UNIVERSITY OF CALIFORNIA
RIVERSIDE

Vision-Based Eco-Oriented Driving Strategies for Freeway Scenarios Using Deep Reinforcement Learning

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Yu Jiang

December 2020

Thesis Committee:
Dr. Guoyuan Wu, Co-Chairperson
Dr. Matthew Barth, Co-Chairperson
Dr. Jun Sheng

The Thesis of Yu Jiang is approved:

Committee Co-Chairperson

Committee Co-Chairperson

University of California, Riverside

Copyright Acknowledgements

The text and figures of this thesis in Chapter 2-4, in part, are a reprint of the material as it appears in the published work: Wei, Z., **Jiang, Y.**, Liao, X., Qi, X., Wang, Z., Wu, G., Hao, P., and Barth, M. (2020). End-to-End Vision-Based Adaptive Cruise Control (ACC) Using Deep Reinforcement Learning. *arXiv preprint arXiv:2001.09181*. Presented in Transportation Research Board (TRB) 2020.

Acknowledgements

First of all, I would like to express my appreciation to my parents for their continuous support during the years of my study. I would also like to express my gratitude to my faculty advisors, Prof. Guoyuan Wu and Prof. Matthew J. Barth. They provided me with encouragement and patience throughout this work. The thesis would not be possibly completed without their continuous guide and instruction. Besides, I would like to extend my sincere thanks to all TSR group members for their assistance and collaboration. Finally, special thanks to former TSR group member Xuewei Qi for the original idea of this work.

ABSTRACT OF THE THESIS

Vision-Based Eco-Oriented Driving Strategies for Freeway Scenarios Using Deep Reinforcement Learning

by

Yu Jiang

Master of Science, Graduate Program in Electrical Engineering
University of California, Riverside, December 2020
Dr. Guoyuan Wu, Co-Chairperson
Dr. Matthew Barth, Co-Chairperson

The rapid development of sensor technologies and machine learning algorithms has prompted the automotive industry to take big steps towards autonomous driving. However, complex driving scenarios make the task extremely challenging. The current technologies are sensors-hungry, and the systems are comprehensive to achieve robust behavior in the real world, which causes an autonomous driving system to be unwillingly expensive. Leveraging the computer vision techniques, the camera as a relatively cheaper sensing option is becoming crucial in the system. Using the emerging game engine-based simulation platform, the development of vision-based autonomous driving achieved by deep reinforcement learning becomes practical and more efficient. Besides, energy consumption and greenhouse gas emissions have been a major concern in recent years due

to the increased travel demand. Environmental sustainability of our transportation system has become a significant topic for researchers and engineers. Motivated by all these, this thesis presented an end-to-end solution for advanced driving assistance strategies for freeway scenarios using vision-based deep reinforcement learning technique. The system is designed and tested in an advanced driving simulation platform based on game engine and can learn optimal control actions directly from raw input images captured by the front on-board cameras. The trained vehicle has robust and safe driving behavior following a preceding vehicle or cruising on the straight freeway. The system can be well adaptive to different speed trajectories of the preceding vehicle and run in real-time. Novel adaptive gap-based and energy model-based reward functions are designed for eco-oriented driving. The method achieved 7.2 – 21.7% energy savings compared with other methods led by strict gap-based reward or force-based reward in the car following strategy.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

As a significant component in Intelligent Transportation Systems (ITS), autonomous vehicles (also known as self-driving cars or automated vehicles) are vehicles that are able to sense their environment and take over the partial or even entire dynamic driving task under various complex conditions. It is honored with promises to revolutionize our life.

The degree of automotive automation can range from fully manual operation to fully automated operation. According to the SAE J3016 standard [1], a ranking from 0 to 5 is defined to describe the degree. In this standard, Level 0 means the driver must continuously monitor all aspects of the dynamic driving task. Level 1 means the system can take over either steering or acceleration/deceleration. The driver must continuously carry out the other tasks and be ready to resume full control. At Level 2, the driver still needs to be ready to resume control anytime, but the system is able to take over both steering and acceleration/deceleration in a defined use case. At Level 3, the system not only has control ability but is also capable of recognizing its limits and notifying the driver. The driver is free from continually monitoring. A vehicle with Level 4 automation can take care of the entire driving task in a defined use case. Within the case, the driver is no longer needed. If the vehicle can take over all driving tasks in any use case, it is at Level 5, the top level of autonomous driving.

For years, automotive and technology companies have been discussing autonomous driving strategies. Now, more and more autonomous functions go on the market every year. Some of the companies have made significant progress. For example, in 2015, Google provided the world's first fully driverless ride on public roads. The car had no steering wheel or floor pedals with Level 4 automation [2]. In December 2016, the project became a stand-alone subsidiary of Alphabet Inc. called Waymo. Soon in October 2017, Waymo began testing autonomous minivans at Level 4 automation without a safety driver on public roads in Chandler, Arizona [3]. Tesla, an American electric vehicle and clean energy company, started their advanced driver assistance system (named Autopilot) from 2013. In September 2020, they released the enhanced version, which provides full autonomy on highways, parking and summons. Eight external cameras, a radar, and twelve ultrasonic sensors and a powerful onboard computer provide a successful vision-based solution for autonomous driver-assist system [4]. Besides, traditional automotive companies have not fallen behind. The BMW Personal CoPilot provided various driving assistance system at Level 3, such as adaptive cruise control, parking assistant, and lane keeping assistant [5].

The achievements in the past few years inspired not only customers but also investors. More and more customers want driving assistance on their cars to reduce long-way commuting pressure. Besides, investors also expect the potential profits and invest a significant amount of money in the development or research of higher-level vehicle automation. However, the final step is always the hardest. Perception and decision planning are two significant challenges in the high-level automation. Perception is the detection, recognition, identification, and interpretation of traffic information in the environment. Based on the acquired surrounding information, driving behaviors are designed and controlled by the decision planning logic. Many tasks may be easy for human drivers but not the same for machines. For instance, the traffic sign recognition as a standard perception task could be challenging in complicated scenarios such as narrow urban streets surrounded by hundreds of shinning billboards. Then the decision planning is also a tricky task when multiple sensors give various and possibly conflicting traffic information. There are still some other technologies to potentially conquer these challenges, such as vehicle-to-everything (V2X) technology enabled by dedicated short-range communications (DSRC) or even 5^{th} generation mobile networks (5G). It is far beyond just a driving problem, but a very complex one combining communication network, urban planning, human factors, and policy making.

Rather than looking forward to a higher-level automation, optimizing the traditional driving assistance system is still a promising research direction. This thesis work focuses on eco-oriented driving assistance strategies for freeway scenarios. The detailed motivations are introduced in the following section.

1.1 Motivation

1.1.1 Environmental Impacts by Transportation

As for either human driving or autonomous driving, energy consumption has been always an issue. In recent years, the fossil fuel consumption and greenhouse gas emissions caused by the traditional internal combustion engine (ICE) vehicles have aroused more and more public attention. According to the statistics provided by the U.S. Environmental Protection Agency [6], transportation-related activities accounted for 28.2% of nationwide greenhouse gas (GHG) emissions in the United States in 2018. As shown in Figure 1.1, the transportation sector generates the largest share of greenhouse gas emissions. The share has been increasing in recent years due, in large part, to the increased demand for travel and goods movement. Figure 1.2 shows the overall trend of GHG emissions from 1990 to 2018.

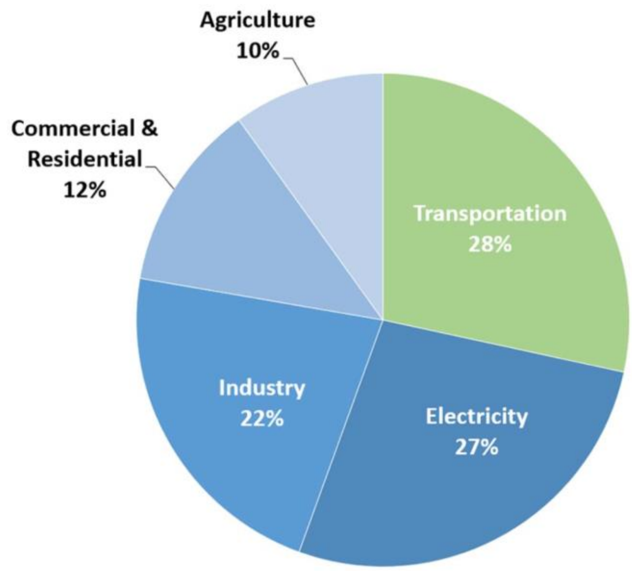

Figure 1.1. Total U.S. Greenhouse Gas Emissions by Economic Sector in 2018. [6]

The development of electric vehicles (EVs) is aimed to be an excellent solution to this problem. EVs transform the electric energy into kinetic energy electrochemically without combustion and therefore have zero emissions of tailpipe carbon dioxide (CO2) or pollutants such as oxides of nitrogen (NOx), non-methane hydrocarbons (NMHC), carbon monoxide (CO), and particulate matter (PM) [7]. However, the sources to generate electricity may not be that eco-friendly. According to the U.S. Energy Information Administration [8], about 62.7% of electricity generation was from fossil fuels, while only 17.5% was from renewable green sources like wind turbines or solar panels. For this new type of vehicle, it is still necessary to optimize energy efficiency to improve the mileage between recharging, reduce battery heat, and reduce the consumption of the non-renewable sources. Therefore, this work focus on eco driving for both ICE vehicles and EVs.

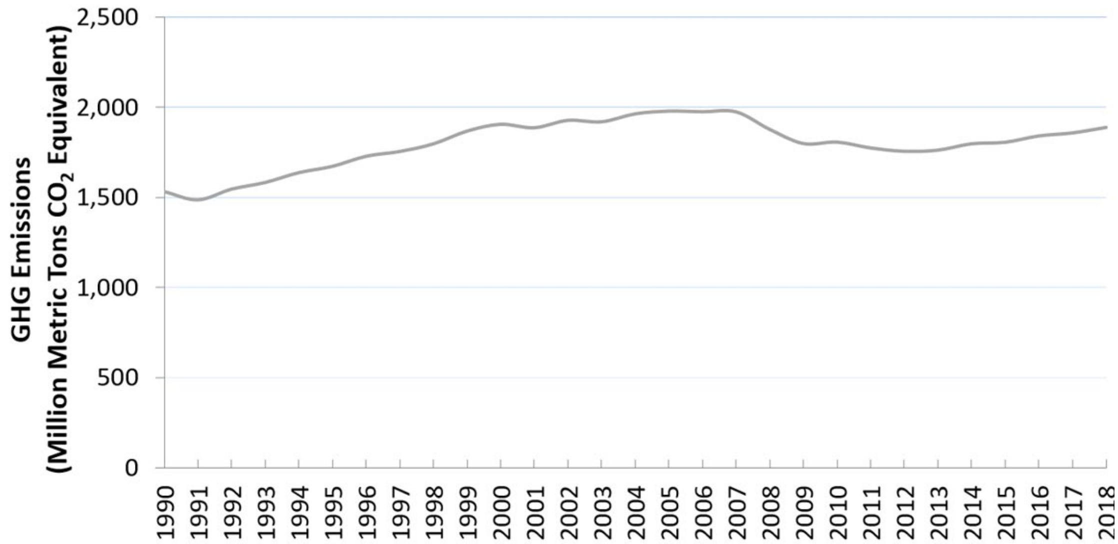

Figure 1.2. Greenhouse Gas Emissions from Transportation, 1990-2018. [6]

1.1.2 Advances in On-board Sensors

For intelligent driving assistance systems, a variety of techniques are used to detect the environment such as radar, LiDAR (Light Detection And Ranging), GPS and computer vision, etc. The robust and reliable sensors become critical hardware. As a fundamental requirement for autonomous driving, the 3D object detection and depth estimation tasks majorly rely on LiDAR, since it can provide accurate 3D point clouds of the detected region. However, high accuracy also means expensive (at the cost of both information storage and computational load). According to the automotive LiDAR market report [9], a 64-beam and 20 Hz model can cost around $75,000, not to mention the 60 W power consumption. Therefore, the alternatives to LiDAR is desirable for avoiding a hefty premium for autonomous driving hardware [10]. As human drivers using eyes to estimate distance, using the stereo or monocular camera is a natural option. A group of stereo cameras within $200-1000 could handle similar functions as the expensive LiDAR system.

The potential of cameras in autonomous driving has been explored in many recent works, such as [10][11][12]. Especially in [10][11], the camera's accuracy and efficiency as pseudo-LiDAR have been revealed. The detailed pros and cons of LiDAR and camera are compared in Table 1-1. Based on the pros of the camera, we believe the potential of the vision-based system is worth exploring, and vision-based architecture is adopted in this work.

Table 1-1. Comparison between LiDAR and camera.

Sensor	LiDAR	Camera
Pros and Cons	√ Accurate depth information	× Less accurate depth information √ Richer semantic information
	√ Works all day and night × Inaccurate under terrible weather or smog environment	× Better at daytime × Sensitive to weather and lighting conditions
	× Expensive (more than $10k) × High energy consumption (10-60W)	√ Cheap ($200-$3000) √ Low energy consumption (2-10W)

1.1.3 Development of Machine Learning based End-to-end Systems

In the deep learning field, “end-to-end” means all parameters are trained jointly rather than in a “step-by-step” manner across the entire pipeline. There are no intermediate variables between the input and the desired output. For example, to implement an emergency brake function with some input observation data, the step-by-step method intuitively gets the distance from the obstacles firstly, and then performs some control strategy based on the distance. In comparison, the end-to-end system is more straightforward. It takes those input data and outputs the control policy directly. As a result, there is no need to specify and tune the parameters of each component individually. This is attractive, since each part of the parameters will be optimal with respect to the desired end task [13]. The reduced parameters also simplify the cost functions in most works and avoid error propagation or error accumulation as well as over-complexity. The accumulated errors sometimes may cause fatal accidents. In the tragic Tesla accident in 2016 [14], an error in the perception module in the form of a misclassification of a white trailer as sky, propagated down the pipeline until failure [15]. The effectiveness of the end-to-end system has been proved in many studies, such as [16][17][18].

Not only the end-to-end training has been proved to be effective, but various deep machine learning (DL) methods have also been widely used in the research of driving strategies. Cai et al. proposed a CNN+LSTM network architecture to map visual perception and state information into future trajectories with the help of high-level turning command [16]. Some deep reinforcement learning (DRL) architectures, such as Deep Q Network (DQN) and Deep Deterministic Policy Gradient (DDPG), also achieved promising results. Okuyama et al. used a DQN to train their autonomous agent, which can learn to drive and avoid obstacles in a simulation based on the input images captured by the on-board monocular camera [12]. Sallab et al. proposed a framework of an end-to-end DRL for autonomous driving and tested the system in a simulated scenario of complex road curvatures [19]. Lillicrap et al. proposed DDPG by adapting the ideas of Deep Q-Learning to the continuous action domain and tested a car driving scenario for learning the end-to-end policies from raw pixel inputs [20]. Despite the various DRL applications, there was not an eco-oriented vision-based driving assistance system using DRL. Therefore, DRL algorithms are applied in this work to seek an eco-oriented vision-based end-to-end solution.

1.2 Contributions

This thesis provides end-to-end solutions for advanced driving assistance strategies in the freeway scenarios using vision-based deep reinforcement learning architectures. The contributions of the thesis are listed below:

- A novel end-to-end deep reinforcement learning based advanced driving assistance system is designed and tested in a realistic simulation platform.
- The proposed system can be applied to ICE vehicles and EVs. The system is capable of learning different driving strategies. The system architecture is simple (using only sensor input from the front on-board cameras) and can work in real-time.
- Novel adaptive gap-based and energy model-based reward functions are designed to guide eco-oriented driving.

1.3 Thesis Organization

The organization of this thesis is shown in Figure 1.3. The structure of this paper is mainly divided into five parts.

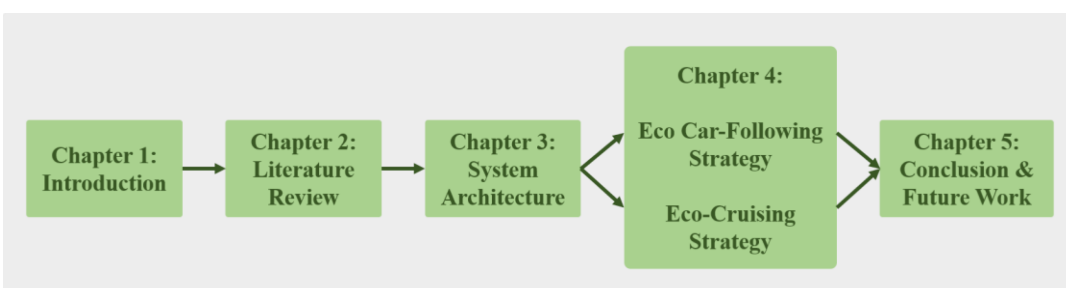

Figure 1.3. The organization of this thesis.

The first chapter introduces the three major motivations of this work, and the contribution as well as roadmap of this work in the field of Intelligent Transportation Systems.

The second chapter summarizes the related work and presents the background information from the following three aspects: *advanced driving assistance techniques*, *energy consumption models*, and *simulation platforms*. Advanced driving assistance techniques are divided by radar/LiDAR-based and vision-based techniques. Several major objectives in this area are discussed, followed by the introduction and comparison among classic deep reinforcement learning (DRL) methods. Besides, the energy consumption models are presented to serve as the foundation for eco-oriented driving. Three prevalent simulation platforms for training are finally illustrated.

The third chapter provides the proposed system architecture of this work, including the system workflow, the network architectures, the simulation environment setup, and data acquisition for reinforcement learning.

Chapter 4 applies two DRL algorithms to the proposed system. The system is used for an eco-oriented car following strategy and an eco-oriented cruising strategy. The details of multiple case studies and results for each strategy are introduced separately in section 4.2 and section 4.3. In addition, the result analysis and possible solutions are illustrated at the end of each section.

Chapter 5 concludes the proposed advanced driving assistance system and its effectiveness for two eco-oriented driving strategies. Two possible future research directions are presented at the end.

Chapter 2: Background and Literature Review

2.1 Radar/LiDAR-based Advanced Driving Assistance System

2.1.1 Adaptive Cruise Control (ACC)

Cruise Control is a driver-assistance system that takes over the throttle of the vehicle to maintain the speed set by the driver automatically. When there is a preceding vehicle slowing down, the driver needs to take back the control of the throttle. Leveraging various on-board sensors, the adaptive cruise control (ACC) technique can free the driver's feet by automatically adjusting the longitudinal vehicle speed to maintain a safe distance from the preceding vehicle. Once the ACC system is implemented in various cars' models on the market, it has been widely acclaimed, especially by the drivers who spend one or more hours commuting.

An ACC system has three major components: perception, planning, and actuation. Perception provides necessary external information, such as the distance to the preceding vehicle. Radar and LiDAR are the most commonly used sensors in an ACC system to detect the depth of the environment. The planning phase indicates the longitudinal control algorithms. It usually calculates a reference acceleration profile according to the perception data. Finally, the low-level controller actuates the driving actions by converting the reference acceleration into throttle and brake [21]. Many longitudinal control algorithms have been proposed and optimized by researchers and engineers. Kesting et al. presented an ACC system adapting an active jam-avoidance strategy that driving style automatically changed w.r.t. different traffic situations [22]. Luo proposed a human-like ACC algorithm using model predictive control (MPC), where safety, comfort, and time-optimality were considered [23]. Shi and Wu designed a target recognition algorithm and a switching logic for their ACC system on the curve [24].

In [25], the authors proposed a distributed consensus algorithm for their ACC system, which provided the reference acceleration based on the speed difference and inter-vehicle gap. The reference acceleration is given by:

$$a_{ref} = \beta \cdot (d_{gap} - d_{ref}) + \gamma \cdot (v_{pre} - v_{host})$$

where β and γ are damping gains; v_{pre} and v_{host} are the velocities of the preceding vehicle and the host vehicle, respectively; d_{gap} is the inter-vehicle gap; d_{ref} is the desired inter-vehicle gap, which is defined by the product of desired time gap and the current velocity of the host vehicle, i.e., $d_{gap} = v_{host} \cdot t_{gap}$. It basically measures the time that the host vehicle takes to reach the current position of the preceding vehicle. d_{ref} is also bounded by a minimum value of d_{safe} to avoid unreasonable small desired inter-vehicle gap when v_{host} approaches to 0. In the simulation testing, this algorithm presented robust and smooth car following behavior. We used this algorithm as a comparison when analyzing the energy efficiency of our methods in Chapter 4.

2.1.2 Cooperative Adaptive Cruise Control (CACC)

CACC, as an extension of the ACC concept, takes advantage of the vehicle-to-vehicle (V2V) technology to allow connected and automated vehicles (CAVs) to form a platoon and cruise at harmonized speeds [21]. Compared with the manually driven string or the traditional ACC-enabled string, the CACC platoon achieves many benefits. Firstly, driving safety is improved since the downstream traffic information can be broadcast to the following vehicles in a timelier manner. Secondly, the time/distance headways between vehicles are significantly reduced which considerably improves the roadway capacity/utilization. Thirdly, the more advanced information shared via communication can achieve better control performance, thus mitigating the unnecessary velocity variations. Therefore, the CACC platoon not only improves the string stability thus avoiding the phantom traffic jams [26], but also saves energy consumption as well as reduces pollutant emissions.

2.1.3 LiDAR Point cloud 3D Object Detection

Point cloud means a set of data points describing the boundaries of objects in space. Utilizing the accurate depth information from laser scanners, detected point cloud might contain the location, orientation, and category information about the objects. Therefore, the existing 3D object detection algorithms focus on extracting that information from the point cloud. Some of the algorithms use support vector machine (SVM) on 3D meshes encoded with geometry features to classify the objects, such as [27][28]. The point cloud analysis and processing are often expensive in computation due to the high dimensionality of the input 3D data. PIXOR decoder [29] only takes the bird's eye view representation as input and uses a novel decoding loss to train the network, improving the efficiency in computation and accuracy in detection. Another drawback of point cloud data is that less semantic information is preserved in it. Sensory-fusion frameworks are used in [30][31] to exploit both LiDAR point cloud and RGB images captured from cameras. A considerable improvement on Average Precision (AP) of 3D localization and detection is obtained in these studies.

2.2 Vision-based Advanced Driving Assistance System

2.2.1 Vision-based Perception

With the rapid development of computer vision techniques, image processing and understanding have become a significant role in autonomous driving. With these techniques, data collected by camera sensors can be well analyzed. For example, Aly et al. introduced a lane detection method. They firstly used Inverse Perspective Mapping (IPM) to transform the camera image into a bird-eye view. Then they implemented a RANSAC spline fitting to fit the lines selected by the Hough transform method [33].

The image segmentation method can improve efficiency of object detection. For instance, Chen et al. improved the performance of image segmentation by introducing the Atrous Convolution [34]. Pan et al. developed Spatial CNN (SCNN) in order to explore the capacity to capture spatial relationships of pixels across rows and columns. The model could be easily incorporated into deep neural networks and trained end-to-end. They evaluated the lane detection performance on the TuSimple benchmark. By introducing SCNN into the LargeFOV model, their 20-layer network outperformed ResNet, MRF, and the very deep ResNet-101 in lane detection. They also showed visual semantic segmentation improvements on the Cityscapes validation set. The total results proved that SCNN could effectively preserve the continuity of long thin structure, while in semantic segmentation, its diffusion effects were also proved to be beneficial for large objects [35].

The methods mentioned above are the applications of computer vision techniques in specific tasks. In the real-world, driving is a comprehensive skill, and a robust autonomous vehicle should be able to handle comprehensive tasks and be aware of its surroundings. For example, human drivers can change their driving decisions based on various information, such as traffic signs, traffic signals, weather, road surface, etc. Therefore, an architecture combining different perception techniques may be needed, and the computational time is essential for running in the real world. Teichmann et al. presented an approach to joint classification, detection, and semantic segmentation via a unified architecture where the encoder is shared amongst the three tasks [36].

2.2.2 Vision-based Decision Making

According to [39], most vision-based autonomous driving systems can be categorized into two major paradigms: mediated perception approaches and behavior reflex approaches. The former approaches involve total scene understanding, such as lanes, signals, obstacles, pedestrians, weather, and etc. The driving decision is generated by leveraging the assistance methods mentioned in the previous sub-sections. The later constructs a direct mapping from the sensory input to driving actions. The authors of [37] proposed a vision-based obstacle avoidance system for off-road driving robot DAVE. It was trained by end-to-end learning methods that learned a mapping from images to steering angles. Hadsell, Raia, et al. presented a self-supervised learning process for an off-road driving robot as well. The system accurately classified complex terrains at distances up to the horizon using a multi-layer convolutional network [38].

NVIDIA trained a CNN (PilotNet) in 2016 to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful in the self-driving problem. Compared to the explicit decomposition of the problem, such as lane marking detection, path planning, and control, their end-to-end system optimized all processing steps simultaneously [17]. With minimum training data from humans, the system learned to drive on local roads and highways with or without lane markings. It also operated in areas with unclear visual guidance, such as parking lots and unpaved roads. However, behavior reflex approaches can only be realized by supervised learning. Whether the learned policies can perform well in an unfamiliar environment remains unknown.

2.3 Reinforcement Learning (RL)

2.3.1 The classic model of RL

Inspired by behavioral psychology, reinforcement learning (RL) proposes a formal framework to handle sequential decision-making [40]. Within this framework, the agent learns a behavior offline from its historical experience. The experience is acquired by interacting with the environment and collecting state changes and reward feedbacks.

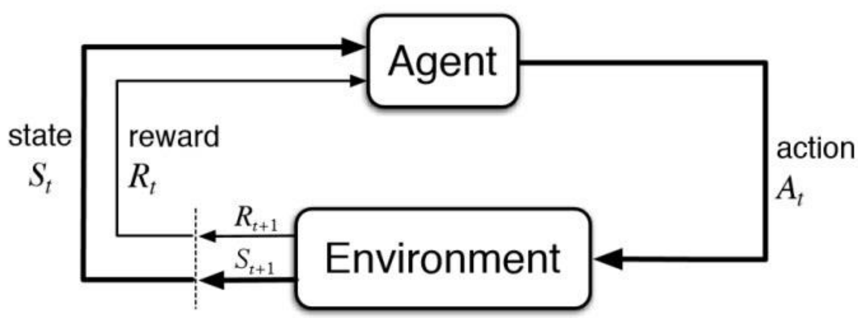

Figure 2.1. The classic model of reinforcement learning [40].

A classic model of reinforcement learning is shown in Figure 2.1. The essential elements are:

1. **Agent**: The agent is the subject of a task. It takes actions to achieve some goals using some policies.

2. **Actions**: The actions are all possible movements that the agent can take. For example, the actions for a vehicle could be throttle, steering angle, etc.
3. **Environment**: The world where the agent is. The agent can observe states and get rewards from the environment.
4. **State**: The observations from the environment. In the transportation system, it could be the data from sensors, for example, the distance to obstacles (LiDAR info), the location (GPS info), or the images (camera info).
5. **Reward**: The reward is the feedback provided by the environment. It is used to measure the performance of the agent's action under a given state at a time step.
6. **Action-value**: The value of an action. To measure the actual value of an action, not only the instant feedback (reward) but also the long-term potential of this action would be needed. Therefore, the action-value is usually defined by the discounted accumulative rewards.
7. **Policy**: The policy is the strategy for the agent to achieve a given goal. It usually maps states to actions, and the best policy is the one that can maximize the rewards during the whole process.

The following sections will introduce several specific RL methods, based on which the algorithms in this work are developed.

2.3.2 Popular RL Methods

2.3.2.1 Q-learning

Q-learning, one of the most popular RL algorithms, was introduced by Chris Watkins in 1989 [41]. A convergence proof was presented by Watkins and Dayan in 1992 [42].

At a time t, the agent observes a state s_t. Then based on the observation, the agent implements an action a_t and observes the next state s_{t+1}. After that the agent gets its reward r_t. This process lasts until the agent achieves a terminal state at time T. The goal is to maximize the total rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ where the discounted factor γ can balance the immediate reward and the later rewards.

The policy π describes the probability distribution of the action at a given state s, i.e. $\pi(a|s) = P(a|s)$. A lookup table $Q(s,a)$ is used to store the action-value, where each entry represents the value of a possible action under a given state. The Q-value is defined by the expectation of R_t i.e.

$$Q^{\pi}(s,a) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots |s_t = s,\ a_t = a,\ \pi] \tag{2.3.1}$$

This function can be rewrite with Bellman equation as below,

$$Q^{\pi}(s,a) = \mathbb{E}[r_t + \gamma Q(s_{t+1}, a_{t+1}) | s_t = s,\ a_t = a,\ \pi] \tag{2.3.2}$$

Then the optimal policy * is the policy that can maximize the achievable expectation of total rewards i.e.

$$Q^{*}(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\,|s_t = s,\ a_t = a,\ \pi =*\right] \tag{2.3.3}$$

2.3.2.2 Deep Q Network (DQN)

As mentioned above, Q-learning stores data in the Q-table. The approach suffers the curse of dimensionality with increasing dimensions of state space and action space. One solution is to combine Q-learning with deep learning techniques. We can use a function approximator, usually an artificial neural network, to approximate the mapping from states/actions to values.

In this method, a neural network $Q(s,a;\theta)$ with some parameters θ is used to approximate the optimal value function $Q^*(s,a)$. Then similarly as Q-learning, the target Q values is

$$Y^Q = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta) \tag{2.3.4}$$

The parameters θ can be updated by applying the gradient descent to the loss function, which is defined by the Mean Square Error (MSE) between the target Q values and the current Q values, i.e.

$$L(\theta) = [Y^Q - Q(s,a;\theta)]^2 \tag{2.3.5}$$

Then the parameters θ at time step t are updated by

$$\theta_{t+1} = \theta_t + \alpha\big(Y^Q - Q(s,a;\theta_t)\big)\nabla_\theta Q(s,a;\theta_t) \tag{2.3.6}$$

where α is the learning rate.

Inspired by this idea, the Deep Q Network (DQN) introduced by Mnih et al. has become a hotspot in the past few years because of the following two heuristics [43]:

1. For *t*-th iteration, the target Q network is replaced by $Q(s_{t+1}, a_{t+1}; \theta_t^-)$ i.e.

   $$Y_t^{DQN} = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_t^-) \tag{2.3.7}$$

   where the parameters θ^- are updated every τ iterations. This prevents the instabilities to propagate quickly and reduces the risk of divergence [40];

2. A replay buffer is used for offline training. The replay buffer stores a set of tuples (s_t, a, r, s_{t+1}) collected from the last N time steps in the online training. Then the offline training can leverage the mini-batch technique, evolving less variance and improving the stability.

By the replay buffer technique, the Deep RL method is more like human learning. It can not only learn from the reward and evaluate the action values, but also learn the knowledge collected in history. It mimics the empiricism part of the human thinking process.

Now we can see the significant advantages using the Deep RL method in the Autonomous Driving field in the simulation environment. In the virtual world, we can collect experience easily. For example, we can penalize the agent when a collision happens and do not need to worry about the safety issue. Besides, we can get as much data as we want into the replay buffer and do not worry about the oil consumption and environmental issues. We can even well craft some corner cases to evaluate the system performance.

2.3.2.3 Double Deep Q Network (DDQN)

Since the Q-learning uses the same network to select an action and evaluate its value, it suffers the overestimation due to the non-uniformed noises in the network. Therefore, enrolling in another network to uncouple the selection and evaluation can handle this problem. Van Hasselt et al. [44] introduced the Double Deep Q Network (DDQN) structure, the target value becomes:

$$Y_t^{DDQN} = r_t + \gamma Q^{-}\left(s_{t+1}, \underset{a_{t+1}}{\operatorname{argmax}}\, Q(s_{t+1}, a_{t+1}, \theta_t)\,, \theta_t^{-}\right) \tag{2.3.8}$$

The greedy action is chosen by the Q network, but the evaluation of this action is done by the target network Q^{-}. The parameters θ^{-} of this network are usually copied from θ after every whole episode of training.

This method is used in the first work in this thesis. During the training, the system will take a random action (randomly chosen in the action space) with a probability of epsilon to ensure the exploration. The epsilon will decrease to zero after a certain number of time steps.

2.3.2.4 Deep Deterministic Policy Gradient (DDPG)

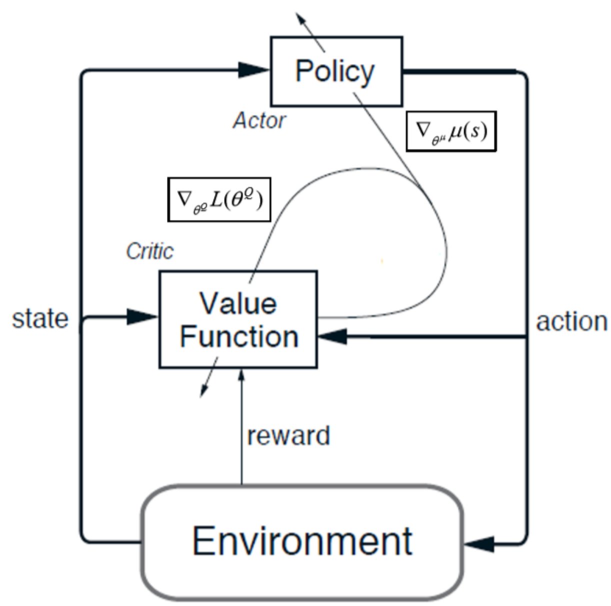

Figure 2.2. The architecture of DDPG [20].

The output dimension of the Q network equals the dimension of the action space. Therefore, the action space is limited and discretized. While in the driving scenario, the actions we need are all represented by continuous variables, such as throttle, braking, and steering angles. Hence, we must discretize the actions to utilize the deep Q learning method for driving tasks. Another option is to use a different RL architecture which can handle continuous action space. The DDPG is an excellent solution introduced below.

We can assume the optimal action is deterministic, so that we do not need the probability distribution $\pi(a|s) = P(a|s)$ to describe the actions. Recall in the Q-learning, $\mu(s) = \underset{a}{\operatorname{argmax}} Q(s, a)$. While in deterministic policy gradient method, we can use a function $a =$

$\mu(s; \theta^\mu)$ to get the desired actions from observations. Similarly, we can utilize the neural network again to approximate this function.

The Figure 2.2 shows the architecture of DDPG. The critic network is updated in the same way as the DQN. The difference here is we get the action from μ network instead of Q network. Thus, the target value becomes:

$$Y_t^{critic} = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}); \theta^Q) \tag{2.3.9}$$

The actor is updated by applying the chain rule to the target value with respect to the actor parameters:

$$\begin{aligned} \nabla_{\theta^\mu} &\approx \mathbb{E}_{\mu'}[\nabla_{\theta^\mu} Q(s, a; \theta^Q)] \\ &= \mathbb{E}_{\mu'}[\nabla_a Q(s, a; \theta^Q) \nabla_{\theta^\mu} \mu(s; \theta^\mu)] \end{aligned} \tag{2.3.10}$$

In practical, for i-th time step under state s_i, the actor is updated by sampled gradients:

$$\nabla_{\theta^\mu} \mu|_{s_i} = \frac{1}{\mathrm{N}} \sum_i \nabla_a Q(s, a; \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s; \theta^\mu)|_{s=s_i} \tag{2.3.11}$$

Algorithm 1 shows detailed procedure of DDPG [20]. Ornstein–Uhlenbeck (OU) process x_t is often used for action exploration.

$$dx_t = -\theta(x_t - \mu)dt + \sigma dW_t \tag{2.3.12}$$

where the W_t denotes the Wiener process, i.e. $dW_t = W(t) - W(t - dt) \sim \mathcal{N}(0, \sigma^2 dt)$.

For the independent random noise, if we want to ensure enough exploration in a small scale of time discretization, we need large enough variance. As a result, vast distances appear between actions in each time step, and it is inappropriate for a real physical body. The motion of real physical body usually is stochastically dependent. Wawrzynski gave well explained mathematical derivation in [32]. Therefore, the OU process is necessary. For example, we use DDPG as the reinforcement learning algorithm to train our autonomous vehicle. We want the vehicle to maintain a safe distance with the preceding vehicle while the output control action is throttle. The throttle directly corresponds to acceleration, and of course, we do not want upheaval changes on acceleration.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i(y_i - Q(s_i, a_i|\theta^Q)^2)$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Algorithm 2.1. The DDPG algorithm [20].

2.3.2.5 Multi-tasks Reinforcement Learning Method

As mentioned above, the neural network as a function approximator is useful especially for finding a non-trivial function or policy. In the multi-tasks learning, one approximator can hardly complete the tasks due to catastrophic interference, i.e., the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information [45]. The following two studies show that how to use multiple approximator for multi-tasks reinforcement learning.

Codevilla et al. proposed an end-to-end imitation learning architecture for multi-tasks of driving based on the input command [46]. By adding a command as input, the system could learn different policies under different commands. They proposed two kinds of architecture. The first one is to treat command c as a state variable and concatenate it with other states. The other is to treat command c as a switch, which controls the activation of different neural networks to learn different policies.

Tessler et al. proposed a Hierarchical Deep RL Network (H-DRLN) for multi-tasks learning [47]. They firstly defined several skills that the agent needs to learn. Each skill represented an optimal policy of a certain task. Then added those skills as a set of new actions into the RL network. If the whole network decided to use a skill rather than the primitive actions, the corresponding pre-trained Deep Skill Network (DSN) would take in charge of a time period until the sub-goal had achieved. Therefore, the experience replay buffer for H-DRLN stores $(s_t, a, \tilde{r}, s_{t+k})$, where a includes primitive actions and the skills. The $k = 1$ when a is a primitive action, and $k \geq 1$ when a is a skill.

2.4 Microscopic Energy Consumption Models

With eco-driving becoming a significant role in the transportation system, various energy consumption models have been developed to measure environmental performance for different types of vehicles. For example, the U.S. EPA's Motor Vehicle Emission Simulator (MOVES) provides greater capability than the MOBILE emission models for estimating the impacts of traffic operational changes [48]. Shibata and Nakagawa constructed a mathematical model which calculates the EV power consumption for both traveling and air-conditioning [49]. In this work, we applied the Comprehensive Modal Emissions Model (CMEM) developed by the researchers from University of California at Riverside to modeling the emissions of internal combustion engine (ICE) vehicles, and a hybrid energy consumption rate model [50] for EVs, respectively. These two models are briefly introduced in the following sections.

2.4.1 Comprehensive Modal Emissions Model (CMEM)

CMEM is designed to estimate the individual ICE vehicle's fuel consumption and emissions [51].

$$\frac{\mathrm{dFuel}}{\mathrm{d}t} \approx \lambda\left(k \cdot N \cdot D + \frac{P_{engine}}{\eta_{engine}}\right) \tag{2.4.1}$$

Based on the fuel consumption and engine power, the emissions can be computed as:

$$\mathrm{Emission} = \frac{\mathrm{dFuel}}{\mathrm{d}t} \cdot \frac{g_{emission}}{g_{fuel}} \cdot CPF \tag{2.4.2}$$

where $\frac{g_{emission}}{g_{fuel}}$ is ratio of engine-out emission index and fuel consumption. CPF is the catalyst pass fraction, which is defined as the ratio of tailpipe to engine-out emissions [64].

2.4.2 Hybrid Energy Consumption Rate Estimation for EVs.

In this work, we implement an EV energy consumption rate estimation model (Type II) adapted from [50] for evaluation. The model is also used to design the eco-oriented reward function in the proposed RL architectures for EVs.

$$P_{est} = l_0 + l_1 v + l_2 v^3 + l_3 va + l_4 v^2 + l_5 v^4 + l_6 v^2 a + l_7 v^3 a + l_8 va^2 \tag{2.4.3}$$

where v is the speed; a is the acceleration; and l_i's is the associated coefficient of the i-th term. Please refer to Table 4 in [50] for the values of associated coefficients.

2.5 The Simulation Platform for Autonomous Driving

Testing new autonomous driving algorithms in the real world could be dangerous. To avoid unnecessary accidents, the algorithm should be tested in a virtual world. Leveraging the development of the game engine, many researchers find it useful for autonomous vehicle's training and simulation. Specifically, the simulation platform can provide complex environment and scenarios depending on the researchers' needs. The following simulation platforms have been widely used in a plenty of research work.

2.5.1 TORCS

TORCS is a highly portable open-source 3D car racing simulation. It is written in C++ and is licensed under the GNU GPL. It runs on multiple computer platform including Linux, MacOS and Windows, etc. As shown in the screenshots in Figure 2.3, TORCS features various types of vehicles and road conditions. The virtual vehicles can be driven with the mouse or the keyboard. Graphic features lighting, smoke, skid marks and glowing brake disks. Many realistic preferences can be set in the simulation such as simple damage model, collisions, tire and wheel properties, etc. [52]. Since 2008, TORCS has also played an important role in various research. For instance, Xiong et al. evaluated their autonomous driving strategy in the TORCS [53]. Compared with other platforms, TORCS is less realistic and cannot present the complexity of other scenarios except for racing.

Figure 2.3. The screenshots of TORCS.

2.5.2 Unity

Unity is a real-time 3D development platform developed by Unity Technologies. Paired with Visual Studio, Unity has been widely used by engineering teams to create games efficiently. It has a friendly and easy-to-use editor, online assets store, and a well-established community for developers and researchers. As of 2018, the engine had been extended to support more than 25 platforms. Recently, more and more researchers have developed simulations for autonomous driving. For example, a simple scenario (see Figure 2.4 (a)) is used in the Udacity Behavioral Cloning project. The complex and realistic scenario could also be present in Unity, such as AirSim (see Figure 2.4 (b)) created by the team at Microsoft AI & Research [54]. Unity also provides an open-source project called The Unity Machine Learning Agents Toolkit (ML-Agents) that enables games and simulations to serve as environment for training intelligent agents [55]. Besides, Automotive companies such as Toyota and Audi are using Unity to improve efficiency for development and innovation.

The Unity standard assets provide car models and controller scripts. Users can easily use them to control a vehicle in a realistic manner. For example, users can set various parameters such as torque over wheels, reverse torque, brake, downforce, top speed, slip limitation, and etc. The traction control script can reduce the power to the wheel if the car is wheel spinning too much. The move function drives a vehicle through the input force. The gear changing, wheel spin checking, and traction control are handled automatically. We used these built-in functions for our project (see Chapter 4).

(a) (b)

Figure 2.4. (a) A sample image of Udacity behavioral cloning project; (b) Windridge City environment in AirSim.

2.5.3 CARLA

An open-source simulator for autonomous driving research called CARLA has been developed from the ground up to support the development, training, and validation of autonomous urban driving systems. The simulation platform supports the flexible specification of sensor suites and environmental conditions. Leveraging the state-of-the-art rendering quality provided by Unreal Engine 4 (UE4), CARLA has been built for flexibility and realism. A third-person view in four weather conditions is shown in Figure 2.5 [56]. Most impressively, it provides several pre-built urban towns, and all the assets are free for non-commercial use. To support the interface between the world and the agent, CARLA is designed as a server-client system, and the socket API in Python is used for implementing the client. Many autonomous driving related research work [46][57][58][59][60] made use of the CARLA simulator and even built their own benchmarks on it.

In this work, we did not consider using CARLA because the community support for the Windows platform has not matured yet in 2019 and building a freeway scenario from the beginning could be a massive work. As introduced in Chapter 5, we may consider CARLA as the simulation platform for future work in the urban scenario.

Figure 2.5. A third person view in four weather conditions in CARLA.

Chapter 3: The Proposed System Architecture

3.1 The System Workflow

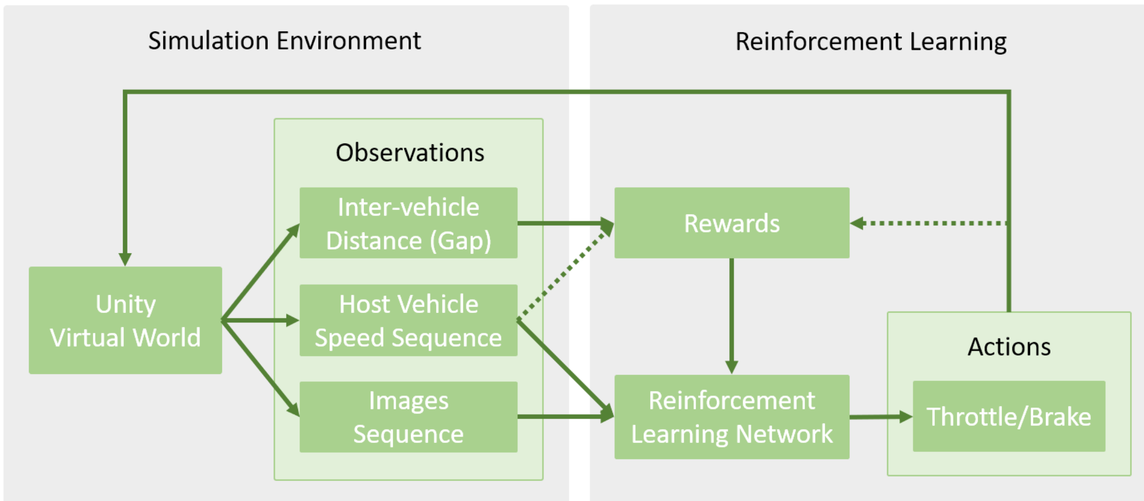

Figure 3.1. System workflow for training. The dashed lines are optional choices depending on the desired tasks.

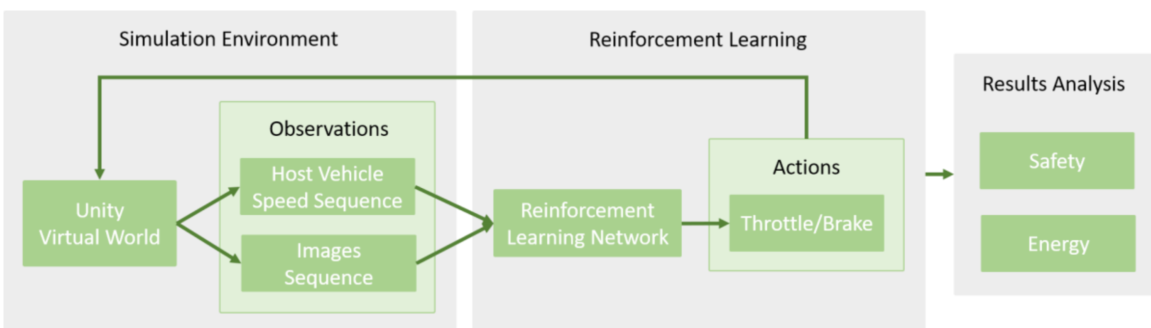

Figure 3.2. System workflow for testing.

As shown in Figure 3.1 and Figure 3.2, the proposed system consists of two stages, training and testing. During the training, the agent vehicle can access different kinds of observations from the simulation environment, depending on the type of tasks and scenarios. Some of them are used as the input of the RL network, such as image sequence, and others may be used for reward calculation. The model of reinforcement learning network should change in order to handle different tasks. The detailed model will be introduced in the following chapters. The simulation environment is frozen when waiting for the output from the reinforcement learning algorithm.

During the testing, at each time step, the agent utilizes only the front camera image sequences and other network-needed variables and sends them to the pre-trained network model to get the optimal control commands. After testing, we can analyze historical data from different aspects. For the safety aspect, the host vehicle should pass all test episodes without collision. For the energy aspect, the energy consumption of the host vehicle is calculated using the energy model mentioned in section 2.4. For the car following strategy we also evaluate the following ability by the root mean square error (RMSE) of inter-vehicle gap.

3.2 The Neural Network Architectures

As shown in Figure 3.1 and Figure 3.2, the core part is the reinforcement learning network. This part handles state value estimation and policy generation. We use the neural network as the function approximator, which mainly has two components. The first one is an image encoder (CNN) that reduces the visual input dimensions and extracts the features of the image sequence. Then the learned visual features and other observed variables are concatenated together as the whole states of the environment. The second component is the fully connected neural network, predicting the state value or the actions we need. We implemented DDQN and DDPG as the reinforcement learning algorithms in our system. The details of network architecture change slightly depending on different algorithms and will be introduced in the following.

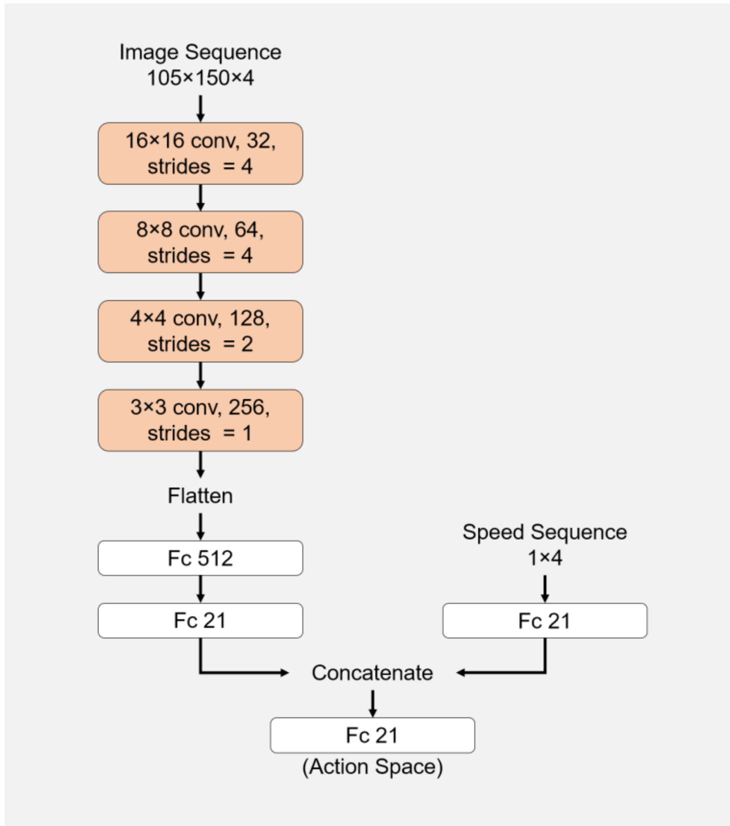

Figure 3.3. The RL network for DDQN.

The RL network for DDQN is shown in Figure 3.3. The image is processed by a convolutional neural network. In the car following scenario, we want the ego vehicle to learn the desired following distance from the image sequence. According to [61], CNN uses the vertical position of objects in the image to estimate their depth. In the freeway scenario, a simple CNN structure should be enough to get the depth feature from the input image sequence. In the network, the image input is a time-series sequence of 4 images with size 105 by 150 in grayscale, and the speed input is a time-series sequence of 4-speed values of the host vehicle. *Conv* represents the convolutional layer, and *fc* represents the fully connected layer. The 21 output values indicate the 21 Q-values of the discretized brake/throttle forces to the vehicle at the given state. We use smaller image input in a later stage to reduce the computational time and a smaller image encoder (see the encoder in Figure 3.4) is used correspondingly. The smaller encoder shows similar capability of feature extracting, and therefore it is used in DDPG algorithm as well.

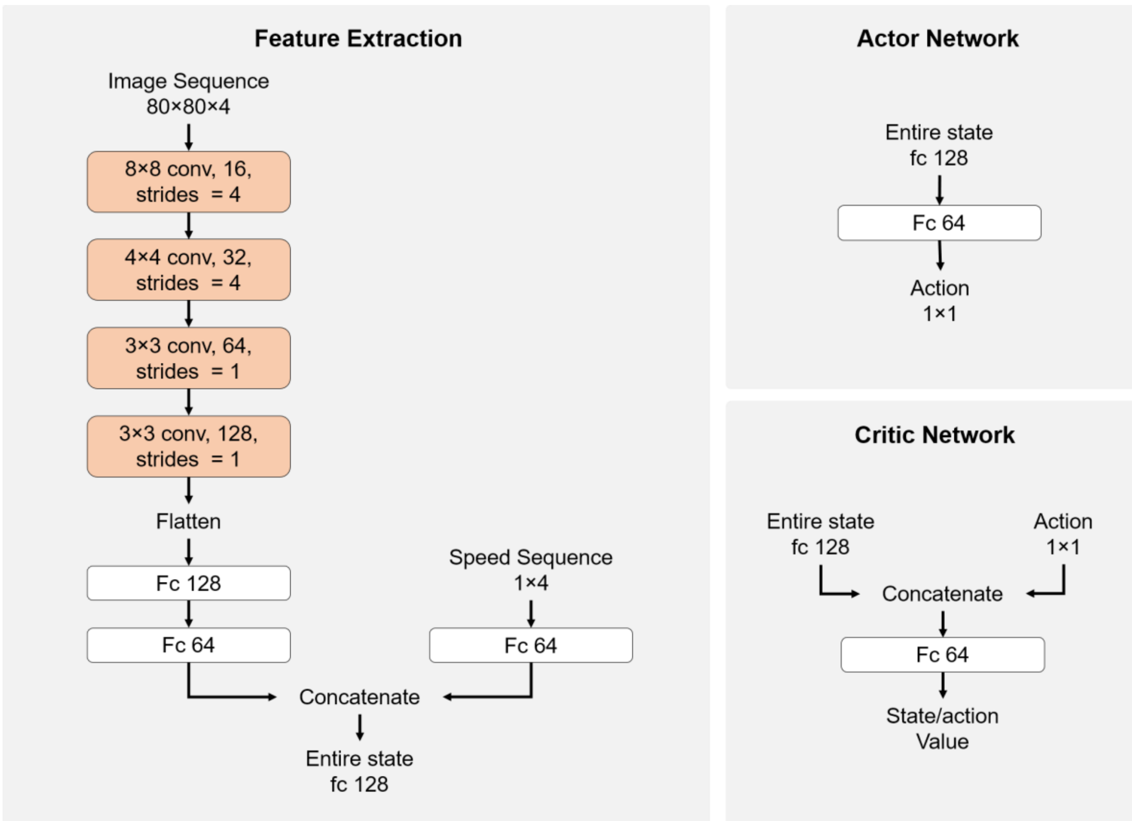

Figure 3.4. The RL network for DDPG.

The RL network for DDPG is shown in Figure 3.4.The same image encoder (*Conv* layers) is used in this algorithm. In the network, the camera inputs a time series sequence of 4 images with size 80-by-80 in grayscale. If two cameras are used in the system, the image sequences are concatenated along the third dimension and the total size of visual input will be $80 \times 80 \times 8$. The velocity state is also a time series of 4 velocity values of the ego vehicle. For actor and critic networks, [62] experimented with neural networks deeper than one hidden layer, but found out that they were unnecessary for small input variables. In this work, we only use a hidden layer containing 64 neurons and find it works well. All the convolutional layers and fully connected layers are followed by a rectified linear unit (ReLU) activation function. The final output layer of the actor network uses a *tanh* activation function to bound the output actions within the range of -1 to 1. Positive action indicates the force on the throttle while the negative indicates the force on the brake.

As mentioned in section 2.3.2.4, we use OU noise to ensure action exploration in the DDPG architecture since it ensures efficient exploration in physical environment. The parameters used in function (2.4.12) is shown in the Table 3-1.

Table 3-1. Parameters of OU noise.

Parameters	Value
$\boldsymbol{\theta}$	0.15
$\boldsymbol{\mu}$	0
$\boldsymbol{dt}$	0.5
$\boldsymbol{\sigma}$	0.125

3.3 The Simulation Environment Setup

In order to train and test our algorithms, we use the well-developed game engine Unity for the simulation environment. To be more realistic, we built a freeway scene with trees, buildings, and traffic flows, as shown in Figure 3.5. Shadows of all the objects in the environment can be changed with the global light. There are three lanes in each direction along the two-way freeway segment. The preceding vehicle and host vehicle are on the middle lane, while faster traffic flow is on the left lane, and slower traffic flow is on the right lane. The host vehicle is designed to always follow the same preceding vehicle, while the lane change maneuver is not allowed. All the objects can be controlled by Unity's C# scripting Application Programming Interface (API) to reach the desired states.

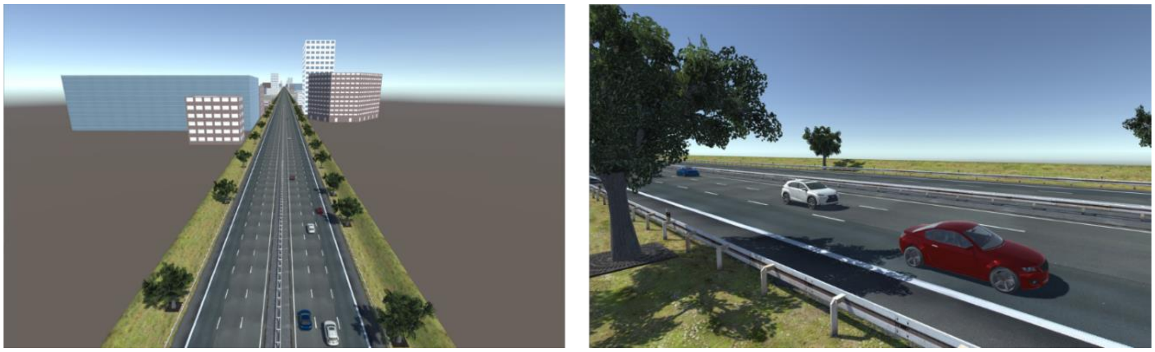

Figure 3.5. Two Sample Images of the Simulation Environment.

Specifically, the simulation we customized mainly provides the following five functions:

1. **Information Communication**: The information, such as velocity, following distance, and images, is transmitted to the reinforcement learning (RL) network by a socket API using the User Datagram Protocol (UDP). The suggested throttle/brake force is calculated and transmitted back to the simulation environment using the same API. The communication procedure is shown in Figure 3.6.
2. **Host Vehicle Control**: The host vehicle can be controlled by the suggested throttle/brake force from the RL network. It also can be controlled by other methods, such as the traditional ACC algorithm or the human-in-the-loop simulation.
3. **Preceding Vehicle Control**: The velocity data of the preceding vehicle is generated from a large pool of trajectories. Both virtual trajectories and real-world data are used during training and testing.
4. **Traffic Flow Control:** A higher constant speed is set to traffic flow on the left lane, and a lower constant speed is set to traffic flow on the right lane.

5. **Simulation Environment Reset**: When a collision occurs or the vehicles reach some preset terminal state, the current episode is terminated, and the simulation environment is reset with all the object settings to be new initial states for the next episode. This function will be executed also when the time step reaches the maximum episode length, since the length of freeway is finite. In this situation, however, the next state for the agent is not the terminal state.

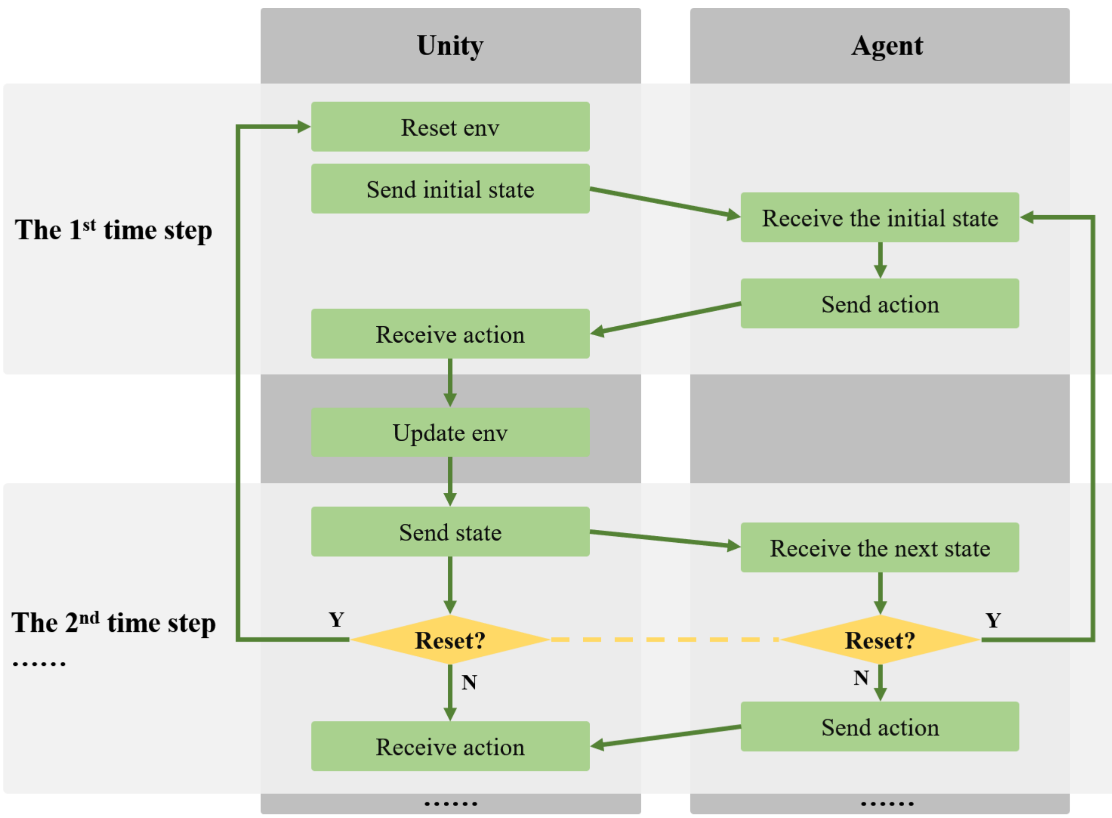

Figure 3.6. Communication procedure.

3.4 Data Acquisition

The offline training data in this system are mainly acquired during exploration in the simulation. The only data that must be preset before simulation is the driving trajectories of the preceding vehicle. For the research on ICE vehicles, we design virtually generated trajectories, which are created by an accumulation of 10 sinusoidal functions, shown as below:

$$v_{lead} = 30 + 0.3\sum_{i=1}^{10} A_i \sin(2\omega_i t) \tag{3.4.1}$$

where $A_i (i = 1,2, \dots ,10)$ are random integer values of 1 or -1, ω_i $(i = 1,2, \dots ,10)$ are random numbers between 0 and 1, and t is the elapsed time in the simulation environment with a resolution of 0.02 sec. During the training, the group of random 10 A_i's and 10 ω_i's will be updated when a new episode begins to make sure that the trajectories are different for each episode. During the testing, a preset 4 groups of A_i's and ω_i's are used to generate the trajectories for comparison.

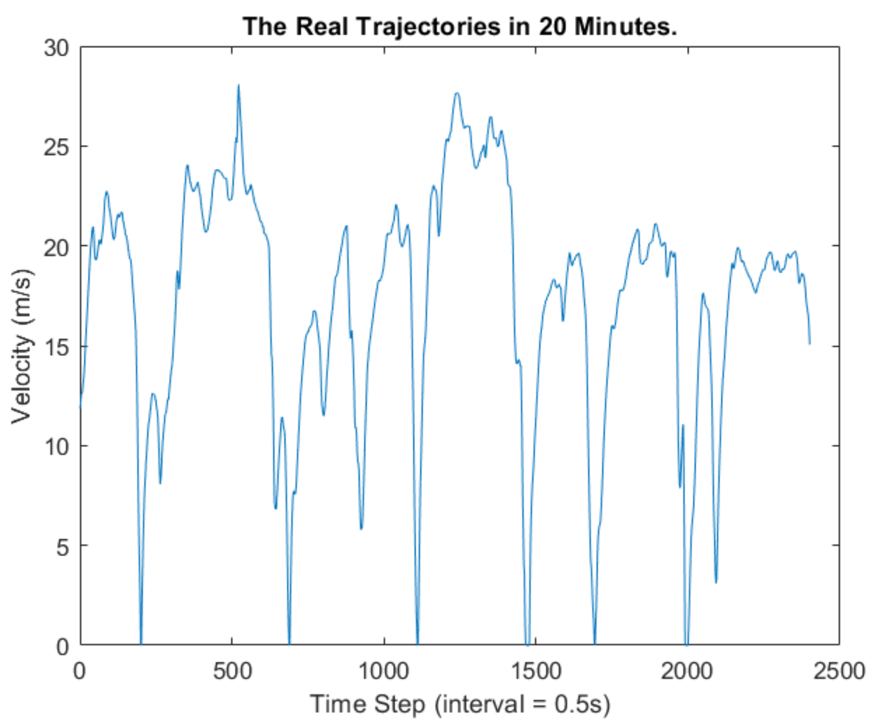

Figure 3.7. The real trajectories used for preceding vehicle.

For the simulation of EVs, the research team [7] collected more than 100 hours of electric vehicle driving data under real-world conditions using a 2012 Nissan Leaf. We make use of the data from a freeway segment (SR-91 in Riverside, California, USA) in the simulation. The original velocity data are segmented and regrouped to match our training and testing episode length. A sample of 20-minute real trajectories is shown in Figure 3.7.

Depending on the different scenarios for ICE vehicles or EVs in the simulation, the parameters of preceding vehicle (PV) and host vehicle (HV) are shown in the following table.

Table 3-2. Trajectory Parameters for ICE Vehicles and EVs in the Simulation.

Parameters	ICE Vehicles	EVs
PV velocity range (m/s)	[27, 33]	[0, 30]
PV acceleration range (m/s^2)	[-3.5, 3.5]	[-5.5, 3.5]
Initial PV velocity range (m/s)	[27, 33]	[11.2, 15.6]
Initial HV velocity range (m/s)	[25, 30]	[11, 16]
Initial following distance range (m)	[25, 35]	[25, 35]

Chapter 4: End-to-end Vision-Based Eco-Oriented Strategies for Freeway Scenarios

4.1 Introduction

In this chapter, end-to-end vision-based eco-driving assistance systems were implemented using DDQN and DDPG algorithms, respectively. We constructed the simulation environment in the Unity game engine. We trained and tested the proposed system for both internal combustion engine (ICE) vehicles and electric vehicles (EVs) to prove the effectiveness. The system only takes image input from one or two on-board front cameras and outputs throttle/brake to control the acceleration. The agent vehicle in our system is assumed to have lane keeping ability so that the throttle/brake is the only control signal (the output actions). Figure 4.1 shows an image captured by an on-board front camera in the freeway scenario.

Figure 4.1. A sample image of the freeway scenario.

We designed different reward functions to achieve two driving strategies. The first car-following strategy requires the agent vehicle to maintain a safe distance with a leading vehicle. We compared the energy efficiency of our system with the traditional ACC system and the human-in-the-loop driving system. The case study and results are illustrated in section 4.2. The other car-cruising strategy requires the agent vehicle to maintain a safe distance with the preceding vehicle and resume the preset speed when the distance is large enough or no predecessor is detected. The case study and results are introduced in section 4.3. An intuitive comparison between the two strategies is shown in Table 4-1 and Table 4-2.

Table 4-1. The simplified strategy for car following.

<table>
<tr><th>Gap</th><th>Policy</th></tr>
<tr><td>Close</td><td>Slow down</td></tr>
<tr><td>Far</td><td>Speed up</td></tr>
<tr><td colspan="2">According to the state, choose brutal force or gentle force.</td></tr>
</table>

Table 4-2. The simplified strategy for car cruising.

<table>
<tr><th>Gap</th><th>Relative motion</th><th>Velocity of agent</th><th>Policy</th></tr>
<tr><td rowspan="2">Close</td><td>Approaching</td><td rowspan="2">-</td><td>Slow down</td></tr>
<tr><td>Receding</td><td>Speed up</td></tr>
<tr><td rowspan="2">Far</td><td rowspan="2">-</td><td>Higher than v_d</td><td>Slow down</td></tr>
<tr><td>Lower than v_d</td><td>Speed up</td></tr>
<tr><td colspan="4">According to the state, choose brutal force or gentle force.
$\boldsymbol{v_d}$ is the preset desired speed.</td></tr>
</table>

4.2 Eco Car-Following Strategy

In this section, we implemented DDQN and DDPG algorithms in the proposed system to achieve an eco car-following strategy. We applied the system for both ICE vehicles and the EVs. The detailed case study and results are presented in the following subsections.

4.2.1 The DDQN Application to ICE Vehicles

Safety is our first concern. Therefore, we designed the gap reward to encourage the gap between 30 to 80m, and the desired gap of 55m gains the highest reward, shown in Figure 4.2 (left). We use this reward function to train our agent vehicle to learn the car following strategy. The trained model is named the *gap-based* DDQN model. During the training, the average reward is gradually increasing to 1, shown in Figure 4.3.a. It proves our model has the learning ability and learns a good (in terms of gap maintaining) following strategy.

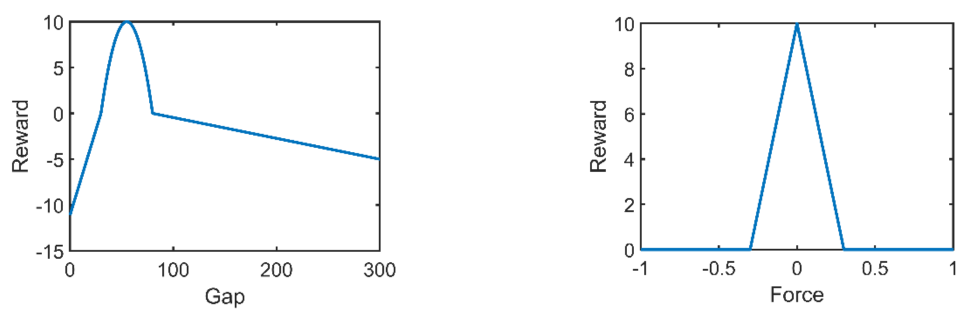

Figure 4.2. The gap reward (left) and force reward (right).

Beyond this safety rule, we designed the force reward to encourage the forces from -0.3 to 0.3, shown in Figure 4.2 (right). Intuitively, the less aggressive driving policy is expected to save more energy. We use the normalized sum of the force reward and gap reward to train our agent vehicle to learn the eco-oriented car following strategy. The trained model is named the *force-based* DDQN model. During the training, the average reward is gradually increasing to 0.9, shown in Figure 4.3.b. The training takes longer to converge due to more complex factors in the reward functions than the gap-based model.

In the testing, four trajectories with 4 minutes each are generated to evaluate our model. Besides, the traditional ACC and human-in-the-loop models are tested for comparison. Figure 4.3.c shows the gap profile of each model and Figure 4.3.d shows the velocity profile. The result shows that the *gap-based* DDQN model follows the desired gap most precisely. The strict driving policy makes the host vehicle more sensitive to speed fluctuation of the predecessor. On the other hand, the *force-based* model learns not to follow that precisely in order to fulfill the force saving objective. In addition, it learns a pulse and glide (PnG) driving pattern, which is considered as an eco-driving strategy for ICE vehicles [63].

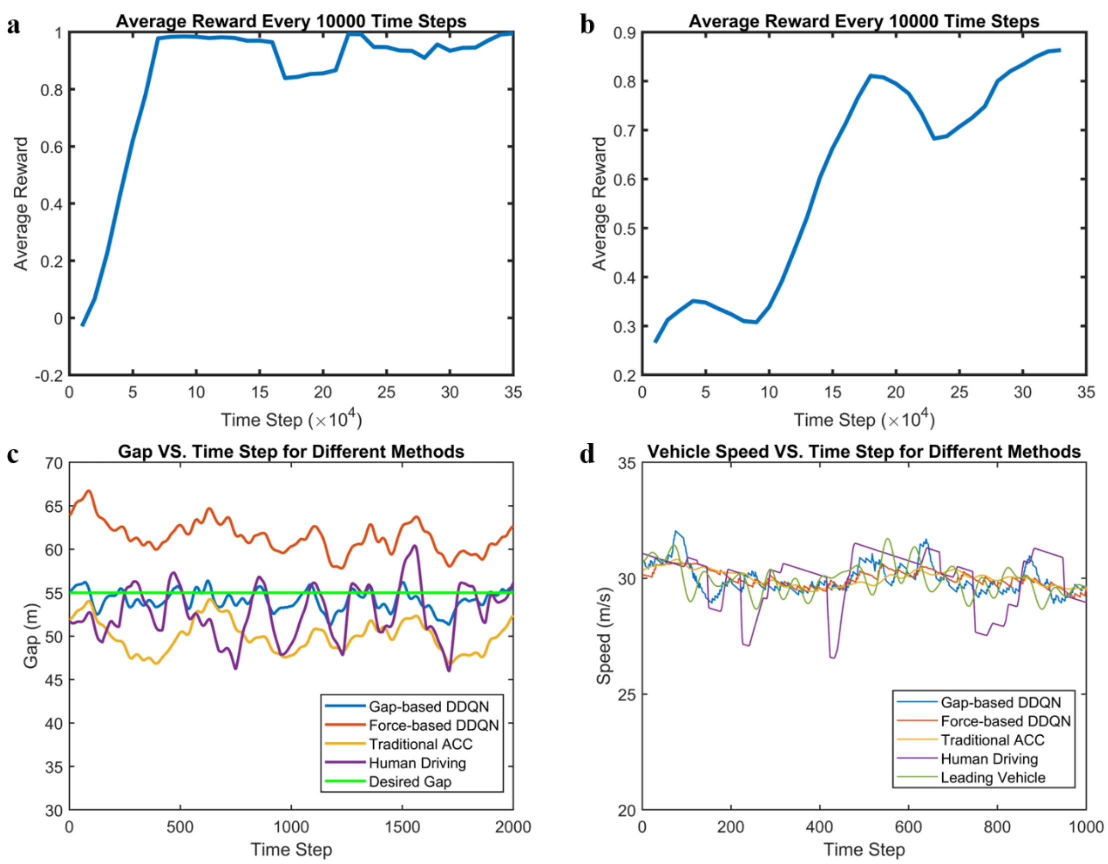

Figure 4.3. Training and Testing Result for ICE Vehicles.

The energy consumption, pollutant emissions, and gap root mean square error (RMSE) of all methods are shown in Table 4-3. As expected, The *gap-based* DDQN model results in the best gap regulation among all the methods. The *force-based* DDQN achieves the best fuel rate but has a similar performance as the traditional ACC method regarding both energy consumption and pollutant emissions.

Table 4-3. Energy Consumption and Pollutant Emission for Combustion Engine Vehicles between different methods.

Method	Leading Vehicle	Gap-based DDQN	Force-based DDQN	Traditional ACC	Human Driving
Fuel Rate (g/mi)	195.393	169.275	**143.754**	143.805	198.586
CO_2 (g/mi)	346.352	339.069	**330.681**	337.124	368.524
CO (g/mi)	143.214	99.003	58.119	**55.433**	143.391
HC (g/mi)	13.613	12.913	9.843	**9.443**	10.673
NO_x (g/mi)	4.656	3.159	1.841	**1.743**	4.500
Gap RMSE (m)	N/A	**3.423**	7.686	6.667	6.591

4.2.2 The DDQN Application to EVs

In this case, the real trajectories are adopted for the preceding vehicle. We use the same reward function designed for ICE vehicles; therefore, a gap-based DDQN model and a *force-based* DDQN model for EV are trained. Figure 4.4.a and b show the average training reward of each model, respectively. Again, we test and compare our trained model with traditional ACC and human-in-the-loop models. Figure 4.4.c and d show the gap profile and velocity profile of each method in testing. We use function (2.5.1) to measure energy consumption. The energy rate is the total energy normalized by the travel distance. The energy consumption and gap RMSE of all methods are shown in Table 4-4. The leading vehicle is the baseline in percentage energy rate comparison.

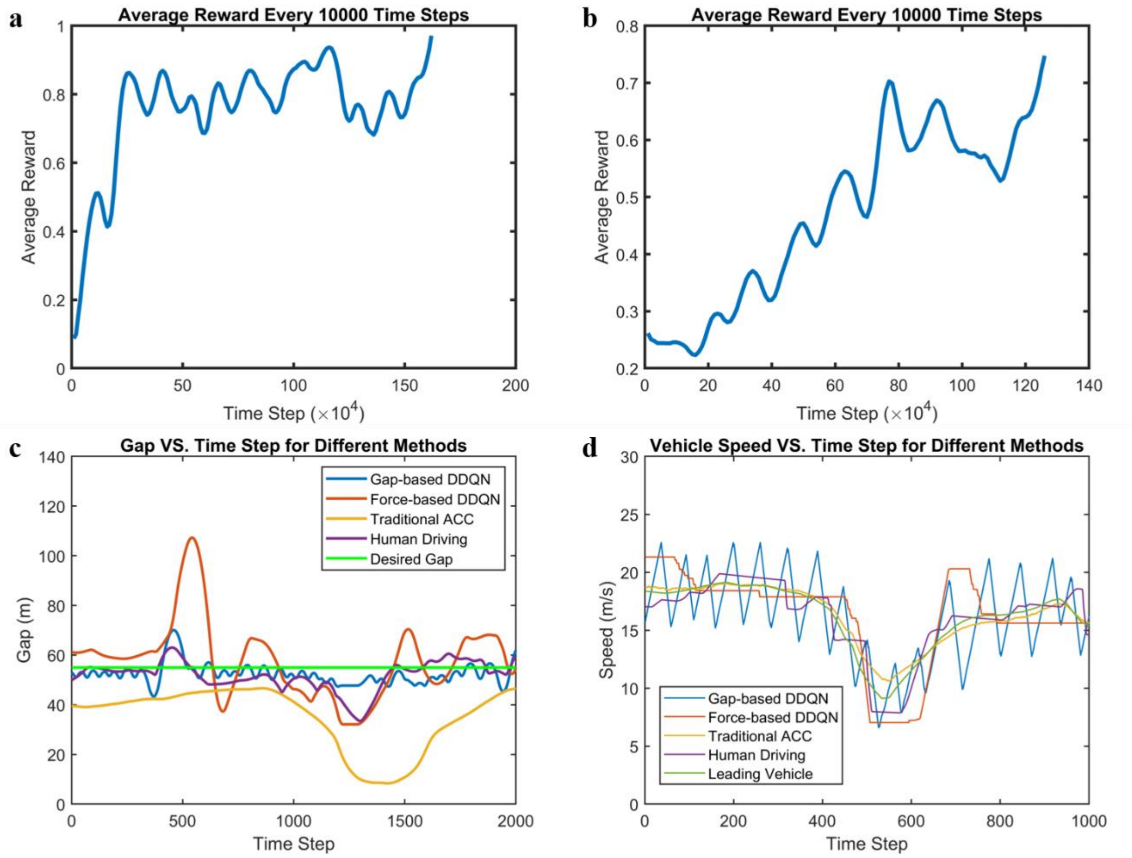

Figure 4.4. Training and Testing Result for EVs.

The same trend of training convergence in the ICE model can be observed in the EVs model. The training takes longer to converge since the real trajectories of the preceding vehicle complicated the experience in the memory buffer. Due to the more abrupt and frequent speed fluctuations, such a strict gap regulation strategy results in the highest EVs energy consumption among all the methods (much higher than others). Unlike the application in ICE vehicles, the *force-based* DDQN model does not result in better energy consumption for EVs compared to the performance of human-in-the-loop simulation or traditional ACC, although the PnG-like behavior is conducted (see Figure 4.4.d). As expected, the gap is quite loosely regulated by the *force-based* DDQN model.

Table 4-4. Energy consumption for EVs between different methods.

Method	Leading Vehicle	Gap-based DDQN	Force-based DDQN	Traditional ACC	Human Driving
Energy Rate (KJ/mile)	1.135e3	4.914e3	2.056e3	**1.128e3**	1.500e3
Percentage Energy Rate	100%	433%	181.1%	**99.4%**	132.2%
Gap RMSE (m)	N/A	**5.083**	10.98	17.440	5.439

Therefore, a new reward function is designed to explore the environmental performance for EVs model. The new reward is based on a looser safety gap reward R_{gap}, shown in Figure 4.5 (left). Rather than setting a fixed desired gap, R_{gap} gives the highest reward when the host vehicle maintains the gap between 40m to 60m and encourages gap between 30m to 100m. In the meantime, the energy-efficient policy is taught by the energy model (EM) reward R_{energy}, shown in Figure 4.5 (right). It is derived directly from the EVs' energy model. Also, the agent will get zero reward when the velocity is higher than 30m/s. When collision happens, or a large gap appears, the training episode will end, and the environment will reset to a random initial state. Finally, the reward in each time step is calculated by the following function:

$$R_t = \begin{cases} -1, & if\ collision\ happens\ or\ a\ large\ gap\ appears. \\ 0, & if\ velocity\ is\ higher\ than\ 30m/s. \\ R_{energy}R_{gap}, & otherwise. \end{cases}$$

The average reward with respect to time steps in the training process is shown in Figure 4.6. The reward is gradually increasing and converging to 0.9, which means that the host vehicle is learning the desired strategy. After the agent successfully learnt the eco-following strategy in the training, we tested the trained model. During a 150,000 time-steps of testing under the leading of a preceding vehicle with a different velocity profile, no collision happened. The ego vehicle can maintain the gap within the desired safe interval (40-60m) in 99.39% of the entire testing time. Result also proved the CNN in the network can extract useful gap information and ignore the motions of buildings, trees and other irrelevant vehicles in the camera view. The mask used in the previous work [64] is redundant.

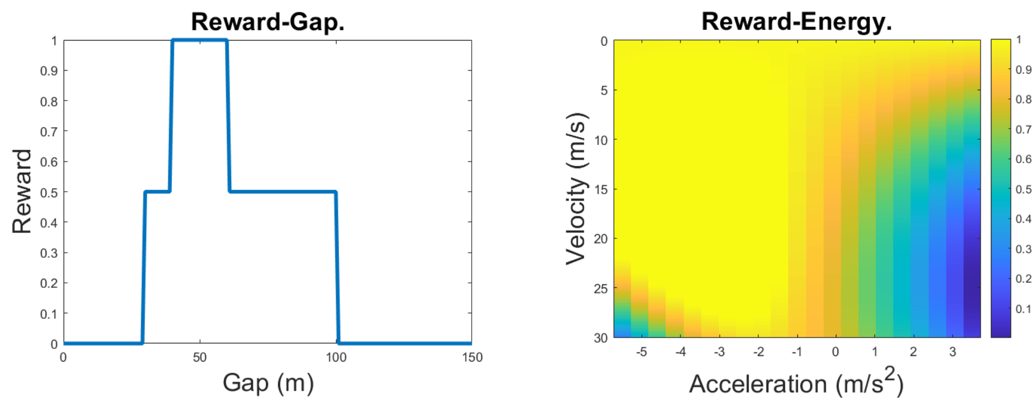

Figure 4.5. The gap reward (left) and energy reward (right) for EM-based DDQN.

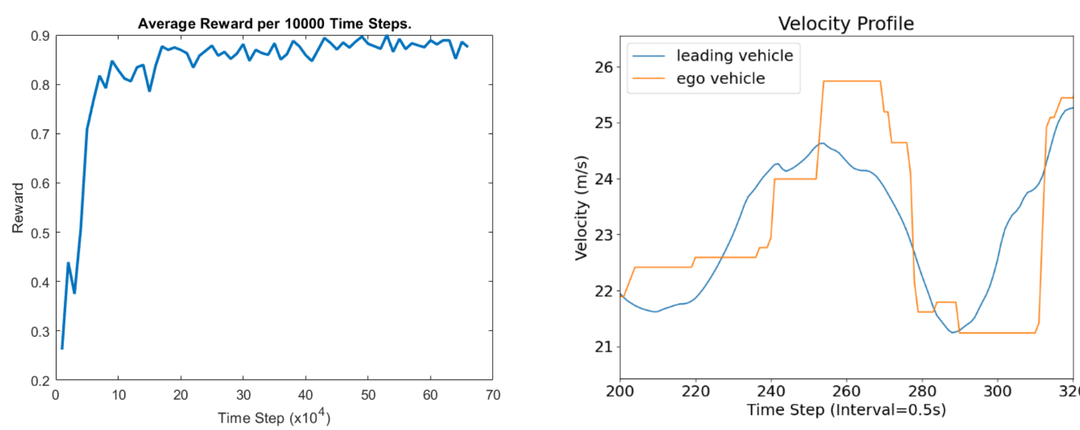

Figure 4.6. The average reward per 1000 time-steps during the training (left); A segment of velocity profile during the testing (right).

The energy consumption rate is shown in Table 4-5. The EM-based DDQN has about 40% improvement compared with the *force-based* DDQN. However, it still cannot beat the leading vehicle or the traditional ACC. From the velocity profile in Figure 4.6, the learnt policy only used brutal force to accelerate or decelerate the agent vehicle. The hypothesis is that this policy could correct the vehicle position immediately. The figure only shows a period. The whole picture of learnt policy can be represented by the force distribution (see Figure 4.7). We compared the force distribution of leading vehicle and EM-based DDQN. Although the agent vehicle learned to keep force to zero to earn a good energy reward, it still needs extreme forces to keep the safety gap. Those extreme forces enforced would explain why the agent vehicle did not have better energy consumption than the leading vehicle.

Table 4-5. Energy Consumption for EVs using EM-based DDQN.

Method	Leading Vehicle	EM-based DDQN
Energy Rate (KJ/mile)	1.243e3	1.793e3
Percentage Energy Rate	100%	144.2%

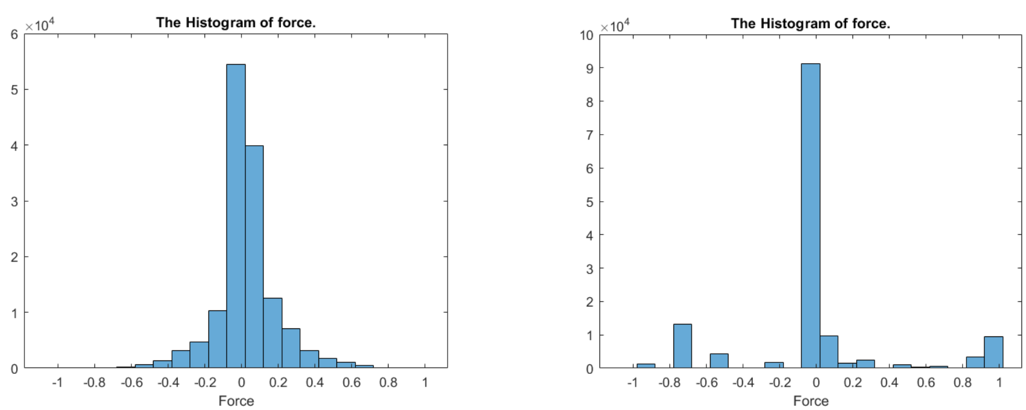

Figure 4.7. The force distribution during testing. The preceding vehicle (left); The agent (right).

4.2.3 The DDPG Application to EVs

To make sure the energy efficient potential of our end-to-end system is not limited by the discretized action space, an Actor-Critic architecture DDPG was implemented.

In the past training, we expected the agent to learn the gap information from visual input. Within a single camera view, the leading vehicle becomes smaller when gap is increasing. Especially when the gap is larger than 50m, the gap changing presents few differences in the image. In a result, when the vehicle is exploring within a large gap range, the memory buffer in the reinforcement learning system may be full of image sequences without any information. This complicated the training and even made the training failure. Although we can adjust the focal length of the single onboard camera, a fixed focal length cannot handle both the close-up view and the far view. We could also track the leading vehicle object and adjust the focal length intelligently. However, the motion of camera view will easily conflict with the relative motion information of two vehicles in the image. Therefore, we use dual-camera setup in the system. We add a telephoto lens on the ego vehicle.

The ego vehicle is driven by the force via the built-in function in Unity mentioned in section 2.5.2. We tested the function and the range of acceleration is $[-10, 5] m/s^2$. Other simulation environment setup is the same as our previous method. Only real trajectories are used for leading vehicle.

Based on the previous experience, the goal that always maintains a strict safety gap is conflicting with the goal to save energy. Therefore, inspired by the CACC system on the market, we designed a new R_{gap} related to both gap and ego velocity:

$$R_{gap} = \begin{cases} 1 & if\ v + safe\_gap < gap < 3v + 2safe\_gap \\ 0 & otherwise \end{cases}$$

We call it adaptive-gap reward, shown in Figure 4.8. The old R_{gap} is called strict gap reward. Then combining these two kinds of R_{gap} with the energy model (EM) reward R_{energy}, we have four different methods. We use the following names to represent each method, (1) Strict gap; (2) Strict gap + EM; (3) Adaptive gap; (4) Adaptive gap + EM. Note, the reward function of Strict gap + EM method is exactly the same as that of EM-based DDQN.

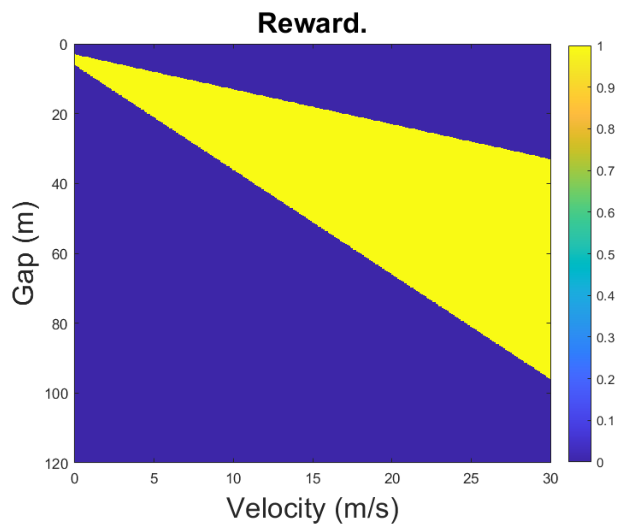

Figure 4.8. The adaptive gap reward function (safe_gap = 3m).

The training result is shown in Figure 4.9. We can see that the dual-camera system learned the desired policy before 40,000 time-steps. It is quite faster than the single camera system, which learnt the policy after around 100,000 time-steps.

The testing has 25 episodes and 600 time steps in each episode, which means 2.08 hours driving on the freeway. The agent vehicle passed all episodes without collision. From the velocity profile shown in Figure 4.10, the traditional ACC and the adaptive gap + EM curves looks smoother than the other two curves. The adaptive gap method failed to learn a smooth velocity curve since the force is not regulated by the EM reward. Although the strict gap + EM method has the EM reward, the agent did not learn a smooth velocity curve since the gap reward is too strict. From the gap profile shown in Figure 4.11, the strict gap + EM achieves the best gap maintaining ability.

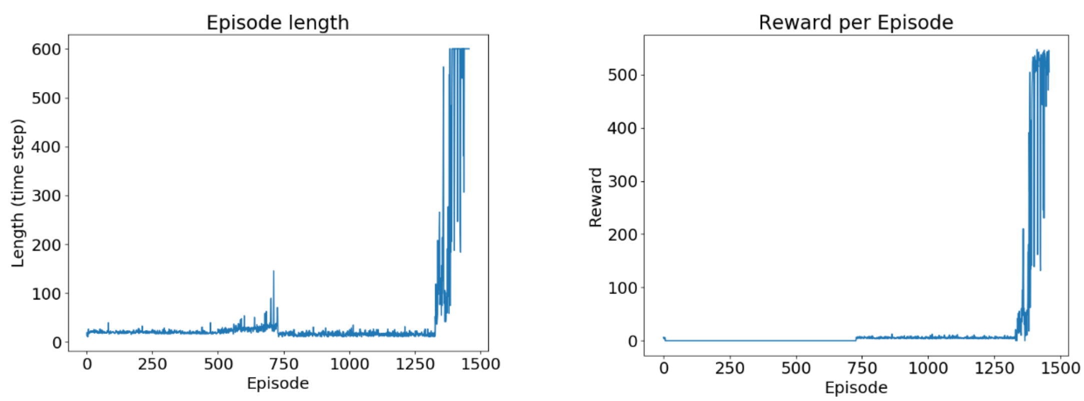

Figure 4.9. The training result of DDPG method.

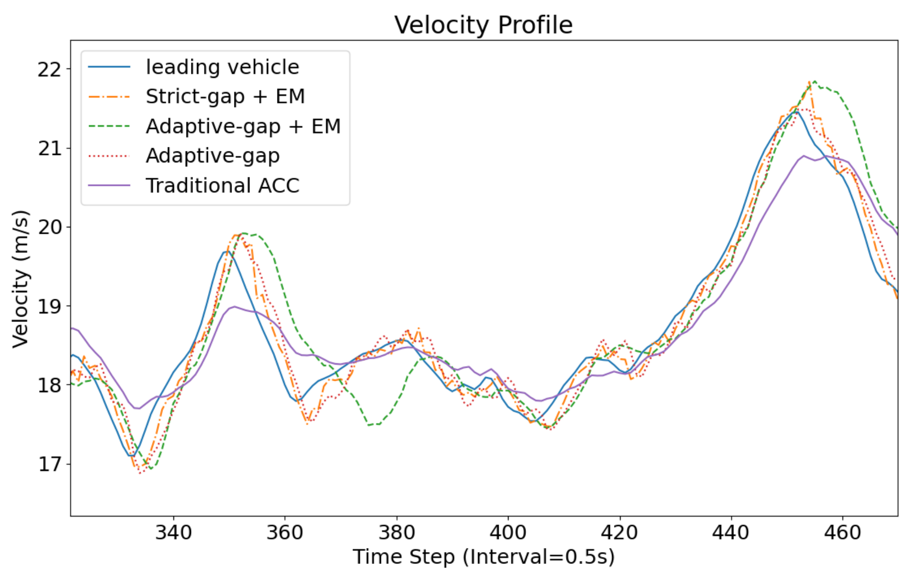

Figure 4.10. The velocity profile of each method during a period of testing.

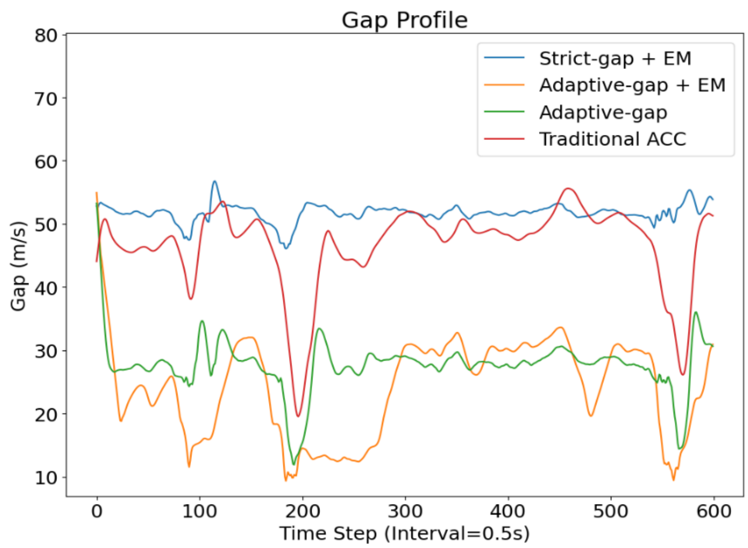

Figure 4.11. The gap profile of each method during an episode in testing.

Again, we use the EVs energy model to calculate the energy consumption normalized by the travel distance. The result is shown in Table 4-6. The adaptive gap + EM method achieves the best energy rate among four DDPG methods. All new methods have lower consumption than the DDQN methods. When we use the same reward function (Strict gap + EM) for DDQN and DDPG system, the later has a 14.5% improvement on the percentage energy rate. However, none of them can beat the traditional ACC system. The acceleration distributions are shown in Figure 4.12. Since the action is continuous and the range of acceleration is within [-10, 5], the bins of histogram are set to 150. The distributions quite match those of energy consumption. Comparing c with e or d with f, the EM reward reduces some of the extreme force. Comparing e with f, the adaptive gap reward narrows the acceleration distributions since the strict gap is more difficult to maintain. While from all figures, the traditional ACC has the most similar pattern as the leading vehicle and even has a bit narrower shape. Few extreme forces are shown in the pattern. Combining these results with the gap profiles in Figure 4.11, relatively flexible gap maintaining seems to be the way for eco-driving. Thus, we did not compare the gap RSME here.

Table 4-6. Energy Consumption for EVs using different DDPG methods.

Method	Leading Vehicle	Strict gap + EM	Adaptive gap	Adaptive gap + EM	Traditional ACC
Energy Rate (KJ/mile)	1.243e3	1.596e3	1.671e3	1.508e3	1.228e3
Percentage Energy Rate	100%	128.4%	134.4%	121.3%	98.7%

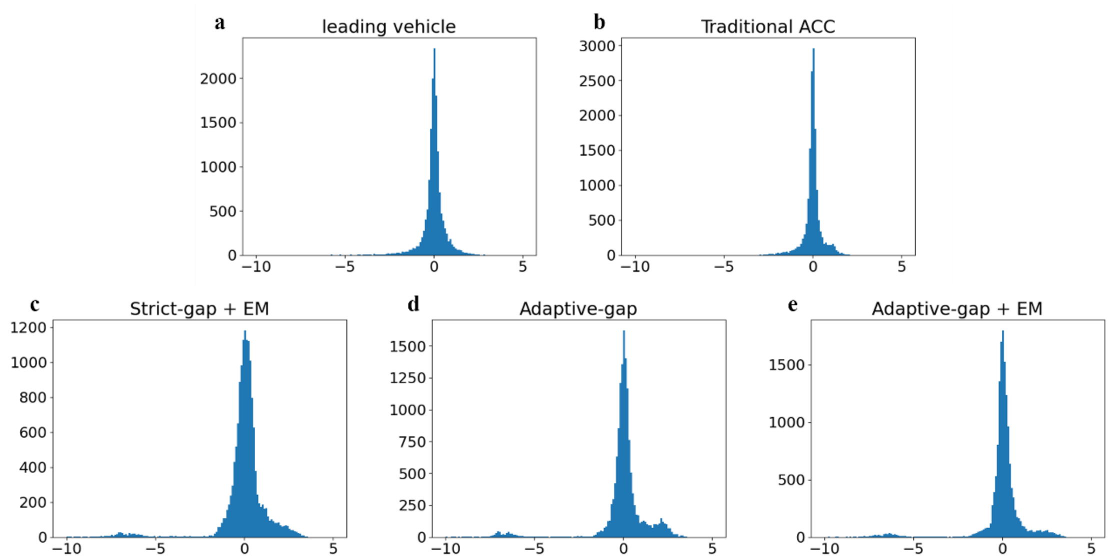

Figure 4.12. The acceleration distribution of different system in testing.

4.2.4 Summary and Discussion

We implemented an end-to-end vision-based car-following system using different reinforcement learning methods. We constructed a simulation environment in the game engine Unity and used socket connection to communicate with our reinforcement learning system. The inference time of 3.58 ms indicates the real-time working ability of the proposed methods.

We implemented DDQN algorithm in the proposed system. We designed gap-based and force-based reward functions for training ICE vehicles and EVs. Training and testing results over virtually generated and real-world driving trajectories show the effectiveness and robustness of the following ability. For the ICE model, the force-based following system only achieves a better energy-saving performance tested on virtually generated trajectories. However, the system failed to save energy for EVs. We designed an energy model-based reward function to constrain the large force output. The method achieved some improvement in energy consumption due to the strict gap constrain.

We implemented DDPG algorithm in the proposed system to further explore the energy efficient potential for EVs. Compared with the previous application, the system improves the smoothness of the velocity due to continuous action space and has a 14.5% improvement in energy rate. We then weakened the gap restriction by involving a new reward function called adaptive gap reward. The model trained by the adaptive gap reward and energy model-based reward has another 7.1% improvement in energy rate. However, the performance is still worse than the traditional ACC system due to more brutal forces. Future efforts on the proposed vision-based end-to-end system are needed to further squeeze the energy efficiency.

The visual input and ego velocity during the past 1~4 seconds may be enough to safely drive the vehicle. However, we expect the system can optimize the energy consumption over long-term historical information. In other words, we hope the system can learn how to optimize the total energy consumption in a whole episode rather than only depending on the instant state. Therefore, we consider using recurrent neural network (RNN) structures, such as Long short-term memory (LSTM).

4.3 Eco-Cruising Strategy

Due to the better result of DDPG algorithm in the previous research work, we implemented it in the proposed system to achieve an eco-cruising strategy. We applied the system for ICE vehicles. Only the real trajectories are used for the preceding vehicle.

The desired cruising speed is set to 17.88 m/s (40 mph) according to the automobile fuel consumption and emission rate curves w.r.t. to the average speeds [65]. The curve indicated the minimum rate at moderate speeds of around 40 to 50 mph. A similar result was also found by Natural Resources Canada. Most cars, vans, pickup trucks, and SUVs are most fuel-efficient when they’re traveling between 30 to 50 mph. Increasing highway cruising speed from 55mph to 75mph can raise fuel consumption by more than 20% [66]. In addition, with the preceding vehicle driving under the real trajectories and the agent vehicle cruising at 17.88 m/s in the simulation, the stop-and-go and cruising situations are balanced. It makes the training set informative enough.

The detailed case study and results are presented in the following section.

4.3.1 The DDPG Application to ICE Vehicles

The first reward is quite straightforward, calculated by the following function:

$$R_t = \begin{cases} 1 & if\ v_d - 0.5 < v < v_d + 0.5 \\ 0 & otherwise \end{cases}$$

where v_d is the desired velocity.

The reward simply encourages the agent vehicle to maintain its velocity to the desired value we set. We expected the agent can learn to avoid collision by the termination and reset in the simulation. However, the agent did not learn the desired strategy and sometimes still collided the leading vehicle. We thought it was because the policy confliction. When the ego vehicle is approaching the leading vehicle, maintaining the desired speed is conflict with avoiding the collision.

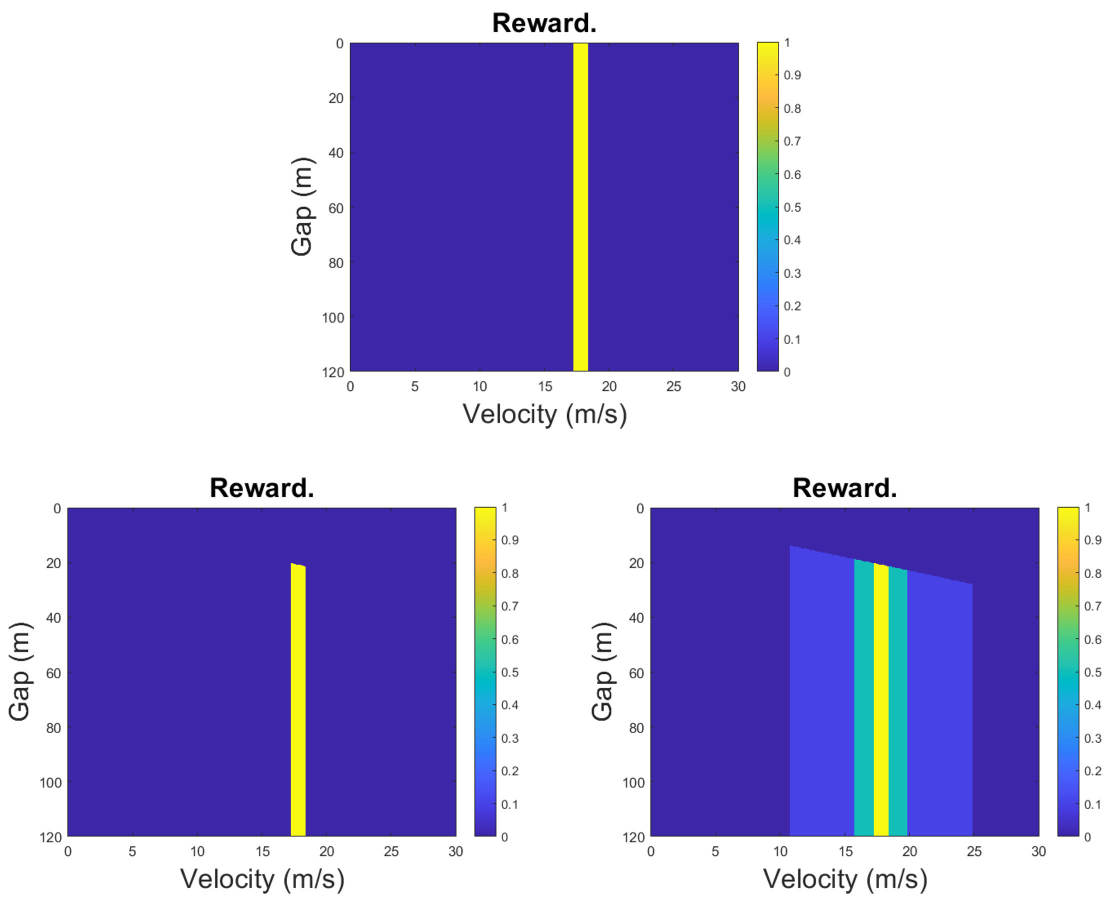

Figure 4.13. The reward functions for eco-cruising strategy.

Therefore, we designed the second reward function:

$$R_t = \begin{cases} 1 & if\ v_d - 0.5 < v < v_d + 0.5 \\ 0 & otherwise\ or\ gap < v + safe\_gap \end{cases}$$

where gap is the distance between leading vehicle and ego vehicle; $safe\_gap$ is the minimum allowed gap, which is 3m in the experiment. Inspired by the adaptive gap reward used in Chapter 4, this reward function enables the agent vehicle to ignore the desired speed when gap is too small. This method failed again due to its sparsity. At the beginning exploration stage of training, the vehicle merely got full reward. As a result, the valuable states were too sparse in the memory making the learning impossible.

To solve the sparse reward problem, we designed the following reward function:

$$R_t = \begin{cases} 1 & if\ v_d - 0.5 < v < v_d + 0.5 \\ 0.5 & else\ if\ v_d - 2 < v < v_d + 2 \\ 0.1 & else\ if\ v_d - 7 < v < v_d + 7 \\ 0 & otherwise\ or\ gap < v + safe\_gap \end{cases}$$

Figure 4.13 visualizes the three reward functions and the sparsity comparison is obvious. As shown in Figure 4.14, the results indicate that the agent vehicle successfully learned the desired driving policy on the training dataset. The driving pattern shown in the velocity profile lasts more than a hundred episodes (around 8 hours driving in real-time) in training. After that, the policy crashed due to catastrophic forgetting. The model is overfitted on the sampled dataset in the memory buffer.

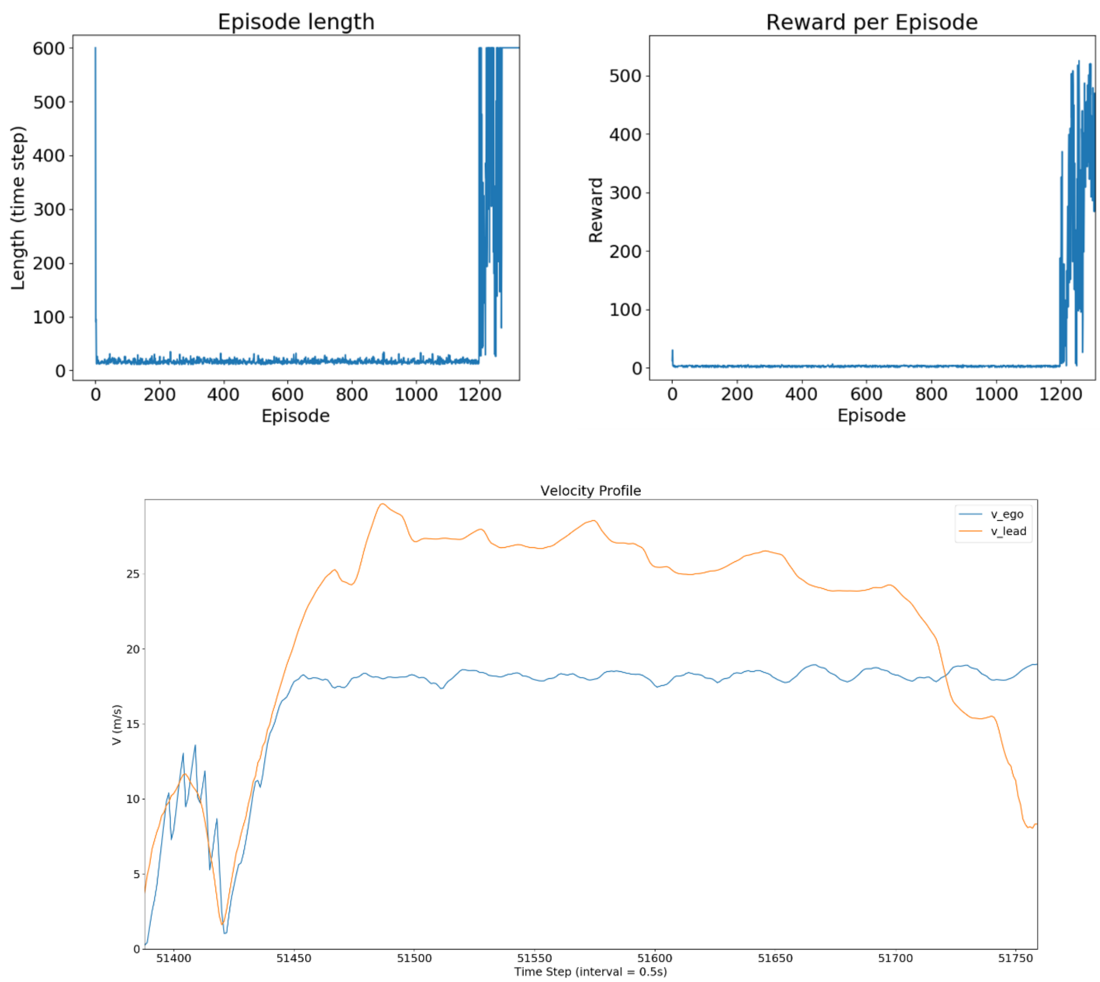

Figure 4.14. The training result for car cruising strategy.

4.3.2 Summary and Discussion

The DDPG algorithm was applied to the proposed system to realize an eco-oriented car cruising strategy. The details are mainly focused on the design of reward functions. The case study shows a robust car cruising behavior on the training set. The validation work and the application to EVs needs some future work.

There are better ways to solve the sparse reward problem besides directly modifying the reward function. Savinov et al. involved the Episodic Curiosity module into the reward calculation, and the new reward was proven to be dense [67]. Andrychowicz et al. proposed the Hindsight experience replay (HER) algorithm, which involved intermediate goals in the experiences so that the agent would learn through the failure as well rather than only from the sparse successful case [68]. Moreover, the priority memory buffer was proposed in [69] to ensure efficient learning. Rather than randomly sampling training data from the replay buffer, it sampled the memory with larger loss value firstly. This solution could ensure the agent to learn the most informative experience.

The proposed system can only achieve cruising strategy at a fixed preset speed. Although we can train multiple models for different speeds, the command-conditional learning architecture [46] may be considered as a better and comprehensive solution. The desired speed is input as a user-defined command (as currently ACC does) into the system, and the agent can learn to change policies depending on that command speed.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

This thesis provides end-to-end solutions for advanced eco-oriented driving assistance strategy on the freeway scenario using vision-based reinforcement learning architectures.

From the end-to-end system aspect, the proposed system learns desired driving strategies from image input directly to the low-level control of throttle/brake actions. We designed workflows for training and testing in the Unity simulation environment and provided complete communication procedures to ensure interactions between the environment and the RL agent. The proposed architecture is compatible with different RL networks and reward functions requiring data from multiple sensors. The memory buffer supports different types and lengths of data for offline training. To improve the training efficiency, offline training and online data collection are running in the meantime. The inference time of 3.58 ms indicates the real-time applicability of the proposed system.

From the advanced driving assistance aspect, the proposed eco-oriented car following strategy and cruising strategy were trained successfully on our system. For the car following strategy, the results show the effectiveness and robustness of the proposed system. We improved the eco-performance of the proposed system after several optimizations on the reward function. The achievement is still less comparable to the traditional ACC system with radar. The possible reasons and solutions are discussed in section 4.2.4. Further research work is needed to explore the environmental potential of the proposed system. For the cruising strategy, the results show the learning ability of the proposed system. Further work is needed to validate the trained model.

From the vision-based aspect, we used one or two front onboard cameras as the major sensor for the environment perception, which is much cheaper than the RADAR/LiDAR system. The necessary gap and relative motion information can be learned from the raw image of the preceding vehicle with the road, sky, trees, buildings, and surrounding vehicles.

From the RL architectures aspect, we implemented DDQN and DDPG algorithms for our system to learn from the experiences offline. The classic RL algorithms are modified to be capable of the vision-based driving system. The performance of two algorithms for the same task is compared in Chapter 4 and DDPG has better performance in our cases due to the continuous action space.

5.2 Future Research Directions

5.2.1 Urban Scenario

The advanced driving assistance system can not only improve the driving experience on the freeway but also in the urban scenario. The urban scenario is much more complicated than the freeway scenario due to the narrower street, denser traffic, interruption of pedestrian, and other unpredictable events. Human drivers pay more attention to urban areas and require more driving assistance to improve their safety and travel efficiency. Besides, the urban miles per gallon (mpg) for an ICE vehicle is usually higher than the freeway mpg due to the frequent stop and go behavior. Consequently, the eco demand in urban is much higher, especially for the city's public transportation system. Therefore, to build an urban scenario in the simulation and to verify the effectiveness of the proposed architecture for comprehensive tasks should be one of the future research directions. The multi-tasks learning architectures mention in section 2.3.2.5 may be considered.

There are some related works for reference. For example, taking advantage of V2X and the signal phase and timing (SPaT) information, Hao et al. proposed an eco-approach and departure (EAD) application for signalized intersections with real-world traffic. The EAD system can save 6% energy for the trip segments when activated within DSRC ranges and 2% energy for entire trips. It can also reduce 7% of CO, 18% of HC, and 13% of NOx for all trips [70]. Bai et al. presented a hybrid reinforcement learning (HRL) framework for connected eco-driving at signalized intersections [71]. The method achieved 1 – 7% travel time saving and 12 – 47% energy consumption saving compared with the intelligent driver model (IDM).

5.2.2 From the Simulation to the Real World

Simulation can be the perfect tool for training and testing the machine learning algorithm for the real-world driving problem. The method succeeded in the simulation, however, still needs the further validation in the real world. If the proposed system cannot work for real-world testing directly, transfer learning may be considered for migration. This future research direction also has related works, such as [72][73].

As aforementioned in Chapter 1, the perception and decision making in the comprehensive scenarios challenge the commercial-grade autonomous driving. Achieving Level 5 automation based on only the host vehicle is nearly impossible. Researchers and engineers are pinning their hope on CAVs using V2V or V2X techniques. Before going to implement the CAVs and the V2X infrastructure, building a mini-city with mobile robots can be a surrogate "real-world simulation platform". The Duckietown project was conceived in 2016 at MIT. The goal was to build a small-scale platform and preserve the real scientific challenges inherent in a full-scale real autonomous robot platform [74]. Similarly, the TSR lab in CE-CERT is going to establish a mini-city starting from the AutoTrac Challenge [75]. Training and validating our system on the mini-city using robots is considered as another future research direction.

Reference

[1] SAE On-Road Automated Vehicle Standards Committee. "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems." *SAE Standard J 3016* (2014): 1-16.

[2] Google self-driving car project, 2015: https://medium.com/waymo/scenes-from-the-street-5bb77046d7ce#.tq11yyoqw

[3] Waymo to start driverless ride sharing in Phoenix area, 2017: https://www.azcentral.com/story/money/business/tech/2018/01/30/waymo-start-driverless-ride-sharing-phoenix-area-year/1078466001/

[4] Tesla Autopilot: https://www.tesla.com/support/autopilot

[5] BMW Personal CoPilot: https://www.bmw.com.mt/en/topics/fascination-bmw/bmw-autonomous-driving.html

[6] U.S. EPA: https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions

[7] Wu, Guoyuan, Kanok Boriboonsomsin, and Matthew J. Barth. "Eco-routing navigation system for electric vehicles." *Center for Environmental Research and Technology, University of California, Riverside, 1084 Columbia Ave., Riverside, CA 92507* (2014).

[8] U.S. EIA: https://www.eia.gov/tools/faqs/faq.php?id=427&t=3

[9] The Automotive LiDAR Market: http://www.woodsidecap.com/wp-content/uploads/2018/04/Yole_WCP-LiDAR-Report_April-2018-FINAL.pdf

[10] Wang, Yan, et al. "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[11] You, Yurong, et al. "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving." *arXiv preprint arXiv:1906.06310* (2019)

[12] Okuyama, Takafumi, Tad Gonsalves, and Jaychand Upadhay. "Autonomous driving system based on deep Q learnig." *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. IEEE, 2018.

[13] McAllister, Rowan, et al. "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning." *International Joint Conferences on Artificial Intelligence, Inc.*, 2017.

[14] Tesla Accident 2016: A Tragic Loss. https://www.tesla.com/blog/tragic-loss

[15] Yurtsever, Ekim, et al. "A survey of autonomous driving: Common practices and emerging technologies." *IEEE Access* 8 (2020): 58443-58469.

[16] Cai, Peide, et al. "Vision-based trajectory planning via imitation learning for autonomous vehicles." *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019.

[17] Bojarski, Mariusz, et al. "Explaining how a deep neural network trained with end-to-end learning steers a car." *arXiv preprint arXiv:1704.07911* (2017).

[18] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." *arXiv preprint arXiv:1604.07316* (2016).

[19] Sallab, Ahmad EL, et al. "Deep reinforcement learning framework for autonomous driving." *Electronic Imaging* 2017.19 (2017): 70-76.

[20] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

[21] Wang, Ziran, Guoyuan Wu, and Matthew J. Barth. "A review on cooperative adaptive cruise control (CACC) systems: Architectures, controls, and applications." *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018.

[22] Kesting, et al. "Adaptive cruise control design for active congestion avoidance." *Transportation Research Part C: Emerging Technologies* 16.6 (2008): 668-683.

[23] Luo, Lihua. "Adaptive cruise control design with consideration of humans' driving psychology." *Proceeding of the 11th World Congress on Intelligent Control and Automation.* IEEE, 2014.

[24] Shi, Jiangchao, and Jian Wu. "Research on Adaptive Cruise Control based on curve radius prediction." *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 2017.

[25] Wang, Ziran, Guoyuan Wu, and Matthew J. Barth. "Developing a distributed consensus-based cooperative adaptive cruise control system for heterogeneous vehicles with predecessor following topology." *Journal of Advanced Transportation 2017* (2017).

[26] Sheikholeslam, Shahab, and Charles A. Desoer. "Longitudinal control of a platoon of vehicles." *1990 American control conference*. IEEE, 1990.

[27] Song, Shuran, and Jianxiong Xiao. "Sliding shapes for 3d object detection in depth images." *European conference on computer vision*. Springer, Cham, 2014.

[28] Wang, Dominic Zeng, and Ingmar Posner. "Voting for voting in online point cloud object detection." *Robotics: Science and Systems*. Vol. 1. No. 3. 2015.

[29] Yang, Bin, Wenjie Luo, and Raquel Urtasun. "Pixor: Real-time 3d object detection from point clouds." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.

[30] Chen, Xiaozhi, et al. "Multi-view 3d object detection network for autonomous driving." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[31] Liang, Ming, et al. "Deep continuous fusion for multi-sensor 3d object detection." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[32] Wawrzynski, Pawel. "Control policy with autocorrelated noise in reinforcement learning for robotics." *International Journal of Machine Learning and Computing* 5.2 (2015): 91.

[33] Aly, Mohamed. "Real time detection of lane markers in urban streets." *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008.

[34] Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

[35] Pan, Xingang, et al. "Spatial as deep: Spatial cnn for traffic scene understanding." *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[36] Teichmann, Marvin, et al. "Multinet: Real-time joint semantic reasoning for autonomous driving." *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018.

[37] Muller, Urs, et al. "Off-road obstacle avoidance through end-to-end learning." *Advances in neural information processing systems*. 2006.

[38] Hadsell, Raia, et al. "Learning long-range vision for autonomous off-road driving." *Journal of Field Robotics* 26.2 (2009): 120-144.

[39] Chen, Chenyi, et al. "Deepdriving: Learning affordance for direct perception in autonomous driving." *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[40] François-Lavet, Vincent, et al. "An introduction to deep reinforcement learning." *arXiv preprint arXiv:1811.12560* (2018).

[41] Watkins, Christopher John Cornish Hellaby. "Learning from delayed rewards." (1989).

[42] Watkins, C. J. and P. Dayan. 1992. “Q-learning”. *Machine learning.* 8(3-4): 279–292.

[43] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.

[44] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." *Thirtieth AAAI conference on artificial intelligence*. 2016.

[45] McCloskey, Michael, and Neal J. Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem." *Psychology of learning and motivation*. Vol. 24. Academic Press, 1989. 109-165.

[46] Codevilla, Felipe, et al. "End-to-end driving via conditional imitation learning." *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[47] Tessler, Chen, et al. "A deep hierarchical approach to lifelong learning in minecraft." *arXiv preprint arXiv:1604.07255* (2016).

[48] Chamberlin, Robert, et al. Analysis of MOVES and CMEM for evaluating the emissions impact of an intersection control change. No. 11-0673. 2011.

[49] Shibata, Seishiro, and Tsuguhiko Nakagawa. "Mathematical model of electric vehicle power consumption for traveling and air-conditioning." *Journal of Energy and Power Engineering* 9.6 (2015): 269-275.

[50] Ye, Fei, et al. "A hybrid approach to estimating electric vehicle energy consumption for ecodriving applications." *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016.

[51] Scora, George, et al. "Real-time roadway emissions estimation using visual traffic measurements." *2011 IEEE Forum on Integrated and Sustainable Transportation Systems*. IEEE, 2011.

[52] B. Wymann, E. Espi´e, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. TORCS, The Open Racing Car Simulator. http://www.torcs.org.

[53] Xiong, Xi, et al. "Combining deep reinforcement learning and safety based control for autonomous driving." *arXiv preprint arXiv:1612.00147* (2016).

[54] Shah, Shital, et al. "Airsim: High-fidelity visual and physical simulation for autonomous vehicles." *Field and service robotics.* Springer, Cham, 2018.

[55] Juliani, Arthur, et al. "Unity: A general platform for intelligent agents." *arXiv preprint arXiv:1809.02627* (2018).

[56] Dosovitskiy, Alexey, et al. "CARLA: An open urban driving simulator." *arXiv preprint arXiv:1711.03938* (2017).

[57] Mohammed, Eslam, et al. "End-to-End Deep Path Planning and Automatic Emergency Braking Camera Cocoon-based Solution." *Neural Information Processing Systems (NeurIPS 2019).* 2019.

[58] Sauer, Axel, Nikolay Savinov, and Andreas Geiger. "Conditional affordance learning for driving in urban environments." *arXiv preprint arXiv:1806.06498* (2018).

[59] Zhang, Enwei, et al. "Learning How to Avoiding Obstacles for End-to-End Driving with Conditional Imitation Learning." *Proceedings of the 2019 2nd International Conference on Signal Processing and Machine Learning*. 2019.

[60] Cai, Peide, et al. "VTGNet: A vision-based trajectory generation network for autonomous vehicles in urban environments." *arXiv preprint arXiv:2004.12591* (2020).

[61] van Dijk, Tom, and Guido CHE de Croon. "How do neural networks see depth in single images?." *arXiv preprint arXiv:1905.07005* (2019).

[62] Zhu, Meixin, Xuesong Wang, and Yinhai Wang. "Human-like autonomous car-following model with deep reinforcement learning." *Transportation research part C: emerging technologies* 97 (2018): 348-368.

[63] Li, Shengbo Eben, et al. "Minimum fuel control strategy in automated car-following scenarios." *IEEE Transactions on Vehicular Technology* 61.3 (2012): 998-1007.

[64] Wei, Zhensong, et al. "End-to-End Vision-Based Adaptive Cruise Control (ACC) Using Deep Reinforcement Learning." *arXiv preprint arXiv:2001.09181* (2020).

[65] BARTH, MATTHEW J. "Energy and Environmental Impacts of Personal Mobility." *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2006 Symposium*. National Academies Press, 2007.

[66] Fuel efficient driving techniques by Natural Resources Canada: https://www.nrcan.gc.ca/energy-efficiency/energy-efficiency-transportation/personal-vehicles/fuel-efficient-driving-techniques/21038

[67] Savinov, Nikolay, et al. "Episodic Curiosity through Reachability." *International Conference on Learning Representations*. 2018.

[68] Andrychowicz, Marcin, et al. "Hindsight experience replay." *Advances in neural information processing systems*. 2017.

[69] Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).

[70] Hao, Peng, et al. "Eco-approach and departure (EAD) application for actuated signals in real-world traffic." *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2018): 30-40.

[71] Bai, Zhengwei, et al. "PengHybrid Reinforcement Learning for Multi-Sensor-Based Connected Eco-Driving Signalized Intersections" (2019).

[72] Bewley, Alex, et al. "Learning to drive from simulation without real world labels." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[73] Kendall, Alex, et al. "Learning to drive in a day." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[74] Duckietown. https://www.duckietown.org/

[75] AUTOTRAC. https://ec.europa.eu/jrc/en/event/other-event/jrc-autotrac-2020-how-future-road-transport-will-look