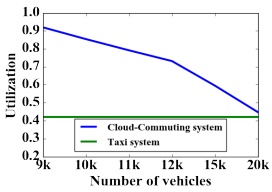


Figure 5.18: utilization of vehicles

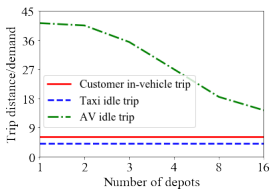


Figure 5.19: Distance per demand in SCCS and taxi system

decreases, which is because the waiting time becomes shorter. However, the idle rate, which characterizes the portion of time that a vehicle stays idle in the depot (Eq (5.4)), increases due to the increasing number of over-deployed AVs.

$$R_{idle} = \frac{\sum_{i=1}^{k} T_{idle}^{i}}{k \cdot T}, \tag{5.4}$$

with T as the total amount of time in a day (i.e., 24 hours), and T_{idle}^{i} is the amount of time the vehicle i spent in depot during the day.

Fig. 5.20 clearly indicates the trade-off between the waiting time and the idle rate when changing the number of vehicles.

The number of depots in our system can also have effects on the customer's experience. Taking $k = 12000$ for example, Fig. 5.15 shows the change of the customer in-system time according to the number of depots, when we fixed the number of AVs

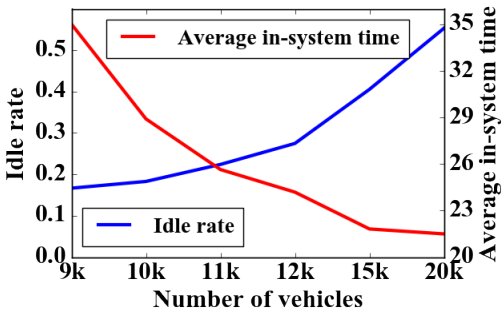


Figure 5.20: Tradeoff

to be 12000. Fig. 5.15(a)–(f) shows that as we increase the number of depots, the passenger in-system time distribution evolutes from high to low in-system time. Moreover, Fig. 5.16–5.17 indicates how the average in-system, waiting time changes, over different numbers of depots.

The phenomena occur because the increase of the number of depots can reduce the picking up time and the waiting time for each service. Moreover, from Fig. 5.15, we can clearly observe that

5.1.4.3 System efficiency gains

By comparing our SCCS with the current taxi system, we now show that the SCCS system can achieve efficiency gains in several aspects, including (1) the higher vehicle utilization, (2)the less number of vehicles needed. Here, the results presented are from the real-world data collected in Shenzhen, China, 2014, the traffic volume does not show significant difference over time, and similar results of vehicle utilization and number of vehicle needed can be obtained with data collected in different time (2016).

Utilization of vehicles. In Fig. 5.1, we show that most of the taxis are idling on the road over days, which means the utilization of the taxis in current taxi system is low. At each time slot, e.g., in 1 hour, we can obtain a ratio of in-service vehicle vs the total number of vehicles. We quantify the utilization of the vehicles as average ratio of in-serve vehicles over all time slots, defined as follows.

$$U = \frac{\sum_{i=1}^{T_{slots}} (N_i^{busy}/N_i)}{T_{slots}}, \tag{5.5}$$

where T_{slots} is the total number of time slots in a day, N_i^{busy} and N_i are the number of in-service and all vehicles at time slot i.

The utilization of the vehicles in our system is shown in Fig. 5.18 when $d = 16$. Taking $k = 11000$ as an example, the utilization is 79.1%, while the utilization of the

Table 5.4: V-values

# Vehicles	7k	7.5k	8k	9k	12k	20k
V	0.465	0.426	0.429	0.462	0.586	0.801

taxis in Shenzhen was 42.02%.

Number of vehicles needed. We can count the number of taxis in Shenzhen taxi system from our trajectory data, which was in total $9,606$ taxis. When using SCCS to serve the same trip demands, the number of vehicles would have impacts on the trade-off between the passenger in-system time and the vehicle idle rate (see Fig. 5.20). We define a measure V-value in Eq.(5.6) as a combination of the two measures to quantify the system performance.

$$V_k = \alpha T_{in-system} + (1-\alpha) \cdot R_{idle}, \tag{5.6}$$

with α as a design trade-off parameter within $[0, 1]$. The smaller V-value indicates better performance. Taking 32 depots and $\alpha = 0.01$ as example, the V-values are listed in Table 5.4. The most appropriate number of vehicles in 32 depots is 7500, which shows a 22% reduction on needed vehicles.

5.1.4.4 Travel Distance Analysis

In this section, we compare the travel distances of AVs in SCCS with those of taxis in current taxi system. Fig.5.19 shows the average distance per demand for the customer in-vehicle trips and idle trips of AVs in SCCS and taxis in current taxi system. As shown in Fig.5.2, the **AV idle trip** includes the picking up trip and the returning trip, and the distances of these trips are obtained via the OSMnx API[149]. The **taxi idle trip** is the trip when the taxi has no passengers onboard. The distance of **customer in-vehicle trip** is the same in SCCS and taxi system. The distances of the later two types of trips are

extracted from the taxi GPS data. From Fig.5.19, we find that, as the number of depots increase in SCCS, the average AV idle trip distance per demand decreases, because the distances of picking up and returning trips decreases when there are more depots over the city. Although the average AV idle trip distance is higher than that of taxis in current taxi system when there are 16 depots in the city, it is safe for us to estimate that when the number of depots is sufficiently large, the average idle distance of AVs in SCCS will be lower than that of taxis in current taxi system.

6

Conclusion and Future Work

6.1 Conclusion

Making sense of *human-generated spatial-temporal data* can benefit people in many aspects. In this dissertation, we illustrate this in the following four domains.

Human Learning Curve Dissection. In this topic, we made the first attempt to employ inverse reinforcement learning to analyze the preferences of taxi drivers when making sequences of decisions to look for passengers. We further studied how the drivers' preferences evolve over time, during the learning processes. This problem is critical to helping new drivers improve performance fast. We extracted different types of interpretable features to represent the potential factors that affect the decisions of taxi drivers and inversely learned the preferences of different groups of drivers. We conducted experiments using large scale taxi trajectory datasets, and the results demonstrated that drivers tend to improve their preferences to habit features to gain more knowledge in the learning phase and keep the preferences to profile features stable over time. Also to study *how* human learns, we propose a novel framework, including trending analysis, learning modeling, and strategy validation. Our experiments on a large-scale real-world taxi trajectory

data prove that the taxi drivers' strategy change follows the learning process of reinforcement learning (RL) and the drivers with different trends of earning efficiency have the different extents to follow RL. Our framework and findings provide an important sight in the fields of human behavior learning and taxi operation management.

Human Behavior Explanation. Generative adversarial imitation learning (GAIL) achieves great success in learning human decision-making strategies from demonstrated data using deep neural networks (DNNs). However, such DNN-based models are hard to explain what aggregate knowledge the models learned from data. To bridge this gap, we propose the explainable generative adversarial imitation learning (xGAIL) framework which includes two novel techniques, namely, Spatial Activation Maximization (SpatialAM) and Spatial Randomize Input Sampling Explanation (SpatialRISE). They can learn global and local explainable spatial-temporal features, respectively. In particular, we take taxi drivers' passenger-seeking strategy as an example to validate the effectiveness of the xGAIL framework. Our analysis of a large-scale real-world taxi trajectory data shows interesting results from two perspectives i) global explainable knowledge of what nearby traffic condition impels a taxi driver to choose a particular direction to find the next passenger, and ii) local explainable knowledge of what key (sometimes hidden) factors a taxi driver considers when making a particular decision. All the knowledge we found sheds light on how to promote taxi drivers' well-being and improve the quality of taxi services, e.g., reducing the waiting time, etc. Moreover, our proposed xGAIL framework can be naturally applied to other urban decision-making processes, such as commuter transit mode choice, and personal vehicle route choice.

Human Mobility Signature Identification. In this topic, we propose the Spatio-temporal Siamese Networks (ST-SiameseNet) to solve the Human Mobility Signature Identification (HuMID) problem. The HuMID problem aims at validating if an income set of trajectories belong to a certain agent based on historical trajectory data. ST-SiameseNet

can deal with large group of agents in a single model. Also, we extract several effective profile features from the trajectories to augment the performance of the ST-SiameseNet. The experimental results illustrate that ST-SiameseNet outperforms state-of-the-art works and achieves an F_1 score of 0.8508 on a real-world taxi trajectory dataset. Our proposed ST-SiameseNet framework can be applied to many other real-world cases with human-generated spatio-temporal data other than the ride-sharing and taxi case. In the future, we will continue studying the driver identification problem with multiple inputs rather than pairwise inputs.

Smart Cloud Commuting System. In this topic, we advocate a Smart Cloud Commuting System (SCCS) for future smart cities with shared AVs to meet daily commuting demands of a large urban city. We have outlined four aspects of system efficiencies that can potentially be attained via the envisaged SCCS. As a first attempt at studying its feasibility, in this work we develop generative models to capture fundamental trip demand arrival and service patterns, and develop a novel framework to explore the impact of design choices on the temporal multiplexing gains (through time-sharing of AVs) that can be achieved by SCCS. We conducted extensive evaluations using a large scale urban taxi trajectory dataset from Shenzhen, China. The results demonstrate that SCCS can reduce the number of vehicles by 22%, and improve the vehicle utilization by 37%.

6.2 Future Work

6.2.1 Novel Spatial-temporal Data Mining Techniques

In this dissertation, the spatial-temporal data are first formulated as sequence of image-like inputs. Then, convolutional neural networks (CNNs) are employed to extract the spatial information, and recurrent neural networks (RNNs), e.g., LSTM, are used to incorporate the temporal dependency in the spatial-temporal data [5, 6]. Recently, Trans-

former networks [150] demonstrates its outstanding performance in analysing sequential data. Moreover, researchers find that Transformer can also perform well in dealing with image tasks comparing with CNNs [151]. These achievements inspire me to think about employing Transformer networks in analyzing human-generated spatial-temporal data.

Besides modeling human-generated spatial-temporal data as sequence of image-like inputs, the intrinsic topology of many spatial-temporal data from urban environment enables us to model them as graphs. Then, graph neural networks (GNNs) can be employed to extract the hidden information. For example, the road networks can be modeled graphs where nodes represent the road segments, and GNNs are employed to predict traffic flow [152, 153, 154, 155, 156, 157]. Inspired by this, GNNs can be employed to understand human behavior from human-generated spatial-temporal data.

6.2.2 Explainable Artificial Intelligence for Human-generated Spatial-temporal Data

As presented in the second topic of this dissertation, we propose the explainable generative imitation learning framework to understand human behavior from human-generated spatial-temporal data. This is an example of developing Explainable Artificial Intelligence (XAI) techniques for human-generated spatial-temporal data. There are plenty of machine learning models developed in analyzing spatial-temporal data, while many of them remains "black-box" models. As a matter of this, in future research, we may develop either intrinsic or post-hoc XAI techniques to make sense of human-generated spatial-temporal data. For instance, the ST-SiameseNet we introduced in the third topic of this dissertation can be further explained to help people understand what characteristics in the mobility data help us identify the human agents. Also, the Transformer networks and GNNs can be further explained to help understand human-generated spatial-temporal

data.

6.2.3 Novel Application Domains in Urban Intelligence

In this dissertation, we mainly study the problems in the domains of urban human behavior analysis and transportation systems. There exist many other application domains of making sense of different sources of human-generated spatial-temporal data in urban intelligence, e.g., analysing the energy (electricity, gas) consuming data to help improve energy supply and consuming efficiency, analysing the patients health care records to help improve the health system, and analysing the crime events data to enhance urban safety, etc.

References

[1] The Population Division of the UN Department of Economic and Social Affairs (UN DESA). *The 2018 Revision of World Urbanization Prospects*. 2018. 1

[2] Menghai Pan, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, and Jun Luo. Dissecting the learning curve of taxi drivers: A data-driven approach. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019. 2, 5, 35, 36, 40, 58, 59, 90, 101, 102

[3] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, Zhihong Tian, and Jun Luo. Dhpa: Dynamic human preference analytics framework: A case study on taxi drivers' learning curve analysis. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(1):1–19, 2020. 2, 5, 40, 59

[4] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, Zhenming Liu, Jie Bao, Yu Zheng, and Jun Luo. Is reinforcement learning the choice of human learners? a case study of taxi drivers. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 357–366, 2020. 2, 5

[5] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. xgail: Explainable generative adversarial imitation learning for explainable human decision

analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1334–1343, 2020. 3, 5, 133

[6] Huimin Ren, Menghai Pan, Yanhua Li, Xun Zhou, and Jun Luo. St-siamesenet: Spatio-temporal siamese networks for human mobility signature identification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1306–1315, 2020. 4, 5, 133

[7] Menghai Pan, Yanhua Li, Zhi-Li Zhang, and Jun Luo. Sccs: Smart cloud commuting system with shared autonomous vehicles. *IEEE Transactions on Big Data*, 2020. 5

[8] Guanxiong Liu, Menghai Pan, Yanhua Li, Zhi-Li Zhang, and Jun Luo. Modeling urban trip demands in cloud-commuting system: A holistic approach. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 857–862. IEEE, 2017. 5

[9] Menghai Pan, Yanhua Li, Taihui Li, Zhi-Li Zhang, and Jun Luo. Smart cloud commuting with shared autonomous vehicles: A first feasibility study. 5

[10] Y. Ge, C. Liu, H. Xiong, J. Chen. A Taxi Business Intelligence System. In *The 17th International Conference on Knowledge Discovery and Data Mining*, pages 735–738, New York, NY, 2011. ACM. 8, 12, 40, 88

[11] Y. Ge and H. Xiong and A. Tuzhilin and K. Xiao and M. Gruteser. An energy-efficient mobile recommender system. In *The the 16th International Conference on Knowledge Discovery and Data Mining*, pages 899–908, New York, NY, 2010. ACM. 8, 12, 40, 88

[12] Huigui Rong, Xun Zhou, Chang Yang, Zubair Shafiq, and Alex Liu. The rich and the poor: A markov decision process approach to optimizing taxi driver revenue

efficiency. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2329–2334. ACM, 2016. 8, 12, 35, 40, 46, 59, 88

[13] M. Qu, H. Zhu, J. Liu, G. Liu, H. Xiong. A Cost-Effective Recommender System for Taxi Drivers. In *The 20th International Conference on Knowledge Discovery and Data Mining (SIGKDD'14)*, pages 45–54, New York, NY, 2014. ACM. 8, 12, 39, 88

[14] OpenStreetMap. `http://www.openstreetmap.org/`. 11, 66, 87, 118

[15] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*, 2015. 12, 14, 38, 39, 88, 90, 114

[16] Bo Lyu, Shijian Li, Yanhua Li, Jie Fu, Andrew C Trapp, Haiyong Xie, and Yong Liao. Scalable user assignment in power grids: a data driven approach. In *SIGSPATIAL GIS*. ACM, 2016. 12, 39, 88, 114

[17] Ye Ding, Yanhua Li, Ke Deng, Haoyu Tan, Mingxuan Yuan, and Lionel M Ni. Detecting and analyzing urban regions with high impact of weather change on transport. *IEEE Transactions on Big Data*, 2016. 12, 39, 88, 114

[18] Chen Liu, Ke Deng, Chaojie Li, Jianxin Li, Yanhua Li, and Jun Luo. The optimal distribution of electric-vehicle chargers across a city. In *ICDM*. IEEE, 2016. 12, 39, 88, 114

[19] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 984–992, 2018. 12, 39, 88

[20] Amin Vahedian Khezerlou, Xun Zhou, Lufan Li, Zubair Shafiq, Alex X Liu, and Fan Zhang. A traffic flow approach to early detection of gathering events: Comprehensive results. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(6):1–24, 2017. 12, 88

[21] Tong Xu, Hengshu Zhu, Xiangyu Zhao, Qi Liu, Hao Zhong, Enhong Chen, and Hui Xiong. Taxi driving behavior analysis in latent vehicle-to-vehicle networks: A social influence perspective. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294. ACM, 2016. 12, 39, 88

[22] Jie Wang, Helai Huang, Pengpeng Xu, Siqi Xie, and S. C. Wong. Random parameter probit models to analyze pedestrian red-light violations and injury severity in pedestrian-motor vehicle crashes at signalized crossings. *Journal of Transportation Safety & Security*, 0(0):1–20, 2019. 12

[23] Zi Yang, Xinpeng Wang, Xin Pei, Shuo Feng, Dajun Wang, Jianqiang Wang, and SC Wong. Longitudinal safety analysis for heterogeneous platoon of automated and human vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3300–3305. IEEE, 2018. 12

[24] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Chao Tian, and Yu Zheng. Detecting vehicle illegal parking events using sharing bikes' trajectories. 2018. 12

[25] Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. Deep distributed fusion network for air quality prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 965–973. ACM, 2018. 12

[26] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang,

Jieping Ye, and Weifeng Lv. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1653–1662. ACM, 2017. 12

[27] Wolfson S. Ma, Y. Zheng. A large-scale dynamic taxi ridesharing service. In *The 29th International Conference on Data Engineering (ICDE'13)*, pages 410–421, New York, NY, 2013. IEEE. 12, 40, 88

[28] J. Yuan, Y. Zheng, L. Zhang, X. Xie. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2390–2403, 2013. 12, 40, 88

[29] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 109–118, New York, NY, 2011. ACM. 12, 40, 88

[30] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. 12, 19

[31] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 13, 20, 25, 59, 65

[32] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011. 13, 20, 22, 59, 65

[33] Yanhua Li, Moritz Steiner, Jie Bao, Limin Wang, and Ting Zhu. Region sampling and estimation of geosocial data with dynamic range calibration. In *ICDE*, 2014. 14, 38, 90

[34] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. 17, 44

[35] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004. 20

[36] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007. 20

[37] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010. 22, 59, 62, 65

[38] R.S. Witte and J.S. Witte. *Statistics, 10th Edition*. John Wiley and Sons, Incorporated, 2013. 23

[39] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016. 35

[40] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2447–2456, 2018. 35

[41] Xun Zhou, Huigui Rong, Chang Yang, Qun Zhang, Amin Vahedian Khezerlou, Hui Zheng, M Zubair Shafiq, and Alex X Liu. Optimizing taxi driver profit efficiency: A spatial network-based markov decision process approach. *IEEE TBD*, 2018. 35, 40, 46, 59

[42] C Zeng and N Oren. Dynamic taxi pricing. *Frontiers in Artificial Intelligence and Applications*, 263:1135–1136, 01 2014. 35, 46, 59

[43] Liang Liu, Clio Andris, Assaf Biderman, and Carlo Ratti. Revealing taxi driver's mobility intelligence through his trace. In *Movement-Aware Applications for Sustainable Mobility: Technologies and Approaches*, pages 105–120. IGI Global, 2010. 35, 59

[44] Jing Fan, Ye Li, Yuanlin Liu, Yu Zhang, and Changxi Ma. Analysis of taxi driving behavior and driving risk based on trajectory data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 220–225. IEEE, 2019. 39

[45] Govind P Yatnalka and Husnu S Narman. A matching model for vehicle sharing based on user characteristics and tolerated-time. In *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI (HONET-ICT)*, pages 143–147. IEEE, 2019. 39

[46] Yanbo Pang, Kota Tsubouchi, Takahiro Yabe, and Yoshihide Sekimoto. Replicating urban dynamics by generating human-like agents from smartphone gps data. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 440–443, 2018. 39

[47] Zipei Fan, Quanjun Chen, Renhe Jiang, Ryosuke Shibasaki, Xuan Song, and Kota Tsubouchi. Deep multiple instance learning for human trajectory identification. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 512–515, 2019. 39

[48] Cong Zhang, Yanhua Li, Jie Bao, Sijie Ruan, Tianfu He, Hui Lu, Zhihong Tian, Cong Liu, Chao Tian, Jianfeng Lin, et al. Effective recycling planning for dockless sharing bikes. In *Proceedings of the 27th ACM SIGSPATIAL International*

Conference on Advances in Geographic Information Systems, pages 62–70, 2019. 39

[49] Jaël Champagne Gareau, Éric Beaudry, and Vladimir Makarenkov. An efficient electric vehicle path-planner that considers the waiting time. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 389–397, 2019. 39

[50] Yubin Duan, Ning Wang, and Jie Wu. Optimizing order dispatch for ride-sharing systems. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2019. 40

[51] Zeyu Wang. Taxi scheduling optimization with incomplete information. In *2019 International Conference on Information Technology and Computer Application (ITCA)*, pages 266–270. IEEE, 2019. 40

[52] Yiwen Song, Ningning Sun, and Huimiao Chen. Demand adaptive multi-objective electric taxi fleet dispatching with carbon emission analysis. In *2019 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, pages 1–5. IEEE, 2019. 40

[53] Qiang Ma, Zhichao Cao, Kebin Liu, and Xin Miao. Qa-share: Toward an efficient qos-aware dispatching approach for urban taxi-sharing. *ACM Transactions on Sensor Networks (TOSN)*, 16(2):1–21, 2020. 40

[54] Zhidan Liu, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. Mobility-aware dynamic taxi ridesharing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 961–972. IEEE, 2020. 40

[55] Xianlei Dong, Min Zhang, Shuang Zhang, Xinyi Shen, and Beibei Hu. The analy-

sis of urban taxi operation efficiency based on gps trajectory big data. *Physica A: Statistical Mechanics and its Applications*, 528:121456, 2019. 40

[56] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, et al. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1983–1992, 2019. 40

[57] Jintao Ke, Feng Xiao, Hai Yang, and Jieping Ye. Optimizing online matching for ride-sourcing services with multi-agent deep reinforcement learning. *arXiv preprint arXiv:1902.06228*, 2019. 40

[58] Jie Shi, Yuanqi Gao, Wei Wang, Nanpeng Yu, and Petros A Ioannou. Operating electric vehicle fleet for ride-hailing services with reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2019. 40

[59] Zhiwei Qin, Jian Tang, and Jieping Ye. Deep reinforcement learning with applications in transportation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3201–3202, 2019. 40

[60] Suining He and Kang G Shin. Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination. In *The World Wide Web Conference*, pages 2806–2813, 2019. 40

[61] Ming Zhou, Jiarui Jin, Weinan Zhang, Zhiwei Qin, Yan Jiao, Chenxi Wang, Guobin Wu, Yong Yu, and Jieping Ye. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2645–2653, 2019. 40

[62] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pages 983–994, 2019. 40

[63] John Holler, Risto Vuorio, Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jieping Ye. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1090–1095. IEEE, 2019. 40

[64] Xiaocheng Tang, Zhiwei Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1780–1790, 2019. 40

[65] Zhenhua Huang, Jinyi Tang, Guangxu Shan, Juan Ni, Yunwen Chen, and Cheng Wang. An efficient passenger-hunting recommendation framework with multitask deep learning. *IEEE Internet of Things Journal*, 6(5):7713–7721, 2019. 40

[66] Suiming Guo, Chao Chen, Jingyuan Wang, Yaxiao Liu, Xu Ke, Zhiwen Yu, Daqing Zhang, and Dah-Ming Chiu. Rod-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data. *IEEE Transactions on Mobile Computing*, 2019. 40

[67] Guojun Wu, Yanhua Li, Jie Bao, Yu Zheng, Jieping Ye, and Jun Luo. Human-centric urban transit evaluation and planning. In *2018 IEEE ICDM*, pages 547–556. IEEE, 2018. 40, 59

[68] Knud Illeris. *How we learn: Learning and non-learning in school and beyond*. Routledge, 2007. 40

[69] Guy R Lefrancois. *Theories of human learning*. Cambridge University Press, 2019. 40

[70] Danielle S Bassett, Nicholas F Wymbs, Mason A Porter, Peter J Mucha, Jean M Carlson, and Scott T Grafton. Dynamic reconfiguration of human brain networks during learning. *Proceedings of the National Academy of Sciences*, 2011. 40

[71] RR Skemp. *The psychology of mathematics learning: Expanded American edition*. Hillsdale, New Jersey: Erlbaum, 1987. 40

[72] IE Rolfe and RW Sanson-Fisher. Translating learning principles into practice: a new strategy for learning clinical skills. *Medical education*, 36(4):345–352, 2002. 40

[73] Khaled H Hamed and A Ramachandra Rao. A modified mann-kendall trend test for autocorrelated data. *Journal of hydrology*, 204(1-4):182–196, 1998. 42

[74] Barry James, Kang Ling James, and David Siegmund. Tests for a change-point. *Biometrika*, 74(1):71–83, 1987. 43

[75] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, NJ, 1994. 43

[76] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 46

[77] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000. 46, 47

[78] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation.

In *Advances in neural information processing systems*, pages 1057–1063, 2000. 46, 47

[79] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009. 49

[80] Gregory W Corder and Dale I Foreman. *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons, 2014. 49

[81] Arthur Getis and J Keith Ord. The analysis of spatial association by use of distance statistics. *Geographical analysis*, 24(3):189–206, 1992. 51

[82] Xin Zhang, Yanhua Li, Xun Zhou, and Jun Luo. Unveiling taxi drivers' strategies via cgail-conditional generative adversarial imitation learning. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019. 59, 65

[83] P Li, Sandjai Bhulai, and JT van Essen. Optimization of the revenue of the new york city taxi service using markov decision processes. In *6th International Conference on Data Analytics, Barcelona (Spain), November 12-16*, pages 47–52. IARIA, 2017. 59

[84] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016. 59, 60, 62, 63, 65, 67

[85] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016. 60, 66, 72

[86] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018. 60, 66, 71, 77

[87] Gilles Vandewiele, Olivier Janssens, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. Genesim: genetic extraction of a single interpretable model. *arXiv preprint arXiv:1611.05722*, 2016. 60

[88] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998. 61, 68

[89] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 63

[90] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017. 65

[91] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019. 65

[92] Ninghao Liu, Mengnan Du, and Xia Hu. Representation interpretation with spatial encoding and multimodal analytics. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 60–68, 2019. 65

[93] Hao Yuan, Yongjun Chen, Xia Hu, and Shuiwang Ji. Interpreting deep models for text analysis via optimization and regularization methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5717–5724, 2019. 65

[94] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 65, 70

[95] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 65, 70

[96] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016. 65, 70

[97] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015. 65

[98] Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780, 2017. 65

[99] M Gethsiyal Augasta and Thangairulappan Kathirvalavakumar. Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150, 2012. 65

[100] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 66

[101] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *NeurIPS*, pages 396–404, 1990. 68

[102] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992. 68

[103] Arthur Getis and J Keith Ord. The analysis of spatial association by use of distance statistics. In *Perspectives on Spatial Data Analysis*, pages 127–145. Springer, 2010. 72

[104] Howard Anton and Chris Rorres. Elementary linear algebra. 10. auflage. hoboken, 2010. 75

[105] Taxi, Uber, and Lyft Usage in New York City. `http://toddwschneider.com/posts/taxi-[]uber-[]lyft-[]usage-[]new-[]york-[]city/`. 83

[106] Faiz Siddiqui. Uber makes changes amid swarm of criticism over rider safety. `https://www.washingtonpost.com/technology`, 19. 83

[107] AARIAN MARSHALL. Uber's New Features Put a Focus on Rider Safety . `https://www.wired.com/story/ubers-[]new-[]features-[]focus-[]rider-[]safety/`, 09 2019. 83

[108] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Jure Leskovec, et al. Driver identification using automobile sensor data from a single turn. In *2016 IEEE 19th ITSC*, pages 953–958. IEEE, 2016. 83, 84, 89

[109] Arijit Chowdhury, Tapas Chakravarty, Avik Ghose, Tanushree Banerjee, and P Balamuralidhar. Investigations on driver unique identification from smartphone's gps data alone. *Journal of Advanced Transportation*, 2018, 2018. 83, 84, 89, 102

[110] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. Distinguishing trajectories from different drivers using incompletely labeled trajectories. In *Proceedings of the 27th ACM International CIKM*, pages 863–872. ACM, 2018. 84, 89

[111] Min-hwan Oh and Garud Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD*, pages 1480–1490, 2019. 84, 89

[112] Xingjian Zhang, Xiaohua Zhao, and Jian Rong. A study of individual characteristics of driving behavior based on hidden markov model. *Sensors & Transducers*, 167(3):194, 2014. 88, 89

[113] José Onate López, Andrés C Cuervo Pinilla, et al. Driver behavior classification model based on an intelligent driving diagnosis system. In *15th ITS*, pages 894–899. IEEE, 2012. 89

[114] Saad Ezzini, Ismail Berrada, and Mounir Ghogho. Who is behind the wheel? driver identification and fingerprinting. *Journal of Big Data*, 5(1):9, 2018. 89

[115] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. Characterizing driving styles with deep learning. *arXiv preprint arXiv:1607.03611*, 2016. 89, 102

[116] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *NIPS*, pages 737–744, 1994. 89

[117] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546. IEEE, 2005. 89, 93, 94, 96, 98

[118] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015. 89

[119] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050, 2014. 89

[120] Gabriel Synnaeve, Thomas Schatz, and Emmanuel Dupoux. Phonetics embedding learning with side information. In *2014 IEEE SLT*, pages 106–111. IEEE, 2014. 89

[121] Herman Kamper, Weiran Wang, and Karen Livescu. Deep convolutional acoustic word embeddings using word-pair side information. In *ICASSP*, pages 4950–4954. IEEE, 2016. 89

[122] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on CVPR*, pages 1701–1708, 2014. 93, 98

[123] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 94

[124] Tushar Nandwana. Autonomous vehicles: Why drive when the vehicle drives you. `https://www.intactspecialty.com/en/technology/whitepaper-[]detail.page?id=db6800071963fe7f72057a25925f49fd`, 2020. 107

[125] Katarzyna Anna Marczuk, Harold Soh Soon Hong, Carlos Miguel Lima Azevedo, Muhammad Adnan, Scott Drew Pendleton, Emilio Frazzoli, et al. Autonomous mobility on demand in simmobility: Case study of the central business district in singapore. In *CIS-RAM 2015*. 113, 114

[126] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 2017. 113, 114

[127] Rick Zhang, Kevin Spieser, Emilio Frazzoli, and Marco Pavone. Models, algorithms, and evaluation for autonomous mobility-on-demand systems. In *American Control Conference (ACC), 2015*, pages 2573–2587. IEEE, 2015. 113, 114

[128] Marco Pavone. Autonomous mobility-on-demand systems for future urban mobility. In *Autonomous Driving*, pages 387–404. Springer, 2016. 113, 114

[129] Ran Wang, Chi-Yin Chow, Yan Lyu, Victor C. S. Lee, Sam Kwong, Yanhua Li, and Jia Zeng. Taxirec: Recommending road clusters to taxi drivers using ranking-based extreme learning machines. In *SIGSPATIAL*, 2015. 113

[130] Xuesong Zhou. Dynamic origin-destination demand estimation and prediction for off-line and on-line dynamic traffic assignment operation. *PhD Thesis Dissertation at University of Maryland College Park*, 2004. 113

[131] Michal Maciejewski, Joschka Bischoff, and Kai Nagel. An assignment-based approach to efficient real-time city-scale taxi dispatching. *IEEE Intelligent Systems*, 2016. 113

[132] Lawrence D Burns, William C Jordan, and Bonnie A Scarborough. Transforming personal mobility. *The Earth Institute*, 431:432, 2013. 114

[133] Joschka Bischoff and Michal Maciejewski. Simulation of city-wide replacement of private cars with autonomous taxis in berlin. *Procedia Computer Science*, 83, 2016. 114

[134] Patrick M Boesch, Francesco Ciari, and Kay W Axhausen. Autonomous vehicle fleet sizes required to serve different levels of demand. *Transportation Research Record*, 2542(1):111–119, 2016. 114

[135] T Donna Chen, Kara M Kockelman, and Josiah P Hanna. Operations of a shared, autonomous, electric vehicle fleet: Implications of vehicle & charging infrastructure decisions. *Transportation Research Part A: Policy and Practice*, 94:243–254, 2016. 114

[136] Aggelos Soteropoulos, Martin Berger, and Francesco Ciari. Impacts of automated vehicles on travel behaviour and land use: an international review of modelling studies. *Transport reviews*, 39(1):29–49, 2019. 114

[137] Ermal Toto, Elke A Rundensteiner, Yanhua Li, Richard Jordan, Mariya Ishutkina, Kajal Claypool, Jun Luo, and Fan Zhang. Pulse: A real time system for crowd flow prediction at metropolitan subway stations. In *ECML-PKDD*. Springer, 2016. 114

[138] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. Discovering urban functional zones using latent activity trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 2015. 114

[139] Xinyue Liu, Xiangnan Kong, and Yanhua Li. Collective traffic prediction with partially observed traffic history using location-base social media. In *CIKM*, 2016. 114

[140] Ye Ding, Yanhua Li, Ke Deng, Haoyu Tan, Mingxuan Yuan, and Lionel M Ni. Dissecting regional weather-traffic sensitivity throughout a city. In *ICDM*, 2015. 114

[141] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path us-

ing sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014. 114

[142] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, and E Chang. Diagnosing New York City's Noises with Ubiquitous Data. In *Ubicomp*, 2014. 114, 115

[143] CV Kumar, Debasis Basu, and Bhargab Maitra. Modeling generalized cost of travel for rural bus users: a case study. *Journal of Public Transportation*, 2004. 114

[144] Google GeoCoding. `https://developers.google.com/maps/documentation/geocoding/`. 117

[145] Hsun-Ping Hsieh, Shou-De Lin, and Yu Zheng. Inferring air quality for station location recommendation based on urban big data. In *SIGKDD*, 2015. 118

[146] Open Source Routing Machine. `https://http://project-[]osrm.org/`. 122

[147] JFC Kingman. The first erlang century—and the next. *Queueing systems*, 63(1-4):3, 2009. 123

[148] AM Lee and PA Longton. Queueing processes associated with airline passenger check-in. *OR*, pages 56–71, 1959. 123

[149] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. 129

[150] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017. 134

[151] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. 134

[152] Zhipu Xie, Weifeng Lv, Shangfo Huang, Zhilong Lu, Bowen Du, and Runhe Huang. Sequential graph neural network for urban road traffic speed prediction. *IEEE Access*, 8:63349–63358, 2019. 134

[153] Rui Dai, Shenkun Xu, Qian Gu, Chenguang Ji, and Kaikui Liu. Hybrid spatio-temporal graph convolutional network: Improving traffic prediction with navigation data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3074–3082, 2020. 134

[154] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4883–4894, 2019. 134

[155] Cen Chen, Kenli Li, Sin G Teo, Xiaofeng Zou, Kang Wang, Jie Wang, and Zeng Zeng. Gated residual recurrent graph neural networks for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 485–492, 2019. 134

[156] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 914–921, 2020. 134

[157] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, volume 34, pages 1177–1185, 2020. 134