北京理工大學

本科生毕业设计（论文）

基于强化学习的无人车高速公路环境换道决策方法研究

Automated Lane Change Strategy in Highway Environment Based on Deep Reinforcement Learning

学　　院：自动化学院

专　　业：自动化

学生姓名：程旭欣

学　　号：1120161412

指导教师：王美玲

校外指导教师：Ching-Yao Chan

2020 年 6 月 10 日

摘　要

在高速公路场景下，驾驶员一般会在超车，汇入另一车道，准备驶出高速等情况时进行换道。不当的换道行为会造成交通紊乱甚至交通事故。目前有很多基于人工规则的换道决策方法，但它们在面对突发交通状况或非常行为时通常只有非常有限的性能。深度强化学习在机器人控制和游戏方面取得了巨大的成果。这给我们提供了另一种思路：通过训练一个智能决策器来完成换道行为的决策。

本研究设计了一种基于近端策略优化的换道策略，定义了强化学习智能体的状态空间和动作空间。此方法具有数据利用率高效和训练过程稳定的优点。

为确保训练过程中的安全性（避免碰撞），设计了一种危险状态检测器和紧急控制器。在即将发生碰撞时，紧急控制器将介入并覆盖智能体的危险动作以避免碰撞发生。通过设置不同的参数，可调整紧急控制器介入的临界点。

为评估智能体策略的表现，提出了多个评估标准。之后，在基于 SUMO 的仿真环境测试了训练完成的智能体策略的表现。训练完成的智能体可以在密度较大的车流中做出相应的决策来执行流畅，安全并且高效的换道行为。

为进一步说明本策略的有效性，设计了一个基于规则的策略，使用碰撞时间作为换道决策的依据。对比了基于强化学习的策略和基于碰撞时间的策略在相同环境下的表现。结果显示基于强化学习的策略在在多个方面的表现超越了基于规则的换道策略。

关键词：自动驾驶；高速公路；换道决策；深度强化学习；近端策略优化；SUMO；碰撞时间

Abstract

In a highway driving scenario, lane-change maneuvers are commonly executed by drivers to overtake a slower vehicle, adapt to a merging lane ahead, prepare to exit highway, etc. However, improper lane change behaviors can be a major cause of traffic flow disruptions and even crashes. While many rule-based methods have been proposed to solve lane change problems for autonomous driving, they tend to exhibit limited performance due to the uncertainty and complexity of the driving environment. Deep reinforcement learning (DRL) offers an alternative approach, which has shown promising success in many application domains including robotic manipulation, and playing games.

In this study, we propose an automated lane change strategy using proximal policy optimization-based deep reinforcement learning, which shows great advantage in learning efficiency while maintaining stable performance. We define state and action spaces of the RL agent. To ensure safety constraints during training, we design a danger detector and an action filter to override unsafe decisions output from RL agent. By setting different values to the parameters of the danger detector, we can alter the sensitivity of the detector. We test performance of our policy on a simulation environment developed based on SUMO(Simulation of Urban Mobility). The trained agent is able to learn a smooth, safe, and efficient driving policy to determine lane-change decisions (i.e. when and how) in dense traffic scenarios with different vehicle speed distributions in target lane. To further demonstrate the effectiveness of our approach, we compare the performance of the proposed policy with a rule-based policy that make lane change decisions based on TTC(Time to Collision) using various evaluation metrics. The results demonstrate the lane change maneuvers can be efficiently learned and executed in a safe, smooth and efficient manner, outperforming the rule-based policy.

Key Words: autonomous driving; lane change; decision making; deep reinforcement learning; Simulation of Urban Mobility; Proximal Policy Optimization

Contents

摘　要 I
Abstract II
Chapter 1　Introduction 1
1. 1　Background 1
1. 2　Related Work 1
1. 3　Thesis Structure 4
Chapter 2　Problem Formulation 5
2. 1　Proximal Policy Optimization 5
2. 1. 1　MDP 5
2. 1. 2　Policy Gradient Methods 6
2. 1. 3　Proximal Policy Optimization 7
2. 1. 4　TD(λ) 9
2. 2　Problem Description 10
2. 3　System Architecture 11
2. 4　State and Action Space 12
2. 5　Reward Function 13
2. 6　Safety Action Filter 16
2. 7　Intelligent Driver Model(IDM) 17
2. 8　Baseline Strategy 17
2. 9　Summary 18
Chapter 3　Simulation Experiment 19
3. 1　Simulation Setup 19
3. 2　Demand Modeling of SUMO 20
3. 3　Evaluation Metrics 22
3. 4　Training and Results 23
3. 4. 1　Training Setup 23
3. 4. 2　Training Results 23
3. 4. 3　Demonstration of a Success Lane Change 25
3. 4. 4　Comparison between Two Policies 27
3. 5　Summary 28
Chapter 4　Summary and Future Work 29
4. 1　Summary 29

4. 2 Future Work 29
References 30
Acknowledgment 33

Chapter 1 Introduction

1. 1 Background

Automated and semi-automated vehicles are considered to have a great potential to improve transportation safety and efficiency, and a considerable amount of studies has been performed with a focus on autonomous driving or advanced driving assistance systems (ADAS). To achieve full autonomous driving, we must consider 3 aspects in an autonomous driving system: perception, decision and execution. For perception, there are many state-of-the-art object detection techniques using multiple sensor inputs such as Lidar and RGB cameras in order to get information of the surrounding environment of ego vehicle[1]. For decision making, the autonomous vehicle (ego vehicle) must make reasonable and safe decisions under uncertain and highly dynamic environment based on the information acquired by the perception layer[2]. As for execution, an ego vehicle needs to have smooth and reliable control in both lateral and longitudinal direction.

Highway environment is the most common scenario where level 3 and level 4 autonomous driving is applied. The ego vehicle needs to make decision to achieve high efficiency as well as absolute safety at all times, especially in more dynamic environment with frequent interactions with surrounding objects such as a lane change scenario. In a highway environment, an accident happens with a higher probability during a lane change maneuver such as ramp merge or preparing to exit the highway[3]. A study showed that roughly 10 percent of all highway accidents are cause by lane change maneuvers[4]. Therefore, a safe, smooth and efficient lane change maneuver is crucial to achieve level 3 or 4 autonomous driving. To satisfy the aforementioned requirements, an ego vehicle must be able to make reasonable decisions in a highly dynamic environment with adversarial and cooperative interactions taken into consideration.

1. 2 Related Work

Even though there have been many mature and applicable autonomous driving systems currently in use on a massive number of vehicles, it is still challenging to achieve automated decision-making and control to drive vehicles smoothly and effectively in some cases.

A set of literature applied rule-based model to address the path-planning and trajectory tracking problems under interactive situations. For example, the approaches of a potential field model and model predictive control are suggested in works by Rasekhipour, Y. et al.[5], Kim, B. et al.[6], and Ji, J., Khajepour. et al.[7]. However, as in a real-world scenario, some irrational and unforeseen behaviors (e.g. uncooperative vehicles) may render the aforementioned methods inefficient. While expert demonstrations from human drivers can be used to learn certain driving tasks via imitation learning[8], a substantial amount of data needs to be collected at all possible conditions (with variation in surrounding traffic, road signs, traffic light) for training, which is costly. Moreover, it requires massive human labor to label such data, but still may not cover all of the complex situations in real-world driving scenarios. Even with such prerequisites of imitation learning, drifts in behaviors can still cause catastrophic results. As Data Aggregation(DAGGER) can mitigate this drift effect, it is still hard to manually label data frame by frame from a simulator, which is against human's intuition. Because driving is a continuous decision making process, with only one or two frame of abstraction states, it is hard for human to label the right action, which can be continuous or discrete.

On the other hand, reinforcement learning (RL) algorithms have shown great potential for handling decision-making and control problems, which enables an agent to learn in a trial-and-error way with interaction with the environment that does not require explicit human labeling or supervision. Instead, it needs a well-defined reward function to determine the objective of the agent. RL has demonstrated significant success for solving complex task in both robotic manipulations[9] and playing video games[10].

However, applying RL to real-world applications is particularly challenging, especially for autonomous driving tasks that involve extensive interactions with other vehicles in a dynamically changing environment. Among a variety of vehicle decision-making and control problems that were tackled using RL algorithms[11-13], facilitating automated lane change maneuvers is of special interests, since improper lane change behaviors can be a major cause of highway crashes and traffic jams[14-17]. An early work of applying deep RL to lane change can be found in[18], where the Q-masking technique is proposed to act as a mask on the output Q-values in a deep Q-learning framework to obtain a high-level policy for tactical lane

change decisions, while still maintaining a tight integration with the prior knowledge, constraints and information from a low-level controller. To overcome the obstacle of rule-based models that are prone to failures in unexpected situations or diverse scenarios, a Q-function approximator with closed form greedy policy is proposed in a recent work[19]. In another work[20], a hierarchical RL based architecture is presented to make lane change decisions and execute control strategies. Specifically, a deep Q-network (DQN) is trained to decide when to conduct the maneuver based on safety considerations, while a deep Q-learning framework with quadratic approximator is designed to complete the maneuver in longitudinal direction.

Additionally, the applications of deep RL algorithms on lane change strategies are often challenged by their slow learning rates. In[21], this problem is addressed by making use of a minimal state representation consisting of only 13 continuous features, which facilitates a faster learning while training a DQN. Moreover, a technique referred to as "multi-objective approximate policy iteration (MO-API)" is presented in[22]. The value and policy approximations are learned using data-driven feature representations, where sparse kernel-based features or manifold-based features can be constructed based on data samples. It is concluded that better learning efficiency can be achieved using the proposed MO-API approach, when compared to benchmark RL algorithms such as multi-objective Q-learning. While these methods are centered around manipulating the feature space, more effective RL algorithms can be also employed to facilitate efficient learning rates and reduce the high variance in policy learning.

To address the above issues, in this work, we apply a proximal policy optimization (PPO)[23] based deep reinforcement learning method with an actor-critic structure, which combines policy gradient methods with a learned value function. The parameterized actor with Adam optimizer can enforce a trust region with clipped objectives that reduce the high computation burden in nonlinear conjugate gradient of TRPO[24]. Moreover, PPO tends to be more efficient in sampling and learning policies than TRPO. Meanwhile, compared to value-based methods, PPO is able to compute actions directly from the policy gradient rather than optimizing from the value function. On the other hand, the merit of the critic is to supply the actor with the knowledge of performance in low variance. All of these nice properties of PPO can improve its capability in real-life application domains.

1. 3 Thesis Structure

The objective of this study is to develop a decision-making strategy to enable automated mandatory lane change maneuvers centered around the objectives of *comfort*, *efficiency*, *speed* and *safety*, using the concept of PPO-based deep reinforcement learning. We test our methods in SUMO, a simulation environment for urban traffic flows.

- **Chapter 1 Introduction**: In this chapter, we introduce autonomous driving and some challenges it is currently facing. Besides, we talk about how reinforcement learning is applied to various of fields including games, robotics and autonomous driving. Inspired by this studies, we propose a new idea of using reinforcement learning to make lane change decisions in a highly dynamic highway environment.

- **Chapter 2 Problem Formulation**: This chapter introduces the Proximal Policy Optimization algorithms and the formulation of reinforcement learning including state and action spaces and reward function design. Besides, we show how the safety action filter work to avoid collision during simulation. We also introduced the IDM car following model. Lastly, we design a rule-based baseline lane change policy in order to compare with the proposed one.

- **Chapter 3 Simulation Experiment**: We show how to setup the simulation environment in this chapter and proposed several evaluation metrics for validating our results. We demonstrate the results from training by showing an example. We also compare our method with the rule-based decision making policy to show the effectiveness of the proposed method.

- **Chapter 4 Summary and Future Work**: In this chapter, we make a conclusion of our work and find some drawbacks of our method. Further more, we propose several direction of future work.

Chapter 2 Problem Formulation

2. 1 Proximal Policy Optimization

In a Reinforcement learning (RL) setup, we assume there is an unknown MDP(Markov Decision Process). An *agent* acts by interacting with the *environment* and collects information about the environment. Then the agent uses the information to improve its policy in order to maximize the expected cumulative rewards returned by the environment. There are two kinds of RL algorithms, model based RL and model free RL based on whether the model of the unknown MDP is approximated and then solved or not. Model based RL needs a large number of samples to infer the parameters of a MDP, which has high variance in some cases where an event with very small probability is hard to get sampled[25]. Then it solves the MDP using value iteration, policy iteration or other dynamic programming methods[26]. However, model-free RL does not need to explicitly know a MDP. For model-free RL, there are two general categories of RL algorithms, namely value-based method and policy-based method. While value-based methods can approximate the value function or action value function(Q-function) using neural networks in an off-policy way such as the state-of the art DQN[27], the primary advantage of policy-based methods, such as the REINFORCE algorithm[28], is that they can directly optimize the quality of policy, have better convergence properties and are effective in high-dimensional or continuous action spaces. Our study thus focuses on policy-based RL methods. By incorporating value-based and policy-based methods, we have actor-critic style methods, where we can approximate the value first and use it to improve our policy.

2. 1. 1 MDP

An MDP(Markov Decision Process) is used to describe an environment used in reinforcement learning, where the states of the environment is fully observable. It consists of a tuple (S, A, T, R, γ), where S is the state space, A is the action space, T is the transition probability, R is the reward function and γ is the discount factor. The objective of reinforcement learning is to find a policy $\pi(a|s)$ that models the conditional distribution over action $a \in A$ given a state $s \in S$. At each timestep of the MDP, the agent observes the current

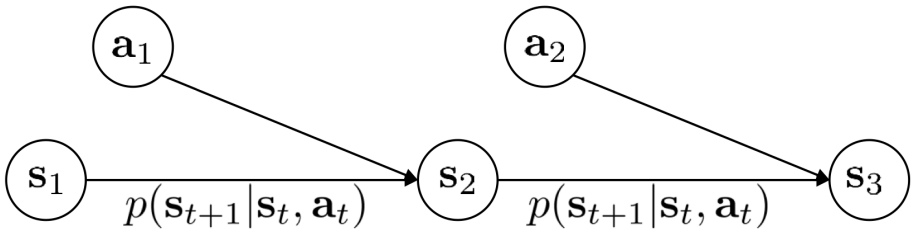


Figure 2-1 Markov Decision Process

state s_t of the environment and then samples an action a_t according to the policy π. The environment will then respond with a new state s_{t+1} according to the transition probability $T(s_{t+1}|s_t, a_t)$ and a scalar reward $r(s_t, a_t)$. For a parametric policy π_θ in which the parameters θ is usually weights in a neural network, the objective of an agent is to find the optimal θ^* that maximize the expected cumulative reward

$$\max_\theta J(\theta) \tag{2-1}$$

where

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{2-2}$$

where $p_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$ is the probability of trajectory $\tau = (s_0, a_0, s_1, a_1, ..., s_{T-1}, a_{T-1}, s_T)$ under policy π_θ, with $p(s_0)$ being the initial state distribution.

2. 1. 2 Policy Gradient Methods

To optimize the objective function in 2.1, we can use a class of methods called "Policy Gradient Methods"[29], where the gradient of the objective function $\Delta J(\theta)$ can be approximated by a gradient estimator. Recall that we want to maximize the objective function $J(\theta)$ in equation 2-2. The most commonly used method is gradient descent methods where we need to take the gradient of the objective function and then apply a gradient step. The gradient can be estimated by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t \sim d_\theta(s_t), a_t \sim \pi_\theta(a_t|s_t)} \left[\nabla_\theta \log(\pi_\theta(a_t|s_t)) \hat{A}_t \right] \tag{2-3}$$

where π_θ is a stochastic policy and $\hat{A}_t = R_t - V(S_t)$ is an estimator of the advantage function at timestep t. $R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$ represents the return received by a particular trajectory starting from state s_t at time t. $V(S_t)$ is a value function that estimates the average return of starting in s_t and following the policy for all subsequent steps.

$$V(S_t) = \mathbb{E}[R_t | s_t, \pi_\theta] \tag{2-4}$$

The expectation $\mathbb{E}$ indicates the empirical average over a finite batch of samples collected in interactions with the environment under current policy. Then we can construct an objective function whose gradient is the gradient estimator. We formulate the estimator as:

$$L^{PG}(\theta) = \mathbb{E}_t \left[\log \pi_\theta \left(a_t | s_t \right) \hat{A}_t \right] \tag{2-5}$$

The policy is trained using the proximal policy optimization algorithm[23], which has shown state-of-the-art results on a number of challenging problems. The value function is trained using multi-step returns with $TD(\lambda)$. The advantages $\hat{A}_t$ is computed using the generalized advantage estimator $GAE(\lambda)$[30]. We will introduce these methods in the following sections.

2. 1. 3 Proximal Policy Optimization

As stated previously, we can estimate the gradient of the objective function $J(\theta)$ by collecting a batch of samples from current policy $\pi(a|s)$. However, in this way the collected samples can only be used once to update the parameters θ. Then we need to discard previous data and use updated policy to collect a new batch of data. Sampling efficiency can be improved by using importance sampling.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t \sim d_{\theta_{old}}(s_t), a_t \sim \pi_{\theta_{old}}(a_t|s_t)} \left[r_t(\theta) \nabla_\theta \log \left(\pi_\theta \left(a_t | s_t \right) \right) \hat{A}_t \right] \tag{2-6}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{2-7}$$

Apply gradient with importance sampling can be seen as maximizing the surrogate

objective:

$$L^{IS} = \mathbb{E}_{s_t \sim d_{\theta_{old}}(s_t), a_t \sim \pi_{\theta_{old}}(a_t|s_t)} \left[r_t(\theta) \hat{A}_t \right] \tag{2-8}$$

While it might be appealing and straightforward to perform multiple steps of optimization on this loss function $L^{PG}(\theta)$ or $L^{IS}(\theta)$, many challenges can arise from the prevalence of sample inefficiency, the balance between exploration and exploitation, and the undesirable high variance of the learned policy. Empirically, it often leads to destructively large policy updates, which are destructive since they can also affect the observation and reward distribution at future time steps.

Compared to this fundamental loss function $L^{PG}(\theta)$ to update a policy, some advanced policy-based algorithms such as TRPO[24] and PPO[23] take the actor-critic structure, which is able to combine advantages of traditional value-based and policy-based approaches. The PPO algorithm is also much simpler to implement, more general, and have better sample complexity (empirically). Specifically, the PPO employs a neural network architecture that shares parameters between the policy and value function, and its loss function also combines the policy surrogate and a value function error term, which is defined as[23]

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{2-9}$$

where ϵ is a hyper-parameter, and $r_t(\theta)$ denotes the probability ratio. In this manner, the probability ratio r is clipped at $1 - \epsilon$ or $1 + \epsilon$ depending on whether the advantage is positive or negative, which forms the clipped objective after multiplying the advantage approximator $\hat{A}_t$. The final value of L_t^{CLIP} takes the minimum of this clipped objective and the unclipped objective $r_t(\theta)\hat{A}_t$, which can effectively avoid taking a large policy update compared to the unclipped version[23], which is also known as the loss function of the conservative policy iteration algorithm[31].

$$L_t^{CLIP+VF}(\theta) = \mathbb{E}_t \left[c_1 L_t^{CLIP}(\theta) - c_2 L_t^{VF}(\theta) \right] \tag{2-10}$$

where L_t^{CLIP} is the clipped surrogate objective, c_1, c_2 are coefficients, L_t^{VF} is the squared-error loss of the value function $\left(V_\theta\left(s_t\right) - V_t^{\text{targ}}\right)^2$. The pseudo code for PPO is shown in Algorithm 1.

Algorithm 1: Proximal Policy Optimization

$\theta \leftarrow random\ weights$;
$\psi \leftarrow random\ weights$;
while *not done* **do**
  $s_0 \leftarrow get\ initial\ states$;
  for $step\ in\ 1, ..., m$ **do**
    $s \leftarrow start\ state$;
    $a \sim \pi_\theta(a|s)$;
    Apply a and simulate for one step;
    $s^{'} \leftarrow endstate$;
    $r \leftarrow reward$;
    record (s, a, r, s') into memory D
  end
  $\theta_{old} \leftarrow \theta$ **for** *each update step* **do**
    Sample minibatch of n samples (s_i, a_i, r_i, s_i) from D;
    Update value function:
    for *each* (s_i, a_i, r_i, s_i) **do**
      $y_i \leftarrow$ compute target value using TD(λ)
    end
    $\psi \leftarrow \psi + \alpha_v \left(\frac{1}{n} \sum_i \nabla_\psi V_\psi\left(s_i\right)\left(y_i - V\left(s_i\right)\right)\right)$
    Update policy:
    for *each* (s_i, a_i, r_i, s_i) **do**
      $\mathcal{A}_i \leftarrow$ compute advantage using V_ψ and GAE);
      $w_i(\theta) \leftarrow \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}$;
    end
    $\theta \leftarrow \theta + \alpha_\pi \frac{1}{n} \sum_i \nabla_\theta \min\left(w_i(\theta)\mathcal{A}_i, \text{clip}\left(w_i(\theta), 1-\epsilon, 1+\epsilon\right)\mathcal{A}_i\right)$;
  end
end

2. 1. 4 TD(λ)

When using a parameterized stochastic policy, it is possible to obtain an unbiased estimation of the gradient of the expected total returns we mentioned earlier[32]. However, the variance of the gradient estimator grows unfavorably with the time horizon expanding to infinity, due to the fact that the action at a timestep t will affect future states and actions. We denote the return $R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$ as the Monte-Carlo return, which will provide an unbiased estimate of the expected return at a given state but can have high variance due to stochasticity of the environment and policy.

Another method is to use a value function rather than empirical returns stated above to provide a lower-variance estimation at the cost of introducing some bias[33]. The n step return

can be obtained by truncating the sum of empirical returns after n steps, and approximate the return for remaining timesteps with a value function $V(s)$:

$$R_t^{(n)} = \sum_{l=0}^{n-1} \gamma^l r_{t+l} + \gamma^n V(s_{t+n}) \tag{2-11}$$

$R_t^{(1)} = r_t + \gamma V(s_{t+1})$ is the 1-step return $TD(1)$ commonly used in Q-learning[34]. $R_t^{(\infty)} = \sum_{l=0}^{T-t} \gamma^l r_{t+l} = R_t$ is exactly the original Monte-Carlo empirical return $TD(\infty)$ with the maximum horizon for each episode is T. While R_t is an unbiased but high variance estimator, $R_t^{(1)}$ reduces the variance but adds bias to the estimation. Thus, we can trade off between variance and bias between variance and bias by alternating n.

However, there is another way to make trade-off between bias and variance that generally has better performance. We want to combine the advantage of low variance of $TD(1)$ and no bias of $TD(\infty)$ by taking exponentially-weighted average of n-step returns with decay parameter λ:

$$R_t(\lambda) = (1-\lambda) \sum_{n=0}^{\infty} \lambda^n R_t^{n+1} \tag{2-12}$$

where R_t^n is $TD(n)$ return and λ is the exponential coefficient. $(1-\lambda)$ is an normalization coefficient due to the fact that $\sum_{n=0}^{\infty} \lambda^n = \frac{1}{1-\lambda}\ s.t. \lambda \in (0,1)$. $R_t(0) = R_t^{(1)}$ and $R_t(1) = R_t^{(\infty)}$. Values of $\lambda \in (0,1)$ result in estimated return that balances between bias and variance. Then we can use the computed λ-return to update the value function, which is the TD(λ) algorithm.

Similarly, we can estimate the advantage with λ-return by subtract the value function, yielding the generalized advantage estimator GAE(λ):

$$\hat{A}_t = R_t(\lambda) - V(s_t) \tag{2-13}$$

Then we can use TD(λ) and GAE(λ) to perform value and policy update in Algorithm 1.

2. 2 Problem Description

A lane change action is conducted when the vehicle needs to exit the highway, overtake a slower vehicle, adapt to the merging lane ahead, etc. Lane change maneuvers are usually

classified into two major categories: mandatory and discretionary. Compared to a discretionary lane change that is intended to achieve faster speed or better driving experience, a mandatory lane change usually occurs when the ego vehicle is forced to make a lane change due to either a lane drop or a highway exit.

In our formulation, we focus on mandatory lane change situations where explicit lane change intentions are already given by the vehicle route planner, and our task is to decide when and how to make the lane change maneuver based on states of surrounding vehicles and the ego vehicle itself. Once a decision is made by the model, a low-level controller is used to generate a corresponding control command to execute the decision. The applicable lane change policy to be learned should incorporate the following three functionalities:

- Avoiding collisions with surrounding vehicles
- Achieving high efficiency
- Executing smooth maneuvers

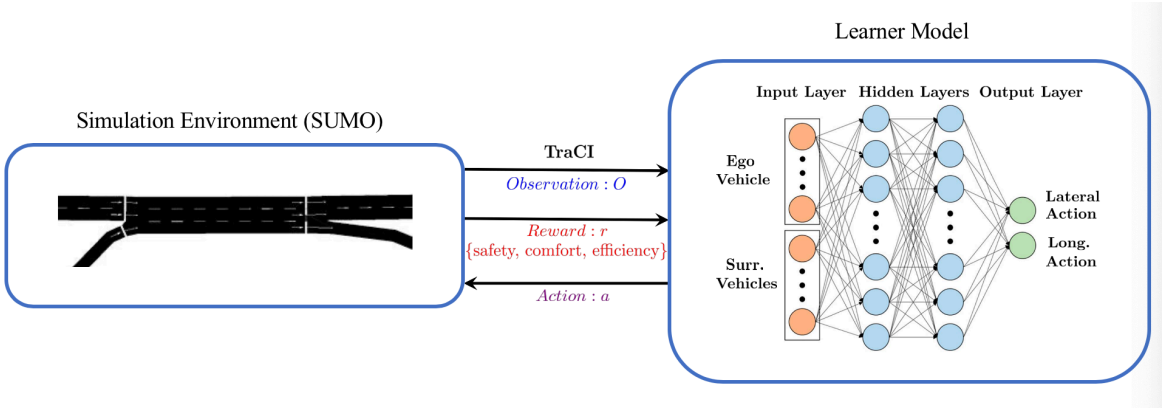


Figure 2-2 System architecture of the proposed PPO-based automated lane change strategy

2. 3 System Architecture

The overall system architecture for enabling automated lane change is shown in Figure 2-2. There are two major components in the system: a learner model and a simulation environment. Specifically, the learner model uses PPO to train the ego-vehicle (agent) to learn a high-level policy for decision making tasks while interacting with the surrounding traffic.

We use two neural networks to parameterize policy $\pi(a|s)$ and value function $V(s_t)$. They have the same structure with the input layer size 21, each of the 2 hidden layer containing 128 neurons. For the value network, there is only one output node without activation function. The output of policy network has a size of 6 with sigmoid as activation function. Simulation environment, which includes the road network, traffic, and different task scenarios, is generated using a high-fidelity microscopic traffic simulation suite SUMO (Simulation of Urban Mobility)[35], and it is used to interact with the training agent. Using SUMO and its associated traffic control interface (TraCI), we can access the vehicle information in the road network, and execute high-level decisions regarding vehicle dynamics given by the learned deep neural network models.

To enable safe, smooth, and efficient driving behaviors on highways, the ego-vehicle first receives its current state and its surrounding vehicles' state from the SUMO environment through TraCI, and these states are fed into the policy and value network. Next, the ego-vehicle determines the longitudinal and lateral actions based on the developed policy network, which then sends the action back to SUMO to model the vehicle's movement in the next time step. Then the environment will return to the agent a scalar reward and observations of next step.

2. 4 State and Action Space

We consider the state or 5 vehicles involved in the lane change decision and execution phase as shown in Figure 2-3: (1) the ego-vehicle C_e; (2) the leading vehicle in the original lane C_0; (3) the leading vehicle in the target lane C_1; (4) the following vehicle in the original lane C_2; and (5) the following vehicle in the target lane C_3.

The state space is composed of a total of 21 continuous state variables from both the ego-vehicle and its surrounding vehicles. Specifically, the ego-vehicle has 5 state variables, including its longitudinal position, speed, acceleration, as well as its lateral position and speed. Additionally, each surrounding vehicle has 4 state variables: relative distance to the ego-vehicle, longitudinal speed, acceleration, and lateral position.

In this study, we design the action space in both the lateral and longitudinal direction, so that an agent can learn when and how to perform a lane change. For the lateral command, we

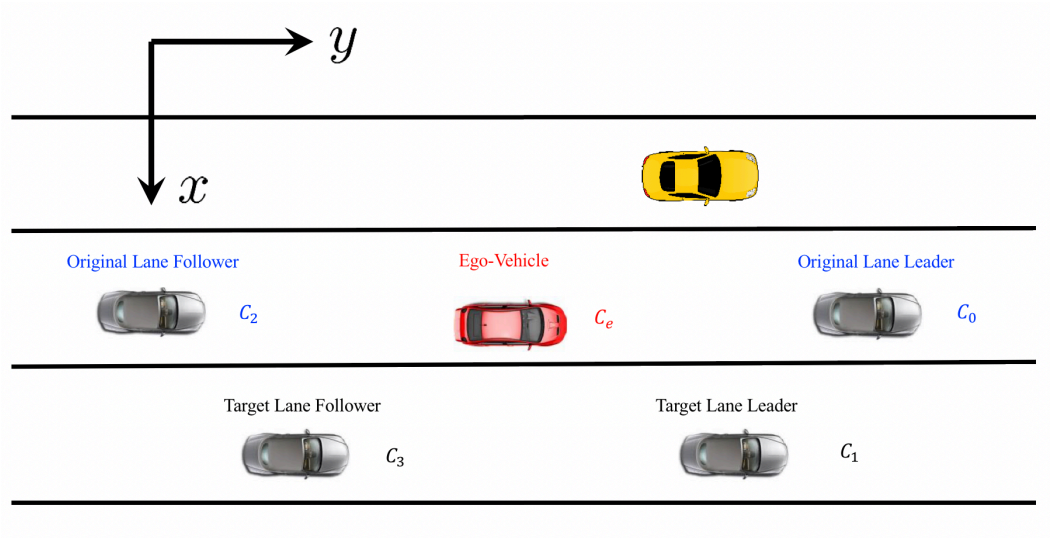


Figure 2-3 A demonstration of a lane change scenario

have two discrete actions {0, 1}, in which "0" represents keeping current lateral position, and "1" indicates performing lane change maneuver at a constant lateral speed $-1m/s$(meaning moving to the target lane laterally). For longitudinal strategy, there are three discrete actions {0, 1, 2}, in which each chosen action denotes the longitudinal acceleration.

Therefore, there are an altogether 6 actions combining different cases of actions in both the lateral and longitudinal direction, which are shown in Table 2-1.

Table 2-1 Action space and corresponding acceleration

	Lateral		**Longitudinal**		
Action	0	1	0	1	2
Corresponding Acceleration m/s	-1	0	-1.5	0	1.5

2. 5 Reward Function

The reward function is designed to incorporate key objectives of this study, which is to develop an automated lane change strategy centered around *safety*, *efficiency*, *speed*, and *comfort*. More specifically, these ideas are explained as follows:

1. *Comfort*: evaluation of jerk (lateral and longitudinal direction);

2. *Efficiency*: evaluation of whether lane change is successful;

3. *Speed*: evaluation of difference of actual and desired speed;

4. *Safety*: evaluation of the risk of collisions and near-collisions.

The reward function representing comfort can be expressed as:

$$R_{comf}(t) = -1 + exp(-\alpha \cdot \dot{a}_y(t)^2 - \beta \cdot a_y(t)^2) \tag{2-14}$$

where $\dot{a}_y$ and a_y are the lateral jerk and the lateral acceleration. This reward function is introduced to avoid sudden acceleration or deceleration of the vehicle that may cause vehicle occupant discomfort. Here we choose $\alpha = 1$ and $\beta = 0.1$.

In terms of efficiency, the ego-vehicle should try to move to the target lane as soon as possible since we are considering mandatory lane change, thus the efficiency reward function can be defined as:

$$R_{eff}(t) = -1 + exp(-d_t) \tag{2-15}$$

where d_t is the ego vehicle's current lateral distance to the center of target lane.

To encourage the ego vehicle complete lane change behavior while achieve its desired speed, we designed the reward for speed as:

$$R_{speed} = -1 + exp(-abs(V_y - V_{desired})) \tag{2-16}$$

where V_y is the current longitudinal speed of ego vehicle.

In order to improve the safety in the automated lane changing process and give the RL agent denser rewards, we introduce a near collision penalty function instead of only returning a penalty when a collision actually happened. In this case, an ego-vehicle can learn to avoid such circumstances either by stop lane change or adjust to a safe distance. The criteria to check if the current situation is dangerous or not is described in section 2. 6.

Thus the designed safety reward takes the form of:

$$R_{safety} = \begin{cases} curr\_timestep - 250 & if\ level-2\ danger\ occurs \\ -1 & else\ if\ level-1\ danger\ occurs \\ -1 + \tanh(\ \min(ttc_{tl},\ ttc_{ol})) & else \end{cases} \tag{2-17}$$

where “level-1” and “level-2” danger is defined in section 2. 6. When “level-2” danger occurs, the current episode ends and we assume the post reward is all -1. ttc_{tl} and ttc_{ol} are the Time to Collision to leader vehicle in target lane and original lane and are computed by:

$$ttc_{tl} = \frac{\Delta d_t}{V_y} \tag{2-18}$$

$$ttc_{ol} = \frac{\Delta d_o}{V_y} \tag{2-19}$$

where Δd_t and Δd_o is the longitudinal distance between ego vehicle and leader vehicle in target lane and original lane.

The total reward is

$$r_{total} = w[R_{comf}, R_{eff}, R_{speed}, R_{safety}]^T \tag{2-20}$$

where

$$r = [0.2,\ 1,\ 0.1,\ 1] \tag{2-21}$$

$$w = \frac{r}{\sum_{r_i \in r} r_i} \tag{2-22}$$

By applying exponential function and a bias term -1 we can bound the reward for each step in $[-1, 0]$. And through normalizing the weight term we can ensure that at each timestep the reward is bounded in $[-1, 0]$, which makes learning easier.

2. 6 Safety Action Filter

To avoid collision during simulation, we designed a safety filter over the actions output by our policy. We consider 2 scenarios where the action of an ego vehicle can be defined as "dangerous", under which the action from the safety controller will replace what comes from the policy network. The first scenario is *rear-end collision*, where two vehicles overlap in lateral direction and have very small longitudinal distance in the meanwhile. The other scenario is *side collision*, when two vehicles are approaching each other in lateral direction while overlap in longitudinal direction. Once any of the aforementioned scenarios occurs, we say that the current state is "dangerous". To be more specific:

$$D_{lat} = \begin{cases} 1 & if \quad W < |x_{ego} - x_{other}| < W + d_{lat} \quad and \quad |y_{ego} - y_{other}| < L + d_{longi} \\ 0 & else \end{cases} \tag{2-23}$$

$$D_{longi} = \begin{cases} 1 & if \quad W \geq |x_{ego} - x_{other}| \quad and \quad |y_{ego} - y_{other}| < L + d_{longi} \\ 0 & else \end{cases} \tag{2-24}$$

where D_{lat} and D_{longi} are flags indicating whether it is dangerous in respective directions. $W = \frac{width_{ego}+width_{other}}{2}$ represents half of sum of width of two vehicles. $L = \frac{length_{ego}+length_{other}}{2}$ represents half of sum of length of two vehicles.

d_{lat} and d_{longi} are the safety distances in lateral and longitudinal directions respectively. By having different d_{lat} and d_{longi} values we can get different levels of danger. We define the situation when $D_{longi} = 1$ *or* $D_{lat} = 1$ with $d_{longi} = 10$ and $d_{lat} = 0.8$ as "level-1 danger" and with $d_{longi} = 5$ and $d_{lat} = 0.3$ as "level-2 danger"

Here, a "level-1" danger is a soft constraint on the ego vehicle which will give a reward of -1 to alert the ego vehicle. However, a "level-2" danger occurs when a collision is about to happen and is very dangerous in real-world scenarios. If a "level-2" danger either in longitudinal or lateral direction is detected, the safety filter will cover the action from the

Table 2-2 Meaning of parameters in IDM

Parameter	Meaning
v_0	the velocity the vehicle would drive at in free traffic
s_0	minimum desired net distance. A car can't move if the distance from the car in the front is not at least s_0
T_m	the minimum possible time to the vehicle in front
a	the maximum vehicle acceleration
b	comfortable braking deceleration; positive

policy with the corresponding emergency behavior to avoid collision.

2. 7 Intelligent Driver Model(IDM)

In traffic flow modeling, the intelligent driver model (IDM) is a time-continuous car-following model for the simulation of freeway and urban traffic.[36] As a car-following model,the IDM describes the dynamics of the positions and velocities of single vehicles. For vehicle α, we denote its position at time t as x_α and velocity as v_α. l_α is the length of the vehicle. Further more, we define $s_\alpha := x_{\alpha-1} - x_\alpha - l_{\alpha-1}$ as the net distance, where $\alpha - 1$ refers to the vehicle directly in front of vehicle α and $\Delta v_\alpha := v_\alpha - v_{\alpha-1}$. The dynamics of vehicle α are described by the following ordinary differential equations:

$$\begin{aligned} \dot{x}_\alpha &= \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha \\ \dot{v}_\alpha &= \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \\ &\text{with } s^*\left(v_\alpha, \Delta v_\alpha\right) = s_0 + v_\alpha T_m + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \end{aligned} \tag{2-25}$$

The meaning of model parameters is listed in Table. 2-2

2. 8 Baseline Strategy

To further demonstrate the effectiveness of our controller, we designed a rule-based baseline lane change decision making strategy. For longitudinal control, rather than using discrete actions as in our RL policy, we choose to use IDM in order to achieve more realistic

longitudinal dynamics. For lateral control, we use TTC(Time to Collision) as the criteria of whether to perform lane change or not.

$$TTC_{tl} = \frac{d_{tl}}{v_{ego}} \tag{2-26}$$

$$TTC_{tf} = \frac{d_{tf}}{v_{tf}} \tag{2-27}$$

where TTC_{tl} and TTC_{tf} are Time to Collision of leader and follower in target lane respectively. d_{tl} and d_{tf} are the distance between ego and leader/follower in target lane. v_{tl} and v_{tf} are the speed of leader/follower in target lane. And v_{ego} is the speed of ego vehicle. Then we define the lane change strategy as

$$a_{lateral} = \begin{cases} 1 & TTC_{tl} > \hat{t}tc \ \ and \ \ TTC_{tf} > \hat{t}tc \\ 0 & else \end{cases} \tag{2-28}$$

where $a_{lateral}$ is the lateral action and $\hat{t}tc$ is the threshold of whether or not to perform lane change. We also use the evaluation metrics in section 3. 3 to evaluate the performance of the baseline policy with different $\hat{t}tc$.

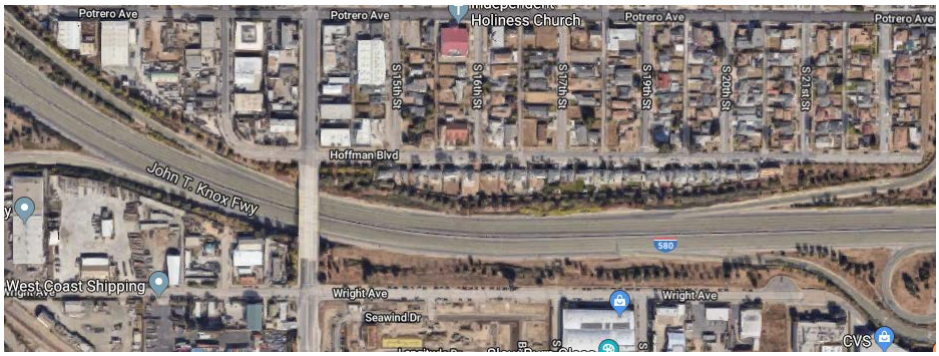
2. 9 Summary

In this chapter, we introduced Markov Decision Process (MDP) and the reinforcement learning setup. We also talked about the state-of-the-art Proximal Policy Optimization (PPO) algorithm derived from actor-critic reinforcement methods, and how to do advantage estimation using TD(λ). Then, we clarified the problem description, formulated our problem as an MDP and defined state and action spaces. Next we defined the reward function which is essential to reinforcement learning. Besides, we introduced a safety action filter and defined 2 types of dangerous situations. We treat the "level-2" danger as an early termination criteria which is important for RL agent not to collect too much useless information and so that there will not be collision during training. IDM is introduced for vehicles other than the ego vehicle. Last, we introduced the rule-based baseline strategy for lane change decision making based on TTC(Time to Collision).

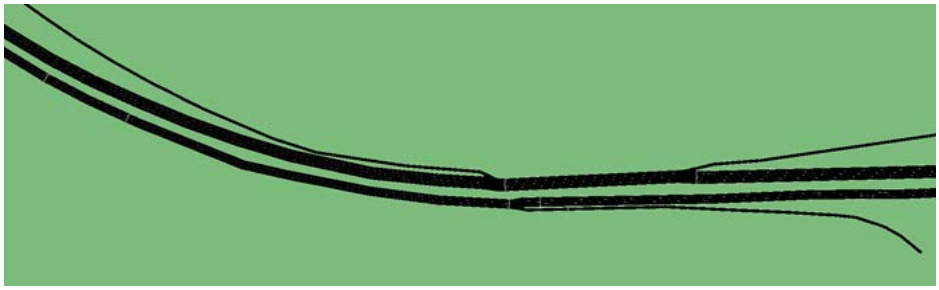
Chapter 3 Simulation Experiment

3. 1 Simulation Setup

The simulation network is modeled using a real-world highway segment with on-ramps and off-ramps as shown in Figure 3-1, which is implemented on SUMO. The highway segment length to the ramp exit is 800 m and each lane is 3.2 m in width. Vehicle counts are generated from a binary probability distribution to simulate the traffic of different density. In this study, we constrain the maximum acceleration and emergency braking deceleration of vehicles as $2.9\ \mathrm{m/s^2}$ and $-4.5\ \mathrm{m/s^2}$, respectively.



(a) Satellite image of selected highway



(b) Extracted highway structure in SUMO

Figure 3-1 Simulation Network

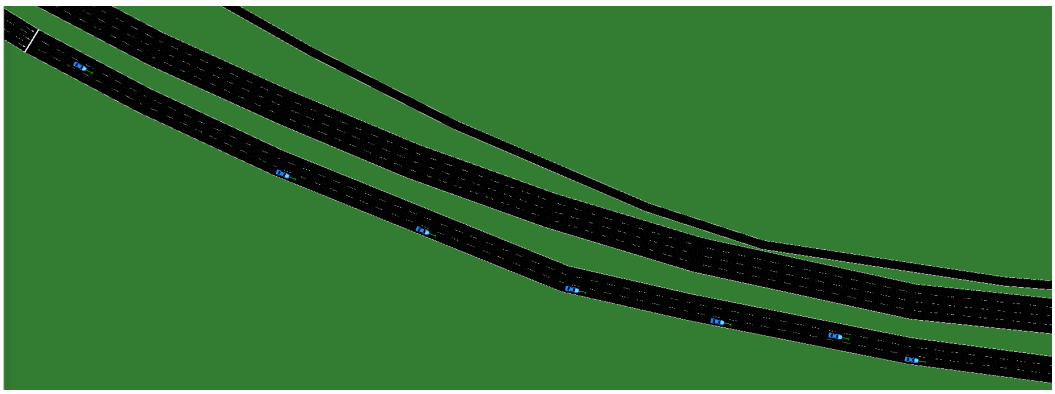
In order to make the proposed simulation network as similar to real traffic as possible, we applied the intelligent driver model (IDM)[37] to other vehicles' longitudinal control which we will cover in the next subsection. In SUMO environment, we also make the lane change mode of other vehicles to be "no-lane-change", meaning that there will only be lane change behaviors of the ego vehicle. When the ego vehicle makes lateral movement to the

target lane, the follower vehicle in the target lane has 2 possible behaviors: observing the lane change maneuver and take the ego vehicle as its leader; not observing the ego vehicle and continuing following its original leader vehicle. In the second case, there is a larger chance of dangerous interaction or even collision. To make the task more challenging and more realistic, we make the follower vehicle in the target lane randomly choose between these two behaviors.

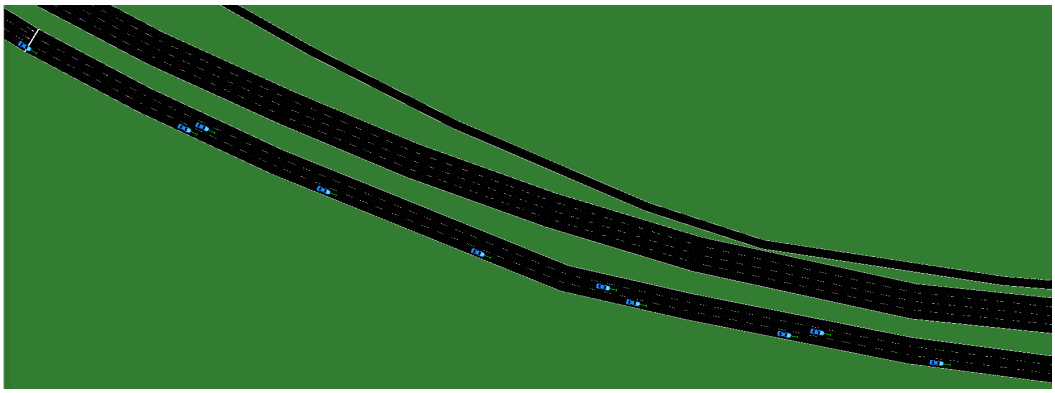
3. 2 Demand Modeling of SUMO

After we have the network extracted from OSM maps, we still cannot run a simulation. The thing that is missing is the definition of behaviors of vehicles, which we define as *Traffic Demand*. In order to generate traffic, we need to first define a route, which delineate the path a vehicle will be driving on. Then we need to define vehicles' departure location and arrival location as well as other properties of the vehicle such as desired driving speed, maximum acceleration and deceleration, etc.

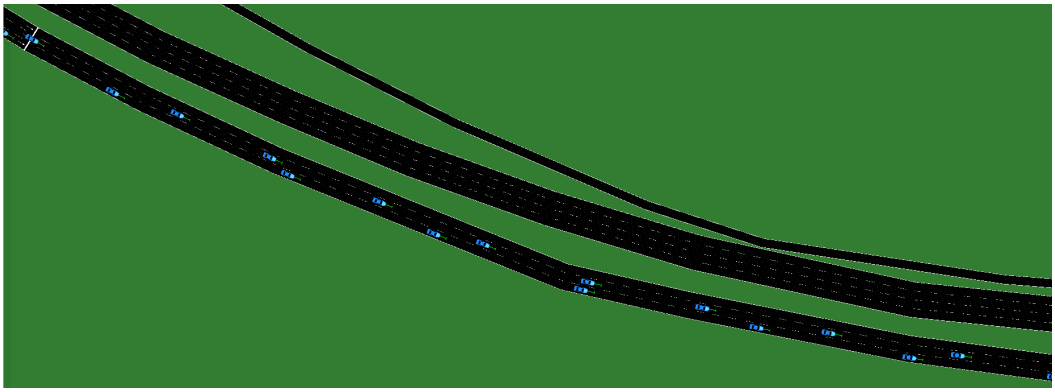
We use *flow* to describe a continuous traffic flow. There are 3 lanes on each direction in the highway. We only model 2 lanes where the lane change behavior occurs, where the *original lane* is the lane the ego vehicle starts from and *target lane* is the lane the ego vehicle is going to change to. A vehicle will be emitted randomly with the given probability p each second. This results in a binomially distributed flow (which approximates a Poisson Distribution for small probabilities). The density of the traffic varies with different probability values. Figure 3-1 demonstrate 3 different traffic density achieved by alternating p.



(a) $p = 0.2$, light traffic



(b) $p = 0.4$, medium traffic



(c) $p = 0.7$, dense traffic

Figure 3-1 Traffic flow of different densities

We found that in medium and light traffic setting, lane change is safe and discretionary in the majority of cases. So in this study we only focus on the case of dense traffic which is more challenging. In addition, we alternate the average speed of vehicles in *target lane* to increase the dynamic of the environment and demonstrate the capability of generalization of our model.

The desired speed $V_{desired}$ for each vehicle is computed using a clipped normal distribution:

$$speedFactor \sim clip(\mathcal{N}(\mu, \sigma^2),\ c_{low},\ c_{high}) \tag{3-1}$$

$$V_{desired} = speedFactor \times V_{limit} \tag{3-2}$$

$V_{limit} = 29m/s$ is the speed limit of the lane. For vehicles in *original lane*, $\mu = 1$. In

terms of vehicles in *target lane*, there are 3 different speed distributions randomly sampled for each episode during training. The parameters of these 3 distributions are shown in Table 3-1.

Table 3-1 Parameters of different speed distributions

Parameter	**Fast**	**Normal**	**Slow**
μ	1.3	1	0.7
σ	0.1	0.1	0.1
c_{low}	1.1	0.8	0.5
c_{high}	1.5	1.2	0.9

3. 3 Evaluation Metrics

To quantify the safety and effectiveness of the proposed PPO-based automated lane change model in both training and testing process, we add five metrics that evaluate the average level-1 danger timesteps(**ADT1**), average level-2 danger timesteps(**ADT2**), average task success rate(**ATSR**), average episode reward(**AER**) and average task completion time(**ATCT**), which are computed under 100 rollouts(episodes). The danger counts both rear-end danger and side-impact danger, which is delineated in section 2. 6. A successfully task in a simulation run is defined as the ego-vehicle having successfully changed to the target lane and maintain on that lane for 1s before reaching the exit and managed to avoid collisions with other vehicles. We can formulate the above three metrics as:

$$ADT1 = \frac{\hat{N}_1}{N_{total}} \tag{3-3}$$

$$ADT2 = \frac{\hat{N}_2}{N_{total}} \tag{3-4}$$

$$ATSR = \frac{N_{succ}}{N_{total}} \tag{3-5}$$

$$AER = \sum_{T_i} \sum_{t=0}^{T_i} r_{i_t} \tag{3-6}$$

$$ATCT = \frac{\sum_{ep_i \in E_{succ}} l_{ep_i}}{N_{total}} \tag{3-7}$$

where $\hat{N}_1$ and $\hat{N}_2$ are the total timesteps with level-1 or level-2 danger in $N_{total} = 100$ episodes. l_{epi} is the length of episode i. E_{succ} is a set containing all the episodes with successful lane change. Note that here $ADT1$ and $ADT2$ can be larger than 1 while $ATSR \in [0, 1]$.

3. 4 Training and Results

3. 4. 1 Training Setup

In terms of training, we utilize the PPO implementation of OpenAI baselines and modify it to suit our task. We use Adam[38] as our optimizer and MPI(Message Passing Interface) to parallel simulation. The hyper-parameters we use during training is shown in Table 3-2.

For training platform, we use an 8-core intel i7-9700k for simulation and Nvidia RTX 2070 for neural network computation.

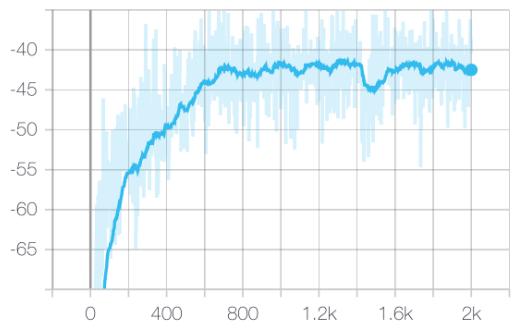
Table 3-2 Hyper-parameters for training

learning rate	2^{-3}
maximum episode length	250
timesteps per actor batch	2048
optimization epochs per actor batch	5
optimization batch size	512
discount factor γ	0.99
decay coefficient λ	0.95

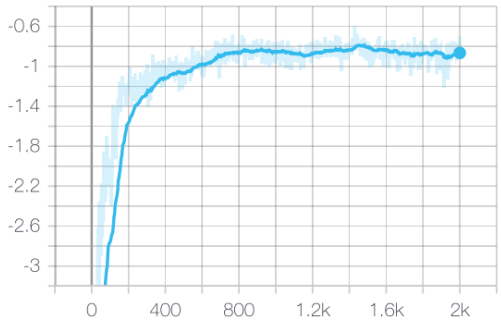
3. 4. 2 Training Results

The training converged after about 5 hours and 17 minutes with 8 parallel threads. The training took 2000 iterations of optimization, collected about 3.3×10^7 samples and completed about 5×10^5 episodes.

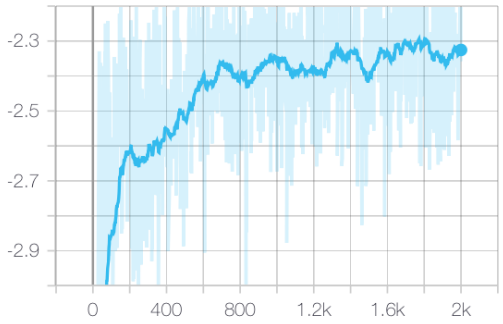
The average cumulative total reward and sub-rewards, as well as some of the evaluation metrics, are shown in Figure 3-2.



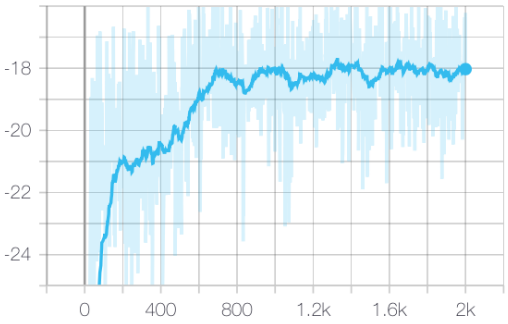
(a) Average episode reward(AER)



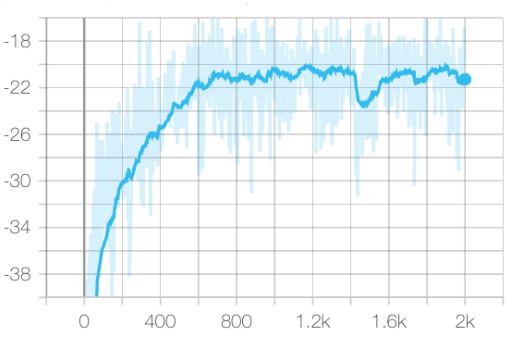
(b) Reward for comfort



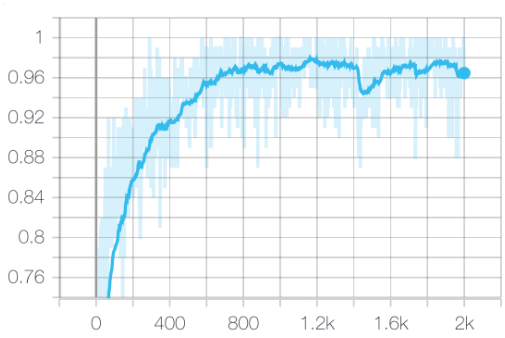
(c) Reward for speed



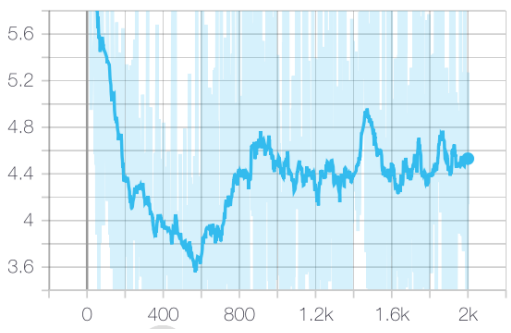
(d) Reward for efficiency



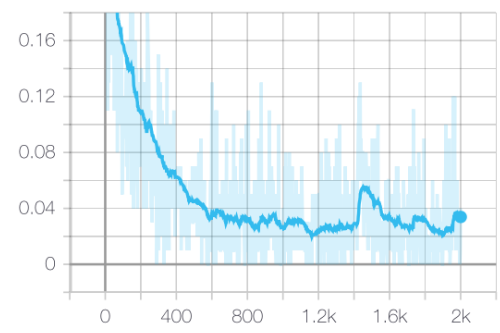
(e) Reward for safety



(f) Average task success rate(ATSR)



(g) Average level-1 danger timesteps(ADT1)



(h) Average level-2 danger timesteps(ADT2)

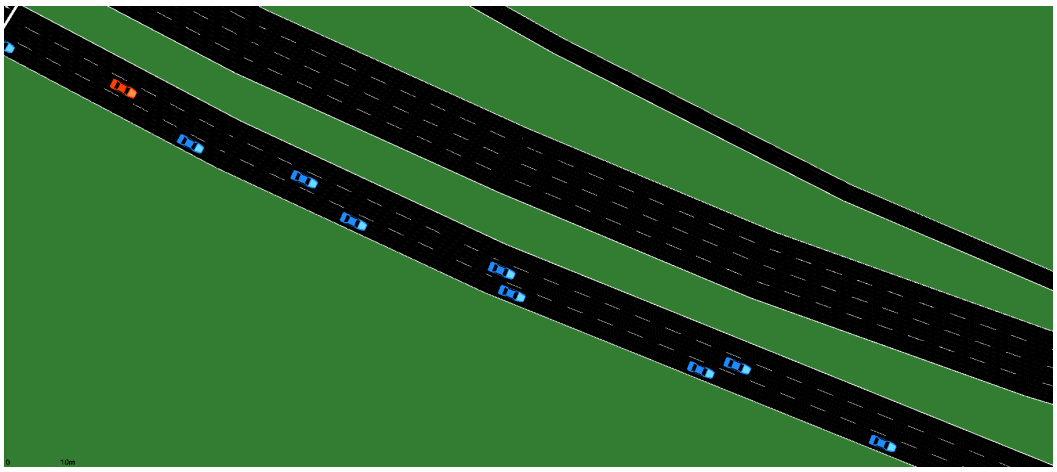
Figure 3-2 Rewards and evaluation metrics during training

The curve of the cumulative reward indicates the ego-vehicle can successfully learn to take actions to maximize the reward. The dark line in Figure 3-2 means smoothed curve for better visualization.

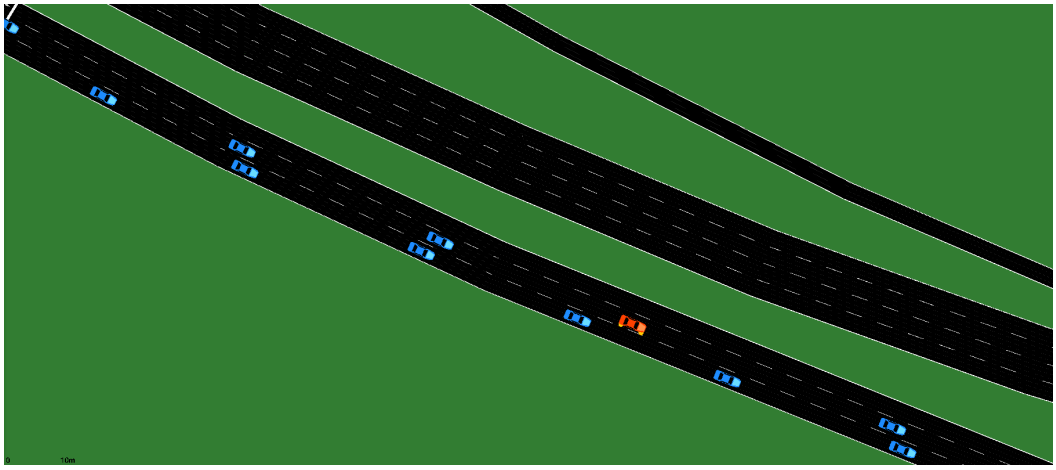
In the training process, a continuously increasing success rate can be observed in Figure 3-2 (f) indicating that the ego vehicle learned to perform lane change maneuver accordingly. On the contrary, the average level-1 danger timesteps shown in Figure 3-2 (g) decreased to its lowest values during the early stage of the training and then gradually grow a little bit and fluctuate with the training process. This can be explained by that the reward for level-1 danger is only -1, acting as a role to alert the ego vehicle the current situation is not desirable. However, the ego vehicle can still perform lane change at this risk in order to get further reward for efficiency. As for the average level-2 danger timesteps shown in Figure 3-2 (h), we can see that it is mostly monotonous because the penalty for a level-2 danger is very high.

3. 4. 3 Demonstration of a Success Lane Change

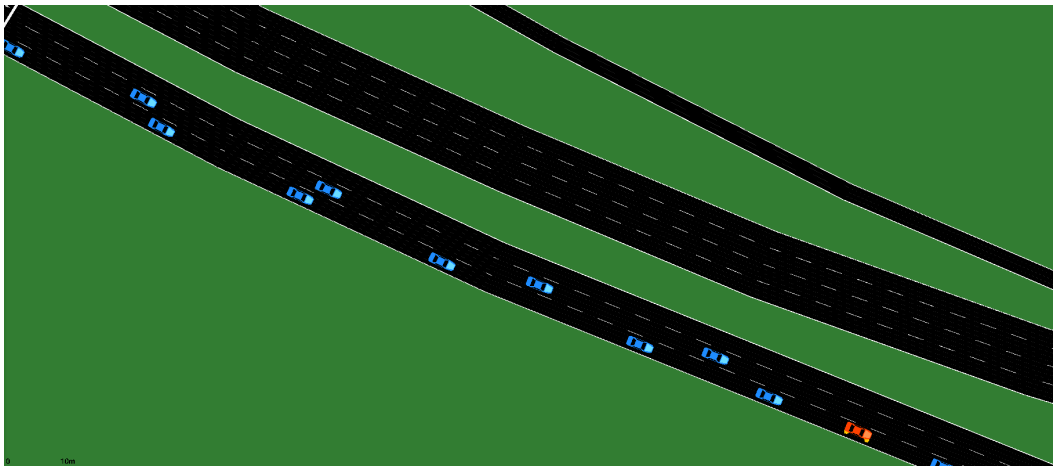
To further demonstrate the effectiveness of our method, we make a challenging environment where the vehicle speed distribution in the target lane is using **Slow** distribution which will result in denser traffic in target lane and is closer to real-world settings. The entire process of the lane change maneuver is shown is Figure 3-3. Initially, as shown in Figure 3-3 (a), the speed of the ego vehicle is about 26 m/s (93.6km/h), but the speed of the leader vehicle in target lane is only 17 m/s (61.2km/h). The ego vehicle did not choose to perform lane change instantly. on the contrary, it chooses to reduce its speed and merge into the next gap to avoid dangerous situation or even collision. By the time the ego vehicle reaches the next gap, the speed has been reduced to about 18m/s (64.8km/h) as shown in Figure3-3 (b). Then it starts to change lane and finally accomplishes the maneuver safely as shown in Figure 3-3 (c).



(a) Initial situation. Speed of ego vehicle is much larger than vehicles in target lane.



(b) After adjustment, the ego vehicle reduced its speed and prepare to perform lane change.



(c) The ego vehicle successfully completed the lane change maneuver.

Figure 3-3 Demonstration of a lane change maneuver performed by trained RL policy

3. 4. 4 Comparison between Two Policies

We evaluate both RL-based policy and rule-based policy using the same evaluation metrics and same set of random seeds to ensure fairness. Unlike during training, during evaluation, the hard safety constraint do not work any more because we want to see the performance of the policies without any external assistance. For RL-based policy, we test the performance of our model with 100 episodes with different random seeds, the trained agent can achieve a 100% success rate, while the average level-2 danger timesteps being only 0.02, meaning that during the 100 episodes there are only 2 timesteps of occurrence of level-2 danger, which can be in the same episode or 2 distinct ones. These statistics indicate the ego-vehicle is capable of learning the mandatory lane change strategy with regard to our designed reward function using the proposed PPO-based model. We compare the performance of the baseline model with the same set of random seeds, the detailed statistics are shown in Table 3-3.

From Table 3-3 we can see that RL policy achieved very high **ATSR** while remain a very low average danger timesteps. Even though the rule-based policy outperform RL policy in terms of **ADT1** with some $\hat{ttc}$ values, other metrics of RL-based policy are significantly better than the ruled-based policy. This can be explained by that the ego vehicle learned a to risk some danger to achieve higher efficiency but never risk a higher cost indicated by **ADT2**. The **ATST** of RL policy is 5.5, indicating that the ego vehicle is able to adjust itself and perform lane change maneuver according to surrounding environment instead of performing the maneuver as soon as possible.

Table 3-3 Comparison of RL-based strategy and rule-based strategy

Policy		**ADT1**	**ADT2**	**ATSR / %**	**AER**	**ATCT**
RL Policy		1.5	0.08	100	-39.8	5.5
Baseline Policy	$\hat{t}tc$					
	0.1	2.13	0.58	99	-89	5.6
	0.3	0.88	0.33	99	-70	6.2
	0.5	0.77	0.5	90	-73	6.8
	1	1.88	1.24	60	-148	9.0
	2	3.82	2.09	8	-228	6.7
	3	3.82	2.09	5	-230	7.3

3. 5 Summary

In this chapter, we introduced our simulation setup using SUMO (Simulation of Urban Mobility). We showed how to use demand modeling to generate different traffics. And we talked about how we evaluate the performance of our policy. In section 3. 4, we showed our training result on this setup and made a demonstration about how the RL policy make decisions and successfully make lane changes with dynamic environments. We also compared the performance of RL policy and rule-based policy and came to a conclusion that our policy outperform the rule-based policy in most critical evaluation metrics for lane change maneuver.

Chapter 4 Summary and Future Work

4. 1 Summary

This thesis proposed an automated mandatory lane change strategy by using proximal policy optimization (PPO)[23] based deep reinforcement learning, which features *safety*, *efficiency*, *speed* and *comfort*. The high-level PPO policy is used to generate lane-change decision (i.e. when and how) at each time step based on the current driving situations of the ego vehicle and its surrounding vehicles, while the lower level control is executed by a pre-defined model. We have shown the ego-vehicle trained using PPO based deep reinforcement learning can take appropriate actions to maximize the accumulated reward and achieve a 100% success rate and low danger rate in dense traffic, which demonstrates the effectiveness of the proposed lane change strategy. We also made comparison between the proposed PPO policy and a baseline policy, demonstrating that our policy is able to outperform the baseline policy.

The code for this project is available at https://github.com/chengxuxin/Lane_change_RL.

4. 2 Future Work

Although we have got a satisfying result for this study, we still need to be aware that our model is simplified in this study. We did not take real-world vehicle dynamics into consideration and simplified ego vehicle's longitudinal and lateral behavior to discrete actions, which is not practical in real world. In terms of observations, we did not consider the noise and delay of processing information in real-world environments. And we did not perform real-world testing of the efficiency ans safety of the algorithm due to many factors including safety concerns. However, from the demonstration we see that the ego vehicle learned some smart decisions on how to make lane change without risk the danger of collision.

The next step of our study is to make more real-world like environments for the ego vehicle. The current environment is simplified and is relatively easy for an RL agent. We can see from the result that even a baseline policy can also achieve 99% of success rate with some parameters, even though it performs less well on other criteria. Through testing on a more

complex and dynamic environment as mentioned earlier can we identify the potential of RL. Another thought is to incorporate human or expert inference by modeling the probability of their lane-changing behaviors to achieve better performance and further reduce danger rate using imitation learning such as Dagger(Data Aggregation). Future research directions may also involve more interdisciplinary work of RL intertwined with other techniques for better generalization, some recent examples can be found in combining transfer learning[39, 40].

References

[1] Chen X, Ma H, Wan J, et al. Multi-view 3d object detection network for autonomous driving[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l. : s.n.], 2017: 1907-1915.

[2] Brechtel S, Gindele T, Dillmann R. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs[C]//17th International IEEE Conference on Intelligent Transportation Systems (ITSC). [S.l. : s.n.], 2014: 392-399.

[3] St-Aubin P G. Traffic safety analysis for urban highway ramps and lane-change bans using accident data and video-based surrogate safety measures[D]. McGill University Libraries, 2011.

[4] Hetrick S. Examination of driver lane change behavior and the potential effectiveness of warning onset rules for lane change or" side" crash avoidance systems[D]. Virginia Tech, 1997.

[5] Rasekhipour Y, Khajepour A, Chen S K, et al. A potential field-based model predictive path-planning controller for autonomous road vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 18(5): 1255-1267.

[6] Kim B, Necsulescu D, Sasiadek J. Model predictive control of an autonomous vehicle[C]//2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556): vol. 2. [S.l. : s.n.], 2001: 1279-1284.

[7] Ji J, Khajepour A, Melek W W, et al. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints[J]. IEEE Transactions on Vehicular Technology, 2016, 66(2): 952-964.

[8] Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars[J]. ArXiv preprint arXiv:1604.07316, 2016.

[9] Finn C, Levine S, Abbeel P. Guided cost learning: Deep inverse optimal control via policy optimization[C]//International Conference on Machine Learning. [S.l. : s.n.], 2016: 49-58.

[10] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J]., 2013. arXiv: 1312.5602 [cs.LG].

[11] Wang P, Li H, Chan C Y. Continuous Control for Automated Lane Change Behavior Based on Deep Deterministic Policy Gradient Algorithm[C]//Proc. IEEE Intell. Veh. Sympo. (IV). [S.l. : s.n.], 2019: 1454-1460.

[12] Sallab A, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving[J]. Electronic Imaging, 2017, 2017(19): 70-76.

[13] Wang P, Li H, Chan C Y. Quadratic Q-network for Learning Continuous Control for Autonomous Vehicles[J]. NIPS Workshop Mach. Learn. Auton. Driving, 2019.

[14] Xu G, Liu L, Ou Y, et al. Dynamic Modeling of Driver Control Strategy of Lane-Change Behavior and Trajectory Planning for Collision Prediction[J]. IEEE Transactions on Intelligent Transportation Systems, 2012, 13(3): 1138-1155.

[15] Yang D, Zheng S, Wen C, et al. A dynamic lane-changing trajectory planning model for automated vehicles[J]. Transp. Research Part C: Emerging Technol., 2018, 95: 228-247.

[16] Ye F, Wu G, Boriboonsomsin K, et al. Development and Evaluation of Lane Hazard Prediction Application for Connected and Automated Vehicles (CAVs)[C]//2018 21st International Conference on Intelligent Transportation Systems (ITSC). [S.l. : s.n.], 2018: 2872-2877.

[17] Wang G, Hu J, Li Z, et al. Cooperative Lane Changing via Deep Reinforcement Learning[J]. ArXiv preprint arXiv:1906.08662, 2019.
[18] Mukadam M, Cosgun A, Nakhaei A, et al. Tactical decision making for lane changing with deep reinforcement learning[J]. NIPS Workshop Mach. Learn. Int. Transp. Syst., 2017.
[19] Wang P, Chan C Y, Fortelle A d L. A reinforcement learning based approach for automated lane change maneuvers[J]. IEEE Intell. Veh. Symp. (IV), 2018.
[20] Shi T, Wang P, Cheng X, et al. Driving decision and control for automated lane change behavior based on deep reinforcement learning[C]//Proc. Int. Conf. Intell. Transp. Syst. (ITSC). [S.l. : s.n.], 2019: 2895-2900.
[21] Mirchevska B, Pek C, Werling M, et al. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning[C]//Proc. Int. Conf. Intell. Transp. Syst. (ITSC). [S.l. : s.n.], 2018: 2156-2162.
[22] Xu X, Zuo L, Li X, et al. A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways[J]. IEEE Trans. Syst., Man, Cybern. Syst, 2018: 1-14.
[23] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. ArXiv preprint arXiv:1707.06347, 2017.
[24] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization[C]//Proc. Int. Conf. Mach. Learn. (ICML). [S.l. : s.n.], 2015: 1889-1897.
[25] Atkeson C G, Santamaria J C. A comparison of direct and model-based reinforcement learning[C]// Proceedings of International Conference on Robotics and Automation: vol. 4. [S.l. : s.n.], 1997: 3557-3564.
[26] Meyn S P. The policy iteration algorithm for average reward Markov decision processes with general state space[J]. IEEE Transactions on Automatic Control, 1997, 42(12): 1663-1680.
[27] Gu S, Lillicrap T, Sutskever I, et al. Continuous deep q-learning with model-based acceleration[C]// International Conference on Machine Learning. [S.l. : s.n.], 2016: 2829-2838.
[28] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3-4): 229-256.
[29] Sutton R S, McAllester D A, Singh S P, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Advances in neural information processing systems. [S.l. : s.n.], 2000: 1057-1063.
[30] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation[J]. ArXiv preprint arXiv:1506.02438, 2015.
[31] Kakade S, Langford J. Approximately optimal approximate reinforcement learning[C]//ICML: vol. 2. [S.l. : s.n.], 2002: 267-274.
[32] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3-4): 229-256.
[33] Konda V R, Tsitsiklis J N. Onactor-critic algorithms[J]. SIAM journal on Control and Optimization, 2003, 42(4): 1143-1166.
[34] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
[35] Lopez P A, Behrisch M, Bieker-Walz L, et al. Microscopic Traffic Simulation using SUMO[C]// The 21st IEEE International Conference on Intelligent Transportation Systems. [S.l.]: IEEE, 2018.

[36] Kesting A, Treiber M, Helbing D. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity[J]. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2010, 368(1928): 4585-4605.
[37] Treiber M, Hennecke A, Helbing D. Congested traffic states in empirical observations and microscopic simulations[J]. Physical Review E, 2000, 62(2): 1805-1824.
[38] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. ArXiv preprint arXiv:1412.6980, 2014.
[39] Hoel C J, Wolff K, Laine L. Automated speed and lane change decision making using deep reinforcement learning[J]. Proc. Int. Conf. Intell. Transp. Syst. (ITSC), 2018.
[40] Xu Z, Tang C, Tomizuka M. Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control[J]. Proc. Int. Conf. Intell. Transp. Syst. (ITSC), 2018.

Acknowledgment

The completion of this work could have not been possible without the assistance and guidance of Prof. Meiling Wang and Prof. Ching-Yao Chan. Their guidance and instructions throughout this work are sincerely appreciated and gratefully acknowledged.

Besides, I wish to record my gratitude to Dr. Pin Wang, Dr. Fei Ye, Huanjie Wang, Wei Wang, Zhongyu Li, Xuebing Peng and many other friends who helped me during the year in UC Berkeley.

It is very lucky of me to have met many friends throughout the four years of undergraduate life. I would like to thank all my friends for having fun with me.

I would like to express my gratitude to my parents and my family for their support, encouragement and love throughout my entire life.

Last but not least, thank you to Lujie, for all her love and support along the way we have traveled.