Confidence-Aware Reinforcement Learning for Self-Driving Cars

Zhong Cao, Shaobing Xu, Huei Peng, Diange Yang, and Robert Zidek

***Abstract*— Reinforcement learning (RL) can be used to design smart driving policies in complex situations where traditional methods cannot. However, they are frequently black-box in nature, and the resulting policy may perform poorly, including in scenarios where few training cases are available. In this paper, we propose a method to use RL under two conditions: (i) RL works together with a baseline rule-based driving policy; and (ii) the RL intervenes only when the rule-based method seems to have difficulty handling and when the confidence of the RL policy is high. Our motivation is to use a not-well trained RL policy to reliably improve AV performance. The confidence of the policy is evaluated by Lindeberg-Levy Theorem using the recorded data distribution in the training process. The overall framework is named "confidence-aware reinforcement learning" (CARL). The condition to switch between the RL policy and the baseline policy is analyzed and presented. Driving in a two-lane roundabout scenario is used as the application case study. Simulation results show the proposed method outperforms the pure RL policy and the baseline rule-based policy.**

***Index Terms*— Automated vehicle, reinforcement learning, motion planning.**

I. Introduction

AUTONOMOUS driving technologies have great potential to improve vehicle safety and mobility in various driving scenarios [1]. Currently, several industrial (confidential) autonomous vehicles have achieved impressive performance. In 2018, Waymo vehicles [9] tested in California achieved an impressive record of driving, more than 17,000 km without disengagement. In the meantime, hundreds of take-over (challenging) cases have been recorded. Let's assume those take-over cases are truly necessary, i.e., engineers must improve their current AV algorithm to handle those hundreds of challenging cases. What would one do? Tuning the current policy trying to handle the challenging cases is not a sure bet, as the vehicle performance may deteriorate for driving in "the rest of 17,000 km", resulting in new challenging cases.

Data-driven methods [2], e.g., reinforcement learning (RL), can learn from collected driving data, are potential ways to design smarter driving policies. The ability of RL-based driving policy has been proved in many scenarios, e.g., on-ramp merging [3], highway exiting [4], intersection driving [5] *et al.* However, fully autonomous driving is not a simple connection of some individual scenarios. There may be various undefined scenarios that the policy should handle at the same time. Directly training a RL policy for all "17,000 km" also seems to be a risky approach, since it requires an increasingly large amount of data and long-time training to cover most scenarios. A not-well-trained RL policy may not be trustworthy and not be better than the current principle-based policy. In fact, according to a research report [6], only the policy evaluation process in the RL model requires at least 8.8 billion miles of driving data. When there is only a limited amount of data, the RL policy may not be well trained and fail in some cases. Most research works for RL-based autonomous driving focus on the learning speed, sample complexity, or computational complexity, but cannot provide a guarantee of the performance [10] with limited training data. The un-explainable and not quantifiable nature of RL training prevents RL technology from mission-critical real-world applications such as autonomous vehicles.

Since current self-driving systems [7], [8] can drive in most cases, using explainable algorithms, e.g., rule-based or model-based, our motivation is: can we guarantee the "trustworthy improvement" by RL technology? Namely, a not-well-trained RL policy still can outperform a given driving policy e.g., industrial autonomous driving system. This work can enable the RL technology to directly improving the fully autonomous driving performance instead of waiting for the ideal training.

In the reinforcement learning domain, the guarantee of policy performance relates to the safe reinforcement learning topic [11]. Typical methods can be briefly divided into 1) Expert policy heuristic, and 2) Dangerous action correction.

The expert policy heuristic method is to first imitate an expert policy, e.g., a human-engineering policy, and then learn for better performance. The expert policy imitation can use supervised learning [2], inverse reinforcement learning [12], or add the expert policy heuristic reward [13]. Then, the reinforcement learning will continue to update this policy for better performance. Ideally, the imitated policy will have the same performance as the human-engineering policies and the RL will further improve these policies. However, both the imitation process and the RL training require massive training data, otherwise, the final policy cannot be always better than the expert policy.

Another way to guarantee the safety of reinforcement learning is to correct dangerous actions. A naïve approach

Manuscript received June 28, 2020; revised December 10, 2020, February 15, 2021, and March 22, 2021; accepted March 25, 2021. The Associate Editor for this article was D. F. Wolf. *(Corresponding author: Diange Yang.)*

Zhong Cao and Diange Yang are with the School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: caozhong@tsinghua.edu.cn; ydg@tsinghua.edu.cn).

Shaobing Xu and Huei Peng are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: xushao@umich.edu; hpeng@umich.edu).

Robert Zidek is with Toyota Research Institute, Ann Arbor, MI 48105 USA (e-mail: robert.zidek@tri.global).

Digital Object Identifier 10.1109/TITS.2021.3069497

is directly adjusting the AV actions by a rule-based safeguard [16]. However, the resulted policy can be too conservative and may not generate an acceptable solution in high-speed scenarios. Some methods also correct actions by designing the safety-oriented reward function and introducing a penalty when the policy leads to a dangerous outcome. For example, in [14] and [15], minimum distance in car-following cases was used to compute the cost function. These methods still require massive training data, or sophisticated safeguards. They cannot guarantee better performance than the given autonomous driving system.

The relevant works to consider the RL training policy confidence is safe policy improvement [18]. Different from the classical RL policy update process, i.e., continuously updating the policy during training, the safe policy improvement methods will prevent updating the policy when the new policy does not have enough confidence. This work provides a way to analyze the confidence of a policy, according to the training data. Following this idea, [17], [19] improve the data sampling efficiency and release the data-collection-policy requirement. [20] further applies this method in non-stationary Markov decision process. [21] calculates the policy confidence, considering the training data error. However, without enough data, the generated policies still may fail in few training cases.

Different from the above methods, this paper proposes a confidence-aware reinforcement learning (CARL) method, involving a principle-based self-driving system. The brief idea is "Not always believe RL model, but activate it where it learns". Namely, given a not-well-trained RL model, the proposed CARL can use a principle-based policy to guarantee AV's statistical lower bound. The generated CARL policy will outperform the principle-based policy as well as the pure RL model. The detailed contributions include:

1) A confidence-aware reinforcement learning framework, containing an RL planner and a baseline principle-based planner. By activating the better planner, the final hybrid policy can be better than either of the planners.

2) A method to estimate confidence levels of both the RL planner and the baseline principle-based planner, according to the distribution of recorded data.

3) A reliable detector to determine when to activate the RL planner.

The remainder of this paper is organized as follows: Section II introduces the preliminaries and formally defines the problem. The policy confidence level is defined in Section III. Section IV designs the confidence-aware reinforcement learning system for AV planning. Section V shows the application scenario and simulation results. Finally, Section VI concludes this work.

II. Preliminaries and Problem Definitions

A. Preliminaries

The trajectory planning problem of autonomous vehicles is formulated as a Markov decision process (MDP) or a partially observation Markov decision process (POMDP) following the process shown in [22]. The MDP assumes the problem satisfies the Markov property: the conditional probability distribution of future states depends only on the present state. In this paper, the MDP notation is adopted, while the algorithm can be translated into a POMDP problem by replacing states with observations [23]. The agent should optimize long-term rewards through a general sequential decision-making setting. Reinforcement learning is used to solve this problem, such that the agent will learn a policy that maps from rich observation to actions.

More specifically, a finite horizon MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, r, \mathcal{P})$ consisting of:

- a context state space $\mathcal{S}$;
- an action space $\mathcal{A}$;
- a reward function r: $\mathcal{S} \rightarrow \mathbf{R}$;
- a transition operator (probability distribution) $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbf{R}$;
- a discount factor $\gamma \in (0, 1]$, set as a fixed value;
- the finite planning horizon defined as $H \in \mathbf{N}$.

A general policy $\pi \in \Pi$ maps each context state to a distribution over actions (i.e., $S \rightarrow P(S)$). Namely, $\pi(a|s)$ denotes the probability (density) of taking action a at state s using the policy π. This paper uses the fixed policy setting for simplification, i.e., $a_\pi(s) = \text{argmax}_a(\pi(a|s))$. Here, Π denotes the set containing all candidate policies π. A policy π has associated value and action-value functions. For a given reward function, these functions are defined as:

$$\begin{aligned} &\forall h \in \mathbf{N}, \\ V_\pi(s) &:= \mathbb{E}_\pi[\sum_{t=h}^{h+H-1} \gamma^{t-h} r_t | s_h = s] \\ Q_\pi(s,a) &:= \mathbb{E}_\pi[\sum_{t=h}^{h+H-1} \gamma^{t-h} r_t | s_h = s, a_h = a] \end{aligned} \tag{1}$$

where $a_t \sim \pi(a_t|s_t)$. $\mathbb{E}[\cdot]$ denotes the expectation with respect to the environment transition distribution. r_t denotes the reward at time t. The subscript π in $V_\pi(s)$ and $Q_\pi(s,a)$ means these functions' value depends on the policy π.

The goal of RL is to find the optimal π to generate the next action that maximizes the expected reward, denoted by:

$$\begin{aligned} &\forall s_t \in \mathcal{S}, \\ V_\pi(s_t) &= \text{argmax}_\pi \mathbb{E}_{s_{t+1} \sim P}[r(s_t, a_\pi) + \gamma V_\pi(s_{t+1})] \end{aligned} \tag{2}$$

where s_{t+1} is the next state generated from s_t, a_π, namely, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. Eq. (2) is the Bellman optimality equation. The collected data is defined as follows:

$$\begin{aligned} \tau_\pi(s) &:= \{s, a_1^\tau, s_2^\tau, a_2^\tau, ..., s_H^\tau\} \\ D_\pi &:= \{\tau_\pi(s_i)\}, s_i \in \mathcal{S} \\ G(\tau_\pi(s)) &:= \sum_i \gamma^n (r(s_i^\tau, a_i^\tau)) \end{aligned} \tag{3}$$

where, $\tau_\pi(s)$ denotes a trajectory of length H, which is a sequence of states and actions. The starting state of $\tau_\pi(s)$ is s. In this trajectory, the return value means the sum of all the discounted reward, denoted by $G(\tau_\pi(s))$. The dataset D_π saves all these trajectories, collected by the policy π to train the reinforcement learning policy.

In this paper, we use the deep Q learning framework as the RL policy generator. Note that other RL framework may also work but were not explored in this work.

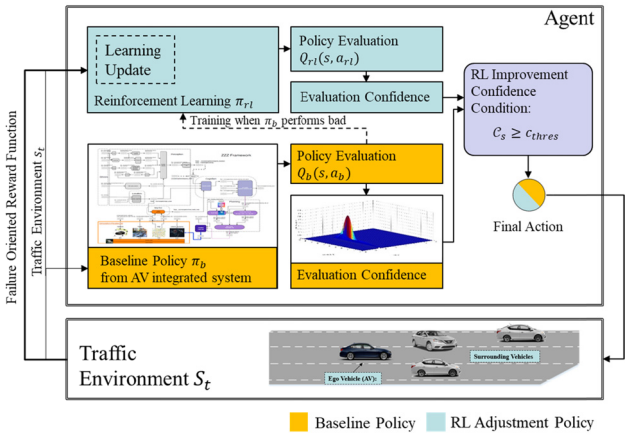


Fig. 1. Confidence-Aware RL Framework.

B. Problem Definitions

The problem studied in this paper is defined as follows: *An AV has a rule-based driving policy, known as the baseline policy. It performs well largely but fails occasionally. The proposed method aims to train an RL policy to improve the baseline policy (when it may fail) but only has limited training data. A confidence index is then designed to evaluate the driving confidence of both the baseline policy and the RL. Finally, by activating the higher confidence policy, the hybrid planner can be better than either of the policies.*

The confidence-aware framework is shown in Fig. 1, which consists of two parts: traffic environment and the agent. The agent generates the action from the observed states, and contains two policies, the baseline principle-based policy π_{b}, and the RL policy π_{rl}. The agent first evaluates the performance of the two policies and then computes confidence according to the collected data. The agent then chooses the better policy with high confidence to drive. As results, the final performance should be better than either of the policies, even with limited training data. The methods of policy evaluation, confidence computation and the RL policy activation condition are introduced in Section III.

In this problem, the input and output of the baseline policy are the same with the state space and action space of the RL policy. Furthermore, the failure cases in the problem description, are defined by the terminal state sets $\mathcal{T}$, i.e., $\mathcal{T} \subset \mathcal{S}$. The reward function for evaluation and training process is largely associated with the failure cases, as shown in Eq. (4). This is a sparse reward setting, which increases the difficulty of RL training.

$$r(s) = \begin{cases} -1, & \text{if } s \in \mathcal{T} \\ 0, & \text{else} \end{cases} \tag{4}$$

III. Driving Policy Performance and Confidence

As we know, a policy in RL is evaluated by the reward expectation $Q_\pi(s, a_\pi)$. The policy evaluation may be less accurate if the training data is not sufficient. We propose a confidence index to quantify the quality of policy evaluation. The policy evaluation and its confidence estimation are discussed in the following subsections.

A. Driving Policy Evaluation

The driving policy evaluation process can use the Monte Carlo policy evaluation method to estimate the state-value function. Firstly, define the sub-dataset as follows

$$D_\pi(s_i) = \{\tau_\pi(s_i)\} \tag{5}$$

where $D_\pi(s_i)$ denotes the dataset starting from the state s_i. According to the Markov property, these collected data, i.e., trajectories defined in Eq. (5), are independent and identically distributed (i.i.d) sampling. The state-value function is then estimated by

$$\begin{aligned} &\forall s_i \in \mathcal{S}, \\ &Q_\pi(s_i, a_\pi) \leftarrow \bar{G}(\tau_\pi(s_i)) = \frac{1}{n}\sum\nolimits_i (G(\tau_\pi(s_i))) \end{aligned} \tag{6}$$

where $Q_\pi(s_i, a_\pi)$ is the true performance value and $\bar{G}(\tau_\pi(s_i))$ is the estimated value. With a large number of samples, $\bar{G}$ approaches Q_π.

However, due to the traffic environment uncertainties and limited collected data, the estimated value $\bar{G}$ may be different from Q_π significantly. This CARL should further consider the policy value function confidence.

B. Driving Policy Confidence

The true value $Q_\pi(s_i, a_\pi)$ only exists in concept and cannot be observed directly. Here we define a distribution of Q_π to describe the probability of Q_π around the estimated value $\bar{G}$, denoted by $P(Q_\pi)$. This distribution is called the true value distribution. The true value distribution is calculated based on Lindeberg-Levy Theorem:

$$\begin{aligned} &\forall z \in \mathbf{R}, s_i \in \mathcal{S} \\ &\lim_{n\to\infty} \mathrm{P}\left[\sqrt{n}(\bar{G}_\pi(s_i) - Q_\pi(s_i, a_\pi(s_i))) \leq z\right] = \Phi\left(\frac{z}{\sigma}\right) \\ &\sigma^2 = \mathrm{Var}(G_\pi(s_i)) \end{aligned} \tag{7}$$

where $G_\pi(s_i)$ is the shorthand of $G(\tau_\pi(s_i))$. $\Phi(z)$ is the standard normal cumulative distribution function (CDF) evaluated at z. As the number of samples increases, the probability, the true value falls near the estimated value, increases.

The CARL framework should activate the policy with higher policy value. Instead of comparing the estimated value $\bar{G}$ explicitly, the proposed CARL will use the policy improvement probability as the index to decide the RL policy activation time. The policy improvement probability $\mathcal{C}(s_i)$ is defined as the probability that the RL policy can improve the baseline policy, shown in Eq. (8).

$$\mathcal{C}(s_i) = P(Q_{\mathrm{rl}}(s_i, a_{\mathrm{rl}}(s_i)) \geq Q_{\mathrm{b}}(s_i, a_{\mathrm{b}}(s_i))) \tag{8}$$

where $s_i \in \mathcal{S}$. Subscript rl, b denote the RL policy and baseline policy, respectively. $\mathcal{C}(s_i)$ denotes the policy improvement confidence. Combining Eqs. (7) and (8), the policy improvement

confidence can be further calculated as:

$$\begin{aligned}\mathcal{C}(s_i) &= P(Q_{\mathrm{rl}}(s_i, a_{\mathrm{rl}}(s_i)) \geq Q_{\mathrm{b}}(s_i, a_{\mathrm{b}}(s_i))) \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{0} \exp(-\frac{\mu}{-2\sigma^2})\mathrm{d}x \\ \mu &= \bar{G}_{\mathrm{rl}}(s_i) - \bar{G}_{\mathrm{b}}(s_i), \sigma^2 = \frac{\sigma^2(G_{\mathrm{rl}}(s_i))}{n_{\mathrm{rl}}} + \frac{\sigma^2(G_{\mathrm{b}}(s_i))}{n_{\mathrm{b}}} \\ n_{\mathrm{rl}}, n_{\mathrm{b}} &\geq n_{\mathrm{thres}}\end{aligned} \tag{9}$$

where $s_i \in \mathcal{S}$. $n_{\mathrm{rl}}, n_{\mathrm{b}}$ denote the number of collected trajectories in the dataset $D_{\mathrm{rl}}(s_i)$ and $D_{\mathrm{b}}(s_i)$, respectively. When the collected trajectories are less than $n_{\mathrm{thres}} = 30$, we set $\mathcal{C}(s_i) = 0$.

In this framework, the baseline policy should guarantee the lower-bound of the whole system, namely, RL policy is used only when it can improve the baseline policy. To this end, we define a confidence threshold c_{thres} such that only when $\mathcal{C}(s_i) \geq c_{\mathrm{thres}}$, the RL policy works to improve the baseline policy performance. The value of c_{thres} should satisfy the following condition:

$$\begin{aligned}&\forall s_i \in \mathcal{S} \\ &\mathbb{E}(Q_{\pi_{\mathrm{rl}}}(s_i, a_{\mathrm{rl}}(s_i))) - \mathbb{E}(Q_{\pi_{\mathrm{b}}}(s_i, a_{\mathrm{b}}(s_i)) \geq 0) \geq 0 \\ &P(Q_{\pi_{\mathrm{rl}}}(s_i, a_{\mathrm{rl}}(s_i)) \geq Q_{\pi_{\mathrm{b}}}(s_i, a_{\mathrm{b}}(s_i))) \geq 0.5 \\ &\Rightarrow c_{\mathrm{thres}} \geq 0.5\end{aligned} \tag{10}$$

where c_{thres} denotes the confidence threshold. c_{thres} =0.5 is the threshold value to use the RL policy instead of the baseline policy.

IV. Confidence Aware Reinforcement Learning

This section describes the data collection process and generates the CARL hybrid policy using the two sub-policies, i.e., $\pi_{\mathrm{b}}, \pi_{\mathrm{rl}}$.

A. Data Collection

We use a fixed horizon sliding window to collect the trajectory $\tau_\pi(s_i)$ and its value return $G(\tau_\pi(s_i))$ as follows:

$$\begin{aligned}\omega_\pi &:= \left\{s_1^\epsilon, a_1^\epsilon, \ldots, s_m^\epsilon \in \mathcal{T}\right\}, \quad s_{1\ldots m-1}^\epsilon \notin \mathcal{T} \\ \tau_\pi(s_i^\epsilon) &:= \{s_i^\epsilon, a_i^\epsilon, s_2^\epsilon, a_2^\epsilon, ..., s_k^\epsilon\}\end{aligned} \tag{11}$$

where $k = \min(i + H - 1, m)$. The ω_π is the set of the collected trajectory using policy π. Different sub-datasets are then defined for policies evaluation and confidence analysis:

$$\begin{aligned}D(s, a) &= D_{\mathrm{b}}(s) \cup D_{\mathrm{rl}}(s) \\ D_{\mathrm{b}}(s) &:= \{\tau(s_1 = s, a_1 = \pi_{\mathrm{b}}(s_1)\} \\ D_{\mathrm{rl}}(s) &:= \{\tau(s_1 = s, a_1 = \pi_{\mathrm{rl}}(s_1)\}\end{aligned} \tag{12}$$

where $D(s, a)$ is divided into two sub-datasets, using the first action. Namely, $D_{\mathrm{b}}(s)$ contains the trajectory, where the first action uses the baseline policy. In the meantime, $D_{\mathrm{rl}}(s)$ contains other trajectories. This is because the most significant influence on the results is usually due to the first action. Then the policy improvement confidence can be calculated from these datasets using Eq. (8).

$$\mathcal{C}(s_i) \leftarrow D_{\mathrm{b}}(s), D_{\mathrm{rl}}(s) \tag{13}$$

The RL policy is updated during driving, which means that the action generated by the RL policy keeps changing and the collected data is not strictly associated with the final RL policy. These problems will be further considered in the RL policy training process (Section IV-B) and the hybrid policy generation module (Section IV-C).

B. RL Policy Generation

According to Eqs. (9) and (13), the calculation of the policy improvement confidence does not rely on the RL training model, but requires a trained RL policy. This work takes the deep Q learning model as an example to train the RL policy, while other RL models, e.g. PPO, SAC or TD3, can also be applied in the following research. A better RL policy could have more opportunities to be activated and further demonstrate better performance. The deep Q learning uses an updated law as follows:

$$\begin{aligned}&Q(s_t, a_t) \\ &\leftarrow Q(s_t, a_t) + \alpha[r(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]\end{aligned} \tag{14}$$

where α denotes the learning rate. Since the state space is continuous and has high dimensions, the deep Q learning uses a neural network to approximate the Q value function. The Q value function in deep Q learning is defined as:

$$\begin{aligned}\theta_{k+1} &\leftarrow \theta_k - \alpha\nabla_\theta\mathbb{E}[(Q(s_t, a_t, \theta) - \hat{Q}(s_t, a_t))^2] \\ \hat{Q}(s_t, a_t) &= r(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a, \theta^-)\end{aligned} \tag{15}$$

where θ and θ^- denote the parameters of the prediction Q value network and the target Q value network, which are updated by the training data. θ^- is updated to θ after $n_{\mathrm{update}} =$ 100 iterations. This setting helps the network improving more consistently. $(Q(s_t, a_t, \theta) - \hat{Q}(s_t, a_t))^2$ is the loss function for training, where $\hat{Q}(s_t, a_t)$ means the value calculated by the bellman function in Eq. (2)

To efficiently collect the required data, the CARL method designs the RL training process into two stages: the baseline policy value estimation stage and the RL model exploration stage. In the first stage, the AV only uses the baseline policy to collect the dataset $D_{\mathrm{b}}(s)$. In the second stage, the RL policy may explore different actions from the baseline policy value for better performance. This paper focuses on when the RL exploration should be triggered, and utilizes the ϵ greedy algorithm for exploration. The overall data collection and RL policy exploration algorithm is described in Alg. 1.

where $\bar{G}_{\pi_{\mathrm{b}}}(s(t))$ denotes the estimated value from the dynamically updated dataset $D(s(t), a_{\mathrm{b}})$. n_{b} denotes the size of $D(s(t), a_{\mathrm{b}})$. Iterations denote the rounds of total training and evaluation.

During the iteration of this algorithm, AV should choose whether to use baseline policy to collect more data or to do RL policy exploration. Two conditions prevent exploration: 1) the baseline policy has not been well evaluated, i.e., $n_{\mathrm{b}} < n_{\mathrm{thres}}$; 2) the baseline policy performs well; The second condition setting uses a uniform distributed random variable $\xi \sim U(0, 1)$,

Algorithm 1: Data Collection and RL Policy Exploration

Input : iterations,Horizon H
Output: RL exploration trigger signal

$i = 0$;
$trajectory = \emptyset$;
$D(s,ab) = \emptyset$;
$D(s,a) = \emptyset$;
while $i < iterations$ **do**
  $i = i + 1$;
  if $s(t) \in T$ **then**
    Sample $\xi \leftarrow U(0, 1)$;
    if $n_b > n_{thres}$ *and* $\xi > \overline{G}_{\pi b}\ (s(t)) + 1$ **then**
      Generate exploration action $a_{explore}$;
      apply $a_{explore}$ to ego vehicle;
    else
      apply a_b to ego vehicle;
    end
    $\tau = \{s_l, a_l, \ldots, s_H, a_H\}$;
    if $a_1 = \pi_b\ (s)$ **then**
      add τ into $D(s_1,a_b)$;
    else
      add τ into $D(s_1, a = a_1)$;
    end
    Remove (s_1, a_1, r_1) from trajectory
  else
    $m = length(trajectory)$;
    for $i = 1$ *to* m **do**
      $\tau = \{s_i, a_i, \ldots, s_m, a_m\}$;
      if $a_i = \pi_b(s)$ **then**
        add τ into $D(s_i, a_b)$;
      else
        add τ into $D(s_i, a = a_1)$;
      end
    end
    $trajectory = \emptyset$;
    for $i = 1$ *to* H **do**
      apply a_b to ego vehicle;
    end
  end
end

which is sampled every step. The AV explores other actions only when $\xi \geq \bar{G}_{\pi_b}(s(t)) + 1$, namely, the exploration probability equals to $-Q(s, a_b)$. These two conditions avoid useless exploration and make the policy training more efficient.

Since the defined state space is continuous, we discretize the sampled data to compute the data distributions. The step for a dimension s^i is $b(\max(s^i) - \min(s^i))$, $b \in (0, 1)$ is an ad-hoc parameter which is set to 0.1. For efficiently getting $D(s, a)$, the algorithm uses R-Tree [24] data structure to store the collected data.

It should be noted that the RL policy keeps updating during training, and the actions generated by the RL policy will change during driving. Therefore, the RL policy evaluation data should be recorded by actions, i.e., $D(s(t), a \in \mathcal{A})$.

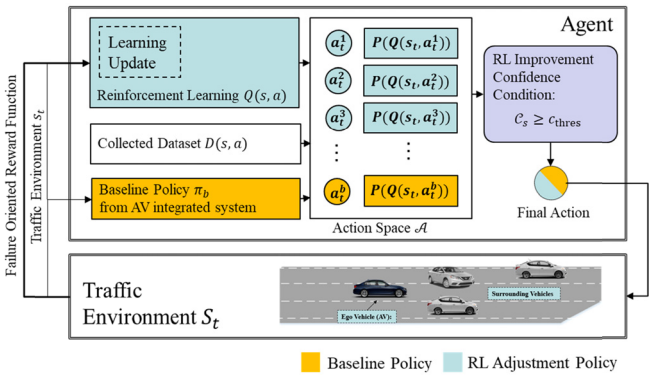


Fig. 2. Hybrid Policy Generation.

C. Hybrid Driving Policy

Once the RL policy is trained, the confidence aware RL framework will activate the RL policy to improve the baseline policy according to policy evaluation and confidence analysis. Fig. (2) shows how the final policy is generated. The details are described in this section.

When visiting a state s_t, the framework will first calculate the baseline policy's true value distribution $P(Q(s_t, a_b(s_t)))$ CDF according to Eq. (7), using the baseline policy evaluation data $D(s_t, a_b(s_t))$

According to the deep Q learning algorithm, $a_{\text{rl}} = \text{argmax}_{a \in \mathcal{A}} Q(s_t, a)$. Therefore, the RL policy's true value distribution should rely on $D(s_t, a_{\text{rl}})$. This strategy is to select the largest Q value but it actually cannot maximize the possibility of improving the baseline policy. Therefore, this framework will adjust the RL policy as follows:

$$a = \begin{cases} a_{\text{rl}}(s_t), \text{if } P(Q(s_t, a_{rl}) - Q(s_t, a_b(s_t) \geq 0)) \geq c_{\text{thres}} \\ a_{\text{b}}(s_t), \text{else} \end{cases} \tag{16}$$

where $a_{\text{rl}}(s_t) = \text{argmax}_{a \in \mathcal{A}} P(Q(s_t, a) - Q(s_t, a_b(s_t) \geq 0))$, the $P(Q(s_t, a))$ is estimated from the collected dataset $D(s_t, a)$. The optimal value of c_{thres} is 0.5, according to the Eq. (10)

Eq. (16) chooses the better policy to drive and avoids activating RL when the confidence is low, i.e., trained by few cases. Therefore, the final performance can be better than the rule-based policy and the RL policy, even with limited data. With more training data, the CARL can activate RL policy in more time, which means the policy can be gradually improved. The following section will implement the CARL in an example scenario to show these benefits.

V. Case Study

A. CALR Evaluation Setting

1) Evaluation Environment Setting: To test our method, we design a roundabout scenario using the Carla simulator [25], as shown in Fig. 3. This roundabout has two lanes, where vehicles can change lanes for safety or mobility, and has eight ramps, where traffic can merge in and exit. The traffic in this simulation roundabout is designed to be

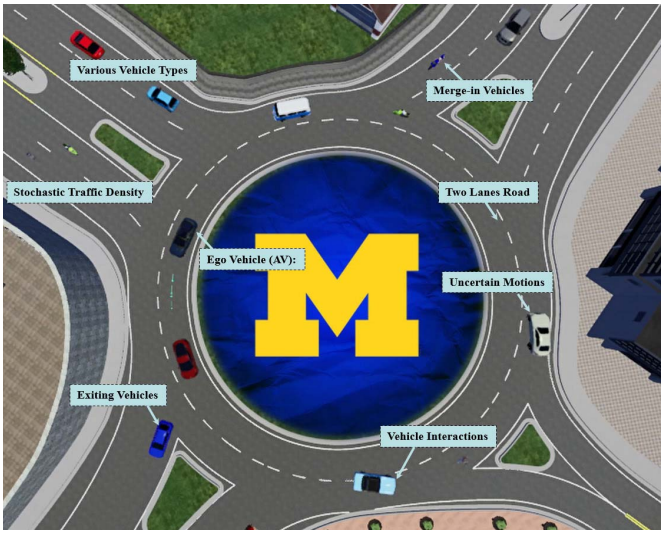


Fig. 3. Roundabout Scenario in Carla.

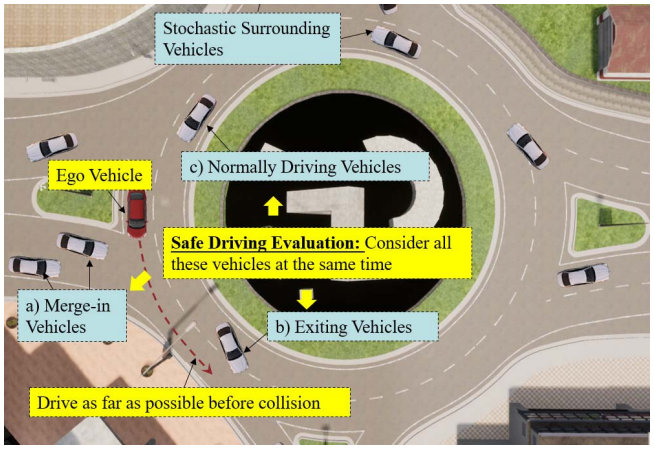


Fig. 4. Safe Driving Evaluation in Roundabout Scenario.

stochastic and aggressive. Surrounding vehicles have various types, including trucks and sedans, and are generated in a random position. Each vehicle is designed to enter and exit this roundabout frequently with aggressive and uncertain motions. The environment continuously generates the vehicles at each time step, forming stochastic traffic density and scenarios. This dangerous environment setting can accelerate the evaluation process. Furthermore, the uncertain environment makes the ego vehicle meet different surrounding traffic in each driving process. It will force the generated policy to adapt to the uncertain surrounding environment and overcome the overfitting problem. This roundabout scenario provides the environment to evaluate the proposed method, but not limits the applications. Other scenarios can also use the CARL method to generate driving policies.

2) Evaluation Method and Performance Metric: This work uses safe driving distance to evaluate the policy performance, shown in Fig. 4. Namely, the ego AV is required to drive in a roundabout as far as possible until it collides with other vehicles. The safety performance metric is the average safe driving distance, quantified by distance per collision,

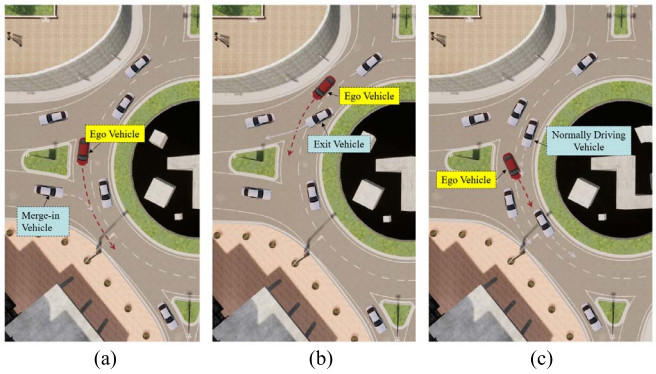


Fig. 5. Roundabout Scenario in Carla. (a) collision caused by a merge-in vehicle; (b) collision caused by an exiting vehicle; (c) collision caused by a normal driving vehicle: ego vehicle is changing lane to avoid a collision with a front vehicle, but a rear vehicle does not slow down in time.

defined as:

$$d_s = \frac{d}{\text{collisions}} \tag{17}$$

where d denotes the total AV driving distance in the ~10 h driving.

This safe-driving evaluation method brings increasingly difficulties to the driving policies and leads to high risk. The ego vehicle should not only react to the surrounding vehicle merging in, but also consider them normally driving, e.g., lane-following and lane-changing, as well as exiting from the roundabout. The combination of these three types of surrounding vehicles and their random initial positions will form more and more complex traffic scenarios. For example, when the ego vehicle drives a single circle in the roundabout, it may experience five merging-ins, three exiting, and six lane-changes of surrounding vehicles at the same time. The collision may happen in various ways. We plot some collision examples in Fig. 5. Such an environment brings a high-dimension driving problem and makes it difficult to design a perfect policy to deal with all these cases. However, this environment is more realistic to the naturistic driving, since the real driving process is not a simple connection of some individual cases.

3) Expected Results for Trustworthy Improvement by CARL Method: The most impressive superiority of this work is "trustworthy improvement" by a not-well-trained RL policy. Therefore, the CARL should be evaluated under the condition that 1) a principle-based policy is given, which may get collisions and 2) the RL policy does not have enough training data. The expected results are that the CARL policy can outperform the principle-based policy, while only the RL policy cannot. These results should come from the comparison of all these three methods' collision risk. If this conclusion holds, further improving the principle-based policy or training RL policy will improve the final performance of the CARL system.

The details of the reinforcement learning framework for this problem are presented in Section V-B including the state space, action space, reward and training parameters. To evaluate the "trustworthy improvement" by a not-well-trained RL policy,

this work designs a fixed baseline principle-based policy in CARL framework, described in Section V-C.

B. Reinforcement Learning Design

1) State Space: For a general state space representation, a scenario translator is designed to convert this roundabout scenario into a two-lane straight road with many on/off ramps. The state space is defined as

$$s \in \mathcal{S}, \quad s = \{q_{\mathrm{e}}, q_{0\mathrm{f}}, q_{0\mathrm{r}}, q_{1\mathrm{f}}, q_{1\mathrm{r}}\} \tag{18}$$

where $q = \{x, y, \dot{x}, \dot{y}\}$, denoting the kinematic state of a vehicle in the converted lane coordinate system. x denotes the distance to the origin point along the lane, y denotes lateral offset to the central lines. The subscript number of q denotes the lane id. The subscript f, r denotes the vehicle immediate to the front/rear of the ego vehicle, respectively. In general, there are as many as five surrounding vehicles. If there is no vehicle at the defined position, a default virtual vehicle that does not affect the ego vehicle's decision will occupy the state space.

The reason to convert the roundabout scenario into a straight road is to unify state-space representation in a concise way, crucial for RL training. In detail, describing a vehicle state requires not only the position and velocity, but also the relationships with the lanes and surrounding vehicles, e.g., which lane it is driving in, or whether it is behind or in front of another vehicle. Using the natural world coordinate system (WCS), without lanes' information, is not efficient to describe these necessary relationships. Therefore, we build a lane coordinate system (LCS). Such a system converts the roundabout's curved lanes into straight lanes, by taking the central line as the reference paths. The values of x, y can then directly represent the longitudinal and lateral relationship between vehicles and lanes. This strategy enables better scalability, i.e., using one unified model to handle various scenarios.

Here, we take one lane as an example to build the LCS. In the roundabout scenario, the starting point and the ending point of the lane are the same, which is the opposite (i.e., farthest) point with the ego vehicle. The central line of this lane is along the driving direction of the vehicles. Then the LCS is built as follows:

$$\begin{aligned} h(t) &= \frac{c_0}{c_i}\int_0^t (\|\mathrm{c}'(\sigma)\|)d\sigma, \quad \vec{c}(s) = \vec{c}(t(h)) \\ \vec{T} &= \frac{d\vec{c}/d\vec{h}}{\|d\vec{c}/d\vec{h}\|}, \quad \vec{N} = \frac{d\vec{T}/d\vec{h}}{\|d\vec{T}/d\vec{h}\|} \\ \frac{\vec{N}}{dh} &= -\kappa\vec{T}, \quad \theta = \arctan\frac{\vec{T}_y}{\vec{T}_x} \end{aligned} \tag{19}$$

where $s(t)$ denotes the length from the starting point to a point along the central line, and the coordinate of this point in WCS is $\vec{h}(s)$. $\vec{T}$ and $\vec{N}$ denote the tangent and normal vectors. c_i denotes the radius of the i^{th} roundabout lane, which makes the lane length of the two lanes the same in LCS. The following equations convert the observed vehicle state from WCS into LCS.

$$\vec{x}(h, l) = \vec{c}(h) + \frac{l(h)}{w}\vec{N}_r(h)$$

$$\begin{aligned} l &= \frac{[\vec{x} - \vec{c}]^T}{w}\vec{N}_c(h) \\ v_x &= ||\dot{\vec{x}}||, \quad \Delta\theta = \theta_x - \theta_c \\ \dot{l} &= v_x \sin\Delta\theta \\ \dot{h} &= \frac{\dot{\vec{x}} - \dot{l}\vec{N}_c}{(1-\kappa_r)l\vec{T}_c} \\ y_l &= l + l_{\mathrm{id}} \end{aligned} \tag{20}$$

where $\vec{x}, \dot{\vec{x}}$ denote the vehicle observed state in WCS and $(h, y_l, \dot{h}, \dot{l})$ denote the corresponding coordinate in LCS. $\vec{c}(s)$ denotes the point on the central line, constrained by $(\vec{c}(s) - \vec{x}) \cdot \vec{T}_c = 0$. The variables with the subscript c denote the attributes of this point. w denotes the lane width, which is 1 in the LCS. This setting makes the y_l are the same when calculating the vehicle coordinates in different LCSs.

The front and rear vehicles in the state space can be calculated as follows:

$$\begin{aligned} \forall q_i &= (h^i, y_l^i, \dot{h}^i, \dot{l}^i) \in \mathbb{L}_{\mathrm{id}}, \text{s.t.} \\ (|y_l^i - l_{\mathrm{id}}| \le 0.5) &\vee |y_l^i - l_{\mathrm{id}}| \in [0.5, 1) \wedge (y_l^i - l_{\mathrm{id}})\dot{l}^i < 0 \\ &\Rightarrow q_{\mathrm{id,f}} = \mathrm{argmin}_{q^i \in \mathbb{L}_{\mathrm{id}}, h^i > h^{\mathrm{e}}}(h^i - h^{\mathrm{e}}) \\ q_{\mathrm{id,r}} &= \mathrm{argmax}_{q^i \in \mathbb{L}_{\mathrm{id}}, h^i < h^{\mathrm{e}}}(h^i - h^{\mathrm{e}}) \end{aligned} \tag{21}$$

where $\mathbb{L}_{\mathrm{id}}$ denotes the vehicles on the i^{th} lane. Vehicles changing into a lane, are also considered.

Using LCS, the policy developed for the straight roads, can be used in the round-about scenario. The vehicles, far away from the lanes, will be deleted.

2) Action Space: The action space in LCS is defined as follows:

$$\begin{aligned} a &= (\ddot{x}_{\mathrm{e}}, y_{\mathrm{lc}} \in \mathcal{A}) \\ a &\in \{a_t^{\mathrm{b}}, \{a_t^{\mathrm{c}}\}\} \end{aligned} \tag{22}$$

where $\ddot{x}_{\mathrm{e}}$ denotes the longitudinal acceleration, selected from the hard brake, an acceleration value, slight deceleration value and zero to adjust or maintain the speed. The values of y_{lc} are assumed to be limited to the set $\{0, 1/k_{\mathrm{lc}}, -1/k_{\mathrm{lc}}\}$, representing lane-keep (0), lane-change right $(-1/k_{\mathrm{lc}})$, and lane-change left $(1/k_{\mathrm{lc}})$. A complete lane change requires at least k_{lc} consecutive lane-change decisions in the same direction. In this work, k_{lc} is set at 6. Combing $\ddot{x}_{\mathrm{e}}$, y_{lc} forms the driving candidate action set. Additionally, the action space also contains the baseline policy action a_t^{b}.

The time step chooses $\Delta t = 0.75$, according to the driver reaction time (about 0.7s-1.5s) and a trade-off between smooth driving and faster calculation. Other values of k_{lc} and Δt can also be selected. Since the time step between actions is quite long, the vehicle should follow a smoothen trajectory, to connect the current state x_0, y_0, $\dot{x}_0$, $\dot{y}_0$ to the desired terminal state x_1, y_1, $\dot{x}_1$, $\dot{y}_1$:

$$\begin{aligned} \vec{p}(t) = &(2t^3 - 3t^2 + 1)\vec{p}_0 + (t^3 - 2t^2 + t)\vec{m}_0 \\ &+ (-2t^3 + 3t^2)\vec{p}_1 + (t^3 - t^2)\vec{m}_1, \quad t \in [0, 1] \end{aligned} \tag{23}$$

where $\vec{p}(t)$ denotes the points on the curve. $\vec{p}_0 = [x_0, y_0]^T$, $\vec{p}_1 = [x_1, y_1]^T$ denote the current position and desired position, respectively. $\vec{m}_0 = \frac{[\dot{x}_0, \dot{y}_0]^T}{|||[\dot{x}_0, \dot{y}_0]|||}$, $\vec{m}_1 = \frac{[\dot{x}_1, \dot{y}_1]^T}{|||[\dot{x}_1, \dot{y}_1]|||}$ denote the ego vehicle starting tangent and desired ending

TABLE I
RL PARAMETERS SETTING

Parameters	Symbol	Value
Exploration Fraction	ϵ	0.1
Network Update Frequency	f_{update}	500
Horizon	H	10 seconds
Trajectory buffer length	ω	20
Training batch size	--	32
Discount factor	γ	0.98
Virtual front vehicle velocity	$\dot{x}_{\text{f}}^{\text{v}}, \dot{y}_{\text{f}}^{\text{v}}$	(50, 0) m/s
Virtual front vehicle position	$x_{\text{f}}^{\text{v}}, y_{\text{f}}^{\text{v}}$	$(x_{\text{e}} + 50, l_{\text{id}})$
Virtual rear vehicle velocity	$\dot{x}_{\text{r}}^{\text{v}}, \dot{y}_{\text{r}}^{\text{v}}$	(0, 0) m/s
Virtual rear vehicle position	$x_{\text{r}}^{\text{v}}, y_{\text{r}}^{\text{v}}$	$(x_{\text{e}} - 50, l_{\text{id}})$

TABLE II
BASELINE POLICY PARAMETERS

Parameters	Symbol	Value
Desired velocity	$\dot{x}_0$	50 km/h
Exponent for velocity	δ	4
Desired time gap	T	1.5 s
Jam distance	g_0	2.0 m
Max acceleration	a	1.4 m/s^2
Desired deceleration	b	2.0 m/s^2
Politeness	p	0.5
Lane change threshold	Δa_{th}	0.1

tangent, respectively. This is the Cubic Hermite spline. Finally, the trajectory should be converted into the world coordinate systems according to Eq. (20).

Furthermore, the action space should be constrained by the ability of the control module to ensure the AV can track the planned trajectories. It can make the evaluation results directly reflect the planner's trajectory planning ability in a stochastic environment. This CARL system adds this controller ability constraints by pruning the illegal actions from the action space, according to the features of planned trajectories. Namely, before calculating the action at each time step, the system will first check all candidate trajectories, and then prune the trajectory that not meeting the controller's requirement, e.g., excessive curvature, from action space. The preview control algorithm [26] can track the trajectory with the lateral acceleration of less than 0.5g in a wide speed range, e.g., from 0 to 50 km/h. In most time of our simulation, it can track the desired trajectories. A barrier control has been designed to guarantee bounded tracking errors [27]. For a fair comparison, this controller ability constraint will be added equally to all the planners during the evaluation.

3) Reward Function: The reward function is defined according to Eq. (4). The terminal set $\mathcal{T}$ contains only the states where the ego vehicle crashes, detected by Carla.

C. Baseline Principle-Based Policy

The baseline principle-based policy uses the intelligent driver model (IDM) [28] for the longitudinal planning and the minimizing overall braking induced by lane change (MOBIL) [29] model for lateral planning. Both are widely used driving policies, but other driving models can also be used. All the states and actions used in the baseline policies are defined in LCS. This setting makes it easy to use the policy in other lane-based scenarios. Eq. (24) shows the IDM models.

$$\ddot{x} = a[1 - (\frac{\dot{x}}{\dot{x}_0})^{\delta} - (\frac{g_0 + T\dot{x} + \frac{\dot{x}\Delta\dot{x}}{2\sqrt{ab}}}{g})^2] \tag{24}$$

where x is the ego vehicle position; $\dot{x}$ and $\ddot{x}$ are vehicles' velocity and acceleration, respectively; $\ddot{x}_0$ is the desired free flow velocity; g is the actual gap between the ego vehicle and its front vehicle and g_0, T, a and b are the parameters determining the desired gap for the vehicle, shown in Table II.

Eq. (25) shows the MOBIL model, which plans the lateral motion. When the following condition is satisfied, the lane change motivation is generated:

$$\bar{\ddot{x}}_{\text{e}} - \ddot{x}_{\text{e}} + p(\bar{\ddot{x}}_{\text{n}} - \ddot{x}_{\text{n}} + \bar{\ddot{x}}_{\text{o}} - \ddot{x}_{\text{o}}) > \Delta a_{\text{th}} \tag{25}$$

where $\ddot{x}$ and $\bar{\ddot{x}}$ are the vehicle state of the current time and the state if the lane changing is made. Subscript e presents the state of ego vehicle. n, o presents the new follower and the old follower, respectively. And p and Δa_{th} are model parameters, which are listed in Table II.

All the parameters are set slightly aggressive, e.g., higher desired velocity than the surrounding traffic. An aggressive baseline policy will increase the driving mobility, in the meantime, the RL can find more opportunities to improve this policy.

D. Data Collection and RL Training

The data collection and RL training follow Alg. 1, which took about 41 hours. During this process, the CARL framework will find opportunities to explore and learn for better performance.

To demonstrate the baseline policy data distribution, we marginalize the 20-dimensional data into 2D, i.e., the ego vehicle velocity and the distance to the roundabout center, indicating the driving lane. The distribution is shown in Fig. (6).

Fig. 7 shows the return value G of all the baseline policy data, where more than 90% of the cases are higher than -0.2 which means the baseline policy drives most of the time safely. However, collisions may occur, e.g., when a vehicle cuts in and brake suddenly. Then the RL policy will explore other actions. In this work, the exploration module activates more than 3,500 times during training.

We also analyzed the 20-dimensional data, collected by the RL policy, into 2D data, shown in Fig. 8. The overall driving policy is generated by Eq. (16), based on the collected data and the confidence index design. Figs. 6 and 8 indicate both baseline policy and RL exploration cannot cover all state space during training. The data-driven RL policy may fail in not-well-trained cases.

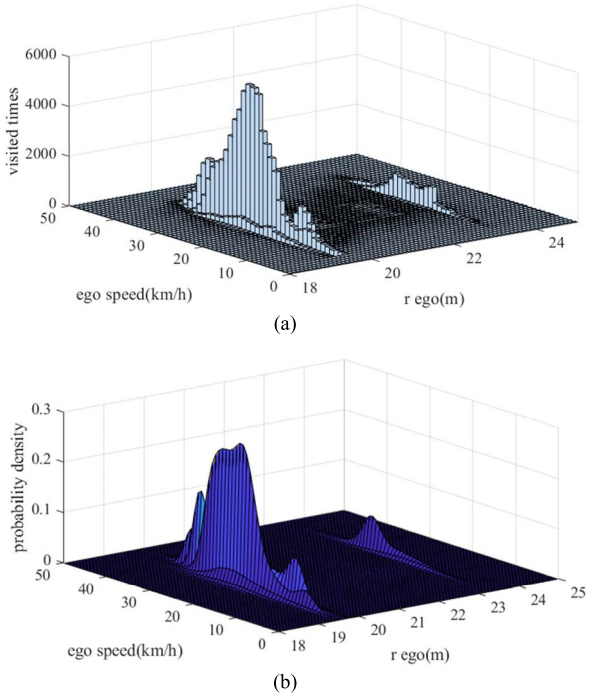


Fig. 6. Collected Date Distribution using the Baseline Policy during CARL training. (a) histogram of the data; (b) probability density function of the data, fit by Gaussian mixture model. The x and y coordinates represent two of twenty-dimension of the state space, i.e., driving lanes and ego vehicle velocity.

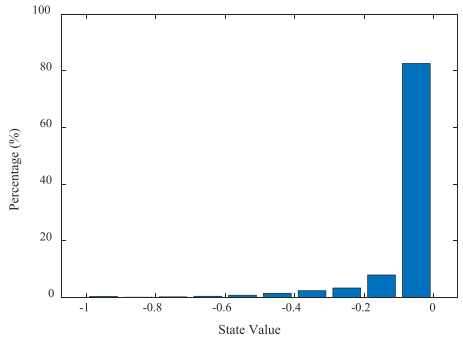


Fig. 7. Reward Return in the Baseline Policy Dataset.

E. Simulation Results

1) CARL Performance With Different Confidence Thresholds: We first design the simulation to test the generated CARL performance with different confidence threshold settings. This threshold determines the RL policy activation conditions. The confidence thresholds c_{thres} are set to 6 different levels, i.e., 0, 0.25, 0.5, 0.75, 0.95, shown in Table III.

The ego vehicle drives in the simulation for about 60 hours and 1,200 km. The simulation results are shown in Table IV, where the safety metric is the average distance per collision, defined in Eq. (17).

In Table IV, as the confidence threshold increases, the active rate of the RL policy decreases from 4.02% to 1.41%, besides the RL policy ($c_{\text{thres}} = 0$) and the baseline policy ($c_{\text{thres}} = 1$). A higher confidence threshold means more stringent requirements in reliability for the RL policy. Fig. 9 shows that the collision rate converges after 100 km testing. The statistical safety metric in Fig. 10 shows the performance of these policies.

The safety performance of the pure rule-based baseline policy is 0.46 km/col., and the collisions usually were caused by the fact that the baseline policy does not respond well to

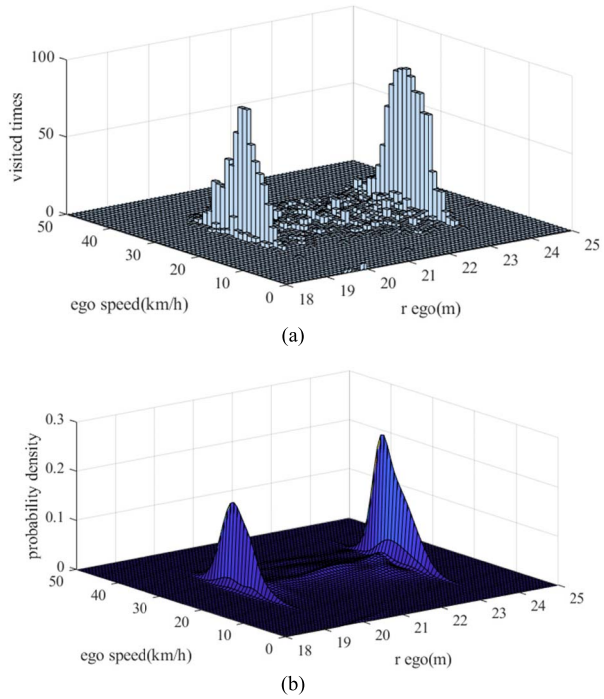


Fig. 8. Collected Date Distribution during RL Policy Exploration Process. (a) histogram of the data; (b) probability density function of the data, fit by Gaussian mixture model. The x and y coordinates represent two of twenty-dimension of the state space, i.e., driving lanes and ego vehicle velocity.

TABLE III
SIMULATION SETTING FOR CARL POLICY TESTING

No.	c_{thres}	Testing Time	Policy
1	0	~10h	RL Policy
2	0.25	~10h	0.25 CARL Policy
3	0.5	~10h	0.50 CARL Policy
4	0.75	~10h	0.75 CARL Policy
5	0.95	~10h	0.95 CARL Policy
6	1.0	~10h	Baseline Policy

TABLE IV
SIMULATION RESULTS IN AGGRESSIVE ENVIRONMENT

c_{thres}	Time (h)	Distance (km)	Ave. Speed (km/h)	RL active rate	Safety (km/col.)
0.00	10.55	149.98	14.21	100 %	0.25
0.25	10.38	230.54	22.21	4.02 %	1.03
0.50	9.75	217.28	22.29	3.86 %	1.10
0.75	8.20	183.10	22.32	1.90 %	0.93
0.95	11.35	255.83	22.53	1.41 %	0.74
1.00	10.28	238.67	23.22	0.00 %	0.46

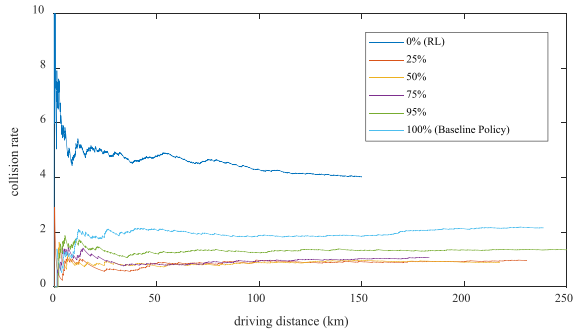


Fig. 9. Convergence of Crash Rate at Different Confidence Thresholds.

the vehicles that were exiting or merging-in. The collision rate of this baseline policy is higher than the common policies, but gives CARL planner more opportunities to assess the weakness cases, further accelerating the validation process. The pure RL policy shows worse performance, i.e., 0.25 km/col., because most cases, that the baseline policy works well, are not involved in the RL training. It confirms that the pure RL policy in the CARL is not well trained.

Using the confidence aware RL, the safety metric is significantly improved from (0.25, 0.46) to (0.74, 0.93, 1.03, 1.10) with different confidence threshold settings. We think this is due to two reasons. Firstly, the CARL method can assess the challenging cases of the baseline policies, and only activates RL policy in these cases. Secondly, the activated RL policy must be well-trained, i.e., has high confidence. In this way, the proposed method well considered the uncertainty of the RL training and utilized its self-learning ability. According to Fig. 10, the CARL planner does not always activate the RL policy, but the performance could still be better than the baseline policy and the pure RL policy.

If this conclusion holds, we can further use a well-tuned principle-based policy in the real application, and train the RL policy with a large amount of data for safer CARL performance. The final performance has potential to outperform the best human-engineering policy and avoid the unexpected behaviors of RL policy.

Furthermore, the confidence threshold setting shows the requirement of the policy improvement probability. Consistent with our analysis, when $c_{\text{thres}} = 0.5$, we obtain the safest results. When $c_{\text{thres}} = 0.95$, the simulation results confirm that the confidence aware RL algorithm can enhance the baseline policy when the confidence of RL is high.

2) CARL Driving Example Cases: During the testing with confidence aware RL policy, we find an example case where the RL policy is improving the baseline policy, shown in Fig. 11 (a).

In this scenario, the baseline policy is driving forward in the lane. At the same time, the inside vehicle is trying to exit the roundabout. In Carla simulation, the surrounding vehicle may exit the roundabout aggressively and the ego vehicle does not have enough distance to respond to this vehicle. It may cause side collisions.

The confidence aware RL policy can find that the baseline policy may fail in this scenario, and slow down the ego AV in advance, shown in Fig. 11 (b). Therefore, the proposed method

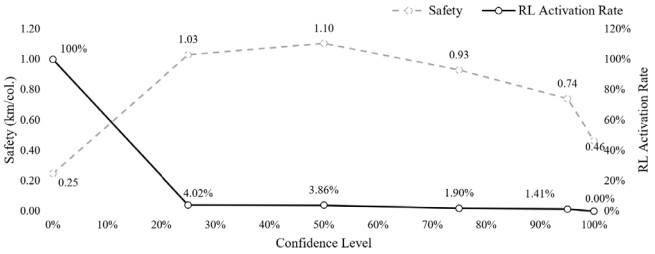


Fig. 10. Simulation Results in Aggressive Environment.

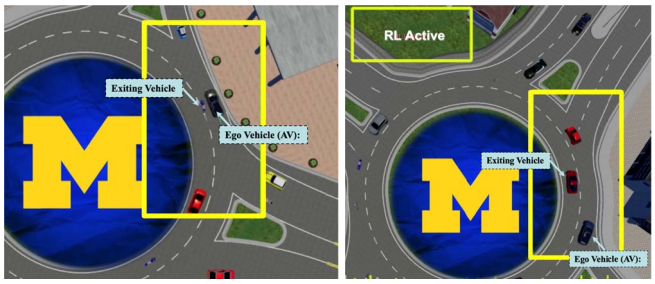


Fig. 11. An example scenario where the confidence aware RL can improve the baseline policy. (a) Baseline policy is hard to respond to the aggressively exiting vehicle. (b) Confidence aware RL slows down the ego AV in advance.

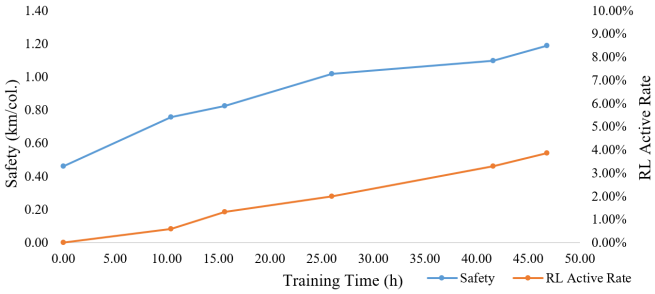


Fig. 12. Testing results using intermediate training models.

can learn from past failures and improve the baseline policy to fit the surrounding environment.

3) CARL Performance Improvement With More Training Data: Another significant benefit of the confidence aware RL is gradually improving the algorithm performance with a reliable performance lower bound. To evaluate this conclusion, we recorded some intermediate trained policies and corresponding training data during the training process. Ego vehicle uses each intermediate trained policy driving in the above-mentioned roundabout environment for about 1 hour. The results are shown in Fig. 12.

In the simulation results, the performance of all the intermediate CARL policies is better than the baseline policy (i.e., training time is 0). It indicates that the CARL can improve the baseline policy, even though the RL policy does not collect enough data, and has not been perfectly trained.

Furthermore, with more collected data and training, the safety performance of the ego AV increases from 0.5 to 1.2 km/col., and RL model has more opportunities to be activated, i.e., the active rate increases from 0% to 4%. This means that the proposed framework can gradually improve the

algorithm performance while always be better than a principle-based self-driving system. This property is extremely valuable when initially applying the RL policies on the real vehicle.

VI. Conclusion

In this paper, we proposed a confidence aware RL framework. This CARL framework contains a principle-based policy as well as an RL policy. An AV always uses the better driving policy, according to both policies' value function. Furthermore, the CARL method can analyze where RL policy learns and avoid activating the RL policy with few training data. Such a setting can avoid the unexpected behavior of RL policy due to inadequate training, further make it reliable to use RL policy. A confidence threshold is also designed to activate the RL policy optimally, and the optimal confidence threshold setting is at 0.5.

The framework is tested in a two-lane roundabout using the open-source simulator Carla. After about 60 hours and 1,200 km of simulated driving, the confidence aware RL achieves better performance than the principle-based policy and the pure RL policy. With more training data, CARL will activate RL policy in more time as well as improve the driving performance.

In general, the confidence-aware RL can analyze the confidence of the learned policy performance according to the training data, and avoid the region where RL policy is ill-prepared. The principle-based baseline policy provides a lower bound of performance and is the default to use when the RL policy has low confidence.

Acknowledgment

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] W. Wang, W. Zhang, and D. Zhao, "Understanding V2 V driving scenarios through traffic primitives," 2018, *arXiv:1807.10422*. [Online]. Available: http://arxiv.org/abs/1807.10422

[2] Z. Cao, D. Yang, K. Jiang, T. Wang, X. Jiao, and Z. Xiao, "End-to end adaptive cruise control based on timing network," in *Society of Automotive Engineers (SAE)-China Congress* (Lecture Notes in Electrical Engineering), vol. 486. Springer, 2017, pp. 839–852.

[3] Y. Lin, J. McPhee, and N. L. Azad, "Anti-jerk on-ramp merging using deep reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 7–14.

[4] Z. Cao *et al.*, "Highway exiting planner for automated vehicles using reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 990–1000, Feb. 2021.

[5] W. Zhou, K. Jiang, Z. Cao, N. Deng, and D. Yang, "Integrating deep reinforcement learning with optimal trajectory planner for automated driving," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–8.

[6] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016.

[7] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2018, pp. 287–296.

[8] H. Fan *et al.*, "Baidu apollo EM motion planner," 2018, *arXiv:1807.08048*. [Online]. Available: http://arxiv.org/abs/1807.08048

[9] M. Herger, (2019). *UPDATE: Disengagement Reports 2018-Final Results*. Dostopno Na, Ogled, vol. 15, no. 8. [Online]. Available: https://thelastdriverlicenseholder.com/2019/02/13/update-disengagement-reports-2018-final-results/

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[11] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[12] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–14.

[13] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–35, Jun. 2017.

[14] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robot. Auto. Syst.*, vol. 114, pp. 1–18, Apr. 2019.

[15] S. Nageshrao, E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," 2019, *arXiv:1904.00035*. [Online]. Available: http://arxiv.org/abs/1904.00035

[16] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 1, pp. 194–206, Jan. 2018.

[17] T. D. Simão and M. T. J. Spaan, "Safe policy improvement with baseline bootstrapping in factored environments," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 4967–4974.

[18] P. Thomas, G. Theocharous, and M. Ghavamzadeh, "High confidence policy improvement," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2380–2388.

[19] T. D. Simão, R. Laroche, and R. T. des Combes, "Safe policy improvement with an estimated baseline policy," 2019, *arXiv:1909.05236*. [Online]. Available: http://arxiv.org/abs/1909.05236

[20] Y. Chandak, S. Jordan, G. Theocharous, M. White, and P. S. Thomas, "Towards safe policy improvement for non-stationary MDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9156–9168.

[21] P. Ozisik and P. S. Thomas, "Security analysis of safe and seldonian reinforcement learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–12.

[22] M. J. Kochenderfer, *Decision Making Under Uncertainty*. Cambridge, MA, USA: MIT Press, 2015.

[23] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 3004–3010.

[24] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *Proc. Int. Conf. Manage. Data*, vol. 19, no. 2, 1990, pp. 322–331.

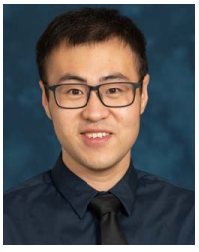
[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[26] S. Xu and H. Peng, "Design, analysis, and experiments of preview path tracking control for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 48–58, Jan. 2020.

[27] S. Xu, H. Peng, P. Lu, M. Zhu, and Y. Tang, "Design and experiments of safeguard protected preview lane keeping control for autonomous vehicles," *IEEE Access*, vol. 8, pp. 29944–29953, 2020, doi: 10.1109/ACCESS.2020.2972329.

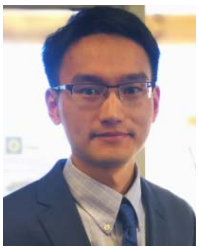
[28] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, p. 1805, 2000.

[29] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1999, no. 1, pp. 86–94, Jan. 2007.



Zhong Cao received the B.S. degree in automotive engineering from Tsinghua University in 2015. He is currently pursuing the Ph.D. degree in automotive engineering with Tsinghua University.

He researches as a Joint Ph.D. Student in mechanical engineering with the University of Michigan, Ann Arbor. Since 2016, he has been a Graduate Researcher with International Science and Technology Cooperation Program of China. His research interests include connected autonomous vehicle, driving environment modeling, and driving cognition.



Shaobing Xu received the Ph.D. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2016.

He is currently a Post-Doctoral Researcher with the Department of Mechanical Engineering and Mcity at the University of Michigan, Ann Arbor. His research interests include vehicle motion control, and decision making and path planning for autonomous vehicles. He was a recipient of an outstanding Ph.D. Dissertation Award of Tsinghua University, the First Prize of the Chinese 4th Mechanical Design Contest, the First Prize of the 19th Advanced Mathematical Contest, and the Best Paper Award of AVEC'2018.

Huei Peng received the Ph.D. degree in mechanical engineering from the University of California, Berkeley, in 1992.

He is currently a Professor with the Department of Mechanical Engineering, University of Michigan, and the Director of Mcity. He is also a Changjiang Scholar at the Tsinghua University, China. His research interests include adaptive control and optimal control, with emphasis on their applications to vehicular and transportation systems. His current research interests include design and control of electrified vehicles, and connected/automated vehicles. He is an SAE Fellow and an ASME Fellow.

Diange Yang received the B.S. and Ph.D. degrees in automotive engineering from Tsinghua University, Beijing, China, in 1996 and 2001, respectively.

He currently serves as the Director of automotive engineering with Tsinghua University. He is also a Professor with the Department of Automotive Engineering, Tsinghua University. He attended the "Ten Thousand Talent Program" in 2016. His research interests include intelligent transport systems, vehicle electronics, and vehicle noise measurement.

Dr. Yang received the Second Prize from the National Technology Invention Rewards of China in 2010 and the Award for Distinguished Young Science and Technology Talent of the China Automobile Industry in 2011.

Robert Zidek received the Ph.D. degree in aerospace engineering from the University of Michigan in 2017. He currently develops autonomous driving software, as a Research Scientist and the Manager at Toyota Research Institute. His team focuses on building decision making capabilities for L4 and L5 autonomous driving systems, including higher-level behavior planning and navigation, generating trajectory planning problems, and final trajectory selection.