Data-Driven Battery Attack Detection and Control Behavior Determination for Vehicle Driving Safety

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

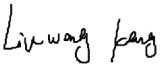
Liuwang Kang

August

2021

APPROVAL SHEET

The dissertation is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Liuwang Kang 

The dissertation has been read and approved by the examining committee:

Haiying Shen, Advisor

Yangfeng Ji, Committee chair

Lu Feng

Brian L. Smith

Haifeng Xu

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

---

August 2021

Abstract

Pure electric vehicles (EVs) have become popular in current transportation systems because of their zero air pollution emissions. The battery management system (BMS) in an EV monitors battery information (current, voltage and temperature) in real time to prevent batteries from overcharging or overheating and also shares these battery information to outside-vehicle environments (e.g., smartphone apps) to enrich vehicle usage experiences. Many researches have studied various aspects regarding vehicle driving safety, few works have comprehensively studied battery security and its effects on driving safety of an EV. Besides, autonomous vehicles (AVs) have been adopted to reduce traffic congestion and multiple AVs will drive on the same road with the AV population growth. Therefore, It is critical to make optimal control decisions of multiple AVs in real time to ensure driving safety.

Motivated by the above scenarios, we focus on three areas to improve vehicle driving safety: (1) battery authentication system for detecting battery attacks (i.e., malicious AC-turn-on requests or battery-charge-stop requests) in electric vehicles; (2) control policy based driving safety system for an individual AV; and (3) multi-AV control decision making system for multiple AVs to ensure their driving safety. *First*, we propose the first battery attack, which can turn on air condition and stop battery charging process by sending requests through a smartphone without being noticed by users, and design a Battery authentication method (Bauth) to detect such battery attacks. Bauth describes a user's habits in turning on air condition and stopping battery charging using a data-driven behavior model. It then applies the behavior model into a reinforcement learning model to judge whether an AC-turn-on or a battery-charge-stop request is from a real user. From real-life daily driving experiments, we find that Bauth can prevent EV batteries from being attacked accurately and its accuracy reaches as high as 95.6%. *Second,* we propose a control policy based driving

safety system (Polsa) to help improve driving safety of a given AV. For a given AV, Polsa extracts its control policies and determines the safest control behavior among multiple control behaviors for each given trigger condition. Accordingly, Polsa has a control policy extraction method using dynamic time warping and k-means clustering technologies to cluster historical driving data with the same control behavior type together and then analyzes positions and driving speeds in each cluster to extract control policies of a target AV. It then develops an optimal control policy determination method to determine the safest control behavior for each given trigger condition by considering time-varying driving state of its nearby vehicle. We use an industry-standard AV platform (Baidu Apollo) to evaluate optimal control policy success rate of Polsa and find that Polsa can extract control policies with as much as 83% accuracy, and improve optimal control policy success rate by 28% compared with existing methods. *Third,* we propose a multi-AV control decision making system (MADM), which considers multi-AV coexistence driving situations. MADM builds a policy formation method to form policies to learn driving behaviors of an expert based on the expert driving trajectory data. It then builds a multi-AV control decision making method, which adjusts the formed policies through a multi-agent reinforcement learning and forms safety driving state of each AV, to make multiple control decisions with safety guarantee. We used a real-world traffic dataset to evaluate optimal control decision making performance of MADM and experimental results show that MADM reduces its emergency rate by as high as 51% compared with existing methods.

I dedicate this thesis to my family. Without their patience, understanding, and support, the completion of this work would not have been possible.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the many people who have helped and supported me through this long Ph.D. journey. First, I thank my advisor, Professor Haiying Shen, whose encouragement, guidance, and support from the initial to the final level of my Ph.D. study enabled me to develop a comprehensive understanding of the subject. I acknowledge my committee members, Professor Yangfeng Ji, Professor Brian L. Smith, Professor Lu Feng, and Professor Haifeng Xu for providing me valuable comments and suggestions in writing this dissertation.

I am proud of my friendships and work relationships with my many colleagues and friends in the Pervasive Communications Laboratory. I am thankful to all of you for supporting my research and providing me with helpful suggestions. Thank you for the laughs and happy days in the office. I would also like to acknowledge my mentor, Ankur Sarker.

All my friends and family have been a source of inspiration in my life. Living outside of my country was tough at the beginning and I would like to thank all my friends for their tremendous help. Lastly, I thank my family for their love and support at all times. None of my achievements would have been possible without their infinite support and steely determination to give me the best possible education.

Chapter 1

Introduction

An pure electric vehicle (EV) consumes electricity in its battery pack to drive the vehicle and auxiliary functions (e.g., air condition (AC) and play music, etc). Compared with traditional vehicles, EVs have less air pollution emission and become popular in recent years [34]. As a key part of pure electrical vehicles (EVs), the battery pack receives energy during the charging process and provides energy in the discharging process. Therefore, the driving range of an EV depends on the total energy stored in its batteries. EV manufacturers have proposed a battery management system (BMS) to monitor temperature, voltage and current of each cell in a battery pack so that these cells can work normally. Besides, BMS shares battery information to outside-vehicle environments (e.g., smartphone apps) to enrich vehicle usage experiences.

Since BMS in EVs allows drivers to turn on AC or stop battery charging remotely by smartphones via wireless communication [52, 70]. Attacks on vehicle batteries can cause vehicle malfunctions and even threats to driving safety and human life. For example, an attacker may modify battery charging settings such as charging current during the charging process to generate over-high battery temperature and even battery explosion [1]. An attacker can also greatly consume battery energy without being noticed by the user to generate driving plan change and driving range anxiety. For

example, a battery attacker may turn on AC or stop battery charging process of an EV through the connected smartphone and cannot be caught by the battery management system, which only monitors battery states. Under this situation, the driving range anxiety affects vehicle driving safety and may even cause driving accidents [60]. There have been no methods for current EVs to detect such battery attacks to ensure battery security. We aim to analyze how to conduct battery attacks through a smartphone and design a battery authentication system to detect malicious AC-turn-on or battery-charge-stop requests from battery attacks.

Autonomous vehicle (AV) technologies have been adopted by many autonomous companies to reduce traffic congestion and improve driving safety [17, 33, 64]. With fast technology development of AVs and their possible popularity in the near future, several countries (e.g., United States, Germany, China and Australia) have stated that testing autonomous vehicles is allowed on public roads and many autonomous vehicle relevant researches [5, 7, 8, 12, 55, 72] are also conducted in recent years. An AV has control policies that specify trigger conditions and the control behavior that the AV should always execute when a trigger condition is satisfied. It is critical to ensure high driving safety performance of control policies in AVs. By now, more than 1,400 AVs have been tested by around 80 companies on public roads in USA [54] and these road tests have resulted in totally 256 accident reports (e.g., rear-end collisions, sideswipe collisions, hitting pedestrians or objects) in 2020. Recently, research works [16, 18, 28, 31, 32, 40, 61, 71] start focusing on the control policies of AVs to improve the driving safety of AVs. Though these methods help ensure driving safety of a target AV under several specific driving scenarios, they may fail to work when AVs drive on other different driving scenarios. In other words, the identified driving scenarios may not cover all possible driving scenarios in public roads by considering highly complex driving environments in practice [39]. Several AV companies develop their own control policies for their AVs and these control policies are usually not open for

the public, which makes it difficult to evaluate driving safety performance of their control policies for further improvement. In this research, we propose a control policy based driving safety system, which can derive control policies and choose the optimal control policy for an AV, to help improve AV driving safety.

With the development of reinforcement learning (RL) technology, RL is a powerful learning framework capable of learning complex policies in a high dimensional environment and has been used in AVs to make control decisions in real time. For an AV, the RL iterates control decisions under different driving states based on to simulation to form policies and its policy performance heavily depends on reward design and exploration coverage of driving state and action spaces. However, driving state and action spaces are huge for an AV considering continuous speed changes, it is difficult for a RL to form accurate policies only based on the simulation. Therefore, it is necessary for AVs to take the advantage of expert relevant knowledge during the policy formation process to obtain accurate policies. For an AV in the multi-AV coexistence situation, driving states of its nearby AVs need to be considered during its control decision making processes to ensure driving safety. Making control decisions simultaneously becomes beneficial for multiple AVs in the multi-AV coexistence situation since such a multiple decision making process considers driving states of AVs driving on the same road and can help make control decisions with higher driving safety. In this research, we will propose a multi-AV control decision making system to make control decisions for multiple AVs simultaneously so that their driving safety can be guaranteed during driving processes.

1.1 Challenges

We propose a Battery Authentication System, a Control Policy based Driving Safety System, and a Multi-AV Control Decision Making System to improve vehicle driving

safety. However, these exist several challenges to develop the above systems and we discuss the challenges below in more details.

1.1.1 Battery Authentication System

The proposed battery attack in this research remotely turns on AC or stops battery charging process through a smartphone, which is registered to connect to the EV. No previous works discuss the possible attacks on the vehicle batteries through the connected smartphone and there have been no methods for current EVs to detect such battery attacks to ensure battery security. One way is to provide authentications on the AC-turn-on and battery-charge-stop requests based on a user's habits to make the EVs resilient to the two battery attacks. However, it is challenging to accurately learn a user's habits due to two reasons and apply them to determine whether AC-turn-on and battery-charge-stop requests come from the driver in practice. Firstly, a user's behaviors are greatly affected by vehicle driving environments such as vehicle indoor temperature and battery State of Charge (SOC). Secondly, the situation where more than one users share one EV exists in the household and becomes popular because of the benefits such as insurance discount, which makes it more difficult to accurately distinguish a user's habits. Besides, the built user behavior model only provides statistical probabilities of user behaviors under different vehicle states. It is a challenge to make a correct authentication decision on a request only based on statistical probabilities.

1.1.2 Control Policy based Driving Safety System

Research works [16, 18, 28, 61, 71] try to keep developing new control strategies to control control behaviors of a target AV under different driving scenarios to improve AV driving safety. Though these methods help ensure driving safety of a target AV

under several specific driving scenarios, they may fail to work when AVs drive on other different driving scenarios. In other words, the identified driving scenarios may not cover all possible driving scenarios in public roads by considering highly complex driving environments in practice [39]. Several AV companies develop their own control policies for their AVs and these control policies are usually not open for the public, which makes it difficult to evaluate driving safety performance of their control policies for further improvement. Therefore, how to obtain control policies of a given AV for driving safety improvement becomes one challenge.

Another group of works try to test driving safety performance of control policies in an AV without knowing the control policies [31, 32] or with their own designed control policies [40]. These methods only check whether an AV will fail to work (i.e., hazard) under a certain driving scenario but cannot know which control policy leads to the hazard and accordingly make corrections. The method in [40] chooses the control behavior that leads to the safest condition measured by its relevant distance to its nearest vehicle. However, the method assumes that the state of its nearby vehicle always keeps constant in one short time period but actually the nearby vehicle usually has time-varying driving state in practice. For example, driving speed of the nearby vehicle may change dynamically or the nearby vehicle may firstly take a left lane change and then return back to its original road lane. Therefore, this assumption may result in wrong control policy selection for a target AV. Therefore, the other challenge is how to choose the optimal control behavior for a given trigger condition for an AV considering time-varying driving state of its nearby vehicle.

1.1.3 Multi-AV Control Decision Making System

RL based decision making methods [14, 15, 44, 74] have been proposed to apply the RL technology to make control decisions for an individual AV. Wolf [74] designed a

reinforcement learning approach using a deep Q network to form policies to control an AV based on the simulation. For the deep Q network, it is able to learn human driving behaviors and its action states are discrete steering angles. Lillicrap [44] proposed a reinforcement learning method which can form a neural policy to control a vehicle so that the vehicle can be kept driving in a simulated racing track. Chen [14] designed a reinforcement learning model to control a vehicle and such a model considers complex temporal delayed problems like traffic light passing scenarios during its decision making process. Yuan [15] applied the RL technology by inputting camera images to make a control decision for a vehicle so that a vehicle drives safely on different driving scenarios. However, these methods have very low policy formation efficiency and may even have policy convergence problems by considering that an AV usually has a huge driving state space. Therefore, how to efficiently form policies for an AV working on different driving scenarios becomes one challenge. Several multi-agent based methods [22, 27, 41, 46, 59] are applied to make control decisions for multi-agent problems. Methods [27, 59] consider join states of all agents as inputs of a centralized controller and output control decisions for all agents in a step. Specifically, they employed the mixed networks to estimate Q-functions for different joint states and made control decisions for multiple agents by maximizing global rewards. Methods [41, 46] tried to build a local reward function for each agent to learn its individual policy or Q-function and then make a control decision for each agent by maximizing each local reward without any explicit coordinations. However, these methods do not have explicit coordinations during the decision making process and their decisions cannot guarantee driving safety when used to control multiple AVs. Therefore, the other challenge is how to make control decisions for multiple AVs with safety guarantee.

1.2 Thesis Statement

The above discussion on the UBI program and CAV systems leads us to the following **thesis statement**:

"*By exploring the user behavior model, spatiotemporal features of control behaviors, and learning expert driving behaviors, one is able to detect malicious action requests on EV batteries and make control decisions for individual or multiple AVs to improve driving safety.*"

To be more specific, we investigated the following suppositions:

(1) The user behavior habits provide useful probability information about AC-turn-on requests or battery-charge-stop requests under different vehicle states. Therefore, we can explore these behavior habits to describe a user's habits in turning on AC and stopping battery charging actions.

(2) The spatiotemporal analysis of control behaviors helps us cluster historical driving data of a target AV with the same control behavior type together to extract its control policies.

(3) The expert driving trajectory data provides us the expert driving behaviors under different driving states to form control decisions for an individual AV.

The null hypothesis is that the introduction of a battery authentication system, a control policy based driving safety system, and a multi-AV control decision making system do not raise attention to the pre-existing vehicle driving safety concerns.

1.3 Contributions

We have developed a set of techniques to address the challenges mentioned in Section 1.1. The major contributions of this dissertation are:

(1) We design a battery authentication system such that it provides authentications on AC-turn-on and battery-charge-stop requests from smartphones to eliminate the effects of battery attacks on EV batteries and ensure battery security.

(2) We develop an effective driving safety technology, which can extract control policies of a target AV based on its historical driving data and determine optimal control policies for a target AV considering the time-varying driving state of the nearby vehicle.

(3) We develop a multi-agent reinforcement learning method to make control decisions for multiple AVs in multi-AV coexistence driving situations with safety guarantee simultaneously.

(4) We design and evaluate extensive data-driven experiments using Matlab Simulation [47], and Baidu Apollo simulation platform [4] to compare the solutions to the previously mentioned challenges.

1.4 Dissertation Outline

The rest of this dissertation is organized as follows.

- Chapter 2 discusses the state-of-the-art works related to electric vehicle battery security, control policy based driving safety analysis, and multi-AV control decision determination.

- Chapter 3 presents the proposed battery attack on EV batteries through a smartphone and a battery authentication system that utilizes a reinforcement learning model to judge if an action request from a smartphone is from battery attacks.

- Chapter 4 describes a control policy based driving safety system for an individual AV, which can derive control policies and choose the optimal control policy for an AV, to help improve AV driving safety.

- Chapter 5 presents a multi-AV control decision making system to make control decisions for multiple AVs simultaneously so that their driving safety can be guaranteed.

- Chapter 6 summarizes the contributions presented in the dissertation and provide future research directions.

Chapter 2

Related Work

2.1 Electric Vehicle Battery Security

Several existing attack defense methods have been proposed to ensure the smartphone security. One group of works [23, 24, 36] focus on improving the defense model in detecting and pruning malicious Apps from centralized mobile marketplaces. Egele et al. [23] statistically analyzed mobile Apps to detect possible privacy leaks of sensitive information. Enck et al. [24] tried to understand the broader security characteristics of existing Apps by studying free Apps from the official Google Play and identify malicious Apps from centralized marketplaces. However, they are far from ideal since malware authors can find new ways to penetrate current marketplaces. Another group of works [21, 25, 67] aim to develop mitigation solution on mobile devices. For example, Davi et al. [21] developed a control flow integrity enforcement framework to prohibit runtime and control-flow attacks on Apple iOS. Enck et al. [25] extended the Android framework to monitor the information flow of privacy-sensitive data. However, they all share a common assumption of a trustworthy framework and this may not be the case for advanced attacks which can directly compromise privileged system daemons. Since battery attack in this paper is introduced into a smartphone

during the update process of the App framework and it conducts the attack without extending vehicle mobile App permissions in a smartphone, it is difficult to detect battery attacks for existing attack defense methods.

Some works [35, 51, 68] discuss EV battery related security problem. Sripad et al. [68] built the battery model to analyze the potential impact of cyber-attacks utilizing the auxiliary components on the EV battery in short and long terms. Hunt [51] et al. controlled vehicle features of Nissan Leafs across the globe via the vulnerable website by knowing the vehicle identification number in advance. However, they both did not discuss the feasibility of attacks through a smartphone and how to eliminate their effects on the remaining battery energy. In this paper, we propose EV battery attacks through a smartphone which consumes battery energy by turning on AC or stop battery charging process. Other works discuss how to detect battery energy related attacks on smartphone. Timothy et al. [10] built a system which correlates smartphone power consumption with its communication activities to provide threshold monitoring and used the system to generate alert notification to users if power consumption does not match communication activities. Caviglione et al. [11] developed a battery energy monitoring system for smartphone so that it can check battery usage and generate issues alerts when anomalous currents are detected. Kim et al. [37] proposed a power-aware malware detection framework which collects application power consumption signatures to detect energy-greedy mobile malwares. However, they assume that battery related attack will result in uncommon phenomena such as anomalous currents. This assumption does not work on EV batteries since AC-turn-on and battery-charge-stop will not result in uncommon phenomena.

Some works [2, 19, 48, 63, 73] discuss how to identify the user based on vehicle related information. These works try to identify the user by analyzing driving behavior signals which can be obtained through sensor measurement or CAN bus message. Wakita et al. [73] firstly built Gaussian Mixture Model and then generated driving

simulation data including vehicle speed, distance from vehicle ahead and accelerator pedal pressure to train the model for user identification. Miyajima et al. [48] collected brake pedal pressure and gas pedal pressure data through in-vehicle's sensor data and applied them into Gaussian Mixture technology to identify the user. Choi et al. [19] obtained steering wheel, vehicle speed, engine speed and brake position signals by reading CAN bus messages and input them into Gaussian Mixture Model and Hidden Markov Model to identify the user. Though these methods provide relatively high identification accuracy, these driving behavior signals cannot be obtained easily in practice.

2.2 Control Policy based Driving Safety Analysis

Methods have been proposed to analyze safety features of AVs to ensure their driving safety. [16, 18, 28, 61, 71] try to keep developing new control strategies to control AV's control behaviors at different driving scenarios so that a target AV can drive safely under these driving scenarios. For example, Rosolia [61] developed a nonlinear control approach to generate a collision-free trajectory for a target AV so that a target AV has the ability of avoiding obstacles when driving on the highway. Chen [16] developed a fuzzy control system which adjusts driving direction and driving speed of a target AV in real time with traffic condition consideration to ensure driving safety. Tian [71] proposed a decision making algorithm which models multi vehicles driving in a roundabout intersection situation with game theory and helps a target AV decide whether it should enter and cross the intersection. Galceran [28] proposed an integrated behavior inference and decision-making approach which models vehicle behavior of a target AV and determines a set of control behaviors for a target AV with considering control behavior of its nearby vehicle to avoid possible collisions with its nearby vehicle. Chen [18] proposed a neural network based control decision-making

system which uses a neural network to learn driving behavior of a human driver and make control decisions based on its trained neural network to avoid possible collisions. Though these methods help to ensure driving safety of a target AV under several specific driving scenarios, they may fail to work for other driving scenarios. In other words, the total number of these specific driving scenarios are limited and these specific driving scenarios cannot cover all possible driving scenarios in public roads by considering highly complex driving environments in practice. Besides, several AV companies develop their own control policies for their AVs and these control policies are usually not open for the public, which makes it difficult to analyze their control policies for further improvement.

Methods [31, 32, 40] try to test driving safety performance of control policies in an AV through driving simulation or road experiments. For example, Jha [32] designed an AV fault injection simulator to test its control policies by injecting fault sensor information into a target AV and simulating how a target AV responses based on its control policies. These methods assume that the state of a nearby vehicle always keeps constant in one short time period, which makes their testing results less reasonable in practice. Road experiments can better test driving safety performance of control policies in a target AV and Hunger [31] developed a driving scenario formalization method to test control policies under different driving scenarios during road experiments. However, existing methods only check whether a target AV with control policies will fail to work under a certain driving scenario but it is difficult for them to figure out which control policy can be selected to ensure driving safety of the target AV under a certain driving scenario.

2.3 Multi-AV Control Decision Determination

Several single-agent RL based methods [14, 15, 44, 74] have been proposed to apply reinforcement learning to make control decisions for an individual AV. Wolf [74] designed a reinforcement learning approach using a deep Q network to form policies to control an AV based on the simulation. For the deep Q network, it is able to learn human driving behaviors and its action states are discrete steering angles. Lillicrap [44] proposed a reinforcement learning method which can form a neural policy to control a vehicle so that the vehicle can be kept driving in a simulated racing track. Chen [14] designed a reinforcement learning model to control a vehicle and such a model considers complex temporal delayed problems like traffic light passing scenarios during its decision making process. Yuan [15] applied the RL technology by inputting camera images to make a control decision for a vehicle so that a vehicle drives safely on different driving scenarios. However, these methods have very low policy formation efficiency and may even have policy convergence problems by considering that an AV usually has a huge driving state space.

Several multi-agent RL methods [22, 27, 41, 46, 56, 59] are developed to make control decisions for multi-agent problems. Methods [27, 59] consider join states of all agents as inputs of a centralized controller and output control decisions for all agents in a step. Specifically, they employed the mixed networks to estimate Q-functions for different joint states and made control decisions for multiple agents by maximizing global rewards. Methods [41, 46] tried to build a local reward function for each agent to learn its individual policy or Q-function and then make a control decision for each agent by maximizing each local reward without any explicit coordinations. However, these methods do not have explicit coordinations during the decision making process and their decisions cannot guarantee driving safety when used to control multiple AVs.

Chapter 3

Battery Authentication System

3.1 Battery Attacks on EVs

Our proposed battery attack remotely turns on AC or stops battery charging process through a smartphone, which is registered to connect to the EV by the user via vehicle mobile App installation. For a smartphone, it can be easily infected by battery attacks after the user installs the vehicle mobile App into it. Among existing malware infection methods [3, 58, 76], we focus on the most popular situation when the framework of smartphone operating system has self-update functionality or the user updates it manually, the malicious App loads additional code for so-claimed benign reasons (e.g., framework update or beta testing) and replaces original code with malicious code which will not be checked by the smartphone operating system [57]. Then, the battery attack infects the smartphone by running malicious code in the context of an application without the user's permission and has full access to the smartphone functions.

Figure 3.1 shows the details of EV battery attack process. In the smartphone operating system, vehicle mobile App and other Apps may be installed at different frameworks or just share the same framework. During the framework self-update

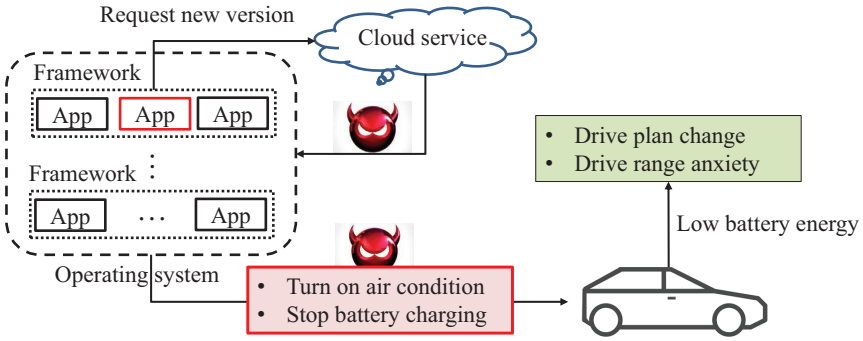


Figure 3.1: Battery attacks on EV batteries through a smartphone.

process, code including battery attack is firstly downloaded from the cloud service and executed in the vehicle mobile App in the smartphone. This way, the battery attack infects the smartphone successfully and reduces EV battery energy remotely by turning on AC or stopping battery charging process of its connected EV. More specifically, the battery attack may result in battery energy loss by turning on AC and stopping battery charging process during battery charging process. The vehicle mobile App provides the user real-time vehicle states and sends out action requests to the EV through mobile communication network such as 3G, 4G or LTE [75]. Therefore, there is no time or distance constraint for launching the battery attacks in practice.

Algorithm 1: Part code in a vehicle mobile App

```
1 client.on_connect = on_connect
2 while True do
3     schedule.run_pending()
4     climate_control(climate_control_instruction)
5     charge_control(charge_control_instruction)
6     mqtt_publish(get_vehicle_status(),get_battery_update())
7     time.sleep(1)
8 end
```

Algorithm 1 shows the code snippet in a vehicle mobile App for controlling AC and battery charge process. The App firstly builds a connection between the smartphone and the EV when the user opens the vehicle mobile App (Line 1). After successful connection, the App starts to check for possible pending requests (Line 3). When

the App receives AC-turn-on and battery-charge-stop commands from the user, it sends these requests to EV through commands *climate_control* and *charge_control* to turn on AC or stop battery charging process (Lines 4 and 5). Lastly, the App updates battery state and vehicle state shown in App interface with frequency of 1 *time/second* (Lines 6 and 7). In the battery attack, the attacker modifies parameter *climate_control_instruction* or *charge_control_instruction* to generate malicious action requests. More specifically, for malicious battery-charge-stop request, it only happens during the battery charging process. For malicious AC-turn-on request, it happens at any time. We divide our proposed battery attacks into No-effort attacks and Smart attacks, which are explained below.

No-effort attacks In a No-effort attack, it sends malicious action requests randomly without considering whether user is in the vehicle. Its objective is to limit the charged energy by stopping charging process or consume EV battery energy as much as possible. Therefore, No-effort attack may stop battery charging batteries during the battery charging process or turn on AC at any time. However, No-effort attack has high possibility of being detected by a user if the user is in the vehicle.

Algorithm 2: Part code in Smart attack

```
1 while User is not in the vehicle do
2 |   climinate_control_instruction=1
3 |   SOC = get_battery_update()
4 |   if SOC > SOC' then
5 |   |   charge_control_instruction=0
6 |   end
7 |   SOC'=SOC
8 end
```

Smart attacks Here, we propose smart attacks to reduce the possibility of being detected by users. Only when the user is not in the vehicle, Smart attack will send malicious battery-charge-stop requests during the battery charging process or malicious AC-turn-on requests. In other words, Smart attack sends malicious battery-charge-stop requests when the user is not in the vehicle during the battery charging process.

For malicious AC-turn-on requests, Smart attack will send them only if the user is not in the vehicle and will not consider whether the vehicle is in the battery charging process. The details about how to conduct Smart attack through the vehicle mobile App are shown in Algorithm 2. When the user is not in the vehicle, Smart attack will send a malicious AC-turn-on request by modifying *climate_control_instruction* (Line 2). For malicious battery-charge-stop requests, Smart attack firstly obtains real-time battery SOC through the command *get_battery_update* (Line 3). And then, Smart attack determines whether the vehicle is in the battery charging process by comparing current battery SOC and its previous value SOC' (Line 4). If SOC is larger than SOC', the vehicle is in the battery charging process and Smart attack will stop battery charging process by modifying *charge_control_instruction* (Line 5). Lastly, SOC is recorded and used for comparison in next loop (Line 7).

3.2 System Design of Bauth

To eliminate the effects of battery attacks on EV batteries and ensure battery security, we propose Bauth for EVs. Bauth runs on a micro-controller in an EV. We present the architecture of Bauth in Figure 3.2. Bauth consists of two major parts: data-driven behavior model and reinforcement learning model. The data-driven behavior model part firstly uses battery state to identify the user for AC-turn-on and battery-charge-stop request authentication. More specifically, it identifies the user currently driving the EV or the user that drove the EV lastly. Here, we assume the user who drove the vehicle lastly is the person to charge the vehicle. And then, for a user, the data-driven behavior model calculates state transform probability of taking an action. Here, we calculate state transform probability by analyzing historical vehicle usage data and use the state transform probability as the reward in training a reinforcement learning model. In the reinforcement learning model part, for a given vehicle state, it makes

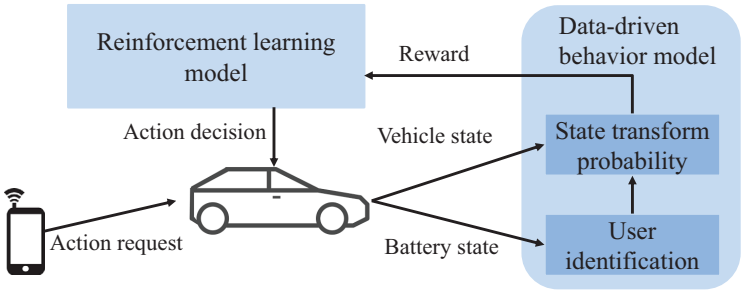


Figure 3.2: The architecture of Bauth for action authentication.

decisions on accepting or rejecting an action request (turning on AC or stopping charging) by maximizing cumulative rewards, i.e. reflecting the user's habits on the actions for the request. When Bauth receives an action request (turning on AC or stopping charging), Bauth first identifies the user. Then, it decides to accept or reject this request based on the current vehicle state. For reading convenience, we list frequently used terms and their corresponding meanings in Table 3.1.

Table 3.1: Terms used in this paper

Terms	Meanings
Vehicle usage data	initial vehicle state, subsequent vehicle state, AC-turn-on/battery-charge-stop
Vehicle state	vehicle indoor temperature (Tem), and battery SOC (SOC)
Battery state	battery current (c), battery power output (p), and battery SOC (SOC)

3.2.1 Data-driven Behavior Model

To judge whether a received AC-turn-on or battery-charge-stop request is authentic, we hope to check if current vehicle usage and battery states match the user's habits in turning on AC or stopping charging battery. However, it is difficult to learn a user's such habits. This is because a user's behaviors are affected by factors including battery SOC. Besides, the situation where more than one users share one EV exists (e.g., in the household) and becomes popular because of benefits such as insurance discount, which makes it more difficult to accurately learn a particular user's habits.

If the driving data of the same vehicle from different users is directly used to build the behavior model, the model accuracy will be decreased greatly. Therefore, we need to identify each user first before learning the user's habits. For this purpose, vehicle mobile App in a smartphone records vehicle usage data. For each sample in vehicle usage data, it includes battery SOC, vehicle indoor temperature and status of AC-turn-on and battery-charge-stop. Both battery SOC and status of AC-turn-on and battery-charge-stop can be obtained by vehicle mobile App. The vehicle indoor temperature is measured by vehicle internal temperature sensor and its value can be sent to the smartphone through bluetooth. This way, the behavior model can be built based on these data samples.

3.2.1.1 User Identification

Several researches [19, 48, 73] use popular classification algorithms to identify users based on driving features such as accelerator, brake and steering handle speed and have advantages such as high identification accuracy. However, the data about these driving features is difficult to be obtained in practice, which limits their application. To solve this problem without compromising the advantage, we develop a user identification method to identify users based on only battery state which is much easier to be obtained.

When a user drives an EV on the road, the user's driving behaviors can be reflected by EV battery state. For example, the battery current is positive when the user accelerates the vehicle and the current magnitude will become larger when the vehicle accelerates with a larger value. Similarly, the battery current becomes negative when the user decelerates the vehicle and its magnitude will also increase as deceleration value becomes larger. Therefore, battery state can be used to describe a user's driving behaviors and identify users. The vehicle mobile App connected to EVs provides real-time battery state. By considering that the battery state can be easily obtained

through vehicle mobile App, we propose a user identification method which uses random forest technology [42] to identify the user using the battery state. Random forest classifies tasks by constructing a multitude of decision trees at training time and outputting the class result, which is the mean prediction of individual trees. Compared with other classification methods such as decision tree, support vector machine, and neural network, random forest overcomes possible over-fitting issues during the training process and has more accurate classification performance for the situation where training data has unbalanced class populations. This situation happens when users sharing the same vehicle drive the vehicle with different frequencies [38].

In the proposed user identification method, the real-time battery state is used by the random forest technology to identify the user. More specifically, the battery state including battery current c, battery power output p, and battery SOC are the inputs of random forest. By considering that battery state has different scales, we need to normalize battery state value x so that these inputs can be equally treated by the random forest during the training process. The normalization process is shown as follows:

$$X_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}, \tag{3.1}$$

where x_i represents the i^{th} battery state sample $[c, p, SOC]$ and X_i is the normalized battery state value. Based on the above equation, the normalized battery state is located between 0 and 1 and is sent to the random forest to identify the user.

Because of the dynamic changes in traffic condition and driving environment including road surface and road gradient, the user usually accelerates and decelerates frequently to ensure driving safety, which results in high fluctuations in battery state values. Such fluctuations affect the identification accuracy of the random forest in training. To eliminate the effects of fluctuations, we use the statistical based method, in which the mean value of the normalized battery state X from time $t - \Delta$ to t

is considered as the statistical battery state $\bar{X}$ at time t. For the same normalized battery state X, the time period Δ determines final statistical battery state $\bar{X}$. How to determine the optimal time period Δ so that the random forest has the highest identification accuracy becomes important for our user identification method. To determine optimal time period Δ, we firstly use the statistical based method to process the same training data and calculate final statistical battery states under different time periods. And then, we train the user identification model with these final statistical battery states and use the trained models to identify users. Lastly, we compare identification accuracies among these models and use the time period of the model with the highest identification accuracy as the optimal time period. Based on the normalization and statistical process, final inputs $\bar{X}$ are obtained and used to train random forest.

The random forest is an ensemble of E trees $T_1(\bar{X}), ..., T_E(\bar{X})$, where T_i is the i^{th} decision tree and $\bar{X}$ represents statistical battery state $[\bar{c}, \bar{p}, \overline{SOC}]$ at each time. Figure 3.3 shows how Bauth applies a random forest method to identify users based on real-time battery state. Given a set of n training samples $(\bar{X}, Y)$ where Y represents user ID of $\bar{X}$, each decision tree T will be firstly trained until the ensemble of E trees is formed. And then, the trained random forest can identify the user based on battery state during the driving process. In practice, road traffic conditions may affect a user's driving features. To eliminate the effects of road traffic condition on user identification accuracy, we use driving data at different traffic level situations to train our model to ensure its identification accuracy. Traffic level situation such as light, heavy and jam can be directly obtained from Google

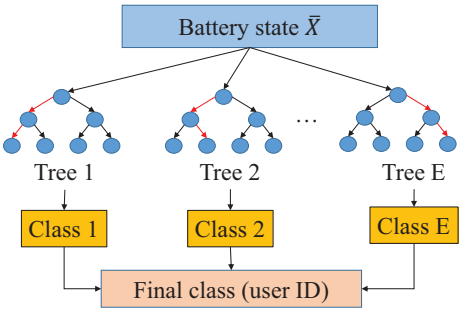


Figure 3.3: Random forest method used for identifying users.

Maps. In summary, for multi-user sharing one vehicle situations, the user can be identified online based on our user identification method and her/his corresponding AC-turn-on behavior data will be saved for the following AC-turn-on behavior analysis. For the battery-charge-stop behavior data, by considering that the user who lastly drives the vehicle at past time periods is more likely to charge batteries, we assume that the battery charging process is always conducted by the user who lastly drives the vehicle at past time periods. Under this condition, the battery-charge-stop behavior data should belong to the user who lastly drives the vehicle and will be used for battery-charge-stop behavior analysis.

3.2.1.2 Calculation of State Transform Probability

A user turns on AC only when (s)he feels that the vehicle's indoor temperature is out of his/her comfort range. Some users may face the balance between the temperature and driving plan because of limited battery energy. In other words, some users would turn on AC only when the battery SOC is high enough to finish their driving plans. Therefore, AC-turn-on action is determined by both vehicle indoor temperature and battery SOC. For the battery charging, a user charges the EV either at public charging parking lot or at private home. The user determines the battery charging time by considering remaining battery energy and his/her vehicle-use time schedule. However, the remaining battery energy is determined by vehicle usages and driving situations such as high road traffic. Therefore, constant battery-charge scheduling of charging EV batteries does not work on EV batteries. Otherwise, a malicious battery-charge-stop request can be easily detected by considering whether it follows a driver's battery-charge schedule.

Based on the above analysis, we find that both a user's AC-turn-on and battery-charge-stop behaviors are mainly affected by vehicle indoor temperature Tem and battery SOC but there are no published researches that discuss the effects of Tem

and SOC on user's AC-turn-on and battery-charge-stop behaviors. Our work is the first that uses Tem and SOC to predict a user's AC-turn-on and battery-charge-stop behaviors. Here, we statistically analyze the dynamics of battery state and vehicle usage data to calculate statistical probabilities of taking actions under certain vehicle state (Tem and SOC). Statistical probabilities describe a user's habits in AC-turn-on and battery-charge-stop behaviors under different vehicle states. If statistical probabilities are used as reward in reinforcement learning model shown in Section 3.2.2, decisions made by reinforcement learning model will follow a user's habits and helps to authorize AC-turn-on and battery-charge-stop requests.

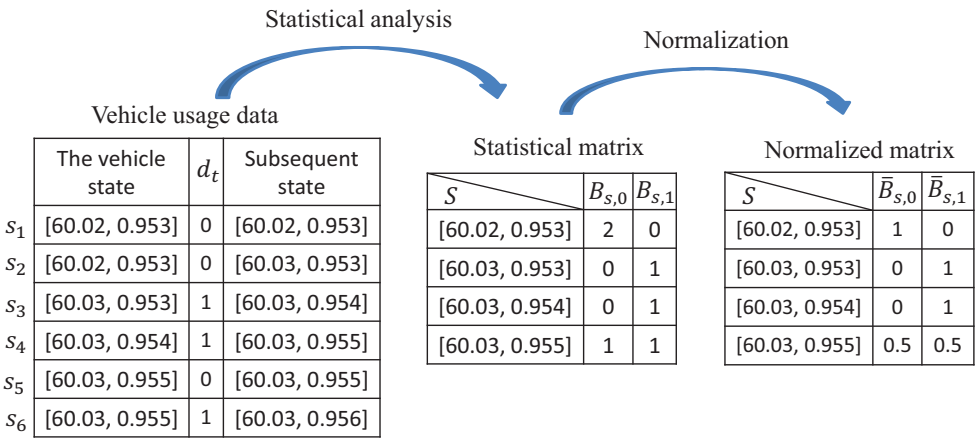
For a given user and a given action request (AC-turn-on or battery-charge-stop), we calculate the statistical probability of taking certain actions under different vehicle states (represented by a matrix $\mathcal{B}$) based on historical vehicle usage data M. In this paper, we use s_t to represent vehicle state (Tem and SOC) and d_t to represent user actions (0 represents reject and 1 represents accept) on the given action request. The element B_{s_t,d_t} in the matrix $\mathcal{B}$ is calculated by the following equation:

$$B_{s_t,d_t} = \sum_{(s',d')\in M} \delta_{s_t,s'}\delta_{d_t,d'}, \tag{3.2}$$

where (s', d') is one data sample which means that the user takes action d' at vehicle state s'. $\delta_{x,y}$ is the Kronecker delta which equals to 1 when x equals to y, and 0 otherwise. Other elements in the matrix equals to zero. The normalization of Equation (3.2) is calculated as follows:

$$\bar{B}_{s_t,d_t} = \frac{B_{s_t,d_t}}{\sum_{d_t\in\{0,1\}} B_{s_t,d_t}}. \tag{3.3}$$

Elements $\bar{B}_{s_t,d_t}$ form a matrix $\bar{\mathcal{B}}$. For each row of $\bar{\mathcal{B}}$, the sum of elements $\sum_{d_t\in\{0,1\}} \bar{B}_{s_t,d_t}$ equals to 1. The matrix element $\bar{B}_{s_t,d_t}$ in the normalized matrix can describe the prob-



	The vehicle state	d_t	Subsequent state
s_1	[60.02, 0.953]	0	[60.02, 0.953]
s_2	[60.02, 0.953]	0	[60.03, 0.953]
s_3	[60.03, 0.953]	1	[60.03, 0.954]
s_4	[60.03, 0.954]	1	[60.03, 0.955]
s_5	[60.03, 0.955]	0	[60.03, 0.955]
s_6	[60.03, 0.955]	1	[60.03, 0.956]

S	$B_{s,0}$	$B_{s,1}$
[60.02, 0.953]	2	0
[60.03, 0.953]	0	1
[60.03, 0.954]	0	1
[60.03, 0.955]	1	1

S	$\bar{B}_{s,0}$	$\bar{B}_{s,1}$
[60.02, 0.953]	1	0
[60.03, 0.953]	0	1
[60.03, 0.954]	0	1
[60.03, 0.955]	0.5	0.5

Figure 3.4: Statistical probability matrix calculation with vehicle usage data.

ability that the given user takes action d_t under vehicle state s_t when the number of samples in vehicle usage data is large enough. Figure 3.4 illustrates an example of calculating the probability. These vehicle state samples over time are expressed by s_1, s_2, s_3, s_4, s_5, and s_6. Note that s_1 is the same as s_2 and s_5 is the same as s_6. We firstly statistically analyze on the samples to calculate the number of taking action d_t at vehicle state s_t through Equation (3.2) and form the statistical matrix $\mathcal{B}$. And then, we normalize the statistical matrix to form statistical probability matrix $\bar{\mathcal{B}}$ through Equation (3.3).

Next, we will calculate the state transform probability based on the normalized matrix $\bar{\mathcal{B}}$.

Based on the normalized matrix $\bar{\mathcal{B}}$, we calculate the normalized state transform probability matrix $\bar{\mathcal{D}}$ from one vehicle state to another vehicle state under action d_t. $\bar{\mathcal{D}}$ indicates how vehicle state would like to be s_{t+1} if a given user takes action d_t under vehicle state s_t.

Given an initial vehicle state s_t under action d_t, we need to calculate the normalized state transform probability from s_t to s_{t+1} (denoted by $\bar{D}_{s_t,d_t,s_{t+1}}$) based on the normalized element $\bar{B}_{s_t,d_t}$. $\bar{D}_{s_t,d_t,s_{t+1}}$ represents the probability when the user takes action d_t and vehicle state transforms from s_t to s_{t+1}. To calculate $\bar{D}_{s_t,d_t,s_{t+1}}$, we

need to figure out state transform probability $D_{s_t,d_t,s_{t+1}}$ which shows the sum of state transform probabilities from s_t to s_{t+1} with action d_t. The state transform probability $D_{s_t,d_t,s_{t+1}}$ can be calculated as follows:

$$D_{s_t,d_t,s_{t+1}} = \sum_{s' \in S} \delta_{(s_t \to s_{t+1}),(s_t \to s')} \bar{B}_{s_t,d_t}, \tag{3.4}$$

where S represents all possible vehicle states in vehicle usage data. Here $(s_t \to s')$ means that vehicle state changes from s_t to s'. We use $\delta_{(s_t \to s_{t+1}),(s_t \to s')}$ to check whether vehicle state s' changing from s_t equals to s_{t+1}. $\delta_{(s_t \to s_{t+1}),(s_t \to s')}$ is 1 when s' equals to s_{t+1}, and 0 otherwise. We normalize state transform probability $D_{s_t,d_t,s_{t+1}}$ to obtain the normalized state transform probability $\bar{D}_{s_t,d_t,s_{t+1}}$ as follows:

$$\bar{D}_{s_t,d_t,s_{t+1}} = \frac{D_{s_t,d_t,s_{t+1}}}{\sum_{s_{t+1} \in S} D_{s_t,d_t,s_{t+1}}}. \tag{3.5}$$

We use the same vehicle usage data in Figure 3.4 to calculate state transform probability and show the calculation process in Figure 3.5. s^1, s^2, s^3, and s^4 represent all possible vehicle states in the sample. We firstly calculate the probability $\mathcal{D}$ of vehicle state transferring from s_t to s_{t+1} with Equation (3.4) based on vehicle usage data and matrix $\bar{\mathcal{B}}$ in Figure 3.4. And then, we normalize the state transform probability $\mathcal{D}$ to obtain the normalized state transform probability $\bar{\mathcal{D}}$.

When building the data-driven behavior model offline, Bauth runs in a microcontroller in the vehicle and collects vehicle usage data using a smartphone registered to connect the vehicle. Note that the smartphone does not need to be in the vehicle. Based on the user identification method in Section 3.2.1.1, Bauth identifies the user currently driving the EV (for AC-turn-on behavior learning) or the user that drove the EV lastly (for battery-charge-stop behavior learning). Then, Bauth conducts statistical analysis on the user's vehicle usage data using the methods introduced in

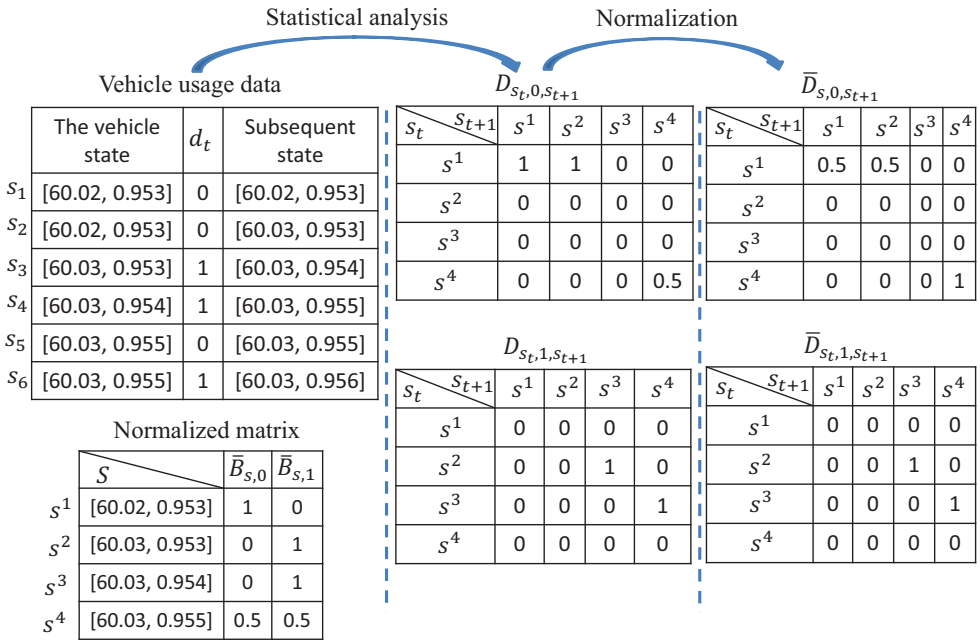


Figure 3.5: State transform probability matrix calculation.

Section 3.2.1.2. And then, $\bar{D}_{s_t,d_t,s_{t+1}}$ will be calculated to show the probability that the user will take action d_t on a given action request at vehicle state s_t and the state is transformed to s_{t+1}. Since the above probability can fully reflect user habits on a given action request (AC-turn-on or battery-charge-stop), we will use this probability into reinforce learning to indicate the reward when Bauth chooses decision a_t under vehicle state s_t. That is, reward function $r(s_t, a_t, s_{t+1})$ equals to $\bar{D}_{s_t,d_t,s_{t+1}}$, where d_t equals to a_t, and indicates the reward after transforming from s_t to s_{t+1} with decision a_t. Here, larger reward means higher probability that the decision a_t from Bauth will follow a user's habits. This way, Bauth chooses decision a_t based on the corresponding reward it will obtain to ensure its decision a_t follow user habits.

3.2.2 Reinforcement Learning based Authentication

Bauth uses the reinforcement learning model to authorize action requests (AC-turn-on and battery-charge-stop) from a smartphone. As shown in Figure 5.1, the current

vehicle state is the input to the reinforcement learning model, which outputs the decision (accept or reject) on the action request. To measure the current vehicle state $s_t = (Tem, SOC)$, the vehicle indoor temperature sensor and vehicle mobile App firstly measure vehicle indoor temperature and battery SOC at time t and send s_t to reinforcement learning model for action authentication. Note that the vehicle mobile App in a smartphone can not only measure battery SOC but also have the function of sending action request.

In the training of the reinforcement learning model, the inputs are vehicle usage data and state transform probability matrix (considered as reward), and the output is the authentication decision a from all possible options (reject or accept) for each vehicle state (which is called optimal policy π^*). The training firstly uses vehicle state s_t and state transform probability matrix to calculate the expected cumulative discounted rewards $E[\sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_{t'}]$ (also called Q-value) for different authentication decisions and then outputs the authentication decision which leads to the maximum value of $E[\sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_{t'}]$, where $\gamma \in [0, 1]$ represents discount factor, t' indicates a variable changing from t to positive infinity (a value which is larger than t in this paper), and $r_{t'}$ means the reward at time t'. Here, the policy is used to choose authentication decisions and calculate the expected cumulative discounted rewards. An optimal policy ensures that its authentication decisions for different vehicle states always conform actions in vehicle usage data.

3.2.2.1 Optimal Policy Formation

The policy π is defined as one map $\pi : s_t \mapsto a_t$ and guides reinforcement learning model to choose authentication decision a_t when vehicle state is s_t in this paper. The policy π keeps being adjusted during the training process until its authentication decisions under different vehicle states conform actions in vehicle usage data to form optimal policy π^*. For π^*, its authentication decision can follow a user's habits with

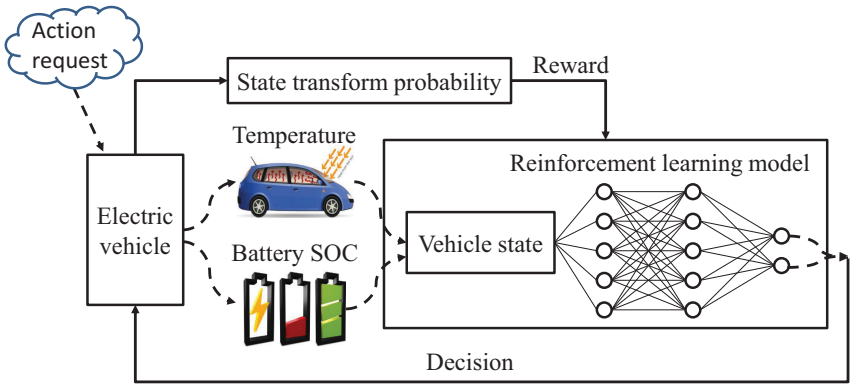


Figure 3.6: Reinforcement learning application in Bauth.

high confidence. During the training process, action-value function Q at initial vehicle state s_t and authentication decision a_t at time t is introduced to indicate the expected cumulative discounted reward $E[\sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_{t'}]$ at vehicle state s_t and given by:

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma \max Q^{\pi}(s_{t+1}, a_{t+1}). \tag{3.6}$$

We see that $Q^{\pi}(s_t, a_t)$ is the reward reinforcement learning model receives for entering current vehicle state s_t plus the maximum future reward for next vehicle state s_{t+1}. For one vehicle state, the policy π chooses authentication decision which can lead to the maximum value of action-value function Q^{π}. Therefore, Q^{π} affects attack detection performance of the policy π and needs to be calculated accurately.

Q^{π} of the policy π will keep being updated through vehicle usage data during the training process until the authentication decision from the policy π maximizes its Q-value, and then this π is the optimal policy π^*. However, it is difficult to calculate Q-values in action-value function and find the optimal policy through Equation (3.6) when the total number of all possible vehicle states is too large. For the vehicle state $s = (Tem, SOC)$, if Tem changes between $40F$ and $90F$ and SOC changes between 0 and 1, by considering that Tem change unit is $0.01F$ and SOC change unit is 0.001, the total number of possible combination of different Tem and SOC reaches at the

10^6 level, which will result in the curse of dimensionality problem in practice [44]. Besides, these many vehicle states can result in huge overload in computing Q-values. Therefore, rather than computing Q-values directly through Equation (3.6), we apply deep neural network into reinforcement learning model to approximate Q-values.

Neural networks form policy π in reinforcement learning model and help to estimate Q-values under different vehicle states and choose corresponding authentication decisions. They contain two main parts: actor neural network with parameters θ^μ and critic neural network with parameters θ^Q. These parameters are interconnection weights in networks and will be adjusted during the policy training process to improve network performance. More specifically, both inputs of two networks are vehicle usage data and state transform probability. The outputs of actor and critic neural networks are action function $\mu(s_t|\theta^\mu)$ and Q-value $Q(s_t, a_t|\theta^Q)$ for authentication decision a_t, respectively. Here, action function $\mu(s_t|\theta^\mu)$ makes decisions based on Q-values under vehicle state s_t and its output is authentication decision a_t which obtains maximum Q-values.

Action function $\mu(s_t|\theta^\mu)$ specifies current policy π by mapping vehicle state s_t to certain authentication decision a_t through $argmax_{a_t \in \{0,1\}} Q(s_t, a_t)$. We utilize actor-critic algorithm [49] to adjust parameters θ^Q and θ^μ during the policy training process so that Q-values are calculated accurately and authentication decision from policy π reflects a user's habits. As one of state-of-the-art methods, the actor-critic algorithm adjusts parameters in actor neural network and critic neural network based on gradient of the expected cumulative discounted reward and approximation error, which are explained in details in the following.

The key idea of actor-critic algorithm is to estimate the gradient of the expected cumulative discounted reward $E[\sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_{t'}]$ so that parameters in actor and critic neural networks can be adjusted. To adjust parameters θ^μ in actor neural network,

the gradient of the cumulative discounted reward with respect to the parameters θ^{μ} is calculated as $\bigtriangledown_{\theta^{\mu}} E[\sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_{t'}]$ [66]. This gradient describes how the expected cumulative discounted reward changes when policy with parameters θ^{μ} selects authentication decision at vehicle state s_t. Parameters θ^{μ} will be optimized to increase the expected cumulative discounted reward by following positive gradient direction.

For critic neural network parameters θ^{Q}, the difference between Q-values approximated by critic neural network and target Q-values calculated through Equation (3.6) is considered as the approximation error $L(\theta^{Q})$ of critic neural network [49] and calculated as follows:

$$L(\theta^{Q}) = E[(Q(s_t, a_t|\theta^{Q}) - Q^{\pi}(s_t, a_t))^2], \tag{3.7}$$

where $Q^{\pi}(s_t, a_t)$ represents Q-value in the policy π and can be calculated through Equation (3.6). $L(\theta^{Q})$ depends on parameters θ^{Q} in critic neural network and θ^{Q} can be updated by minimizing $L(\theta^{Q})$ in Equation (3.7).

The above two paragraphes show how actor-critic algorithm uses the gradient and the approximation error to adjust parameters in networks. More details about how to adjust parameters in actor and critic neural networks can be found in [49, 66]. Based on actor-critic algorithm, parameters in the policy will be adjusted in the policy training process and the policy after the training process will better approximate Q-values and make authentication decisions on requests. The optimal policy makes authentication decisions, which can follow a user's habits so that the battery attacks will not be authenticated.

3.2.2.2 GAN-based Vehicle Usage Data Generator

For an action request from a smartphone, Bauth firstly obtains vehicle state and then generates an authentication decision based on its reward function. Reward function is

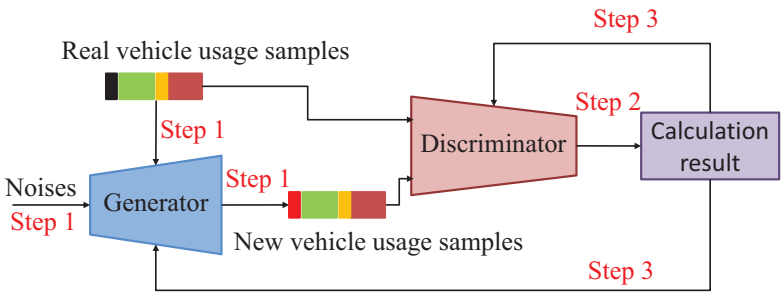


Figure 3.7: The architecture of a GAN-based vehicle usage data generator in Bauth.

built by statistically analyzing vehicle usage data. The more historical vehicle usage data is used for statistical analysis, the more accurate the reward function describes a user's habits. However, vehicle usage data is collected from real EVs and it is difficult to collect a large number of vehicle usage samples, which can cover possible different action-vehicle state pairs, for statistical analysis. To ensure attack detection performance of Bauth, we develop a GAN-based vehicle usage data generator to generate vehicle usage data. The basic idea of the GAN-based vehicle usage data generator is to generate new vehicle usage samples, which match the user's habits. Bauth firstly collects real vehicle usage data from a vehicle and then uses the GAN-based vehicle usage data generator to generate new vehicle usage data (user's action decision on an AC-turn-on or battery-charge-stop under different vehicle states). Bauth will keep generating new vehicle usage samples until the total number of vehicle usage samples is large enough to cover possible different action-vehicle state pairs.

Figure 3.7 shows how the GAN-based vehicle usage data generator in Bauth forms new vehicle usage samples, which can accurately match the user's habits. Here, the GAN-based vehicle usage data generator includes a gain part and a discriminator part and works by following total three steps. In step 1, for a given user's action decision, the generator part adds random noises into its corresponding vehicle state (vehicle indoor temperature and battery SOC) to generate a new vehicle usage sample. In step 2, the discriminator part firstly checks whether vehicle state in the new vehicle usage

sample stays in its statistical vehicle state range and then determines whether such a new vehicle state sample should be stored or ignored. In step 3, the generator part and the discriminator part influence each other and iteratively update themselves. Specifically, the generator part updates itself incrementally so that the generated vehicle usage samples are increasingly more similar to real vehicle usage samples. Once the discriminator determines that the vehicle state in the new vehicle usage sample stays in the statistical vehicle state range, the discriminator will store such a new vehicle usage sample. This iterative updating process will continue until the total number of new and real vehicle usage samples are large enough to cover all possible action-vehicle state pairs.

To generate new vehicle usage samples, which match the user's habits, the generator firstly needs to calculate statistical vehicle state ranges when the user conducts a action decision (AC-turn-on or battery-charge-stop) based on real vehicle usage data. For a given action decision a in the real vehicle usage data, its statistical vehicle indoor temperature range is indicated with $(\mu^a_{Tem}, \sigma^a_{Tem})$, where μ^a_{Tem} and σ^a_{Tem} indicate the mean value and the standard deviation value of Tem. Similarly, the statistical battery SOC range is indicated with $(\mu^a_{SOC}, \sigma^a_{SOC})$, where μ^a_{SOC} and σ^a_{SOC} indicate the mean value and the standard deviation value of SOC. And then, the generator forms a new vehicle usage sample (d', Tem', SOC'). Specifically, for a given user's action decision d under vehicle state (Tem, SOC), the generator forms a new vehicle usage sample by adding random noises θ_{Tem} and θ_{SOC} in Tem and SOC. Under this situation, the new vehicle usage sample (d', Tem', SOC') equals to $(d, Tem + \theta_{Tem}, SOC + \theta_{Tem})$. The discriminator will calculate vehicle indoor temperature difference as $Tem + \theta_{Tem} - \mu^a_{Tem}$ and battery SOC difference as $SOC + \theta_{SOC} - \mu^a_{SOC}$. If $Tem + \theta_{Tem} - \mu^a_{Tem}$ stays in the range $[-3\sigma^a_{Tem}, 3\sigma^a_{Tem}]$ and $SOC + \theta_{SOC} - \mu^a_{SOC}$ is located at the range $[-3\sigma^a_{SOC}, 3\sigma^a_{SOC}]$, the discriminator will consider this new vehicle usage sample (d', Tem', SOC') matches the user's habits

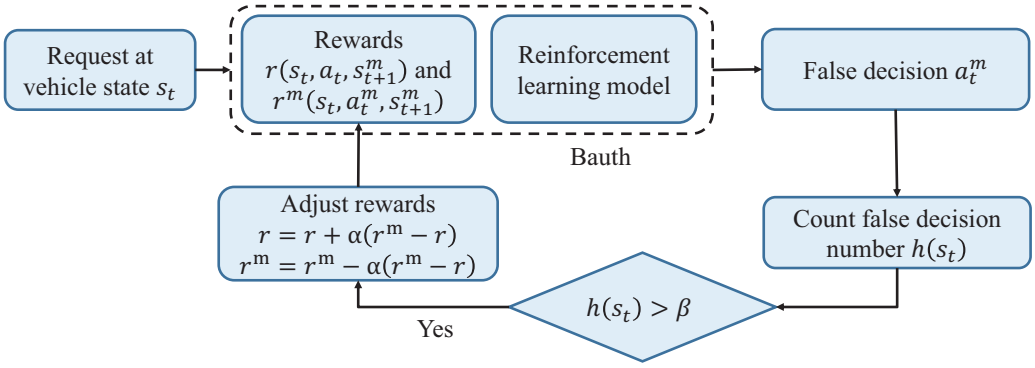


Figure 3.8: The reward function update process with our reward self-adjustment model.

and save this sample into the new vehicle usage dataset. Otherwise, the discriminator will ignore this new vehicle usage sample (d', Tem', SOC') and start the next iteration process. Bauth will keep the above iteration process with the GAN-based vehicle usage data generator to form new vehicle usage samples until the total number of new vehicle usage samples and real vehicle usage samples are large enough so that they can cover all possible action-vehicle state pairs.

3.2.2.3 Self-Adjustment in Reward Function

Since reward function in this paper is built by conducting a statistical analysis on vehicle usage data, it will cause less accurate reward function calculation result by considering that vehicle usage data may contain wrong user's action decisions because of user's operation errors, which results in false authentication (false alarm or missed detection) of Bauth. Here, we develop a reward self-adjustment method to check whether false authentications are caused by the reward function and update the reward function based on the user action on false authentication decisions. Figure 3.8 shows how our reward self-adjustment model updates the reward function. For a request at vehicle state s_t, we assume that Bauth selects false authentication decision a_t^m and vehicle state changes from s_t to s_{t+1}^m at a false authentication situa-

tion, where s_t and s_{t+1}^m represent vehicle states at time t and $t+1$, respectively. For this false situation, a_t represents its correct authentication decision. The probability that Bauth selects authentication decision a_t^m at vehicle state s_t is calculated as $\bar{D}_{s_t,a_t^m,s_{t+1}^m}$. Here we use p_{a_t} to represent $\bar{D}_{s_t,a_t^m,s_{t+1}^m}$ for simplification and $h(s_t)$ to represent total times of this false authentication at vehicle state s_t. The probability p_{a_t} of selecting correct authentication decision a_t equals to $1-p_{a_t^m}$. And then, p_{a_t} will be compared with a threshold value ε and $h(s_t)$ will be compared with a threshold value β to determine whether the reward needs to be updated. ε is used to indicate the minimum probability that Bauth selects authentication decision a_t at vehicle state s_t. β indicates total times of false authentication and can be set by users. If p_{a_t} is less than ε and $h(s_t)$ is larger than β, it means that Bauth will never select correct authentication decision and the reward of selecting correct authentication decision at this vehicle state needs to be updated to train reinforcement learning model so that reinforcement learning model can select correct authentication decision. For the new reward, p'_{a_t} and $p'_{a_t^m}$ are updated to $p_{a_t}+\alpha(p_{a_t^m}-p_{a_t})$ and $p_{a_t^m}-\alpha(p_{a_t^m}-p_{a_t})$, where α is an update coefficient and determined based on Bauth's performance in practice. The reward in Bauth keeps being updated until authentication action from Bauth conforms action in vehicle usage data. By this way, the reward self-adjustment method helps to update the reward based on the user action on false authentication decisions and eliminate negative effects of statistical analysis on reward function.

3.2.2.4 Authentication on Action Requests from Smartphone

The battery attack tries to attack the EV batteries by generating malicious action requests from a smartphone. For an action request from a smartphone, Bauth firstly obtains vehicle state and makes its authentication decision with maximum Q-value. And then, Bauth sends its authentication decision to the Electronic Control Units (ECUs) in the EV, which control vehicle actuators to execute vehicle functions includ-

ing AC-turn-on and battery-charge-stop. More details about vehicle control through ECUs can be found in [6]. Therefore, Bauth can improve EV battery security and help eliminate negative effects of battery attacks such as driving plan change and driving range anxiety. The pseudocode of Bauth is shown in Algorithm 3.

When Bauth receives an action request from a smartphone, it firstly observes vehicle state $s_t = [Tem_t, SOC_t]$ and selects authentication decision a_t based on vehicle state s_t and policy π^* (Line 2). And then, Bauth sends its authentication decision to ECUs in the EV to finish this action authentication process (Line 3). If the user takes action d_t and d_t is not the same as authentication decision a_t, total times $h(s_t^m)$ of this false authentication will increase by one. If $h(s_t^m)$ is larger than β and probability is less than ε, reward $r(s_t, a_t, s_{t+1})$ will be updated to train reinforcement learning model until new authentication decision a_t conforms the user action d_t (Lines 4-11).

Algorithm 3: Request authentication with Bauth

```
   Data: Trained reinforcement learning model
1  while Receive an action request from a smartphone do
2      Observe vehicle state s_t = [Tem_t, SOC_t];
3      Make authentication decision a_t on action request based on vehicle state a_t and policy
       π* and send it to ECUs in EV;
4      if d_t ≠ a_t then
5          if P_{a_t} ≤ ε and h(s_t^m) ≥ β then
6              Update reward r(s_t, a_t, s_{t+1}) to train reinforcement learning model and output
               new authentication decision a_t;
7              Go back to Line 4;
8          end
9      else
10         h(s_t^m) = h(s_t^m) + 1;
11     end
12 end
```

3.3 Performance Evaluation

We conducted real-world experiments based on real-life EV usage to evaluate the attack detection performance of Bauth. In the experiment, we firstly conducted EV driving experiments using total 10 participants to test identification accuracy of our

user identification method. And then, we recorded AC-turn-on and battery-charge-stop behaviors of total seven EVs and utilized probability based reward function into reinforcement learning model to learn these behaviors by training reinforcement learning model with record data. Lastly, based on the established reinforcement learning model, Bauth authorized action requests from a smartphone and we compared battery attack detection results between Bauth and a statistical method (explained in details in Section 3.3.1 and abbreviated as SM) for performance evaluation.

3.3.1 Experiment Settings

We implemented battery attacks through a smartphone and run Bauth in a laptop to authorize action requests from a smartphone. To conduct battery attacks through a smartphone, we installed one App including malicious code into one Android smartphone to remotely attack EV batteries by turning on AC or stopping battery charging process. For such an App, it can conduct commands from users successfully when working normally while it can also conduct battery attacks without being noticed by users. In the experiment malicious requests are automatically conducted using a smartphone with frequency 0.1 $time/second$ and these requests are directly sent to both EVs and Bauth. More specifically, malicious AC-turn-on requests from No-effort attack are sent out randomly without considering whether a user is in the vehicle while malicious AC-turn-on requests from Smart attack are sent out only when the user is not in the vehicle. For malicious battery-charge-stop requests, No-effort attack and Smart attack also need to ensure that the vehicle is in the battery charging process. Here, names of the App and vehicle brand in the experiment are anonymous to avoid the possibility that others may attack these vehicles with this App through our method.

In EV driving experiments, total 6 male and 4 female participants (with ages

(a) Driving experiment map.

(b) Average battery output power.

Figure 3.9: EV driving experiments and battery energy information.

Table 3.2: Vehicle usage case

EV	The number of users	Driving route type
#1	1	Home-Office
#2	1	Home-Office-Mall
#3	1	Home-Mall-Supermarket
#4	2	Home-Office-Mall-Supermarket
#5	1	Home-Mall
#6	3	Home-Office-Mall-Supermarket
#7	1	Home-Office-Supermarket

range from 20 to 35 years) drove one EV on a 7.4 *mile* long road shown in Figure 3.9(a). During the driving process, battery state of EV batteries shown in the App was recorded with the frequency of 1 *time*/*second* to evaluate the identification performance of our user identification method. The daily driving of total 7 EVs was used for EV usage experiments and EV usage experiments lasted total 25 days long. 21-day data is used to train reinforcement learning model. 4-day data including malicious requests is used for attack detection performance evaluation. More specifically, malicious requests in a 2-day data belongs to No-effort attack and malicious requests in another 2-day data belongs to Smart attack. In the experiment, we applied Bauth and SM to authorize action requests from a smartphone in the last 4-day experiments, respectively. Besides, the user would turn off AC or restart the battery charging process if the user found attacks which were not detected successfully by Bauth and SM.

In the experiment, vehicle states and AC-turn-on and battery-charge-stop behaviors were recorded through the vehicle mobile App. The vehicle indoor temperature was measured by vehicle internal temperature sensor and its real-time measurement value was sent to the smartphone through bluetooth. Battery state was measured and recorded by the vehicle mobile App. The vehicle usage cases such as the number of users sharing the same EV and driving route types in the EV usage experiments are shown in Table 3.2. Here, driving route type is determined based on locations where an EV arrives in one day. All users of these EVs charge their EVs at private homes and may have different habits in AC-turn-on and battery-charge-stop.

We experimentally evaluated Bauth and our experiments covered a broad set of vehicle usage cases (i.e., different numbers of users per vehicle and driving route types). Our results answer the following questions:

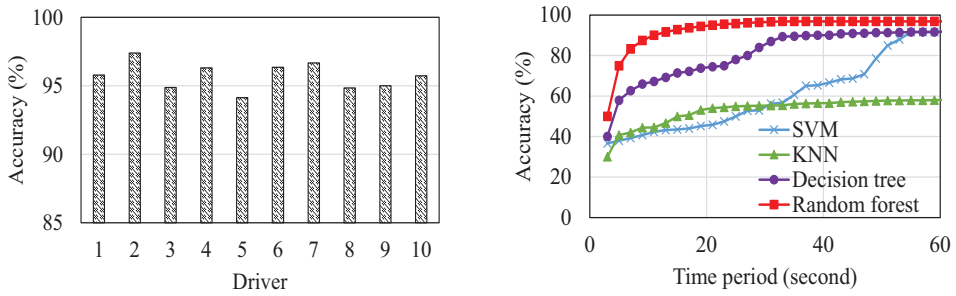
- How is Bauth's performance in identifying different users? Our experimental results demonstrate that Bauth has high average identification accuracy. Figure 3.10 shows identification accuracy comparisons among different users.
- How is Bauth's performance compared to other methods in terms of attack detection accuracy and precision? Attack detection accuracy describes conformity degree between detection results and true attack situations and is calculated as the rate of $true\,positives + true\,negatives$ over $total\,requests$. Attack detection precision describes how closely correct detected attacks agree with detected attacks and is calculated as the rate of $true\,positives$ over $true\,positives + false\,positives$. Here, we define that battery attack is detected successfully in the experiment only when user or Bauth or SM detects it in $10s$. There exist no available methods for authorizing action requests (to turn on AC or stop battery charging process) to prevent EV batteries from battery attacks as the battery attack is firstly proposed in this paper. The most direct way is to analyze vehicle usage data to obtain statistical probabilities of taking AC-turn-on and battery-charge-stop actions under different vehicle states and

utilize statistical probabilities to select decisions for action requests, which forms the statistical method (SM). For one action request at given vehicle state, SM accepts this action request if statistical probability of accepting AC-turn-on or battery-charge-stop action is higher than statistical probability of rejecting this action. Otherwise, SM rejects this action request. Figure 3.11 and Figure 3.12 show statistical probability and state transform probability results based on given vehicle usage data, respectively. We compared attack detection results between Bauth and SM in Figure 3.13.

- How is the effectiveness of GAN based vehicle usage data generator and self-adjustment algorithm to improve Bauth's attack detection accuracy? Figure 3.14 shows the effects of GAN based vehicle usage data generator and reward self-adjustment process on attack detection performance of Bauth.
- Is Bauth effective in different vehicle usage cases? We find that Bauth is able to maintain high levels of performance both in the presence of multi-user sharing one EV situations and different driving route types. Figure 3.15 shows Bauth's performances for different vehicle usage cases.
- Can Bauth efficiently avoid the effects of battery attacks on battery energy loss? Experimental results demonstrate that Bauth helps to avoid battery energy loss and Figure 3.16 shows battery energy loss results in the experiment.
- Does Bauth have a computation overload problem when authorizing action requests? Experimental results demonstrate that Bauth has low computation time and Figure 3.17 shows computation time of Bauth on different action requests.

3.3.2 User Identification Evaluation

To verify the user identification method, we did EV driving experiments on the road shown in Figure 3.9(a) and total 10 participants drove the same EV from *Startpoint* to *Endpoint*. During the driving process, the App in the smartphone recorded real-time

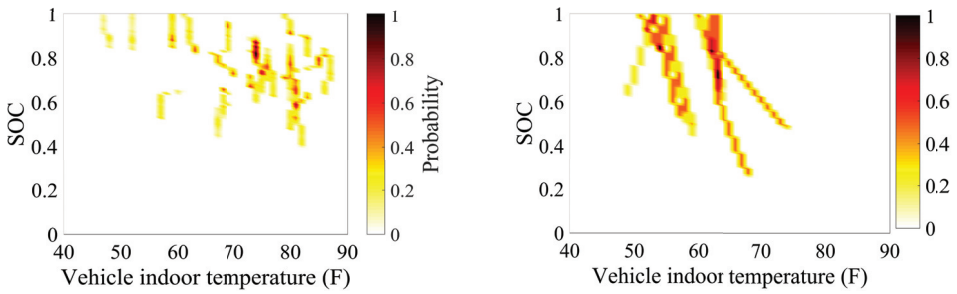


(a) Identification accuracies. (b) Different identification methods.

Figure 3.10: User identification accuracy performance evaluation.

EV battery states including battery current, battery power output and battery SOC. For the battery power output information, its value is positive when the vehicle accelerates on the road while it becomes negative when the vehicle brakes. Figure 3.9(b) shows average absolute battery output values during driving and braking processes for 10 participants. We see that both values are different among these participants and reflect participant's driving habits (acceleration and brake levels), which explains why these battery states can be used to identify drivers.

The user identification method in Section 3.2.1.1 uses random forest technology to identify the driver based on the recorded battery states. We utilize 4-fold cross-validation method [9] to evaluate user identification performance and identification results among participants are compared in Figure 3.10(a). The average value of identification accuracies among these drivers reaches 95.7%. Besides, we compare identification performances among random forest and existing identification technologies including support vector machines (SVM), Decision Tree, and k-nearest neighbors (KNN) with different time periods Δ and show them in Figure 3.10(b). We see that random forest has higher identification accuracy compared with other existing technologies as time period Δ increases from 1 *second* to 60 *seconds* with a 2 *seconds* increasing size. More specifically, the identification accuracy for the random forest keeps almost constant when time period Δ equals to 20*seconds*, which is less than



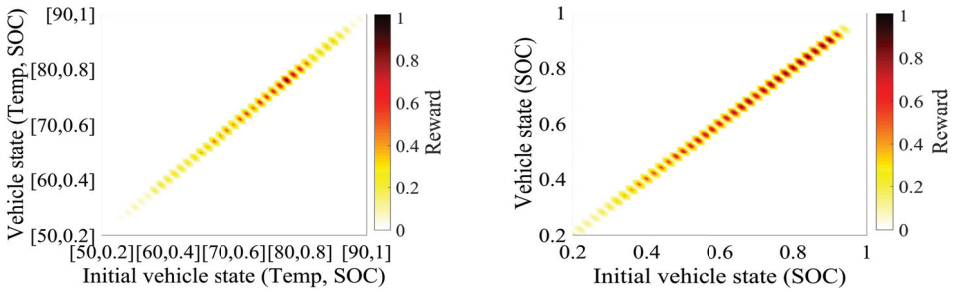
(a) AC-turn-on probability. (b) Battery-charge-stop probability.

Figure 3.11: Statistical probabilities of AC-turn-on and battery-charge-stop.

corresponding time period values of other technologies. It means that random forest can use less data to identify drivers with the same identification accuracy compared with other existing technologies.

3.3.3 Authentication Performance Evaluation

To evaluate attack detection performance of Bauth, we apply Bauth and SM to authorize action requests from a smartphone, respectively. SM is formed by statistically analyzing situations where the driver turns on AC or charges EV batteries based on the data and Figure 3.11 shows statistical probabilities of turning on AC and stoping charging batteries under different vehicle states for #4 EV. Color bars in the right part of figures show the mapping from statistical probabilities to color degrees. Here we choose #4 EV for attack detection performance evaluation mainly because this EV is shared by two drivers and its driving route type contains Home, Supermarket, Office and Mall, which makes #4 EV better represent different vehicle usage cases. We see that drivers would prefer turning on AC when vehicle indoor temperature is higher than $55F$ and SOC is more than 0.5. For the battery charging process, the driver would often charge EV batteries when battery SOC is located at 0.3-1. This is reasonable because the driver always charges batteries after EV parks at home or near the office.



(a) Reward of AC-turn-on.

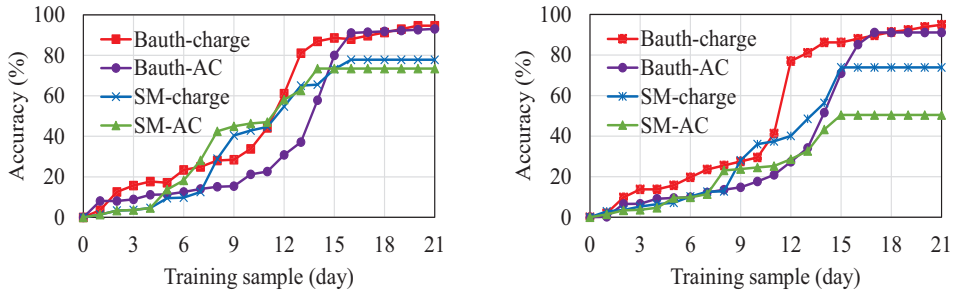
(b) Reward of battery-charge-stop.

Figure 3.12: The reward of AC-turn-on and battery-charge-stop for different vehicle states (vehicle indoor temperature (F) and battery SOC).

To implement Bauth, we firstly need to calculate the normalized state transform probability matrices $\bar{\mathcal{D}}$s based on the training data and apply these matrices as rewards to train networks in reinforcement learning. The matrices $\bar{\mathcal{D}}$s of turning on AC and stoping charging batteries are shown in Figure 3.12. X-axle represents initial vehicle state and Y-axle represents the vehicle state after taking action (turning on AC or stop charging batteries). We see that the reward reaches the maximum value only when turning on AC happens at around vehicle state $(80, 0.8)$. For battery-charge-stop action, the reward reaches the maximum value when battery SOC is large than 0.8. This is mainly because the driver decides to stop charging batteries only when battery SOC is large enough to finish an user's driving plan.

3.3.3.1 Performance for different attacks

We firstly trained Bauth and SM based on rewards shown in Figure 3.12 and then used them to authorize action requests. Specifically, we used different numbers of training samples in the first 21-day data(changing from 1 day to 21 days) for training and used the last 4-day data to evaluate the effects of training sample numbers on their attack detection accuracies. Figure 3.13 shows attack detection accuracies on No-effort attack and Smart attack for Bauth and SM under different numbers of

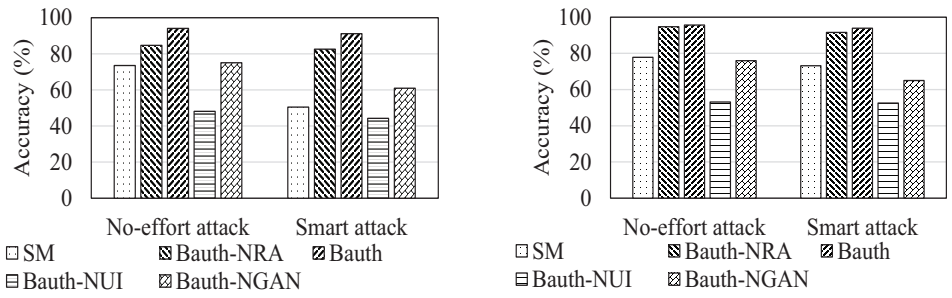


(a) Accuracy for No-effort Attacks. (b) Accuracy for Smart Attacks.

Figure 3.13: Attack detection accuracy comparisons for No-effort attacks and Smart attacks.

samples (changes from 1 day to 21 days). For both No-effort attack and Smart attack, Bauth has higher attack detection accuracies on battery-charge-stop and AC-turn-on requests compared with SM. More specifically, attack detection accuracy of Bauth is higher than the value of SM when the total training sample is more than 15 days. Besides, Bauth can still keep increasing its attack detection accuracy even while SM has almost constant authentication accuracy as the training data is more 15 days, which demonstrates good balance between exploitation and exploration for reinforcement learning.

Here, we trained Bauth and SM with total 21-day samples and used their attack detection results as examples to better understand attack detection performance of Bauth on different battery attacks. Figure 3.14 shows attack detection accuracies on No-effort attack and Smart attack for Bauth and SM. For SM, the maximum attack detection accuracy on Smart attack and No-effort attack is less than 80% for two different kinds of action requests (turning on AC and stop charging batteries) As the number of training samples increases. SM has lower attack detection accuracy on AC-turn-on requests because the driver decides to turn on AC or not based on both the remaining battery energy and her/his individual preference on comfortable temperatures which will not be considered during the battery charging process. More



(a) AC-turn-on request.

(b) Battery-charge-stop request.

Figure 3.14: Attack detection accuracy comparisons on AC-turn-on and battery-charge-stop requests.

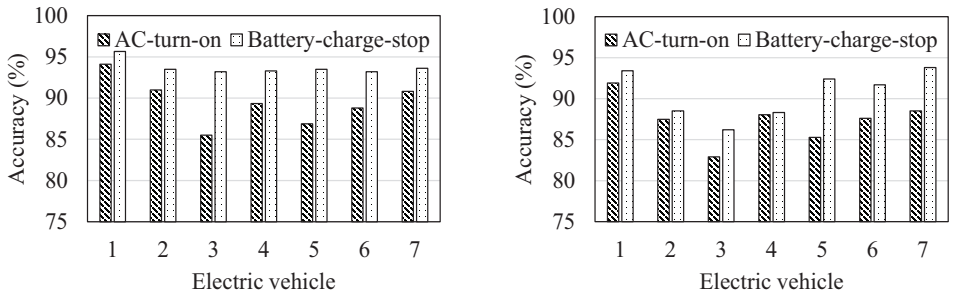
specifically, SM has lower attack detection accuracies on Smart attack compared with No-effort attack, especially for AC-turn-on requests. This is because Smart attack happens only when the driver is not in the vehicle and the driver may turn on AC remotely to ensure vehicle indoor temperature is at the comfortable level, which makes it difficult to detect Smart attack with SM.

For Bauth, it has higher attack detection accuracy than SM and its maximum attack detection accuracy reaches 95.6%, which demonstrates that Bauth has good attack detection performance on both AC-turn-on and battery-charge-stop action requests. Besides, Figure 3.14 shows the effects of user identification (UI), GAN-based vehicle usage data generator (GAN), and reward-adjustment (RA) on Bauth's attack detection performance. For Bauth without UI (Bauth-NUI), 21-day samples of all users are used to train reinforcement learning model without identifying users. We see that Bauth-NUI has lower attack detection accuracies when authorizing action requests. This is because different drivers may share the same vehicle randomly and usually have different habits on AC-turn-on and battery-charge-stop, which makes it hard to accurately learn habits of each driver for an reinforcement learning model. Bauth without RA (Bauth-NRA) has similar training and testing processes compared with Bauth and the only difference between Bauth-NRA and Bauth is that Bauth-

NRA does not adjust rewards when authentication decision is not correct. In the experiment, a user's action was recorded and sent to Bauth to adjust rewards if it was not the same as the authentication decision. Based on Figure 3.14, we see that attack detection accuracies on No-effort attack and Smart attack of Bauth are larger than values of Bauth-NRA because of conducting RA processes. For Bauth without GAN (Bauth-NGAN), 21-day samples of all users are directly used to build reward functions without generating new vehicle usage data through the GAN-based vehicle usage data generator. We see that Bauth-NGAN has much lower attack detection accuracies on both AC-turn-on and battery-charge-stop requests than Bauth, which demonstrates that the GAN-based vehicle usage data generator helps to improve Bauth's attack detection performance.

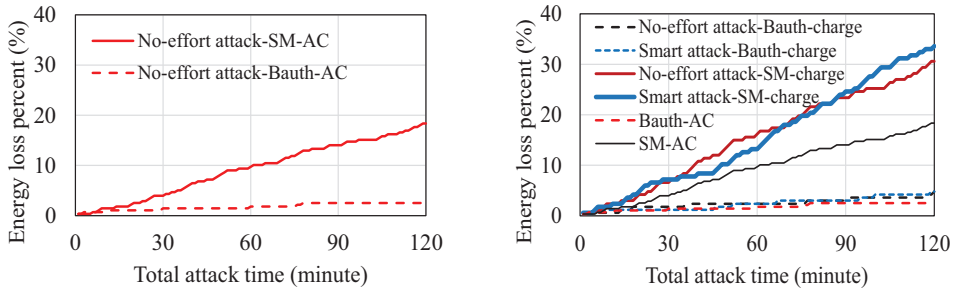
3.3.3.2 Performance for different vehicle usage situations

To evaluate attack detection performance of Bauth on different vehicle usage situations, we implemented Bauth on different vehicle usage situations. Vehicle usage situations shown in Table 3.2 are different because of numbers of sharing drivers and driving route types. Figure 3.15 shows attack detection accuracy results on No-effort attack and Smart attack at different vehicle usage situations. The average attack detection accuracies for No-effort attack and Smart attack are 91.6% and 88.9% respectively, which demonstrates good attack detection performance of Bauth. Bauth has the lowest attack detection accuracy at the usage situation (#3 EV) because #3 EV (vehicle usage frequency is around 10 $times/week$) was not used as frequency as other vehicles and provided limited data to train Bauth. Besides, Bauth has higher attack detection accuracy on battery-charge-stop requests compared with AC-turn-on requests. This is because the driver stop charging EV batteries by considering only battery SOC while the driver turns on AC or not by considering not only vehicle indoor temperature but also battery SOC, which are more difficult to learn than



(a) Accuracy for No-effort Attacks. (b) Accuracy for Smart Attacks.

Figure 3.15: Bauth's attack detection accuracy performance for different vehicle usage cases.



(a) Battery not-charging situation. (b) Battery charging situation.

Figure 3.16: The percent of battery energy consumed by battery attacks in two-hour experiments.

battery-charge-stop actions in practice.

3.3.3.3 Battery energy consumption with authentication

To evaluate the effects of Bauth on avoiding battery energy loss caused by battery attack in practice, we did vehicle usage experiments on #1 EV and implemented Smart attacks and No-effort attacks, respectively. Here we choose #1 EV because this EV only drives from Home to Office and its user is not in the vehicle in most cases, which provides many chances of conducting Smart attack in the experiment. Figure 3.16 shows how total battery energy loss percentage changes as total attack time increases at battery not-charging situations and battery charging situations. For the battery not-charging situation, only No-effort attack sent malicious AC-turn-on

requests in the experiment since the vehicle was not in the battery charging process and a user was driving the vehicle during the whole battery not-charging situation. Figure 3.16(a) compares total battery energy loss percentages at the battery not-charging situation for Bauth and SM. Here, total battery energy loss percentage is calculated as the rate of total battery SOC loss over initial battery SOC. Based on Figure 3.16(a), total battery energy loss percentage for Bauth is around 4% while the value of SM reaches as much as 20% if total attack time lasts two hours. This is because Bauth has higher attack detection accuracy compared with SM. Figure 3.16(b) shows results at the battery charging situation. For the battery charging situation, No-effort attack sent malicious requests randomly while Smart attack would happen only when the user was not in the vehicle. Though stopping battery charging processes will not consume battery energy, battery SOC will not increase as the user expects. The difference between real battery SOC change and expected battery SOC change is caused by the battery attacks and is considered as battery energy loss. The expected battery SOC change is calculated as the difference between final battery SOC and the battery SOC when attack happens. We see that EV with Bauth has much less battery energy loss compared with values of SM. More specifically, the battery energy loss after two hours reaches around 30% for SM. This is to say, driving range of EV will be reduced from 310 *mile* into 210 *mile* if EV suffers 30% energy energy loss. Based on Figure 3.16(b) we see that Smart attack can result in more energy loss compared with No-effort attack. This is because detection accuracy on Smart-attack is lower than value on No-effort attack. This section demonstrates the effectiveness of Bauth on avoiding battery energy loss caused by malicious action requests and the importance of researching on EV energy security.

3.3.3.4 Computation complexity analysis of Bauth

We implemented Bauth on the laptop which has Intel i5 CPU and 4 gigabyte memory. When Bauth receives an action request (AC-turn-on or battery-charge-stop) from a smartphone, Bauth firstly observes vehicle states to identify the user and then chooses an authentication action based on its policy. Computation time per each action request authentication process is recorded to analyze computation complexity of Bauth. Figure 3.17 shows average computation time of Bauth on different types of action requests. We see that average computation time on different action requests equal to 0.72 *seconds* and the maximum value of these average computation time for Bauth is 0.79 *seconds*, which is less than both time period (1 *second*) of measuring battery states and time period (10 *seconds*) of sending action requests. Though both our user identification model and reinforcement learning model in Bauth need a training process and such a training process usually takes a long time to obtain good performance, Bauth applies the well-trained user identification model and reinforcement learning model to make action request authentication processes and does not need much time to authorize an action request.

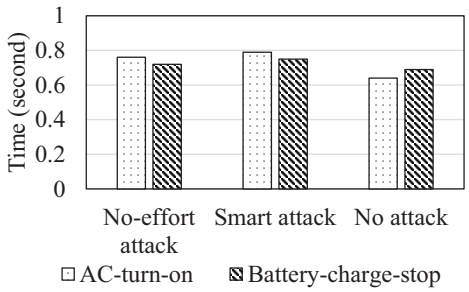


Figure 3.17: Computation time of Bauth on different action request types.

Chapter 4

Control Policy based Driving Safety System

4.1 System Design

We propose Polsa to extract control policies of a target AV and determine optimal control policies under different driving scenarios for the target AV to improve its driving safety. In this paper, we follow safety criteria in References [69, 77] and use *Stop distance* and *Actual distance* to test whether a control policy is optimal under a driving scenario. Figure 5.4 shows *Stop distance* and *Actual distance* in both longitudinal and latitudinal directions. Specifically, *Stop distance* d_{stop} indicates the maximum distance a target AV will travel when it decelerates to a completed stop with the maximum acceptable deceleration. *Actual distance* d_{actual} is defined as the distance a target AV can travel without having a possible collision with its nearby vehicle. We introduce potential distance d_{pot} to check whether there exists a possible collision and calculate d_{pot} as $d_{actual} - d_{stop}$. If d_{pot} is negative in either longitudinal or latitudinal direction, we conclude that this target AV will have a collision with its nearby vehicle. Therefore, an optimal control policy should ensure that d_{pot} is always

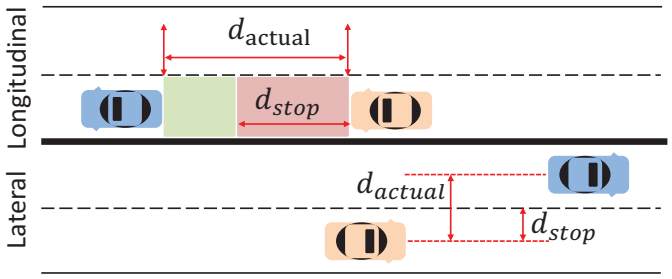


Figure 4.1: Stop distance and actual distance in longitudinal and latitudinal directions.

positive when executed on a target AV.

We present the architecture of Polsa in Figure 4.2. Polsa consists of two major parts: control policy extraction and optimal control policy determination. The control policy extraction part firstly analyzes historical driving data of a target AV to cluster driving events with the same control behavior type together and then extracts control policies based on driving event cluster results. Here, historical driving data includes states (position, driving speed) of both a target AV and its nearby vehicle. Since historical driving data of a target AV collected from driving simulations or road experiments can be saved in its data center, Polsa can easily access historical driving data to extract control policies of a target AV. In the optimal control policy determination part, Polsa firstly predicts driving state of a nearby vehicle in a driving scenario and then applies its driving state prediction result to determine which control policy can ensure driving safety of a target AV in a driving scenario. Lastly, Polsa will send its optimal control policy determination results to AVs as references to improve their driving safety.

4.1.1 Control Policy Extraction

When a target AV drives on the road, its sensors (e.g., cameras and Lidar) measure the state of its nearby vehicle and send measurement information to the target AV in

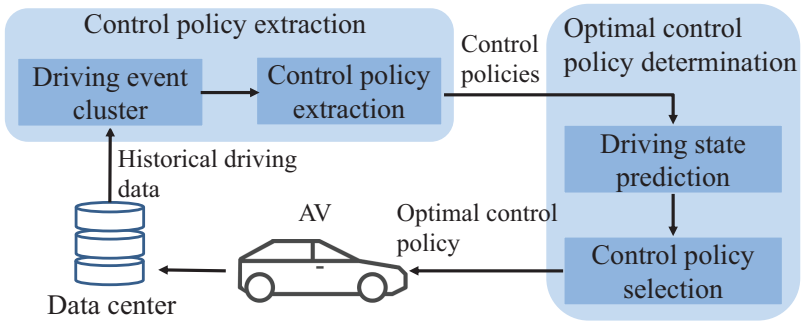


Figure 4.2: The architecture of Polsa for driving safety analysis on a target AV.

real time. Based on state of its nearby vehicle, a target AV will determine its control behaviors in longitudinal direction (Acceleration, Deceleration and Constant speed) and latitudinal direction (Left-lane-change, Right-lane-change and Intersection-turn) based on its control policies. Based on accident reports of self-driving cars from California DMV, we find that collisions between a target AV and its nearby vehicle dominates all different kinds of accidents in accident reports. Therefore, we only focus on driving scenarios including a target AV and its nearby vehicle in this paper and discuss how to improve driving safety of a target AV when its control policies are implemented under these driving scenarios. Here, the nearest vehicle of a target AV is defined as a vehicle which is the closest to the target AV and their position difference should be no more than $100m$ by considering sensor measurement ability [53]. Here, we use *nearby vehicle* to indicate the nearest vehicle of a target AV and a nearby vehicle can be autonomous driving or human-driving type.

By considering that a target AV measures its driving environments through on-board sensors and these sensors can only measure vehicle features like position and driving speed, we describe a control policy π of a target AV with a trigger condition x and a control behavior A. Specifically, trigger condition x represents position and driving speed related information of the target AV and its nearby vehicle (e.g., absolute speed difference between the target AV and its nearby vehicle is more than $20km/h$ or their absolute position difference is less than $5m$) and time duration of

executing a control policy π depends on its trigger condition x and states of the target AV and its nearby vehicle. Control behavior A represents control behavior of a target AV and changes in $\{$*Acceleration*, *Deceleration*, *Constant-speed*, *Left-lane-change*, *Right-lane-change*, *Intersection-turn*$\}$.

For historical driving data of a target AV, it usually contains many different control behaviors and these control behaviors result in different driving events. Here, a driving event of a target AV represents a process where the target AV executes one of its control policies and includes states of the target AV and its nearby vehicle during a control policy execution process. Specifically, start-time and end-time of a driving event indicate specific time when a target AV starts and stops executing a control policy. Therefore, Polsa firstly needs to find driving events in the historical driving data and then clusters these driving events based on states of a target AV and its nearby vehicle to obtain different driving event clusters. Lastly, Polsa analyzes driving data of each driving event cluster and extracts trigger information and the corresponding control behavior to form a control policy.

4.1.1.1 Driving Event Clustering in Historical Driving Data

As mentioned above, the historical driving data of a target AV usually includes different types of driving events. For example, when a target AV drives behind its nearby vehicle on the same road lane, it will need to choose a deceleration or lane-change action to avoid a possible collision when its nearby vehicle suddenly decelerates. Above control behaviors (deceleration or lane-change) form different driving events. In practice, states of a target AV and its nearby vehicle in different driving events usually have different changes, which results in different time durations. For example, a target AV takes around $5seconds$ to reduce its driving speed from $60km/h$ to $30km/h$ during a deceleration process but takes only $3seconds$ to decelerate from $60km/h$ to $30km/h$ during a lane change process. Therefore, it is difficult for Polsa to extract

control policies of a target AV by directly analyzing state information in its historical driving data. Here Polsa does the following steps on the historical driving data to obtain driving event clusters and then analyzes driving data of each driving event cluster to extract control policies:

- *Step 1:* Polsa calculates position difference and speed difference of a target AV to determine control behavior type and time duration of each driving event in historical driving data.
- *Step 2:* Polsa analyzes position trajectory difference and speed trajectory difference between a target AV and its nearby vehicle to extract driving event features.
- *Step 3:* Polsa clusters driving events with the same control behavior based on driving event features to form different driving event clusters for control policy extraction.

Driving Event Determination For a historical driving dataset of a target AV, it includes states of both a target AV and its nearby vehicle. Specifically, each state contains both position information (latitude and longitude) and driving speed information. Therefore, given a driving scenario Y with total T time stamps, its driving data can be mathematically indicated with $\{y_1, ..., y_t, ..., y_T\}$, where $y_t = [p_t, v_t, p'_t, v'_t]$ represents states of a target AV and its nearby vehicle at time $t \in \{1, ..., T\}$. $p_t \in \Re^2$ and $p'_t \in \Re^2$ indicate position information (longitudes and latitudes) of a target AV and its nearby vehicle. Similarly, $v_t \in \Re$ and $v'_t \in \Re$ indicate driving speeds of a target AV and its nearby vehicle at time t.

As mentioned earlier, a driving event shows a time period when a control policy is executed and can be treated as fundamental building parts of a driving scenario (contains one or more driving events). In other words, the historical driving data of a driving scenario Y can be divided into a number of driving events $\bar{Y}$s and these driving events do not have temporal overlaps with each other in the same driving scenario. Therefore, a driving event can be described as $\bar{Y} = \{y_m, ..., y_n\}$, where

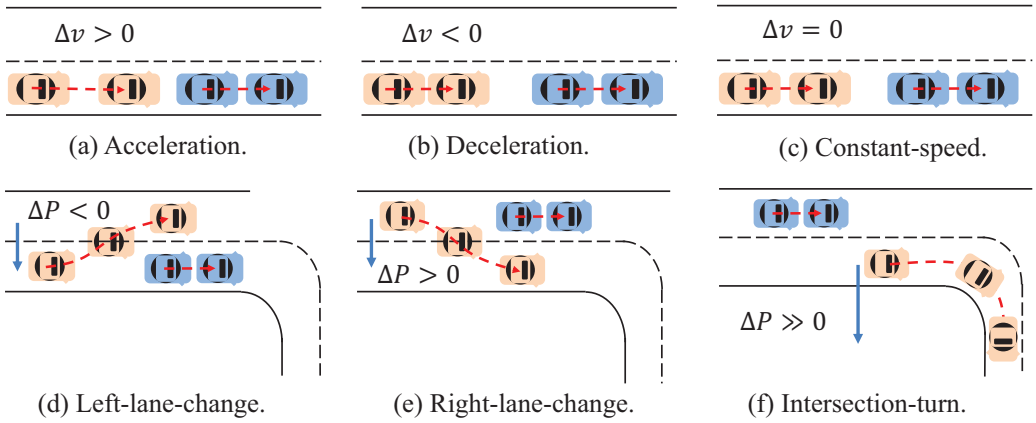


Figure 4.3: Control behavior types of a target AV when one of its control policies is executed.

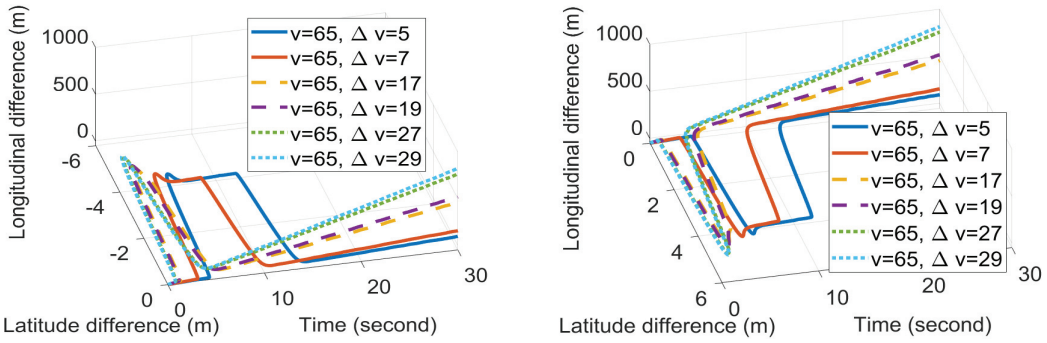
$\bar{Y} \subseteq Y$ and $1 \leq m \leq n \leq T$ and its time length equals to $n - m + 1$. Since Polsa has limited prior knowledge on driving event types, it needs to segment a historical driving dataset into small pieces and analyze the target AV's control behavior at each piece of the historical driving data to extract driving event features.

Based on historical driving data of a target AV, Polsa analyzes possible control behaviors of a target AV and uses them to segment the historical driving data into different driving events. Here, a control behavior of a target AV represents its driving actions when one of its control policies is executed. By analyzing position and speed information, Polsa divides control behaviors of a target AV into the following types: *Acceleration*, *Deceleration*, *Constant-speed*, *Left-lane-change*, *Right-lane-change* and *Intersection-turn*. Figure 4.3 shows these different control behavior types. Red vehicle and blue vehicle represent a target AV and a nearby vehicle, respectively. We see that Polsa can generally determine specific control behavior types of a target AV by checking its speed change Δv and its latitude difference ΔP in one short time period. Specifically, a control behavior will be classified as *Acceleration*, *Deceleration* and *Constant-speed* if Δv equals to a positive value, a negative value and zero, respectively. Besides, Polsa infers whether a target AV has a *Left-lane-change*, *Right-lane-change*

or *Intersection-turn* based on ΔP. As shown in Figure 4.3(d), Figure 4.3(e) and Figure 4.3(f), ΔP will become positive and reach around the width of one road lane when a target AV makes a right lane change. Similarly, ΔP will become negative and reach around the width of one road lane when the target AV makes a left lane change. Besides, ΔP will become much larger than the width of one road lane if the target AV makes an intersection turn. By this way, Polsa can generally divide historical driving data of a target AV into different driving events by checking Δv and ΔP.

Since Polsa divides historical driving data of a target AV into driving events based on the above six control behavior types, each control behavior type may still result in different driving events for a target AV. For example, the target AV in Figure 4.3(d) makes a right lane change maybe because its nearby vehicle makes a sudden deceleration and the target AV has to make a left lane change to avoid a possible collision with its nearby vehicle or maybe because its nearby vehicle drives too slow and the target AV wants to pass through its nearby vehicle. Therefore, a control behavior of a target AV may result in many different driving events in the historical driving dataset and it is necessary for Polsa to cluster these driving events. Here, we introduce driving event features by considering both temporal and spatial spaces in one driving event to ensure that Polsa can cluster driving events accurately for control policy extraction in Section 4.1.1.2.

Driving Event Features As shown in Figure 4.3, position and driving speed information of a target AV usually change with a certain regularity when a target AV conducts the same control behavior in different driving events. For example, a target AV will always have a large speed change when it conducts a heavy deceleration or will have a large latitude difference if it makes a fast lane change. Besides, a target AV needs to consider movements of its nearby vehicle when determining its control behavior to ensure driving safety. For example, a target AV will decelerate or make



(a) *Left-lane-change* situation.

(b) *Right-lane-change* situation.

Figure 4.4: Position (latitude and longitude) trajectory differences between a target AV and its nearby vehicle when speed difference Δv changes from $5km/h$ to $29km/h$ during one left/right lane change process.

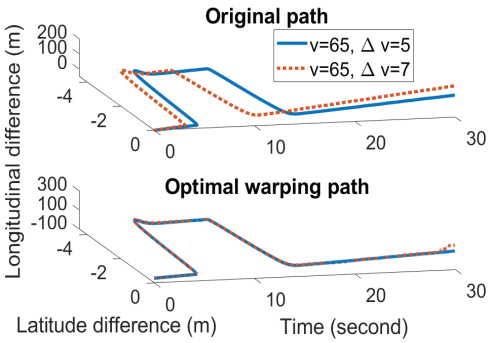
a lane change if its nearby vehicle decelerates suddenly. Based on the above analysis, we take advantage of position and driving speed information of a target AV and its nearby vehicle and use their position trajectory difference and speed trajectory difference as driving event features to cluster these driving events. By this way, Polsa can cluster driving events with the same control behavior type based on these driving event features. Given historical driving data $\{y_m, ..., y_n\}$ of a target AV and its nearby vehicle in a driving event $\bar{Y}$, Polsa calculates their position trajectory difference and speed trajectory difference as $\{\Delta p_m, ..., \Delta p_n\}$ and $\{\Delta v_m, ..., \Delta v_n\}$, where Δp_m equals to $p_m - p'_m$ at time m and Δv_m equals to $v_n - v'_n$ at time n.

Here, we take *Left-lane-change* and *right-lane-change* situations where a target AV and its nearby vehicle drive with different constant speeds as examples to show how their position trajectory difference changes under different driving events. Figure 4.4 shows position trajectory differences (include latitude trajectory difference and longitude trajectory difference) between a target AV and its nearby vehicle when the target AV drives with $65km/h$ and its nearby vehicle drives with $60km/h$, $58km/h$, $48km/h$, $46km/h$, $38km/h$ and $36km/h$, respectively. Their corresponding speed differences equal to $5km/h$, $7km/h$, $17km/h$, $19km/h$, $27km/h$ and $29km/h$. We see

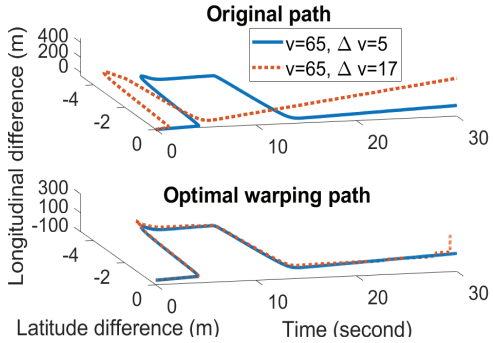
that latitude difference between the target AV and its nearby vehicle becomes negative during a left lane change process and positive during a left lane change process. Specifically, for any two driving events, their position trajectory differences change more similarly as their speed differences are close to each other. For example, position trajectory difference with $\Delta v = 5km/h$ changes more similarly to position trajectory difference with $\Delta v = 7km/h$ than position trajectory difference with $\Delta v = 17km/h$. Based on the above analysis, we conclude that position trajectory differences or speed trajectory differences are not the same for different driving events, which explains why we select position difference Δp and speed difference Δv as driving event features for Polsa.

Driving Event Clustering For given historical driving data of a target AV, it includes many different driving events and these driving events may have different data lengths. For example, a target AV usually takes longer time to finish a deceleration process if its deceleration value is small. In other words, driving events even with the same control behavior type may have different data lengths. Therefore, Polsa may fail to cluster driving events if it directly compares their position trajectory differences and speed trajectory differences. Here, we uses the dynamic time warping (DTW) method to align position trajectory differences and speed trajectory differences of two driving events $\bar{Y}$ and $\bar{Y}'$ by considering that the DTW method belongs to a pattern matching method and has good performance in clustering time series sequences with different sequence lengths [29]. For driving events $\bar{Y}$ and $\bar{Y}'$, the DTW method in Polsa firstly calculates their position trajectory matrix D_p and speed trajectory matrix D_v and then utilizes D_p and D_v to calculate event dissimilarity degree between $\bar{Y}$ and $\bar{Y}'$ for driving event clustering.

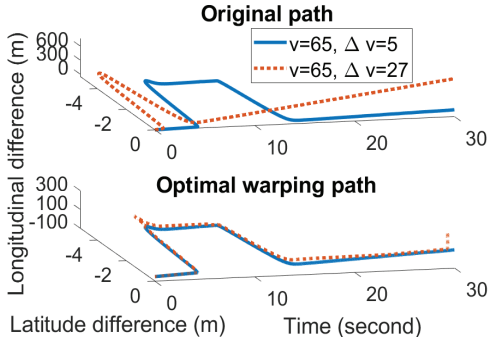
Given driving event $\bar{Y}$ with position differences $\{\Delta p_m, ..., \Delta p_n\}$ and speed differences $\{\Delta v_m, ..., \Delta v_n\}$ and driving event $\bar{Y}'$ with position differences $\{\Delta p_{m'}, ..., \Delta p_{n'}\}$



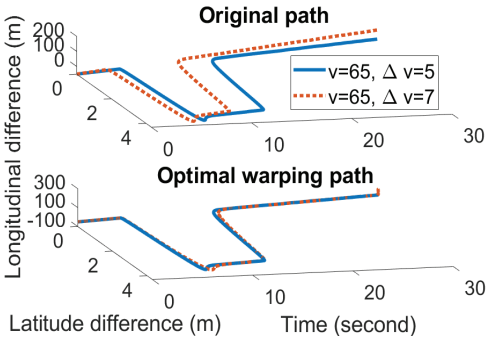
(a) $\Delta v = 7$ for *Left-lane-change*.



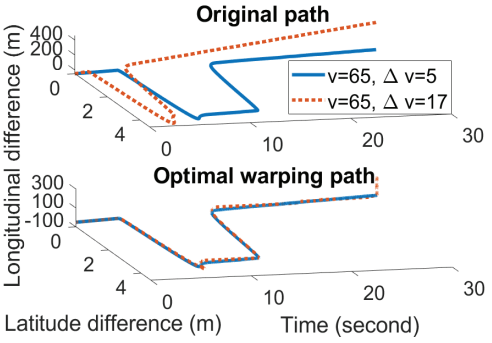
(b) $\Delta v = 17$ for *Left-lane-change*.



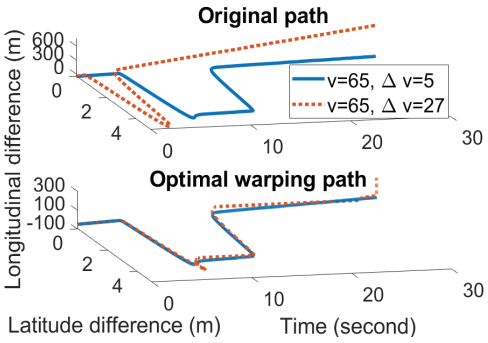
(c) $\Delta v = 27$ for *Left-lane-change*.



(d) $\Delta v = 7$ for *Right-lane-change*.



(e) $\Delta v = 17$ for *Right-lane-change*.



(f) $\Delta v = 27$ for *Right-lane-change*.

Figure 4.5: Optimal warping paths of position differences in two driving events when both a target AV and its nearby vehicle drive with constant speeds but different speed differences Δ during a left/right lane change process.

and speed differences $\{\Delta v_{m'}, ..., \Delta v_{n'}\}$, Polsa calculates position trajectory matrix $D_p \in \Re^{(n-m+1)\times(n'-m'+1)}$ and speed trajectory matrix $D_v \in \Re^{(n-m+1)\times(n'-m'+1)}$. Here, position trajectory matrix D_p is defined to represent point-to-point relationships between position differences $\{\Delta p_m, ..., \Delta p_n\}$ and position differences $\{\Delta p_{m'}, ..., \Delta p_{n'}\}$, where element $D_p(i, j)$ indicates alignment distance between position differences $\{\Delta p_m, ..., \Delta p_{m+i}\}$ and position differences $\{\Delta p_{m'}, ..., \Delta p_{m'+j}\}$. Similarly, speed trajectory matrix M_v represents point-to-point relationships between speed differences $\{v_m, ..., v_n\}$ and speed differences $\{\Delta v_{m'}, ..., \Delta v_{n'}\}$, where element $D_v(i, j)$ indicates alignment distance between position differences $\{\Delta v_m, ..., \Delta v_{m+i}\}$ and position difference $\{\Delta v_{m'}, ..., \Delta v_{m'+j}\}$.

In Polsa, the DTW method calculates position trajectory matrix D_p or speed trajectory matrix D_v as a cumulative error matrix of position differences or speed differences. An element in a cumulative error matrix represents total alignment distance indicating the sum of errors along the optimal warping path. Figure 4.5 shows optimal warping paths when position differences with different speed differences are aligned with position differences with $\Delta = 5km/h$. For *Left-lane-change* cases in Figure 4.5(a), Figure 4.5(b) and Figure 4.5(c), total alignment distances for $\Delta = 7km/h$, $\Delta = 17km/h$ and $\Delta = 27km/h$ situations equal to 5.1e+2, 3.5e+4 and 1.7e+5, respectively. For *Right-lane-change* cases in Figure 4.5(d), Figure 4.5(e) and Figure 4.5(f), total alignment distances for $\Delta = 7km/h$,$\Delta = 17km/h$ and $\Delta = 27km/h$ situations equal to 4.5e+2, 3.6e+4 and 1.7e+5, respectively. We calculate optimal warping paths when position differences with different speed differences ($\Delta = 7km/h$, $\Delta = 17km/h$ and $\Delta = 27km/h$) are aligned with position differences with $\Delta = 5km/h$. For *Left-lane-change* cases, total alignment distances for $\Delta = 7km/h$, $\Delta = 17km/h$ and $\Delta = 27km/h$ situations equal to 5.1e+2, 3.5e+4 and 1.7e+5, respectively. For *Right-lane-change* cases, total alignment distances for

$\Delta = 7km/h$,$\Delta = 17km/h$ and $\Delta = 27km/h$ situations equal to 4.5e+2, 3.6e+4 and 1.7e+5, respectively. We see that, for both *Left-lane-change* and *Right-lane-change* situations, total alignment distance will become larger as speed difference Δv increases from 7 km/h to 27 km/h. Here, Polsa clusters driving events by calculating total alignment distances in position differences and speed differences. Specifically, the DTW method calculates element $D_p(i,j)$ in the position trajectory matrix D_p as $\min(D_p(i,j-1), D_p(i-1,j), D_p(i-1,j-1))+r_p(i,j)$ and element $D_v(i,j)$ in the speed trajectory matrix D_v as $\min(D_v(i,j-1), D_v(i-1,j), D_v(i-1,j-1))+r_v(i,j)$, where $r_p(i,j)$ represents alignment distance between the i^{th} position difference Δp_{m+i} in driving event $\bar{Y}$ and the j^{th} position difference $\Delta p_{m'+j}$ in driving event $\bar{Y}'$. Similarly, $r_v(i,j)$ represents alignment distance between the i^{th} speed difference Δv_{m+i} in driving event $\bar{Y}$ and the j^{th} speed difference $\Delta v_{m'+j}$ in driving event $\bar{Y}'$. Compared with speed differences in driving events $\bar{Y}$ and $\bar{Y}'$, each position difference includes both longitude position difference and latitude position difference and hence belongs to a two dimensional data. Here, Polsa calculates $r_p(i,j)$ and $r_v(i,j)$ through Euclidean distance and Manhattan distance as follows:

$$r_p(i,j) = ||\Delta p_{m+i} - \Delta p_{m'+j}||_2 \tag{4.1a}$$

$$r_v(i,j) = |\Delta v_{m+i} - \Delta v_{m'+j}| \tag{4.1b}$$

where Δp_{m+i} and Δv_{m+i} represent the i^{th} position difference and the i^{th} speed difference in the driving event $\bar{Y}$. Similarly, $\Delta p_{m'+j}$ and $\Delta v_{m'+j}$ represent the j^{th} position difference and the j^{th} speed difference in the driving event $\bar{Y}'$. Based on position trajectory matrix D_p and speed trajectory matrix D_v, Polsa can calculate driving event dissimilarity degree $M_{\bar{Y}\to\bar{Y}'}$ between driving events $\bar{Y}$ and $\bar{Y}'$ as $\frac{D_p(n-m,n'-m')}{max(n-m,n'-m')} + \frac{D_v(n-m,n'-m')}{max(n-m,n'-m')}$ and uses the driving event dissimilarity degree as an interpretable representation to cluster driving events. For example, for driving events

$\bar{Y}$ and $\bar{Y}'$, smaller driving event dissimilarity degree indicates both their position differences and their speed differences change more similarly.

Based on driving event dissimilarity degree calculation results, Polsa randomly picks up the k-means clustering method from existing clustering methods to cluster driving events in the historical driving data of a target AV by considering its good classification performance in unlabelled data sets and fast converge speed during the training process [78]. Given a group of total N driving events $\{\Phi_1, ..., \Phi_N\}$ with the same control behavior type, Polsa partitions these driving events into k clusters with optimal cluster centers $\{C_1, ..., C_k\}$ and the j^{th} cluster includes one or more driving events $\{\phi_1^{C_j}, ..., \phi_{N_{C_j}}^{C_j}\}$, where N_{C_j} represents the number of driving events in cluster C_j. For the k-means clustering method, the optimal driving event cluster results should minimize the total sum of dissimilarity degrees between a driving event and the center in its corresponding each driving event cluster for all N driving events. However, the k-means clustering method is prone to produce inconsistent and sub-optimal cluster solutions (hence sub-optimal k) because of large solution spaces and needs to consider how to initialize centers of these k driving event clusters during its classification process.

To generate global optimal cluster solutions, we combine a greedy algorithm and the k-means clustering method together to form a greedy k-means clustering method for clustering driving events. The basic idea of the greedy k-means clustering method is to firstly extend the total cluster numbers K into ρK with an enlargement parameter and initialize the center of each cluster with a random sample in the training data. And then, the greedy k-means clustering method eliminates these cluster centers one-by-one until the total number of cluster centers keeps constant. During a cluster elimination iteration, the greedy k-means clustering method uses $C^*(k) = \{C_1, ..., C_k\}$ to indicate optimal cluster centers of total k clusters and $C^*(k-1)$ to indicate optimal

Algorithm 4: Pseudocode of clustering driving events with the greedy k-means clustering method

Data: Total N driving events are known in advance and total cluster numbers are initialized as k

Result: The optimal cluster numbers k^* and optimal cluster centers $C^*(k^*)$

1 Enlarge k into ρk and initialize these cluster centers with random samples in the training data;
2 Assign k^* as ρk and calculate the sum of driving event dissimilarity degrees $\delta(k^*)$
3 Assign $\delta(k^*)$ as a template value Γ
4 **while** $k^* > \frac{k}{\rho}$ **do**
5 $k^* = k^* - 1$
6 Apply the k-means clustering method to calculate $\delta(k^*)$
7 **if** $\delta(k^*) < \Gamma$ **then**
8 $\Gamma = \delta(k^*)$
9 **end**
10 **end**
11 Output the number k^* of optimal clusters with the minimum Γ
12 Apply the k-means clustering method to determine optimal cluster centers $C^*(k^*)$ through Equation (4.2)

$k-1$ cluster centers when the k-means clustering method classifies driving events into total $k-1$ clusters. For the given total k clusters, the k-means clustering method determines optimal cluster centers $C^*(k)$ by minimizing the sum of driving event dissimilarity degrees $\delta(k^*)$ as follows:

$$\arg\min_{C_j \in C} \sum_{i=1}^{N} \sum_{j=1}^{k} \varphi_{\Phi_i, C_j} M_{\Phi_i \to C_j} \tag{4.2}$$

where $M_{\Phi_i \to C_j}$ represents driving event dissimilarity degrees between the i^{th} driving event and the center in the cluster C_j; φ_{Φ_i, C_j} is a binary indicator and equals to 1 if Φ_i is in C_j. Otherwise, φ_{Φ_i, C_j} becomes 0. Based on Equation (4.2), the k-means clustering method obtains optimal k driving event cluster centers. Algorithm 4 shows how to cluster driving events through the greedy k-means clustering method. For the given total N driving events and initial k clusters, the greedy k-means clustering method firstly enlarge the total number of clusters into ρk=k^* and calculates the sum of driving event dissimilarity degrees as $C^*(\rho k)$ as a template value Γ (Lines $1-3$). And then, the greedy k-means clustering method applies the greedy algorithm

to decrease the total number of clusters one-by-one and calculates the sum of driving event dissimilarity degrees. If the sum value is less than the template value Γ, Γ will be replaced with sum value (Lines 5−9). The greedy k-means clustering method will keep repeating this process until the clusters with the minimum driving event dissimilarity degrees are obtained. Under this condition, the greedy k-means clustering methods can cluster driving events with global optimal results.

4.1.1.2 Control Policy Extraction

For a given driving event cluster c, Polsa firstly needs to know its control behavior type A a target AV executes and states $x = [v, v', p, p']$ of a target AV and its nearby vehicle which triggers control behavior type A. And then, Polsa combines control behavior type A and state related function $f(x)$ together to form a control policy $\pi = [f(x), A]$ for the driving event cluster c. For example, a target AV will choose the control behavior type (*Deceleration*) to avoid a collision with its front nearby vehicle when the nearby vehicle decelerates and their relevant distance is less than stop distance d_{stop}. Under this condition, $f(x)$ becomes $\{\Delta v' \leq \delta_1 \; and \; ||p'-p||_2 \leq \delta_2\}$ and A is described as $\{Deceleration\}$, where δ_1 and δ_2 are two threshold values.

To describe state related function $f(x)$ accurately, we define T_c to indicate the set of time stamps when a driving event happens and introduce a linear regression model to figure out state related function $f(x)$ which triggers control behavior A in the driving event cluster c. In our linear regression model, for each state x_t^i with x_t^1=v, x_t^2=v', x_t^3=p and x_t^4=p', Polsa estimates its specific value at time-stamp $t \in T_c$ with other three states as follows:

$$\widehat{x}_t^i = \sum_{j=1 \wedge i \neq j}^{4} \alpha_j x_t^j + \alpha_0 \tag{4.3}$$

where α is used to describe the effect of state x_t^j on state x_t^i at t. Based on Equation

(4.3), Polsa obtains the estimated states and uses these estimated values and their true values to calculate the mean square error $\frac{1}{T_c}\sum_{t=1}^{T_c}(\widehat{x}_t^i - x_t^i)^2$ in the whole driving event. For the well trained model with R_c^i, absolute error $|\widehat{x}_t^i - x_t^i|$ at time t should be less than a small threshold value ρ_i if a driving event in c is triggered. R_c^i represents a set of states which co-trigger a driving event in c together with state x_t^i and can be an empty set when only state x_t^i triggers a driving event. In other words, a driving event in c will happen if the following trigger conditions are satisfied: $|\sum_{j\in R_c^i}\alpha_j x_t^j + \alpha_0 - x_t^i| \leq \rho_i$. By this way, we can extract a control policy of a target AV for the driving event cluster c and describe it as $\{|\sum_{j\in R_c^i}\alpha_j x_t^j + \alpha_0 - x_t^i| \leq \rho_i; A\}$.

4.1.2 Optimal Control Policy Determination

In this section, we will discuss how Polsa takes advantage of the extracted control policies to determine optimal control policies of a target AV under different driving scenarios. When a target AV drives on the road, it needs to know states of its nearby vehicle at next several time-stamps so that it can determine which control policy should be executed to ensure its driving safety. However, it is difficult for a target AV to know states of its nearby vehicle at next time-stamps by considering irregular control behaviors of a nearby vehicle. Existing methods [28, 40] just assume that states of a nearby vehicle always keep constant in the next several time-stamps for simplicity. For example, they will consider speed of its nearby vehicle constant at next time stamps, which is contrary to real driving situations and result in the target AV's wrong control policy selection.

To handle this problem, we develop two different driving state prediction models for Polsa. The first driving state prediction model works through a Support Vector Machine and a Bayesian filter (SBF) and the other driving state prediction model predicts driving state based on a long short term memory (LSTM) based recurrent

neural network. For the SBF based driving state prediction model, it firstly uses a Support Vector Machine to predict the possibility of being a certain driving state for a nearby vehicle based on driving states of the ego AV and its nearby vehicle. and then it applies a Bayesian filter to smooth the probability outputs to improve driving state prediction accuracy. For the LSTM-neural network based driving state prediction model, it inputs driving state relevant information of the ego AV and its nearby vehicle into a LSTM-neural network to predict driving state of the nearby vehicle. We will explain details of the SBF based driving state prediction model and the LSTM-neural network based driving state prediction model in Section 4.1.2.1.

4.1.2.1 Driving State Prediction

Here, we propose the SBF based driving state prediction model and the LSTM based driving state prediction model for Polsa to predict driving state of a nearby vehicle.

SBF based Driving State Prediction Model: For driving events in a driving event cluster, the SBF based driving state prediction model uses position and driving speed related information of a target AV and its nearby vehicle to predict driving state (*Acceleration*, *Deceleration*, *Constant-speed*, *Left-lane-change*, *Right-lane-change*, or *Intersection-turn*) of a nearby vehicle at next time stamps. Specifically, Support Vector Machine uses a feature map to transform position and driving speed related information from a low dimension to a high dimension and finds an optimal hyper-plane which can maximize the separation margin between samples in two classes. For driving events with total M samples $\{(x_1, o_1), ..., (x_M, o_M)\}$, observation o_m represents one driving state type in $O = \{$*Acceleration*, *Deceleration*, *Constant-speed*, *Left-lane-change*, *Right-lane-change*, *Intersection-turn*$\}$ and $m \in \{1, ..., M\}$. Support Vector Machine formulates the following constrained optimization problem to find an optimal hyper-

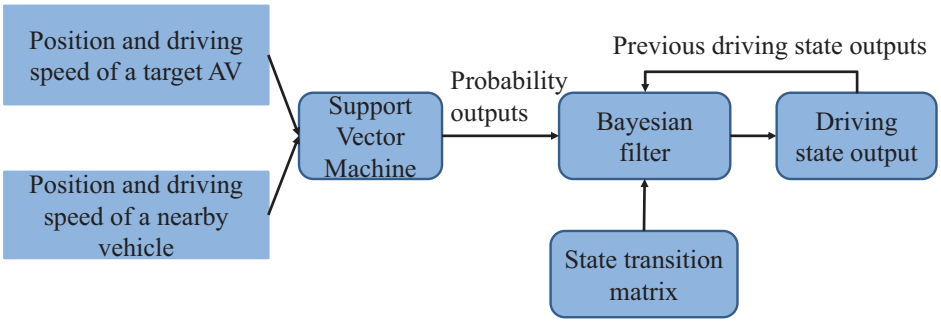


Figure 4.6: The SBF based driving state prediction model with a Support Vector Machine and a Bayesian filter.

plane:

$$\arg\min_{W} \frac{1}{2} W^T W + \frac{C}{M} \sum_{m=1}^{M} \eta_m \tag{4.4a}$$

$$s.t. \quad 1 - o_m(x_m^T W + b) \geq \eta_m \ \wedge \ \eta_m \geq 0 \tag{4.4b}$$

where W represents the normal vector to a hyperplane and η_m represents a nonnegative variable changing between 0 and $1 - o_m(x_m^T W + b)$; C indicates a parameter used to control the trade-off between maximizing the margin and minimizing the training error. Specifically, its small value tends to emphasize the margin while ignoring the outliers in the training data but a large value tends to overfit the training data. Based on Equation (5.7), Polsa obtains the optimal normal vector W^* which forms an optimal hyperplane to classify any two driving states.

Figure 4.6 how the SBF based driving state prediction model predicts driving state of a nearby vehicle based on SBF. Support Vector Machine chooses position and driving speed relevant features to form a feature vector and uses this feature vector as the input x_m of Support Vector Machine: position p'_m and driving speed v'_m of the nearby vehicle, position difference $p_m - p'_m$ and driving speed difference $v_m - v'_m$ of the target AV and the nearby vehicle and their first derivatives $\frac{\partial(p_m - p'_m)}{\partial t}$ and $\frac{\partial(v_m - v'_m)}{\partial t}$. Here, we choose these features mainly because patterns of these features

are remarkably different for each driving state type. Based on feature vectors in total m time-stamps, Support Vector Machine outputs the probability of an input x_m belonging to driving state o_m.

As shown in Figure 4.6, the driving state prediction model uses a Bayesian filter to smooth probability output $P(o_m|x_m)$ from Support Vector Machine by fusing a sequence of observed featured factors $\{x_1, ..., x_m\}$ together and the final probability output $P(o_m|x_1, ..., x_m)$ that the nearby AV's driving state belongs to o_m can be smoothed as:

$$P(o_m|x_1, ..., x_m) = P(o_m|x_m) \sum_{o_m \in O} P(o_m|o_{m-1}) P(o_{m-1}|x_1, ..., x_{m-1}) \tag{4.5}$$

where x_m represents feature vector at time m and o_m indicates driving state at time m; $P(o_m|o_{m-1})$ represents state transition probability from o_{m-1} to o_m and can be learned during the training process. To calculate these state transition probabilities, we firstly count the total number $ST(o_i, o_j)$ of transitions from driving state o_i to driving state o_j in the training data and then normalize these counted numbers as follows:

$$P(o_j|o_i) = \frac{ST(o_i, o_j)}{\sum_{o_k \in O} ST(o_i, o_j)} \tag{4.6}$$

based on Bayesian filter and Support Vector Machine, the driving state prediction model can predict driving state of a nearby vehicle. And then, Polsa will infer position and driving speed of the nearby vehicle based on the predicted driving state and apply them in its optimal control policy determination process.

LSTM-Neural Network based Driving State Prediction Model: Though the SBF based driving state prediction model can predict driving state of a nearby vehicle in the next few time stamps, it face an information theoretically infeasible problem in practice by considering that Bayesian filter needs specific prior knowledge of a nearby vehicle to tune parameters during the model training process. As one type of deep neural networks, the long short term memory (LSTM) based recurrent neural network

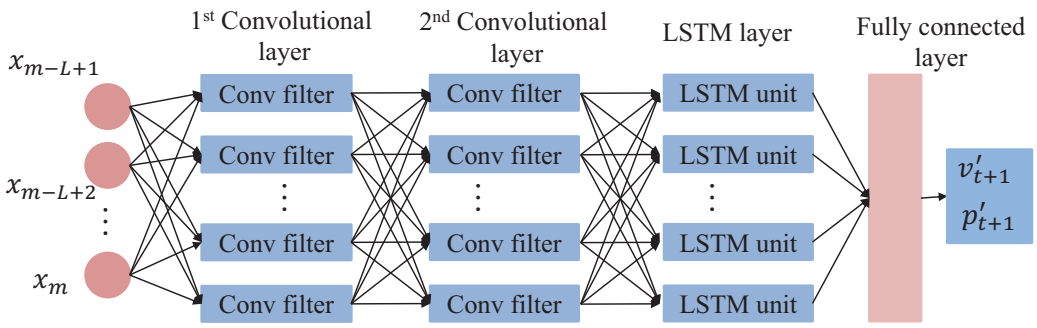


Figure 4.7: The architecture of LSTM-neural network based driving state prediction model.

has been widely used in learning temporal dynamics and analyzing the sequential structure in the time-series data [62]. Here, we also develop a LSTM-neural network based driving state prediction model to predict driving state of a nearby vehicle to avoid the above problems and compared their driving state prediction performance between SBF and LSTM-neural network in Section 5.2.

For the LSTM-neural network based driving state prediction model, it inputs position and driving speed relevant information of the ego AV and its nearby vehicle into a LSTM-neural network to predict driving state of the nearby AV at next time-stamps. The LSTM-NN based driving state prediction model works by putting convolutional layers and LSTM layers as its front layers and a connected layer as its end layer. Specifically, convolutional layers are used to extract useful information from its input data and LSTM layers help to estimate SOH based on the extracted useful information from convolutional layers.

Figure 4.7 shows the architecture of the LSTM-neural network based driving state prediction model. The LSTM based neural network includes an input layer, two convolutional layers, a LSTM layer and a fully connected layer. For the LSTM based neural network, the input layer inputs $[y_{m-L+1}; y_{m-L+2}; ...; y_m]$ at time stamp m and the fully connected layer outputs $[v'(m+1), p'(m+1)]$, where L represents the length of inputs and affects driving state prediction performance of the

LSTM-neural network based driving state prediction model. y_{m-L+1} indicates values of v, v', p, p' and $v \dot{-} v'$ at time stamp $m - L + 1$ and can be described as $[v_{m-L+1}, v'_{m-L+1}, p_{m-L+1}, p'_{m-L+1}, v_{m-L+1}-v'_{m-L+1}, p_{m-L+1}-p'_{m-L+1}, \Delta v_{t-L+1}, \Delta p_{t-L+1}]$. As shown in Figure 4.7, input data firstly passes through two convolutional layers for useful information extraction and then the extracted useful information will be sent into the LSTM layer to conduct regression processes. Lastly, the fully connected layer will convert outputs from the LSTM layer into a final driving state prediction result (position and driving speed).

Input length L affects driving state prediction accuracy of the LSTM-neural network based driving state prediction model because an input with a small input length may lead to information loss of the entire time-series data but an input with a large input length may result in high computation load. We apply a False Nearest Neighbors (FNN) method [13] to determine optimal input length L^*. FNN is an approach of optimizing dimensions of time-series data by assuming that the optimal input length can minimize the percentage of false nearest neighbors for an input. Here, we analyze how the percentage of false nearest neighbors for an input changes as input length L increases to determine the optimal input length. For the input $Y^m = [y^{m-L+1}, y^{m-L+2}, ..., y^m]$ and its nearest neighbor $Y^k = [y^{k-L+1}, y^{k-L+2}, ..., y^k]$, the FNN method calculates Euclidean distance to indicate their distance as follows:

$$D_L(m,k) = \sum_{j=1}^{L} ||y^{m+j-1} - y^{k+j-1}|| \tag{4.7}$$

where $D(m,k)$ indicates Euclidean distance between X^m and X^k. When input length changes from L to $L+1$, their Euclidean distance $D_{L+1}(m,k)$ will become $D_L(m,k) + ||y^{m+L} - y^{k+L}||$. Based on the above distances under different length changes, the FNN method checks whether X^m is a false nearest neighbor of X^k by comparing $\frac{|D_{L+1}(m,k)-D_L(m,k)|}{D_L(m,k)}$ and a threshold τ_D. X^m will be detected as a false nearest neighbor

of input X^k if $\frac{|D_{L+1}(m,k)-D_L(m,k)|}{D_L(m,k)}$ is larger than τ_D. Otherwise, it will be considered as its true nearest neighbor. By counting the number of false and true nearest neighbors under different input lengths, the FNN method calculates corresponding percentages of false nearest neighbors and determines the optimal input length L^*. And then, the input $[y_{m-L^*+1}; y_{m-L^*+2}; ...; y_m]$ at time stamp m will be sent into the LSTM-neural network based driving state model to predict driving state of the nearby vehicle at next time stamp.

4.1.2.2 Optimal Control Policy Determination

For a driving event, a target AV measures current state of its nearby vehicle to predict its states at next time stamps so that the target AV can select an optimal control policy to ensure its driving safety. Based on Section 4.1.1, Polsa extracts control policies $\Pi = \{\pi_1, ..., \pi_{M_A}\}$ of a target AV, where M_A represents the total number of extracted control policies for the control behavior type A. Given state $y = [p, v]$ of a target AV and $y' = [p', v']$ of its nearby vehicle, we propose an optimal control policy determination method for Polsa to select an optimal control policy. Specifically, the proposed optimal control policy determination method determines an optimal control policy by considering states of both the target AV and its nearby vehicle in the near future. Therefore, the goal of the proposed optimal control policy determination method is to find an optimal control policy π^* which maximizes total rewards over its time period T and this optimal control policy determination problem can be described as follows:

$$\arg\max_{\pi_i \in \{\pi_1,...,\pi_{M_A}\}} \sum_{t=1}^{T} \gamma^t R(y_t, y'_t) P(y_t, y'_t | \pi_i) \tag{4.8}$$

where γ represents a reward discount rate; $R(y_t, y'_t)$ is defined as a reward function representing the reward when the state of a target AV is y_t and state of its nearby vehicle is y'_t at time t and can be calculated as $e^{\mu_1||p_t - p'_t|| + \mu_2|v_t - v'_t|}$, where μ_1 is a

positive constant and μ_2 is a negative constant. Based on the reward function, Polsa will obtain a high reward if position difference between the target AV and its nearby vehicle becomes large or their speed difference becomes small.

$P(y_t, y'_t|\pi_i)$ in Equation (4.8) represents the probability of state of the target AV becoming y_t and state of its nearby vehicle becoming y'_t when the target AV conducts control policy π_i at time t. Since a target AV and its nearby vehicle are independent on each other, state of the target AV or its nearby vehicle at next time stamp only depends on its current state and control policy. Therefore, the probability $P(y_t, y'_t|\pi_i)$ can be calculated as $P(y_t|\pi_i)P(y'_t)$, where $P(y_t|\pi_i)$ and $P(y'_t)$ represent the probability of state of the target AV becoming y_t under control policy π_i and the probability of state of its nearby vehicle becoming y'_t. For $P(y_t|\pi_i)$, Polsa can calculate it by considering previous state of the target AV and its executed control policy π_i. For $P(y'_t)$, Polsa uses the proposed driving state prediction model to obtain the probability of nearby vehicle's state becoming y'_t at time t. By this way, Polsa calculates the cumulative rewards in the given time period T for different control policies and selects optimal control policy π^* which can maximizes cumulative rewards. Since Polsa uses the safety criteria to check optimal control policy π^* before executing it on a target AV to ensure that the target AV always drives at a safe region.

4.2 Performance Evaluation

We implemented Polsa by running MatLab on one laptop (Intel i5 CPU and 16 gigabyte memory) and used driving datasets from Baidu Apollo simulation platform [4], by running two existing popular autonomous driving control methods [28, 40] to extract control policies and test optimal control policy determination performance of Polsa. As indicated in [26], Apollo simulation platform is developed by Baidu company to simulate autonomous driving environments based on real-world traffic

driving data and has good control policy testing performance. Specifically, we firstly compared control policies extracted by Polsa and real control policies of two existing autonomous driving control methods to test control policy extraction performance of Polsa and then compared optimal control policy determination results to evaluate optimal control policy determination performance of Polsa. For simplicity, we use ACC, DEC, COS, LLC, RLC, and INT to represent *Acceleration*, *Deceleration*, *Constant-speed*, *Left-lane-change*, *Right-lane-change* and *Intersection-turn*.

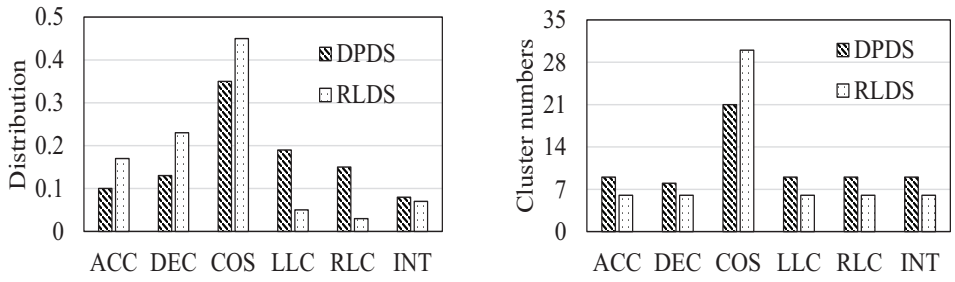
4.2.1 Experiment Settings

Comparison Methods: We implemented two existing popular autonomous driving control methods into Baidu Apollo simulation platform to generate driving scenario training datasets, which are used by Polsa to extract control policies for optimal control policy determination result comparisons. One method is a designed-policy based driving control system (DPDS) [28] and the other method is a reinforcement learning based driving control system (RLDS) [40]. DPDS adjusts control behaviors of a target AV by firstly measuring its states and its nearby vehicle states and then choosing control policies among the designed control policies to ensure driving safety of a target AV. DPDP adopts the following designed control policy types and uses them to control a target AV: lane-maintain, right/left lane change, turn left, turn right, go straight, and yield at an intersection. RLDS builds a reward function with driving comfort consideration and uses a reinforcement learning method to determine a control policy among the designed control policies for a target AV by maximizing its cumulative rewards. For RLDS, its designed control policies include speed maintain, lane main, accelerate, hard accelerate, decelerate, hard decelerate, right lane change and left lane change. Compared with Polsa which considers time-varying driving state of a nearby vehicle and extracts control policies from driving scenario datasets, DPDS

and RLDS assume that states of a nearby AV keep constant in the next time-stamps and control policies of a target AV are known in advance.

Driving Scenario Dataset: In the experiment, we used three driving scenario datasets (two driving scenario training datasets and a driving scenario testing dataset) from Baidu Apollo AV simulation platform for performance evaluation. In this simulation platform, we designed two driving scenario training datasets (each dataset includes total 218 driving scenarios) for control policy extraction and a driving scenario testing dataset (total 200 driving scenarios) for optimal control policy determination performance evaluation. For each driving scenario training dataset, it simulates how a target AV with DPDS or RLDS executes control behaviors under different driving scenarios and each driving scenario includes a target AV and its nearby vehicle driving on the same road. Specifically, for driving scenarios with the same control behavior type, both the target AV and its nearby vehicle may drive with variable or constant speeds and have different initial relevant positions. For the driving scenario testing dataset, it contains different driving scenarios to test whether optimal control policies can ensure driving safety of a target AV.

Evaluation Aspects: In this section, we evaluated Polsa's performance in extracting control policies and determining optimal control policies. For the control policy extraction part, we used Polsa to extract control policies of DPDS and RLDS based on two driving scenario training datasets and showed its control policy extraction results in terms of policy time duration distributions and control policy extraction accuracy. Here, control policy extraction accuracy represents the rate of control policies whose trigger conditions and control behaviors are correctly extracted over all control policies under different position and driving speed differences. We also tested driving state prediction accuracy of Polsa and driving state prediction accuracy is calculated as the rate of driving scenarios where driving state of a nearby vehicle is correctly predicted over all driving scenarios. For the optimal control policy deter-



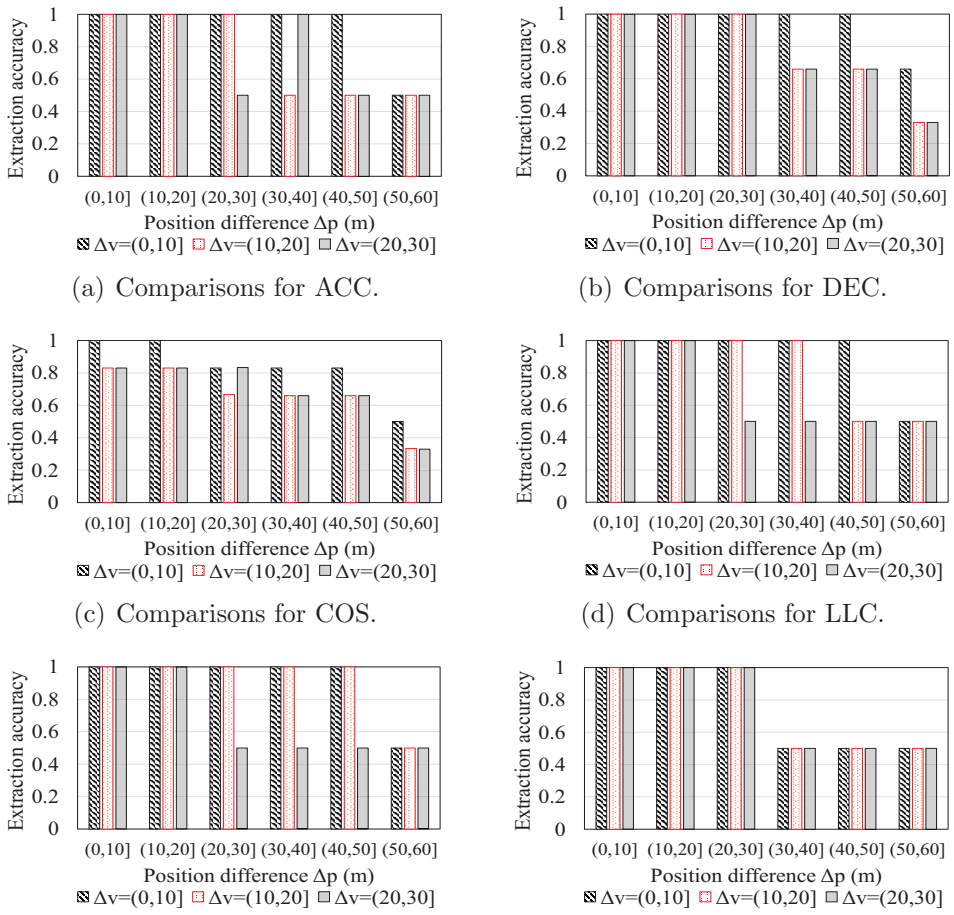
(a) Distributions of driving event types. (b) Driving event cluster numbers.

Figure 4.8: Number distributions of driving events with the same control behavior type and numbers of driving event clusters for each control behavior type in driving scenario training datasets of DPDS and RLDS.

mination evaluation part, we firstly used DPDS, RLDS and Polsa to determine their optimal control polices for driving scenarios in the driving scenario testing dataset and then checked whether the distance between the target AV and its nearby vehicle is larger than the defined safety criteria in Figure 5.4 to calculate optimal control policy success rate. Here, optimal control policy success rate indicates the rate of driving scenarios where a target AV executes the suggested optimal control policy and its position relevant to its nearby vehicle is always larger than safety criteria among all driving scenarios.

4.2.2 Evaluation Results

Here, we firstly used Polsa to extract control policies of DPDS and RLDS based on two driving scenario training datasets. And then, we applied Polsa, DPDS and RLDS to determine optimal control policies for each driving scenario in the driving scenario testing dataset. Lastly, we calculated optimal control policy success rates of Polsa, DPDS and RLDS for optimal control policy determination performance evaluation.



(a) Comparisons for ACC.

(b) Comparisons for DEC.

(c) Comparisons for COS.

(d) Comparisons for LLC.

(e) Comparisons for RLC.

(f) Comparisons for INT.

Figure 4.9: Control policy extraction accuracies of Polsa on DPDS's control policies with control behaviors as following types: (a) ACC, (b) DEC, (c) COS, (d) LLC, (e) RLC and (f) INT.

4.2.2.1 Driving Event Clustering Results

To extract control policies of DPDS or RLDS based on its driving scenario training dataset, Polsa firstly needs to cluster driving events by calculating their event dissimilarity degrees and then compares these event dissimilarity degrees with a threshold to determine which driving event cluster a driving event belongs to. Figure 4.8(a) shows distributions of driving event numbers with the same control behavior type in driving scenario training datasets of DPDS and RLDS, respectively. Figure 4.8(b) compares the number of driving event clusters for each control behavior type. As shown in Fig-

ure 4.8(a), Polsa clusters driving events into total six different types, which matches the number of control behavior types defined in Section 4.1.1.1. We see that driving events with ACC, DEC, and COS domonate the driving scenario training dataset for DPDS and RLDS. This is mainly because a target AV meets driving events (i.e., ACC, DEC, and COS) more frequently than other driving events (i.e., LLC, RLC and INT) when driving on the road. As shown in Figure 4.8(b), we see that the number of driving event clusters for COS is larger than numbers of driving event clusters for other control behavior types. This is because, compared with other control behavior types, both latitude and longitude position differences between a target AV and its nearby vehicle change more dynamically, which requires more driving event clusters to describe COS accurately. Based on driving event clustering results, Polsa analyzes the control behavior of the target AV and driving states of the target AV and its nearby vehicle to form a control policy for each driving event cluster. Therefore, the number of control policies for each control behavior type should equal to the number of its corresponding driving event clusters.

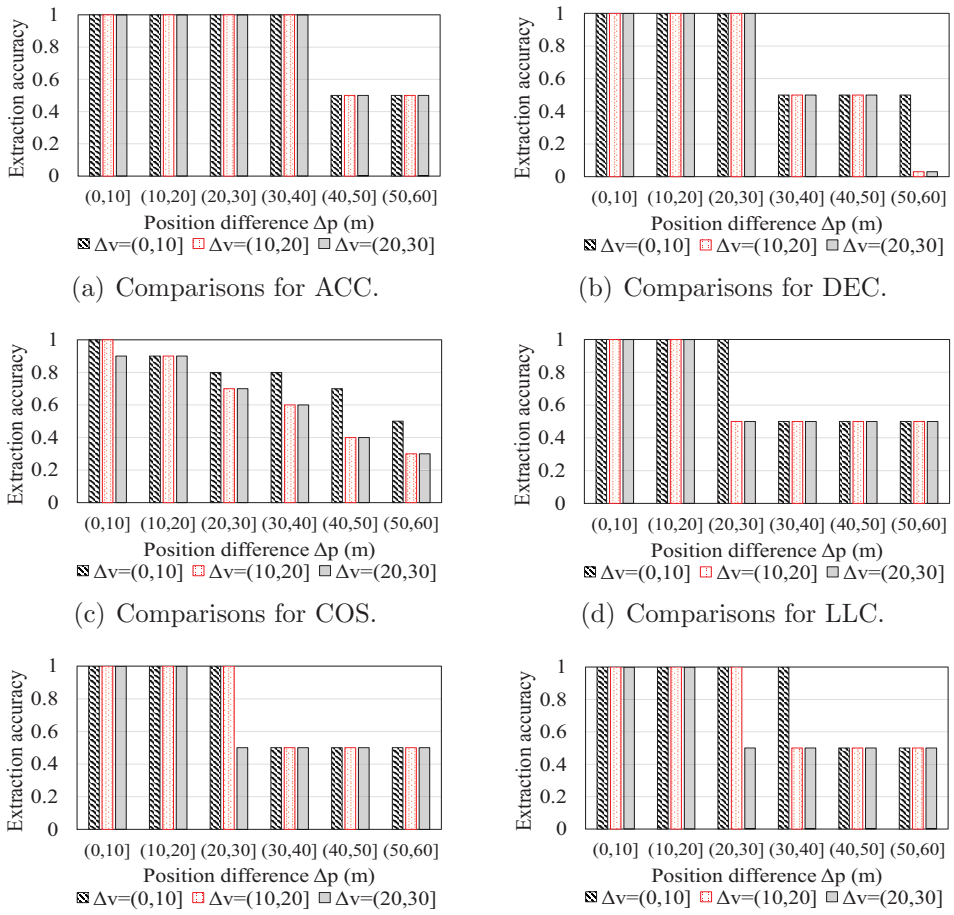
4.2.2.2 Control Policy Extraction Results

Based on driving event clustering results, Polsa analyzes control behavior of a target AV and its trigger condition to extract one control policy for each driving event cluster. Based on Figure 4.8(b), Polsa extracts total 66 control policies from the driving scenario training dataset of DPDS and total 60 control policies from the driving scenario training dataset of DPDS. For DPDS, numbers of control policies for ACC, DEC, COS, LLC, RLC and INT equal to 9, 9, 21, 9, 9 and 9. For RLDS, numbers of control policies for ACC, DEC, COS, LLC, RLC and INT equal to 6, 6, 30, 6, 6 and 6. Based on the extracted control policies, we did statistical analysis on time durations of executing a control policy and find that time durations of many control

policies are less than $10seconds$. In other words, for many control policies, it takes less than $10seconds$ to finish its execution process, which explains why a target AV needs to predict driving state of its nearby vehicle at next time stamps so that it can make a correct control behavior. Specifically, around 80% of control behaviors like LLC and RLC are accomplished within less than $5seconds$. This is because position difference between a target AV and its nearby vehicle changes greatly and the target AV needs to finish a whole lane change process quickly to ensure its driving safety.

Here, we calculated control policy extraction accuracies of Polsa on DPDS's control policies under different Δvs and Δps and showed control policy extraction accuracies in Figure 4.9. Figure 4.9(a), Figure 4.9(b), Figure 4.9(c), Figure 4.9(d), Figure 4.9(e) and Figure 4.9(f) show extraction accuracies of Polsa on control policies with control behavior as ACC, DEC, COS, LLC, RLC and INT, respectively. We see that both Δp and Δv affect control policy extraction accuracies of Polsa on all control policies with different control behaviors. Specifically, Polsa keeps high control policy extraction accuracy (near to 100%) when Δp is less than $30m$ but its control policy extraction accuracy reduces as Δp increases from $30m$ to $60m$. Besides, Polsa has lower control policy as Δv increases from 0 to $30km/h$. Here, large Δp or Δv reduces Polsa's control policy extraction accuracy because driving scenario training datasets contain less driving events with large Δp or Δv for extracting control policies, which results in low control policy extraction accuracy.

We also used Polsa to extract control policies of RLDS and showed its control policy extraction accuracies under different Δvs and Δps in Figure 4.10. Similar to the DPDS situation, Δp and Δv also affect control policy extraction accuracies of Polsa in the RLDS situation. Specifically, control policy extraction accuracy of Polsa firstly keeps stable and then decreases as Δp increases from 0 to $60m$. Besides, control policy extraction accuracy of Polsa generally becomes lower as Δv increases from 0 to $30km/h$. Compared with the DPDS situation, Polsa extracts control policies of RLDS



(a) Comparisons for ACC.

(b) Comparisons for DEC.

(c) Comparisons for COS.

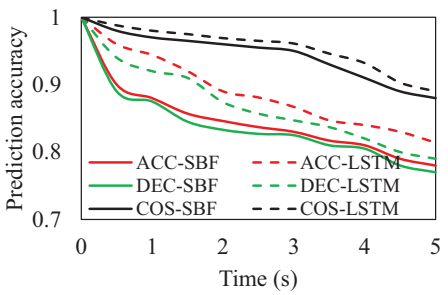
(d) Comparisons for LLC.

(e) Comparisons for RLC.

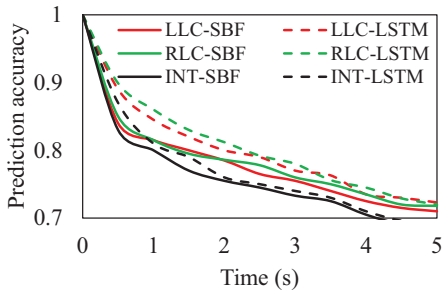
(f) Comparisons for INT.

Figure 4.10: Control policy extraction accuracies of Polsa on RLDS's control policies with control behaviors as following types: (a) ACC, (b) DEC, (c) COS, (d) LLC, (e) RLC and (f) INT.

less accurately when Δp is larger than $30m$. This is mainly because RLDS considers driving comfort with a high priority and always tries to avoid large deceleration or acceleration actions when making control policies for a target AV, which results in less chances for a target AV to execute control behaviors (e.g., ACC, DEC, LLC and RLC) and hence less driving events with larger Δp in the training dataset. Based on above analysis, we conclude that Polsa has acceptable control policy extraction accuracies on both DPDS and RLDS and can be used to extract control policies of a target AV based on their historical driving data.



(a) Prediction accuracy comparisons on ACC, DEC and COS.



(b) Prediction accuracy comparisons on LLC, RLC and INT.

Figure 4.11: Driving state prediction accuracy comparisons on ACC, DEC, COS, LLC, RLC and INT as time period increases from 1 to 5*seconds*.

4.2.2.3 Driving State Prediction

Based on driving event clustering results for each control policy type, Polsa firstly trained the SBF based driving state prediction model [45] (SBF) and the LSTM-neural network based driving state prediction model (LSTM) with 70% samples in the driving scenario training dataset and then used them to predict driving state of a nearby vehicle at the next time stamp (i.e., 1 *second*). Here, we compared the predicted driving states and real driving states to test their driving state prediction accuracy performance. For control behavior types (i.e., ACC, DEC, COS, LLC, RLC and INT), average driving state prediction accuracies of SBF equal to (88%, 87%, 96%, 81%, 79%, 92%) while average driving state prediction accuracies of LSTM equal to (94%, 92%, 97%, 84%, 83%, 87%). Here, LSTM has higher driving state prediction accuracy than SBF mainly because LSTM has no information theoretically infeasible problems during the model training process.

Here, we also used SBF and LSTM to predict driving states of a nearby vehicle when time period increases from 1 to 5 *seconds*. Figure 4.11 shows driving state prediction accuracy comparisons between SBF and LSTM on different control behavior types. Overall, LSTM has higher driving state prediction accuracy than SBF as time period increases. Therefore, we applies LSTM into Polsa and Polsa applies

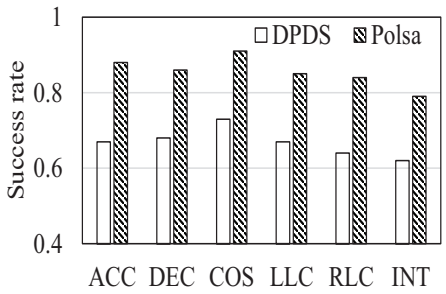
its driving state prediction results to determine optimal control policies. Specifically, for both SBF and LSTM, their driving state prediction accuracies on all six control behavior types firstly decrease greatly and then gradually as time period increases. This is because large time period usually results in less observation validations during driving state prediction processes and state prediction processes with less observation validations will reduce driving state prediction accuracy for SBF and LSTM.

4.2.2.4 Optimal Control Policy Determination

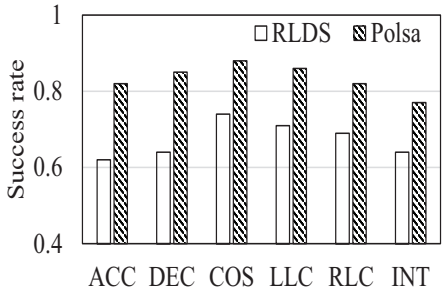
For each driving event in the driving scenario testing dataset, we applied Polsa, DPDS and RLDS to determine their own optimal control policies and then used the developed safety criteria to check whether their optimal control policies will guarantee driving safety of the target AV. Figure 4.12 shows optimal control policy success rate comparisons between Polsa, DPDS and RLDS. For Polsa, its optimal control policy success rates on all six control behavior types are more than 80% while values of DPDS are (67%, 68%, 73%, 67%, 64%, 62%) and values of RLDS are (62%, 64%, 74%, 71%, 69%, 64%). We see that Polsa has higher optimal control policy success rate than DPDS and RLDS. This is mainly because Polsa predicts driving state of a nearby vehicle and applies the driving state prediction result during its optimal control policy determination process to determine optimal control policies for a target AV. The above analysis result shows that Polsa can determine optimal control policies with high optimal control policy success rate.

4.2.2.5 Computational Time analysis

We also tested computational time of Polsa when determining optimal control policies for different numbers of driving events and made computational time comparisons between Polsa, DPDS and RLDS in Figure 4.13. We see that both Polsa, DPDS



(a) Comparisons between Polsa and DPDS.



(b) Comparisons between Polsa and RLDS.

Figure 4.12: Optimal control policy success rate comparisons among Polsa, DPDS and RLDS.

and RLDS have higher computational time as the number of driving events increases from 1 to 10. Specifically, their computation time equal to 0.096*second*, 0.067*second* and 0.083*second* as the number of driving events becomes to 5. Here, Polsa has the highest computation time than DPDS and RLDS mainly because Polsa takes time to predict driving state of the nearby vehicle for determining optimal control policies. The maximum computation time of Polsa is still less than 0.1*second* and it is small enough for a target AV to execute control policies to ensure driving safety.

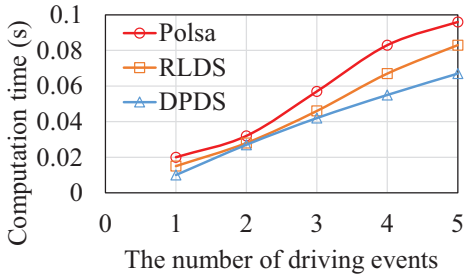


Figure 4.13: Computational time comparisons between Polsa, DPDS and RLDS.

4.3 Related Work

Methods have been proposed to analyze safety features of AVs to ensure their driving safety. [16, 18, 28, 61, 71] try to keep developing new control strategies to control AV's control behaviors at different driving scenarios so that a target AV can drive safely under these driving scenarios. For example, Rosolia [61] developed a nonlinear

control approach to generate a collision-free trajectory for a target AV so that a target AV has the ability of avoiding obstacles when driving on the highway. Chen [16] developed a fuzzy control system which adjusts driving direction and driving speed of a target AV in real time with traffic condition consideration to ensure driving safety. Tian [71] proposed a decision making algorithm which models multi vehicles driving in a roundabout intersection situation with game theory and helps a target AV decide whether it should enter and cross the intersection. Galceran [28] proposed an integrated behavior inference and decision-making approach which models vehicle behavior of a target AV and determines a set of control behaviors for a target AV with considering control behavior of its nearby vehicle to avoid possible collisions with its nearby vehicle. Chen [18] proposed a neural network based control decision-making system which uses a neural network to learn driving behavior of a human driver and make control decisions based on its trained neural network to avoid possible collisions. Though these methods help to ensure driving safety of a target AV under several specific driving scenarios, they may fail to work for other driving scenarios. In other words, the total number of these specific driving scenarios are limited and these specific driving scenarios cannot cover all possible driving scenarios in public roads by considering highly complex driving environments in practice. Besides, several AV companies develop their own control policies for their AVs and these control policies are usually not open for the public, which makes it difficult to analyze their control policies for further improvement. To handle this problem, we built a control policy extraction method to extract control policies of an AV from its historical driving data.

Methods [31, 32, 40] try to test driving safety performance of control policies in an AV through driving simulation or road experiments. For example, Jha [32] designed an AV fault injection simulator to test its control policies by injecting fault sensor information into a target AV and simulating how a target AV responses based on its control policies. These methods assume that the state of a nearby vehicle always

keeps constant in one short time period, which makes their testing results less reasonable in practice. Road experiments can better test driving safety performance of control policies in a target AV and Hunger [31] developed a driving scenario formalization method to test control policies under different driving scenarios during road experiments. However, existing methods only check whether a target AV with control policies will fail to work under a certain driving scenario but it is difficult for them to figure out which control policy can be selected to ensure driving safety of the target AV under a certain driving scenario. To handle this problem, we designed an optimal control policy determination method to determine an optimal control policy from the extracted control policies.

Chapter 5

Multi-AV Control Decision Making System

5.1 System Design

We present the architecture of MADM in Figure 5.1. MADM consists of two major parts: the policy formation method and the multi-AV control decision making method. The policy formation method combines imitation learning and reinforcement learning together to form policies to learn driving behaviors of an expert based on the expert driving data. In the multi-AV control decision making part, MADM firstly uses a multi-agent reinforcement learning model to adjust the learned policy of each individual AV from the policy formation method to ensure that MADM can make control decisions for multiple AVs simultaneously. Besides, based on driving states of each individual AV, MADM forms its backup policy and applies them during the decision making process to guarantee driving safety of each individual AV. Lastly, MADM sends its control decisions to multiple AVs and each individual AV will drive by following its control decision to ensure driving safety.

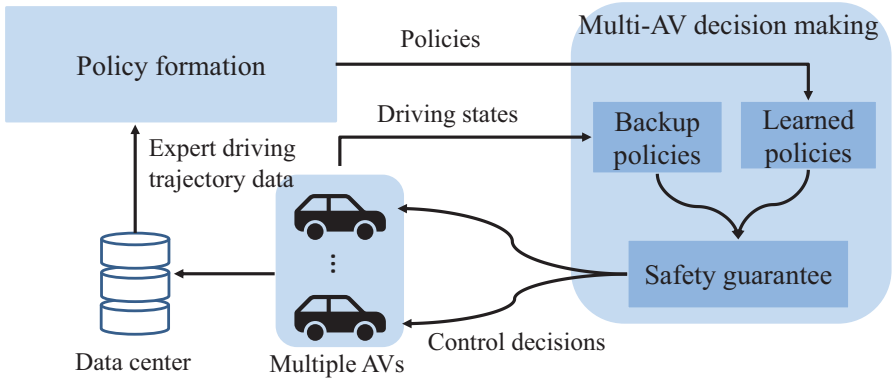


Figure 5.1: The architecture of MADM for making control decisions in multi-AV coexistence situations.

5.1.1 Imitation and reinforcement learning based Policy Formation

An AV contains several subsystems (e.g., sensor measurement system, intelligent decision making system and decision execution system). These subsystems firstly measure the real-time driving environment to make control decisions based on its policies and then execute these control decisions to obtain fully autonomous driving functions. Manually developing policies for an AV and testing the developed policies in a real driving environment consume huge time and any a poor control decision may cause physical vehicle damages. Existing RL based control decision making methods have low efficiency in forming policies and may even face policy convergence problems because of a huge driving state space. The above two limitations motivate us to take the advantage of imitation learning for the policy formation.

For the imitation learning, it forms policies by learning driving behaviors of an expert based on the expert's driving data and achieves a similar driving performance as the expert. In practice, the expert driving data only covers a limited number of different driving situations. Under this situation, the learned policies from imitation learning will fail to work if the AV drives in driving situations, which are not included

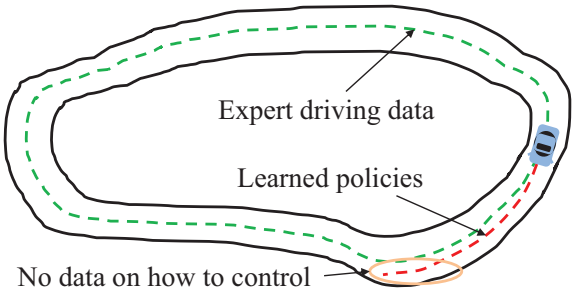


Figure 5.2: The necessary combination of imitation learning and reinforcement learning.

in the expert's driving data. Figure 5.2 shows an example where policies of imitation learning fail to ensure driving safety of an AV during the decision making process. The green dot line indicates one expert driving trajectory sample and the red dot line indicates the trajectory when the AV follows the learned policies at a certain driving situation. We see that the AV drives away from the road centerline as it keeps following the learned policies, which means that the learned policies fail to make correct control decisions to prevent the AV from driving away from the road centerline when the AV is in such a situation (indicated with red line). Therefore, the policies from the imitation learning has the sub-optimal policy formation problem and need to be improved to ensure these policies work on different driving situations. Based on the above analysis, we take the advantage of both imitation learning and reinforcement learning and combine them together into MADM to efficiently form policies for an individual AV based on the expert driving data. Specifically, the imitation learning specifies state and action spaces which are worth exploring firstly by reinforcement learning so that policies working on all possible driving situations can be formed efficiently.

In the proposed policy formation method, MADM models an AV as $\{S; A; \pi_{\theta,\mu}; r; \gamma\}$. Here, S indicates the driving state space of an AV; A represents its action space (i.e., control decisions); $\pi_{\theta,\mu}$ indicates AV's policy used for determining action a at state s; $r : s \times a \rightarrow r$ indicates the reward when an AV takes action a at state s; γ is a

discount reward factor and changes in $[0, 1]$. MADM describes driving behaviors of an expert with π_E and the expert driving data with a sequence of state-action pairs $\{< s^1, a^1 >, < s^2, a^2 >, ..., < s^T, a^T >\}$, where $< s^1, a^1 >$ indicates a state-action pair and shows driving state and driving action at a time-stamp. Based on the expert driving data, MADM firstly calculates driving state distribution as follows:

$$p_{\pi_E}(s) = \frac{1}{T} \sum_{t=1}^{T} \delta^t_{\pi_E}(s) \tag{5.1}$$

where T indicates the total number of state-action pairs in the expert driving data and $p_{\pi_E}(s)$ indicates the average probability of driving state being s when the driving behavior of an expert follows π_E. $\delta^t_{\pi_E}(s)$ indicates whether driving state equals to s at time t. $\delta^t_{\pi_E}(s)$ equals to 1 if driving state equals to s at time t. Otherwise, $\delta^t_{\pi_E}(s)$ equals to 0. For any two pairs of state-actions (s^i, a^i) and (s^j, a^j) with $0 \leq i, j \leq T$, a loss function $L(a^i, a^j)$ in imitation learning [20] is introduced to indicate the difference between these two state-action pairs and defined as $|s^i - s^j|^2 + \eta|a^i - a^j|^2$, where η is used to balance the effects of driving state difference $|s^i - s^j|^2$ and driving action difference $|a^i - a^j|^2$ on $L(a^i, a^j)$.

Based on all state-action pairs in the expert driving data, MADM forms a policy so that the policy can map driving state s to driving action a to make control decisions for an AV. Here, MADM describes the policy $\pi_{\theta,\mu}$ of an AV with parameters θ and μ, where θ and μ are used to indicate parameters in an action network and a reward network. MADM obtains parameters θ and μ by considering both expected cumulative rewards and loss function values between the predicted driving actions and the expert driving actions as follows:

$$\pi_{\theta,\mu} = \arg\max_{\theta,\mu} \sum_{i=1}^{N} \sum_{j=1}^{T_i} (R_i^{\pi_{\theta,\mu}(s^j)} - \beta L_i(\pi_{\theta,\mu}(s^j), a^j)) \tag{5.2}$$

where N denotes the total number of driving trajectories in the expert driving data and T_i indicates the total number of state-action pairs in the i^{th} driving trajectory; s^j indicates the j^{th} driving state in a driving trajectory and follows the expert driving state distribution p_{π_E} during the training process. For the i^{th} driving trajectory, $\pi_{\theta,\mu}(s^j)$ indicates driving action determined by $\pi_{\theta,\mu}(s^j)$ at driving state s^j; $R_i^{\pi_{\theta,\mu}(s^j)}$ represents expected cumulative reward MADM receive will when selecting driving action $\pi_{\theta,\mu}(s^j)$ at driving state s^j in reinforcement learning and equals to $\mathbb{E}\sum_{m=0}^{\infty}\gamma^m r_i(s^{t+m}, a^{t+m})$; $\sum_{j=1}^{T_i} L_i(\pi_{\theta,\mu}(s^j), a^j)$ indicates the sum of loss function values in imitation learning. Here, we use the parameter β to balance imitation learning and reinforcement learning during the policy formalization process.

Solving Equation (5.2) can be interpreted as combining reinforcement learning and imitation learning together and then running them simultaneously. During the policy formation process, $R_i^{\pi_{\theta,\mu}(s^j)}$ helps to converge the policy to achieve as many rewards as possible. Besides, $\sum_{j=1}^{T_i} L_i(\pi_{\theta,\mu}(s^j), a^j)$ encourages the policy to make control decisions, which are the same as driving behaviors of the expert. $\pi^*_{\theta,\mu}$ indicates the optimal policy in Equation (5.2) when MADM is trained with total N driving trajectories. MADM forms policies based on expert driving data and considers loss functions during the policy formation process, which ensures $\pi^*_{\theta,\mu}$ performs as good driving behaviors of the expert at different driving situations and can be formed efficiently.

5.1.2 Multi-AV Optimal Policy Adjustment

For the multi-AV coexistence situation, these individual AVs belong to a competitive and cooperative relationship. Here, the competitive relationship means that each individual AV tries to finish its own driving trip successfully and safely, which is considered as a local task in MADM. Similarly, the cooperative relationship indi-

cates that these individual AVs on the same road need to avoid unexpected driving behaviors (e.g., sudden stop, hard brake and sharp turn) during the driving process to provide a safe driving environment so that there exist no collisions between AVs, which forms a global task in MADM. Therefore, MADM needs to adjust the learned policies of each individual AV so that these adjusted policies consider both local and global tasks during the decision making process.

5.1.2.1 System modeling in multiple-AV coexistence situations

For an AV, its policy $\pi_{\theta,\mu}$ can be obtained through imitation and reinforcement learning based policy formation method in Section 5.1.1. For the multiple AV coexistence situation, MADM considers multiple AVs as a dynamic system and models such a dynamic system as $\{N'; S_i; A_i; \pi_{\theta_i,\mu_i}; r_i; \gamma_i\}$ with $i \in [1, N']$, where N' indicates the total number of multiple AVs; S_i, A_i, π_{θ_i,μ_i} and r_i represent driving state space, driving action space, policy and reward of the i^{th} AV. Here, MADM uses the multi-agent reinforcement learning method to update the learned policy of each individual AV for making multi-AV control decisions. In the multi-agent reinforcement learning method, the learned policies of multiple AVs are indicated as $\{\pi_{\theta_1,\mu_1}; \pi_{\theta_2,,\mu_2}; ...; \pi_{\theta_{N'},\mu_{N'}}\}$ with parameters $\{< \theta_1, \mu_1 >; < \theta_2, \mu_2 >; ...; < \theta_{N'}, \mu_{N'} >\}$.

5.1.2.2 Centralized Multi-AV Control Decision Learning with Decentralized Execution

For the multi-AV coexistence situation, MADM is considered as a centralized controller and takes the joint states of multiple AVs as its inputs to output a multi-AV control decision (i.e., control decision for each individual AV) at each time-stamp. And then, the multi-AV control decision will be executed on individual AVs in a decentralized way and each AV will execute its corresponding control decision individually. By this way, MADM belongs to a centralized decision learning but decentralized

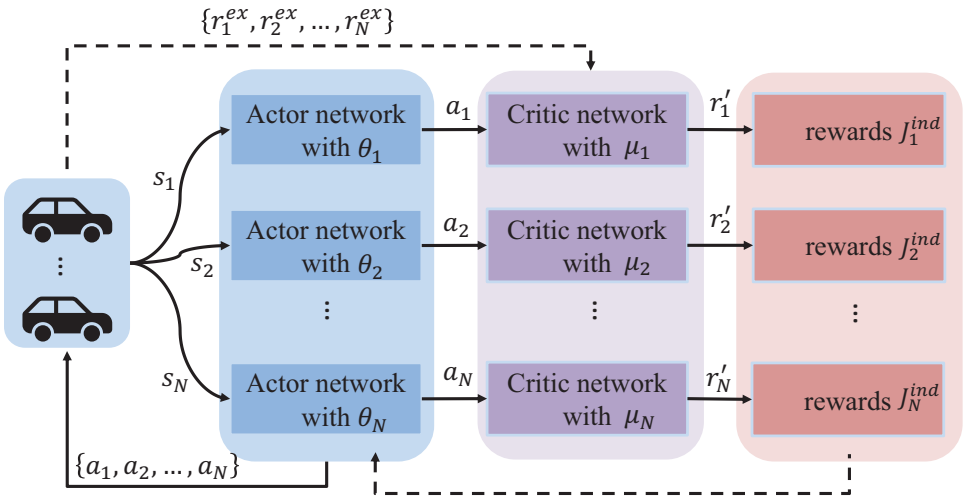


Figure 5.3: Policy adjustment process for multiple AVs in MADM.

execution system. Such a centralized decision learning with decentralized execution function is important since it not only allows MADM to make multi-AV control decisions for multiple AVs but also avoids large computing burdens on an individual AV because of decision making processes.

Figure 5.3 shows how MADM adjusts the policy of each AV in a centralized decision making learning with decentralized execution way. MADM uses π_{θ_i,μ_i} with $i \in [1, N]$ to indicate policies of N AVs. Driving states and driving actions of these N AVs are indicated as $< s_1, s_2, ..., s_N >$ and $< a_1, a_2, ..., a_N >$, respectively. For each AV, MADM adjusts its policy (containing an actor network with parameter θ_i and a critic network with parameter μ_i) to account for the presence of other AVs. Each AV is rewarded based on its local task and global task. Specifically, it will be assigned a large reward if it drives safely and has no unexpected behaviors after following its control decision. Otherwise, it will be punished and assigned with a small reward. Here, MADM adjust the original reward $r_i^{ex}(s^t, a^t)$ as $r_i'(s^t, a^t)$ to indicate the final reward the i^{th} AV receives when the AV executes driving action a^t at driving state s^t and an extrinsic reward $r_i^{ex}(s^t, a^t)$ to indicate the effect of the presence of other AVs. Therefore, $r_i'(s^t, a^t)$ should have a high relationship with extrinsic reward $r_i^{ex}(s^t, a^t)$

and initial reward $r_i(s^t, a^t)$. MADM calculates $r_i^{'}(s^t, a^t)$ of the i^{th} AV as follows:

$$r_i^{'}(s^t, a^t) = \lambda r_i(s^t, a^t) + (1 - \lambda) r_i^{ex}(s^t, a^t) \tag{5.3}$$

here, parameter λ is used to balance the effect of $r_i(s^t, a^t)$ and $r_i^{ex}(s^t, a^t)$. Based on $r_i^{'}(s^t, a^t)$ at time stamp t, MADM calculates the expected cumulative rewards $J_i^{ind}(\theta_i, \mu_i)$ of the i^{th} AV as $\mathbb{E}\sum_{m=0}^{\infty} \gamma^m r_i^{'}(s^{t+m}, a^{t+m})$, where s^{t+m} and a^{t+m} form a state-action pair of the i^{th} AV at the time-stamp $t + m$.

As shown in Figure 5.3, MADM uses $r_i^{ex}(s^t, a^t)$ to update parameter μ_i in the actor network of the i^{th} AV during the policy adjustment process. Besides, MADM applies $J_i^{ind}(\theta_i, \mu_i)$ to update parameter θ_i in the actor network of the i^{th} AV for its local task. MADM also calculates the total expected cumulative rewards $J^{mul}(\boldsymbol{\theta}, \boldsymbol{\mu})$ of total N AVs as $\sum_{i=1}^{N} J_i^{ind}(\theta_i, \mu_i)$ and uses it to update $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$ in all actor networks for considering the global task. Here, $\boldsymbol{\theta}$ indicates the actor network parameter set and $\boldsymbol{\mu}$ represents the critic network parameter set. Since $J^{mul}(\theta, \mu)$ highly depends on $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$, MADM forms the following parameter optimization problem to adjust the policy of each AV:

$$\begin{aligned} \arg\max_{\boldsymbol{\theta}, \boldsymbol{\mu}} \quad & J^{mul}(\boldsymbol{\theta}, \boldsymbol{\mu}) \\ \text{s.t.} \quad & \theta_i = \arg\max_{\theta_i} J_i^{ind}(\theta_i, \mu_i),\ i \in \{1, 2, ..., N\} \end{aligned} \tag{5.4}$$

optimal parameters in Equation (5.4) should not only maximize the expected cumulative rewards of each AV but also maximize $J^{mul}(\boldsymbol{\theta}, \boldsymbol{\mu})$ of N AVs. To satisfy the above requirements, MADM firstly maximizes $J_i^{ind}(\theta_i, \mu_i)$ by optimizing the parameter set $\boldsymbol{\theta}$ and then determines the optimal parameter $\boldsymbol{\mu}$ by maximizing the expected cumulative rewards $J^{mul}(\boldsymbol{\theta}, \boldsymbol{\mu})$ based on the optimal parameter set $\boldsymbol{\theta}$. Based on Equation (5.4), the optimal policy $\tilde{\pi}_{\boldsymbol{\theta}, \boldsymbol{\mu}}$ can consider both total expected cumulative

rewards of multiple AVs and the expected cumulative rewards of each individual AV simultaneously.

Here, we apply a policy gradient method [65] to determine optimal parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ in Equation (5.4). Algorithm 5 shows how to determine optimal parameters with the policy gradient method in MADM. MADM inputs the learned policies $\pi_{\boldsymbol{\theta},\boldsymbol{\mu}}$ of N AVs in Section 5.1.1 to update parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ with adjustment rates α_1 and α_2 until optimal parameter sets are obtained. Specifically, MADM firstly obtains driving state of each AV and makes a multi-AV control decision for N AVs based on their policies (*Line* 2). And then, MADM obtains driving states of N AVs to form state-action pairs $\{s^1, a^1, ..., s^N, a^N\}$ after the multi-AV control decision is executed on each AV (*Line* 3). Lastly, MADM will adjust two parameter sets with adjustment rates α_1 and α_2. Specifically, for the i^{th} AV, its parameter θ_i is updated as $\theta_i^{'} = \theta_i + \alpha \Delta_{\theta_i} J_i^{ex}(\theta, \mu)$. Similarly, the parameter $\mu_i^{'}$ will be updated as $\mu_i + \alpha_2 \Delta_{\mu_i} \theta_i^{'} \Delta_{\theta_i^{'}} J^{ex}(\theta^{'}, \mu)$. MADM will keep repeating this parameter adjustment process until the termination condition is satisfied. Here, we the define a termination condition as follows: Euclidean distance $d_{\boldsymbol{\theta}}$ between $\{\theta_1, ..., \theta_N\}$ and $\{\theta_1^{'}, ..., \theta_N^{'}\}$ should be less than $\Gamma_{\boldsymbol{\theta}}$ and Euclidean distance $d_{\boldsymbol{\mu}}$ between $\{\mu_1, ..., \mu_N\}$ and $\{\mu_1^{'}, ..., \mu_N^{'}\}$ should be less than $\Gamma_{\boldsymbol{\mu}}$. More details of the parameter gradient method can be found in reference [65].

Algorithm 5: The optimal parameter determination process with the policy gradient method in MADM

Input: The learned policies $\pi_{\boldsymbol{\theta},\boldsymbol{\mu}}$ of N AVs with parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$.
Output: Optimal parameter sets $\{\theta_1, ..., \theta_N\}$ and $\{\mu_1, ..., \mu_N\}$.
1 **while** *Termination condition is not satisfied* **do**
2 Make multi-AV control decisions for multiple AVs based on their driving states;
3 Execute multi-AV control decisions on N AVs and generate a set of state-action pairs $\{s^1, a^1, ..., s^N, a^N\}$;
4 Update parameter set $\{\theta_1, ..., \theta_N\}$ with α_1 as $\{\theta_1^{'}, ..., \theta_N^{'}\}$;
5 Update parameter set $\{\mu_1, ..., \mu_N\}$ with α_2 as $\{\mu_1^{'}, ..., \mu_N^{'}\}$;
6 Calculate $d_{\boldsymbol{\theta}}$ and $d_{\boldsymbol{\mu}}$ and compare them with $\Gamma_{\boldsymbol{\theta}}$ and $\Gamma_{\boldsymbol{\mu}}$ to check whether the termination condition is satisfied;
7 **end**

5.1.3 Multi-AV Control Decision Making with Safety Guarantee

In this section, we will discuss how MADM makes multi-AV control decisions with providing a safety guarantee function for multiple AVs based on the adjusted policies. Here, N AVs are considered as a whole system and we use $\boldsymbol{s}^{t+1} = g(\boldsymbol{s}^t, \pi_{\theta_1,\mu_1}, \pi_{\theta_2,\mu_2}, ..., \pi_{\theta_N,\mu_N})$ to denote its dynamics process, where $\boldsymbol{s}^t = [s_1^t, ..., s_N^t]$ represents driving states of N AVs at time-stamp t. For the i^{th} AV, its driving state s_i is described as $(x_i, y_i, v_i, \varphi_i)$ and its driving action a_i is indicated with $(acc_i, steer_i)$. Here, x_i and y_i indicate latitude position and longitude position, v_i and φ_i represent driving speed and heading direction, acc_i and $steer_i$ represent acceleration value and steering angle. For such a system running for total T time-stamps, its driving trajectory with an initial driving state $\boldsymbol{s}^1$ can be described as $(\boldsymbol{s}^1, \boldsymbol{s}^2, ..., \boldsymbol{s}^T)$.

5.1.3.1 Safe Driving State Definition

In MADM, an AV is considered to be in a safe driving state if it has no collisions with other AVs when driving on the same road. Specifically, safe driving state of the i^{th} AV should satisfy the following constraint: $f(s_i, s_j) > 0$ for $j \in [1, N]$ and $j \neq i$, where $f(s_i, s_j)$ indicates the safety index of the i^{th} AV relevant to the j^{th} AV. The safety index $f(s_i, s_j)$ should consider both driving position and driving speed aspects and we define $f(s_i, s_j)$ as follows:

$$f(s_i, s_j) = d(s_i, s_j) - \kappa \dot{d}(s_i, s_j) - D \tag{5.5}$$

where κ is a positive constant factor; D represents safe vehicle distance between two AVs and equals to a constant value; $d(s_i, s_j)$ indicates vehicle distance between the i^{th} AV and the j^{th} AV and can be calculated as $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2)}$. $\dot{d}(s_i, s_j)$

indicates driving speed difference between the i^{th} AV and the j^{th} AV and equals to $v_i - v_j$. Based on Equation (5.5), we see that both vehicle distance and speed difference affect $f(s_i, s_j)$. If vehicle distance decreases or speed difference increases, $f(s_i, s_j)$ will become smaller, which means that the i^{th} AV will have a less possibility of being in a safe driving state. Therefore, the safety index $f(s_i, s_j)$ should be positive if the i^{th} AV is in a safe state. For the i^{th} AV, all driving states with positive safety index values are considered as safe driving states and these safe driving states are combined together to form a safe driving state space indicated as χ_i. Similarly, the system is considered as safe if N AVs are all in their safe driving states and driving state of each AV should be always in its own safe driving state space in T time stamps. Therefore, the safe driving state space of N AVs can be described as $\boldsymbol{\chi} = \chi_1 \bigcap \chi_2 \bigcap ... \bigcap \chi_N$. For N AVs with a driving trajectory $(\boldsymbol{s}^1, \boldsymbol{s}^2, ..., \boldsymbol{s}^T)$, its driving state is considered as in a safe driving state if $\boldsymbol{s}^t \in \boldsymbol{\chi}$ when t changes from 1 to N.

5.1.3.2 Safety Guarantee

Based on Section 5.1.2, MADM adjusts the learned policy into the optimal policy $\tilde{\pi}_{\boldsymbol{\theta},\boldsymbol{\mu}}$ so that $\tilde{\pi}_{\boldsymbol{\theta},\boldsymbol{\mu}}$ can be used to make decisions for multiple AVs. However, $\tilde{\pi}_{\boldsymbol{\theta},\boldsymbol{\mu}}$ is formed based on the simulation and cannot fully guarantee driving safety of multiple AVs by considering that the driving environment in the simulation is not always the same as the real situation. To provide safety guarantee for total N AVs, we need to figure out a backup policy π_{back} for MADM, which can help each AV reach into a safe driving state. Suppose a system with total N AVs starts with initial driving state $\boldsymbol{s}^1 \in \chi$, MADM needs to decide whether the system should make a multi-AV control decision with π_{back}. If $s^{i+1} = g(s^i, \tilde{\pi}_{\boldsymbol{\theta},\boldsymbol{\mu}}) \in \chi$, it is safe to continue making a multi-AV control decision with $\tilde{\pi}_{\boldsymbol{\theta},\boldsymbol{\mu}}$. Otherwise, MADM will switch to π_{back} for making a decision.

Backup policy. Backup policy π_{back} can transmit the system with N AVs from any

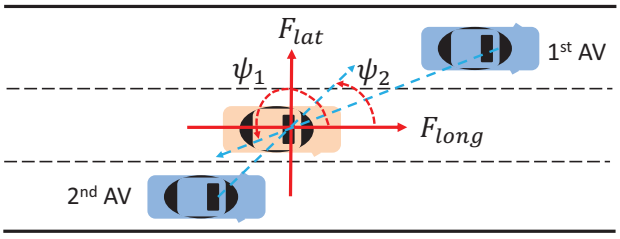


Figure 5.4: Repulsive potential forces of nearby AVs on the ego AV.

driving state $\boldsymbol{s^t}$ to a safe driving state $\boldsymbol{s}^{t+1} \in \chi$. One direct way is to manually design a backup policy for an AV, which adjusts its driving speed and steering angle until its all safety index values become positive. Specifically, for the i^{th} AV, its backup policy π^i_{back} adjusts its driving speed and steering angle until all safety index values between the i^{th} AV and other $N-1$ AVs are positive.

Here, we use the repulsive potential field method [43] to form a backup policy for an AV. The repulsive potential field method is a popular method used for planning robotic paths and its goal is to control a robot to reach a destination without collisions with other obstacles. When the repulsive potential field method is applied into MADM, MADM considers other AVs as obstacles and calculates repulsive potential fields between the ego AV and other AVs for driving speed and steering angle adjustments. Figure 5.4 shows repulsive potential fields existing between the ego AV and its two nearby AVs. We see that the repulsive potential filed has a high relationship between AV positions. Here, the repulsive potential field $U^i_{s_i \to s_j}$ of the j^{th} AV on the i^{th} AV can be calculated as:

$$U^i_{s_i \to s_j} = \begin{cases} \epsilon(\frac{1}{D} - \frac{1}{d(s_i,s_j)}), & \text{if } d(s_i,s_j) \leq D \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

where ϵ is a positive constant factor showing the effect of AV position on repulsive potential field. $d(s_i, s_j)$ indicates AV distance between the i^{th} AV and the j^{th} AV. D indicates the safety vehicle distance between two AVs. Based on Equation (5.6), we

see that $U^i_{s_i \to s_j}$ is positive as $d(s_i, s_j)$ is larger than D. Otherwise, $U^i_{s_i \to s_j}$ becomes zero. Based on the repulsive potential field $U^i_{s_i \to s_j}$, we calculate its derivative value to obtain repulsive potential force as $\dot{U}^j_{s_i \to s_j}$. As shown in Figure 5.4, MADM calculates total repulsive potential forces from all other AVs on the i^{th} AV and divides them into longitudinal and lateral directions:

$$F_{long} = \sum_{j=1 \ and \ j \neq i}^{N} |\dot{U}^i_{s_i \to s_j}| \sin(\psi_j) \tag{5.7a}$$

$$F_{lat} = \sum_{j=1 \ and \ j \neq i}^{N} |\dot{U}^i_{s_i \to s_j}| \cos(\psi_j) \tag{5.7b}$$

where F_{long} and F_{lat} indicate total repulsive potential forces in longitudinal and lateral directions, respectively. ψ_1 or ψ_2 represents repulsive potential force direction from the 1^{st} AV or the 2^{nd} AV relevant to longitudinal direction. Based on F_{long} and F_{lat} of the i^{th} AV, MADM adjusts its driving speed and steering angle to ensure the i^{th} AV drive in a safe driving state. The i^{th} AV is expected to accelerate or decelerate if F_{long} is a positive or negative value. Otherwise, the i^{th} AV is expected to keep driving with a constant speed. Similarly, The i^{th} AV is expected to make a left lane change or a right lane change if F_{lat} is a positive or negative value. Otherwise, the i^{th} AV is expected to stay on the same lane. By calculating the repulsive potential force, MADM forms the backup policy π^i_{back} for the i^{th} AV and applies π^i_{back} to adjust its driving speed and steering angle to ensure that its safety index values are positive, which means that the i^{th} AV stays in a safe driving state. Similarly, MADM does the same repulsive potential force calculation processes on other $N-1$ AVs to form their backup policies. By this way, we can form backup policies $\pi_{back} = (\pi^1_{back}, \pi^2_{back}, ..., \pi^N_{back})$ for total N AVs.

5.1.3.3 Decision Making with Safety Guarantee

To make multi-AV control decisions for total N AVs with safety guarantee, MADM selects optimal policy or backup policy so that each AV after executing the multi-AV control decision will still stay in a safe driving state. Here, MADM does not keep using backup policy to make multi-AV control decisions mainly because backup policy can only ensure each AV stay in a safe driving safety without maximizing their expected accumulative rewards. Instead of keeping using N backup policies for N AVs, MADM considers all possible combinations of optimal policies and backup policies. Here, we use a set of policy indexes $\mathbf{b} = \{b_1, b_2, ..., b_N\}$ to indicate the final selected policy type. Specifically, b_i indicates the selected policy type for the i^{th} AV. $b_i = 1$ means that optimal policy $\tilde{\pi}_{\theta_i,\mu_i}$ is selected to make a control decision while $b_i = 0$ represents that MADM selects backup policy π^i_{back}. The goal of MADM is to maximize the number of the selected optimal policies so that MADM can obtain higher expected cumulative rewards with providing safety guarantee. Here, we form the following equation for MADM to determine optimal policy indexes:

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \sum_{i=1}^{N} |b_i| \digamma(s_i, \chi_i) \tag{5.8}$$

where $|b_i|$ indicates the absolute value of policy index b_i and $\digamma(s_i, \chi_i)$ indicates whether driving state s_i of the i^{th} AV stays in the safe driving state space χ_i. $\digamma(s_i, \chi_i)$ equals to 1 if $s_i \in \chi_i$. Otherwise, $\digamma(s_i, \chi_i)$ equals to 0.

For total N AVs, iterating over all possible different combinations of optimal policies and backup policies to determine optimal policy indexes can be computation consuming by considering that the total number of different combinations changes in an exponential growth $\mathbf{b} \in 2^N$. Here, we propose a local searching method to determine the optimal policy indexes $\mathbf{b}^*$. Algorithm 6 shows how to determine optimal policy indexes through a local searching method in MADM. The local searching

Algorithm 6: The optimal policy index determination process in MADM

Input: Safe driving state spaces $\{\chi_1, \chi_2, ..., \chi_N\}$ of N AVs are known.
Output: Optimal policy indexes $\mathbf{b}^* = \{b_1^*, ..., b_N^*\}$.

1 Initialize optimal policy indexes as $\mathbf{b}^* = \{1, ..., 1\}$ and hence the number k of zero policy indexes in $\mathbf{b}^*$ equals to 0;
2 $weight\_index\_number$=$\sum_{i=1}^{N} |b_i^*| F(s_i, \chi_i)$;
3 **while** *Termination criterion is not reached* **do**
4 k=k+1;
5 **while** $m \leq \binom{k}{N}$ **do**
6 Rebuild a new policy index sequence $\boldsymbol{b}^m$ with m zero policy indexes;
7 **if** $weight\_index\_number \leq \sum_{i=1}^{N} |b_i^m| F(s_i, \chi_i)$ **then**
8 $weight\_index\_number$=$\sum_{i=1}^{N} |b_i^m| F(s_i, \chi_i)$;
9 **end**
10 Update optimal policy index sequence $\mathbf{b}^*$ as $\boldsymbol{b}^m$;
11 m=m+1;
12 **end**
13 Calculate Euclidean distance $d_{\boldsymbol{b}}$ and compare it with $\Gamma_{\boldsymbol{b}}$ to check whether the termination condition is satisfied;
14 **end**
15 Output $\mathbf{b}^*$ to MADM to make a multiple-AV control decision;

method begins from an initial policy index sequence $\{1, 1, ..., 1\}$ and calculates the value in Equation (5.8) as $weight\_index\_number$ (Lines $1-2$). And then, the local searching method goes through a sequence reconstruction loop to determine optimal policy indexes (Lines $3-13$). Specifically, for situations with k zero policy indexes in the sequence, the local searching method firstly forms different new sequences and calculates its corresponding $weight_index_number$s to choose the sequence $\boldsymbol{b}^m$ with the maximum value of $weight\_index\_number$. And then the local searching method will updates $\mathbf{b}^*$ as $\boldsymbol{b}^m$ (Lines $5-12$). The local searching method will keep repeating the above process on situations with different zero policy indexes until the termination condition is satisfied. Here, we the define the termination condition as follows: Euclidean distance $d_{\boldsymbol{b}}$ between $\mathbf{b}^*$ and $\boldsymbol{b}^m$ should be less than $\Gamma_{\boldsymbol{b}}$. By this way, MADM can find the optimal policy index sequence to determine whether a learned policy or a backup policy should be selected for each AV to make a decision.

5.2 Performance Evaluation

We implemented MADM by running MatLab on one laptop (Intel i5 CPU and 16 gigabyte memory) and used the real-world traffic dataset [50] as the expert driving data to train MADM and test its performance in learning expert control behaviors and making multi-AV control decisions.

5.2.1 Experiment Settings

Expert Driving Dataset: The expert driving dataset [50] contains total $1,048,575$ vehicle trajectory samples from $3,366$ vehicles. These vehicle trajectories are collected with a sampling frequency of 10 $times/second$ on the Hollywood Freeway in Los Angeles at three time periods (7:50-8:05 a.m., 8:05-8:20 a.m. and 8:20-8:35 a.m.) at different days. Specifically, the driving road is approximately around $640m$ long and has total five mainline lanes throughout this road section. The vehicle trajectory sample of each vehicle includes trajectory information (vehicle latitude, vehicle longitude, specific lane ID where a vehicle drives, driving speed, acceleration or deceleration value). Based on trajectory information, we can easily figure out the total number of vehicles in a road section and their relevant positions. Here, we clustered vehicle trajectories with similar driving behaviors by considering driving speed, maximum acceleration/deceleration values and lane change frequency and selected the cluster with the maximum number of vehicle trajectories as the expert driving data so that the expert driving data has enough vehicle trajectories for modeling training.

Comparison Methods: We implemented total four different optimal control decision methods for optimal control decision making performance comparisons. These four methods include a reinforcement learning based control decision system (RLS) [15], an imitation learning based control decision system (ILS) [30], a multi-agent RL based control decision system (MRLS) [56] and MADM without considering safety

guarantee (MADM-NSG). RLS makes optimal control decisions for an AV by forming a value network and a policy network with deep neural networks and using the formed networks to select optimal control decisions with the maximum cumulative expected rewards. ILS learns control behaviors of an expert based on its driving data and makes control decisions by imitating the learned control behaviors. Since RLS and ILS focus on making optimal control decision for an individual AV, more RLS or ILS are needed in multiple AV situations so that each AV will receive its control decision from its corresponding RLS or ILS simultaneously. MRLS develops a multi-agent connected autonomous driving platform to simulate multi-AV coexistence driving environments and then trains a multi-agent reinforcement learning model based on the driving platform to form control policies through reinforcement learning so that the trained multi-agent reinforcement learning model can make control decisions for multiple AVs simultaneously. Here, both MRLS and MADM-NSG do not consider safety guarantee during their control decision making processes. Compared with M-RLS, MADM-NSG combines imitation learning and reinforcement learning together to form policies to learn driving behaviors of an expert based on the expert driving data.

Evaluation Aspects: In this experiment section, for the control behavior learning part, we used MADM, RLS and ILS to learn control behaviors of an expert based on the expert driving data under different traffic volume levels and showed their control behavior learning performances in terms of the normalized reward value. Here, the expert driving data is divided into three different levels (low, medium and high) based on traffic volume (vehicles/hour) and their corresponding traffic volume ranges equal to $[0, 840]$, $[841, 1080]$ and $[1081, +\infty]$, respectively. The normalized reward value indicates the rate of the expected cumulative rewards received by MADM, RLS or ILS over the maximum value of their expected cumulative rewards during their whole

model training processes. For the optimal control decision making part, we used MADM, RLS, ILS, MRLS and MADM-NSG to make optimal control decisions for multiple AVs under driving situations with a high traffic volume level and introduced emergency rate and average trip speed to evaluate their optimal control decision making performance. Here, emergency rate is used as a driving safety criterion and indicates the percentage of simulations, where one or more AVs have an offroad driving behavior, a collision or a hard brake event, over total 1000 simulations. Here, an offroad driving behavior means that an AV drives off the road shoulder during the driving process. A collision means that the AV has a collision with its nearby AVs. A hard brake event indicates that an AV brakes with more than a certain deceleration value (3 m/s^2) during its driving process. Average trip speed is used as a driving efficiency criterion and represents average driving speed of multiple AVs during the whole trip when they follow multi-AV control decisions. For an AV with an offroad driving behavior, a collision or a hard brake event, its driving speed is considered to be zero during its remaining trip. Lastly, we introduced average computation time per each update (TPU) to indicate computation load of a control decision making process and calculated it as average computation time of all control decision making processes in the whole trip.

5.2.2 Evaluation Results

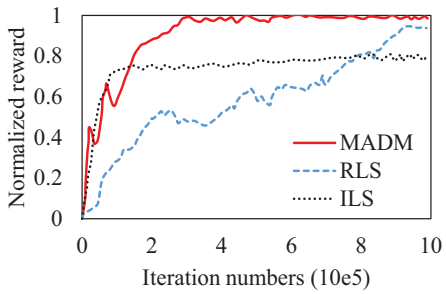
We firstly used MADM, ILS and RLS to learn control behaviors of an expert to evaluate control behavior learning performance. Then, we applied the trained MADM, ILS, RLS, MARS and MADM-NSG to make multi-AV control decisions and calculated their emergency rates, average trip speeds, and TPU for optimal control decision making performance evaluation.

5.2.2.1 Control behavior learning performance

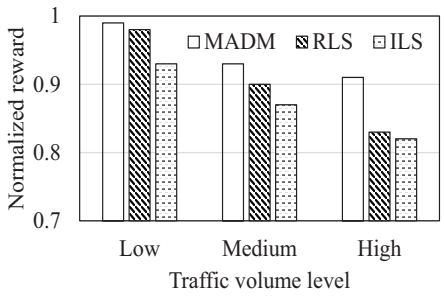
Based on the expert driving data, we trained MADM, ILS and RLS and calculated their normalized reward values during their training processes for control behavior learning performance evaluation. Figure 5.5(a) compares the normalized reward values between MADM, ILS and RLS as the number of training iterations increases from 1 to 1e6. Compared with ILS, the maximum normalized reward values of MADM and RLS are higher mainly because they used the reinforcement learning technology to form control policies and have no sub-optimal policy formation problem during the training process. We also see that the normalized reward values of MADM and ILS increase much faster than RLS when the number of training iterations is less than 1e6. This is because imitation learning in MADM and ILS forms control policies by imitating driving behaviors of an expert based on the expert driving data so that it can quickly achieve good driving performance as the expert, which helps to increase control behavior learning speed.

Vehicle trajectories in the expert driving data also contain nearby vehicle information, vehicle trajectories collected under different traffic volumes may affect control behavior learning performance. Here, we also used vehicle trajectories under different traffic volume levels to train MADM, ILS and RLS and compared their control behavior learning performance in Figure 5.5(b). We see that MADM has higher normalized reward values than ILS and RLS for all three situations, which demonstrates that MADM has better control behavior learning performance. Besides, the normalized reward values of MADM, ILS and RLS will become smaller as the number of traffic volume becomes larger. This is reasonable because more nearby vehicles will stay around an AV and there exist more possible different driving situations during the model training process, which causes low control behavior learning performance. Based on the above analysis, we can conclude that MADM can learn control behaviors

more accurately and efficiently.



(a) Normalized reward value comparisons as the number of training iterations increases.



(b) Normalized reward value comparisons when trained with vehicle trajectories under different traffic volume levels.

Figure 5.5: Control behavior learning performance comparisons between MADS, RLS and ILS as the number of training iterations increases or traffic volume changes with different levels.

5.2.2.2 Control decision making performance

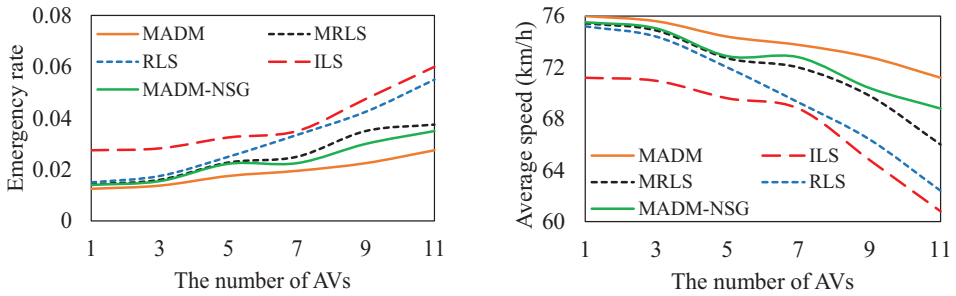
We trained MADM, MRLS, RLS, ILS and MADM-NSG based on the expert driving data for total 1e6 iterations so that they can achieve their best performance for making control decision processes. And then, we used the above trained methods to make control decisions for one or more AVs with a high traffic volume level and calculated their emergency rates and average trip speeds for control decisions making performance evaluation.

Under different numbers of AVs: Figure 5.6(a) compares emergency rates between MADM, MRLS, RLS, ILS and MADM-NSG as different numbers of AVs finish a trip with 10 km trip distance. Compared with other methods, MADM has smaller emergency rates and the maximum value of its emergency rates reaches around 97% when the number of AVs equals to 11, which means that MADM can help 11 AVs finish the whole trip successfully in 97% of total 1000 simulations. Besides, for MADM, MRLS, RLS, ILS and MADM-NSG, their emergency rates become larger as the

number of AVs increases. This is because each multi-AV control decision contains a set of control decisions corresponding to multiple AVs and it will cause an offroad driving behavior, a collision or a hard brake event if the multi-AV control decision fail to ensure one or more of multiple AVs driving safety. Here, emergency rates of MADM, MRLS and MADM increase slower than RLS or ILS mainly because they consider joint states of multiple AVs as inputs to making control decisions for multiple AVs simultaneously to ensure driving safety. Compared with MRLS and MADM-NSG, MADM also considers safety guarantee during its control decision making process and its control decisions can help to ensure each AV always drive in a safe driving state, which explains why MADM has the smallest emergency rates.

Figure 5.6(b) shows average trip speed comparisons between MADM, MRLS, RLS, ILS and MADM-NSG as the number of AVs increases from 1 to 11. We see that their average trip speeds become smaller as more AVs share the same road. This is reasonable since their emergency rates becomes larger with AV number increase and an AV with high emergency rates usually has a lower average trip speed. Since MADM has the smallest emergency rates for all situations with different AV numbers, it average trip speed is always larger compared with values of MRLS, RLS, ILS and MADM-NSG and its minimum value reaches around 72 km/h, which is still much larger than values of MRLS, RLS, ILS and MADM-NSG (66 km/h, 62 km/h, 61 km/h, 69 km/h). Therefore, we conclude that MADM can make control decisions for multiple AVs with smaller emergency rates and larger average trip speeds even when the number of AVs increases to 11.

Under different trip distances: Here, we also explored the effect of trip distances on control decision making performance and analyzed how emergency rate and average trip speed change as total 11 AVs finish different trip distances in the simulation. Figure 5.7(a) shows emergency rate comparisons between MADM, MRLS, RLS, ILS and MADM-NSG as the trip distance increases from 10 km to 60 km. We see that

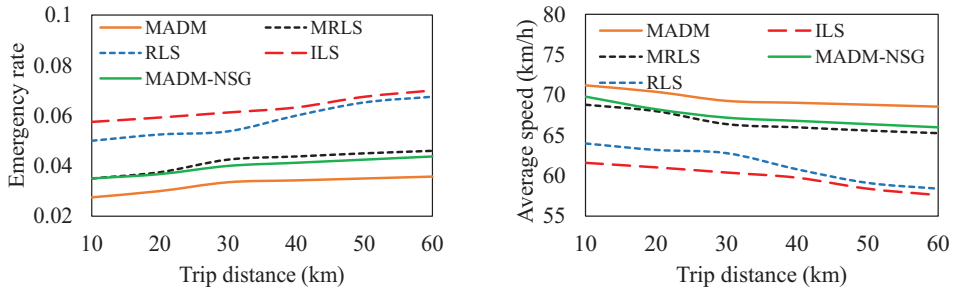


(a) Emergency rates as the number of AVs increases.

(b) Average trip speeds as the number of AVs increases.

Figure 5.6: Emergency rate and average trip speed comparisons between MADM and existing methods as the number of AVs increases from 1 to 11.

MADM has the smallest emergency rate than other methods and its maximum emergency rate becomes 3.6%, which is smaller than values of MRLS, RLS, ILS and MADM-NSG (4.6%, 7.0%, 7.2% and 4.4%). Besides, emergency rates of MADM, MRLS, RLS, ILS and MADM-NSG keep increasing slowly as trip distance changes from 10 km to 60 km, which demonstrates that trip distance has a small effect on emergency rates.

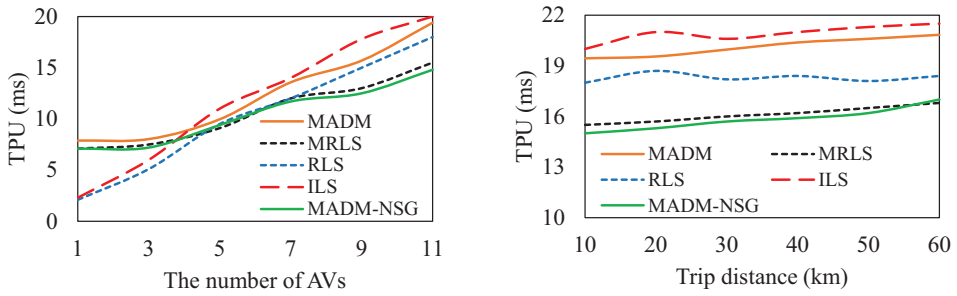


(a) Collision existence rates as trip distance increases.

(b) Average trip speeds as trip distance increases.

Figure 5.7: Emergency rate and average trip speed comparisons between MADM and existing methods as the trip distance increases from 10 km to 60 km.

Figure 5.7(a) compares average trip speeds between MADM, MRLS, RLS, ILS and MADM-NSG as trip distance increases from 10 km to 60 km. Similarly, average trip speeds of MADM, MRLS, RLS, ILS and MADM-NSG decrease with a small change as trip distance increases. For MADM, its average trip speed is larger than average



(a) TPU comparisons as the number of AVs increases.

(b) TPU comparisons as trip distance increases.

Figure 5.8: TPU comparisons between MADM and other methods as the number of AVs or trip distance increases.

trip speeds of other methods and its range equals to 2.7 km/h, which is smaller than ranges of MRLS, RLS, ILS and MADM-NSG (3.6 km/h, 5.6 km/h, 4.1 km/h and 3.8 km/h). Based on emergency rate and average trip speeds in Figure 5.7, we conclude that MADM keeps stable control decision making performance and trip difference has little effects on it.

5.2.2.3 Computation cost analysis

We implemented MADM, MRLS, RLS, ILS and MADM-NSG by running MATLAB on one laptop mentioned above and compared their TPUs during their control decision making processes for computation load evaluation. Figure 5.8 shows how their TPUs change as the number of AVs or trip distance increases. Based on Figure 5.8(a), we see that TPUs of MADM, MRLS, RLS, ILS and MADM-NSG keep increasing as more AVs need to be considered during the control decision making process. Specifically, their GPUs reaches the maximum value as the number of AVs increases to 11 and their maximum values equal to 19.4 ms, 15.5 ms, 18.1 ms, 20.0 ms and 14.8 ms, respectively. Figure 5.8(b) shows TPUs of MADM, MRLS, RLS, ILS and MADM-NSG as trip distance increases. Overall, GPUs of all methods become larger but increase in a small range. In other words, computation load during a control decision

making process keeps almost constant for even when trip distance changes and their maximum TPUs equal to 20.8 ms, 16.8 ms, 18.4 ms, 21.5 ms and 17.0 ms, which are acceptable for AVs in practice.

Chapter 6

Summary and Future Work

In this chapter, we first summarize battery authentication, control policy based driving safety system and multi-AV control decision making system presented in this dissertation. Next, we discuss the possible impacts of such methodologies on improving driving safety. We also present the limitations of our methodologies. We conclude this dissertation by outlining the future research directions we would like to focus on in the near future.

6.1 Summary

6.1.1 Battery Authentication System

- We analyze the feasibility of launching the two EV battery attacks through a smartphone and how to implement the two attacks in practice, which has not been studied in previous works.
- We firstly propose a data-driven behavior model to learn a user's habits of turning on AC and stopping battery charging process based on vehicle usage data. We apply the random forest to identify users based on battery state to improve the behavior model's learning accuracy. Then, we use the behavior

model to train a reinforcement learning model, which decides the action that mostly conform to the user's actual action and thus can detect the battery attacks from a smartphone.

- We conducted the first battery attack experiment to show the feasibility of such attacks on EVs. To verify the attack detection performance of Bauth, we conducted real-world experiments using total 7 EVs in daily driving and compared attack detection results between Bauth and the statistical method. The experimental results demonstrate that the accuracy of the random forest based user identification method is around 95.7% and Bauth authorizes action requests from a smartphone with accuracy as high as 95.6%.

6.1.2 Control Policy based Driving Safety System

- We propose a control policy extraction method to extract control policies of a target AV based on its historical driving data. Specifically, the method firstly uses dynamic time warping and k-means clustering technologies to cluster historical driving data with the same control behavior type together and then analyzes positions and driving speeds in each cluster to extract control policies of a target AV.

- We propose an optimal control policy determination method to determine an optimal control behavior for a given trigger condition for a target AV considering time-varying driving state of its nearby vehicle. This method firstly predicts the driving state of its nearby vehicle and then applies its driving state prediction result into the proposed optimal control policy determination method to determine an optimal control policy which maximizes the sum of relative distances between the target AV and its nearby vehicle in the subsequent time stamps.

- We used an industry-standard AV platform (Baidu Apollo) to evaluate the control policy extraction and optimal control policy determination performance of Polsa in comparison with with other two state-of-the-art methods[28, 40]. The comparison results show that Polsa can extract control policies with 83% accuracy, and improve optimal control policy success rate by 28% compared with existing methods.

6.1.3 Multi-AV Control Decision Making System

- We propose a policy formation method to form policies for an individual AV. By taking the advantage of expert's driving data, the policy formation method in MADM combines imitation learning and reinforcement learning together to specify state and action spaces which are worth exploring by reinforcement learning so that MADM can learn policies efficiently and achieve good control performance as the expert.

- We develop a multi-agent reinforcement learning method to make control decisions with safety guarantee for multiple AVs in multi-AV coexistence driving situations. Specifically, MADM uses a multi-agent reinforcement learning model to adjust the learned policy of each individual AV so that these policies can work together to make control decisions. Lastly, MADM develops a safety guarantee method to form backup policies and apply them during the decision making process to guarantee driving safety of each AV.

- We use a real-world traffic dataset from the United States Department of Transportation Federal Highway Administration [50] to evaluate optimal control decision making performance of MADM in comparison with the state-of-the-art methods [15, 30, 56]. The experimental results demonstrate that MADM reduces its emergency rate by as high as 51% compared with existing methods.

6.1.4 Limitations

One obvious limitation of our proposed methodologies is that we verified the optimal control decision making performance of the control policy based driving safety system and the multi-AV control decision making system based on Baidu Apollo simulation platform. Though the Baidu Apollo simulation platform can simulate vehicle driving environments based on real-world traffic driving data, responses of nearby vehicles do not always match the real situations because of different users' driving habits, which makes the performance evaluation results less convincing.

Another limitation of our proposed methodologies is that we used machine learning technologies (e.g., neural network, random forest and reinforcement learning) into the proposed methodologies to predict vehicle state of nearby vehicle, identify EV users, and make control decision, which results in computational overhead problems.

6.2 Future Work

Research is a continuous process, and there is always room for improvement in terms of theories and experimental evaluations. In this dissertation, we have presented three data-driven driving safety approaches that are more effective compared with the other existing approaches. However, there are certain limitations in our proposed methodologies, as we discussed in Section 6.1.4. Therefore, these limitations can lead us to future research directions.

Here, we list some directions for future research works.

- In the future, we will consider other user information (e.g., the number of passengers) during the action request authorization process of the battery authentication system presented in this dissertation. Different numbers of passengers usually have different total vehicle weights and hence affect battery energy con-

sumption greatly. We can consider this kind of factor to improve the authentication performance of the proposed methodology presented in this dissertation.

- In this proposed control policy based driving safety system, we only consider the vehicle, which stays nearest to the target AV, during the control policy selection process and other vehicles driving in the same road may have effects on the target AV in practice. We can consider situations with more than one nearby vehicles to make control policy selection outputs from the proposed system more reasonable.

- The proposed multi-AV control decision making system considers multi-AV situations by assuming that the number of multiple AVs keep constant during its whole control decision making process. However, vehicles may drive on or off the road randomly in practice and our method will focus on this kind of situations to make its control decision making results convincing.

Bibliography

[1] S Abada, G. Marlair, A Lecocq, M Petit, V. Sauvant-Moynot, and F. Huet. "Safety focused modeling of lithium-ion batteries: A review". In: *J. Power Sources* 306 (2016).

[2] N Abdennour, T Ouni, and N. B. Amor. "Driver identification using only the CAN-Bus vehicle data through an RCN deep learning approach". In: *Robotics and Autonomous Systems* 136 (2021), p. 103707.

[3] S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi. "Droidnative: Semantic-based detection of android native code malware". In: *arXiv preprint* (2016).

[4] *Apollo Simulation Platform*. `http://apollo.auto/synthetic.html`. 2019.

[5] J. Basl and J. Behrends. "Why Everyone Has It Wrong About the Ethics of Autonomous Vehicles". In: *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2019 Symposium*. 2020.

[6] K. Ç. Bayindir, M. A. Gözüküçük, and A. Teke. "A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units". In: *Energy Convers. Manag.* 52.2 (2011).

[7] C. Berger and M. Dukaczewski. "Comparison of architectural design decisions for resource-constrained self-driving cars - A multiple case-study". In: *Informatik 2014*. 2014.

[8] J.-F. Bonnefon, A. Shariff, and I. Rahwan. “Autonomous vehicles need experimental ethics: Are we ready for utilitarian cars”. In: *arXiv preprint* (2015).

[9] M. W. Browne. “Cross-validation methods”. In: *J. Math. Psycho.* 44.1 (2000).

[10] T. K. Buennemeyer, M. Gora, R. C. Marchany, and J. G. Tront. “Battery exhaustion attack detection with small handheld mobile computers”. In: *Proc. of Portable Information Devices*. 2007.

[11] L. Caviglione and A. Merlo. “The energy impact of security mechanisms in modern mobile devices”. In: *Network Security* 2012.2 (2012).

[12] A. Chatham. “Google’s self-driving cars: the technology, capabilities, and challenges”. In: *Proc. of Embedded Linux* (2013).

[13] D. Chelidze. “Reliable estimation of minimum embedding dimension through statistical analysis of nearest neighbors”. In: *Journal of Computational and Nonlinear Dynamics* 12.5 (2017).

[14] J. Chen, Z. Wang, and M. Tomizuka. “Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1239–1244.

[15] J. Chen, B. Yuan, and M. Tomizuka. “Model-free deep reinforcement learning for urban autonomous driving”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 2765–2771.

[16] J. Chen and T. Gonsalves. “Autonomous highway car following system based on fuzzy control”. In: *Proc. of HPCCTC*. 2018, pp. 98–101.

[17] Q. Chen, S. Tang, Q. Yang, and S. Fu. “Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds”. In: *Proc. of ICDCS*. 2019, pp. 514–524.

[18] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng. "Brain-Inspired Cognitive Model WithAttention for Self-Driving Cars". In: *IEEE TCDS* 11.1 (2019).

[19] S. Choi, J. Kim, D. Kwak, P. Angkititrakul, and J. H. Hansen. "Analysis and classification of driver behavior using in-vehicle can-bus information". In: *Proc. of DSP*. 2007.

[20] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. "End-to-end driving via conditional imitation learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 4693–4700.

[21] L. Davi, A. Dmitrienko, M. Egele, T. Fischer, T. Holz, R. Hund, S. Nürnberger, and A.-R. Sadeghi. "MoCFI: A Framework to Mitigate Control-Flow Attacks on Smartphones." In: *Proc. of NDSS*. Vol. 26. 2012.

[22] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao. "LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning." In: *NeurIPS*. 2019, pp. 4405–4416.

[23] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. "PiOS: Detecting Privacy Leaks in iOS Applications." In: *Proc. of NDSS*. 2011.

[24] W. Enck, D. Octeau, P. D. McDaniel, and S. Chaudhuri. "A study of android application security." In: *Proc. of USENIX Security Symposium*. Vol. 2. 2011.

[25] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones". In: *ACM TOCS* 32.2 (2014).

[26] R. Fei, S. Li, X. Hei, Q. Xu, F. Liu, and B. Hu. "The Driver Time Memory Car-Following Model Simulating in Apollo Platform with GRU and Real Road Traffic Data". In: *Mathematical Problems in Engineering* 2020 (2020).

[27] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. "Counterfactual multi-agent policy gradients". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[28] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson. "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment". In: *Autonomous Robots* 41.6 (2017).

[29] K. Ghanem. "Towards more accurate clustering method by using dynamic time warping". In: *arXiv preprint* (2013).

[30] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, et al. "Urban driving with conditional imitation learning". In: *Proc. of ICRA*. IEEE. 2020, pp. 251–257.

[31] H. Hungar, F. Köster, and J. Mazzega. "Test specifications for highly automated driving functions: Highway pilot". In: *Presentation at Vehicle Test & Development Symposium*. 2017.

[32] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer. "AVFI: Fault injection for autonomous vehicles". In: *Proc. of DSN*. 2018.

[33] J. Kang and D. Lin. "DASH: A Universal Intersection Traffic Management System for Autonomous Vehicles". In: *Proc. of ICDCS*. 2020.

[34] L. Kang and H. Shen. "Electric Vehicle Battery Energy Information is Enough to Track You". In: *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE. 2020, pp. 217–228.

[35] A. Khalid, A. Sundararajan, A. Hernandez, and A. I. Sarwat. "Facts approach to address cybersecurity issues in electric vehicle battery systems". In: *2019 IEEE Technology & Engineering Management Conference (TEMSCON)*. IEEE. 2019, pp. 1–6.

[36] M Khan, R Tripathi, and A. Kumar. "A Malicious Attack and Defense Techniques on Android-Based Smartphone Platform". In: *Internation Journal of Innovative Technol-ogy and Exploring Engineering (IJITEE)* 8 (2019).

[37] H. Kim, J. Smith, and K. G. Shin. "Detecting energy-greedy anomalies and mobile malware variants". In: *Proc. of MobiSys*. 2008.

[38] V. Y. Kulkarni and P. K. Sinha. "Random forest classifiers: a survey and future research directions". In: *Int J Adv Comput* 36.1 (2013).

[39] S. Kuutti, S. Fallah, R. Bowden, and P. Barber. "Deep Learning for Autonomous Vehicle Control: Algorithms, State-of-the-Art, and Future Prospects". In: *Synthesis Lectures on Advances in Automotive Technology* 3.4 (2019).

[40] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard. "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems". In: *IEEE Transactions on control systems technology* 26.5 (2017).

[41] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell. "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4213–4220.

[42] A. Liaw, M. Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002).

[43] L. Lifen, S. Ruoxin, L. Shuandao, and W. Jiang. "Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function". In: *Proc. of CGNCC*. IEEE. 2016, pp. 2011–2015.

[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv preprint* (2015).

[45] S. Liu, K. Zheng, L. Zhao, and P. Fan. "A driving intention prediction method based on hidden markov model for autonomous driving". In: *arXiv preprint* (2019).

[46] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *arXiv preprint arXiv:1706.02275* (2017).

[47] *Matlab Simulation*. `https://www.mathworks.com/products/simulink.html`. 2016.

[48] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura. "Driver modeling based on driving behavior and its evaluation in driver identification". In: *Proc. of IEEE* 95.2 (2007).

[49] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *Proc. of ICML*. 2016.

[50] *Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data*. `https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories`. 2020.

[51] *Nissan Leaf Electric Cars Susceptible to Hacking Attacks*. `https://securityzap.com/nissan-leaf-electric-cars-susceptible-to-`hacking-attacks/. 2016.

[52] *NissanConnect EV and Services*. `https://www.nissanusa.com/connect/features-app/system-requirements/nissan-connect-ev/`. 2018.

[53] T. Ogawa, H. Sakai, Y. Suzuki, K. Takagi, and K. Morikawa. "Pedestrian detection and tracking using in-vehicle lidar for automotive application". In: *Proc. of IV*. 2011.

[54] *Over 1,400 self-driving vehicles are now in testing by 80+ companies across the US*. https://techcrunch.com/2019/06/11/over-1400-self-driving-vehicles-are-now-in-testing-by-80-companies-across-the-u-s/. 2019.

[55] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE T-IV* 1.1 (2016).

[56] P. Palanisamy. "Multi-agent connected autonomous driving using deep reinforcement learning". In: *Proc. of JCNN*. IEEE. 2020, pp. 1–7.

[57] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna. "Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications." In: *Proc. of NDSS*. 2014.

[58] C. Qian, X. Luo, Y. Shao, and A. T. Chan. "On tracking information flows through jni in android applications". In: *Proc. of DSN*. 2014.

[59] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4295–4304.

[60] N. Rauh, T. Franke, and J. F. Krems. "Understanding the impact of electric vehicle driving experience on range anxiety". In: *Human factors* 57.1 (2015).

[61] U. Rosolia, S. De Bruyne, and A. G. Alleyne. "Autonomous vehicle control: A nonconvex approach for obstacle avoidance". In: *IEEE TCST* 25.2 (2016).

[62] B. B. Sahoo, R. Jha, A. Singh, and D. Kumar. "Long short-term memory (LSTM) recurrent neural network for low-flow hydrological time series forecasting". In: *Acta Geophysica* 67.5 (2019), pp. 1471–1481.

[63] S. H. Sánchez, R. F. Pozo, and L. A. H. Gómez. "Driver Identification and Verification From Smartphone Accelerometers Using Deep Neural Networks". In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[64] A. Sarker and H. Shen. "A Data-Driven Misbehavior Detection System for Connected Autonomous Vehicles". In: *Proc. of Ubicomp* 2.4 (2018).

[65] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[66] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. "Deterministic policy gradient algorithms". In: *Proc. of ICML*. 2014.

[67] C. Song, H.-P. Cheng, H. Yang, S. Li, C. Wu, Q. Wu, and H. Li. "Adversarial Attack: A New Threat to Smart Devices and How to Defend It". In: *IEEE Consumer Electronics Magazine* 9.4 (2020), pp. 49–55.

[68] S. Sripad, S. Kulandaivel, V. Pande, V. Sekar, and V. Viswanathan. "Vulnerabilities of Electric Vehicle Battery Packs to Cyberattacks on Auxiliary Components". In: *arXiv preprint* (2017).

[69] J. Suh, K. Yi, J. Jung, K. Lee, H. Chong, and B. Ko. "Design and evaluation of a model predictive vehicle control algorithm for automated driving using a vehicle traffic simulator". In: *Control Engineering Practice* 51 (2016).

[70] *Tesla Android and iPhone app*. `https://www.tesla.com/support/android-and-iphone-app/`. 2018.

[71] R. Tian, S. Li, N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz. "Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts". In: *Proc. of CDC*. 2018.

[72] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* 25.8 (2008).

[73] T. Wakita, K. Ozawa, C. Miyajima, K. Igarashi, K. Itou, K. Takeda, and F. Itakura. "Driver identification using driving behavior signals". In: *IEICE Trans. on Information and Systems* 89.3 (2006).

[74] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, J. Härtl, F. Dürr, and J. M. Zöllner. "Learning how to drive in a real world simulation with deep q-networks". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 244–250.

[75] S. Woo, H. J. Jo, and D. H. Lee. "A practical wireless attack on the connected car and security protocol for in-vehicle CAN". In: *IEEE Trans. on ITS* 16.2 (2015).

[76] L. Xue, X. Luo, L. Yu, S. Wang, and D. Wu. "Adaptive unpacking of Android apps". In: *Proc. of ICSE*. 2017.

[77] T. H. Yimer, C. Wen, X. Yu, and C. Jiang. "A Study of the Minimum Safe Distance between Human Driven and Driverless Cars Using Safe Distance Model". In: *arXiv preprint* (2020).

[78] C. Yuan and H. Yang. "Research on K-value selection method of K-means clustering algorithm". In: *JMultidisciplinary Scientific Journal* 2.2 (2019).