Safety21
INNOVATING FOR SAFETY

US DOT National
University Transportation Center for Safety

Carnegie Mellon University

Penn
University of Pennsylvania

The Ohio State University

Morgan State University

CCAC Community College of Allegheny County

The University of Texas Rio Grande Valley

Structured Exploration: Bootstrapping Reinforcement Learning with Sub-optimal Policies for Autonomous Driving in Complex Traffic Scenarios

Keith Redmill (PI) (https://orcid.org/0000-0003-1332-1332)
Zhihao Zhang (https://orcid.org/0009-0009-7932-2128)
Ekim Yurtsever (https://orcid.org/0000-0002-3103-6052)

FINAL RESEARCH REPORT - August 31, 2026

Contract # 69A3552344811/69A3552348316

Technical Report Documentation Page

1. Report No.	2. Government Accession No	3. Recipient's Catalog No.
Pending assignment.	n/a	n/a

4. Title and Subtitle	5. Report Date
Structured Exploration: Bootstrapping Reinforcement Learning with Suboptimal Policies for Autonomous Driving in Complex Traffic Scenarios	August 31, 2026
	6. Performing Organization Code n/a

7. Author(s)		8. Performing Organization Report No.
Keith A. Redmill, Ph.D.	https://orcid.org/0000-0003-1332-1332	n/a
Zhihao Zhang	https://orcid.org/0009-0009-7932-2128	
Ekim Yurtsever, Ph.D.	https://orcid.org/0000-0002-3103-6052	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
The Ohio State University Center for Automotive Research 930 Kinnear Road Columbus, OH 43212	n/a
	11. Contract or Grant No. 69A3552344811

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Safety21 University Transportation Center Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213	Final Report: July 1, 2025 to July 31, 2026
	14. Sponsoring Agency Code USDOT

15. Supplementary Notes

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. The U.S. Government assumes no liability for the contents or use thereof.

16. Abstract

Automated vehicle control using reinforcement learning (RL) has attracted significant attention due to its potential to learn driving policies through environment interaction. However, RL agents often face training challenges in sample efficiency and effective exploration, especially in complex driving scenarios with delayed rewards, making it difficult to discover an optimal driving strategy that can escape myopic behaviors. To address these issues, we propose guiding the training of the RL driving agent with a demonstration policy that need not be a highly optimized or expert-level controller. Specifically, we integrate a rule-based lane-change controller, which embeds human-like driving heuristics, with the Soft Actor-Critic (SAC) algorithm to enhance exploration and learning efficiency. The suboptimal controller is employed both as a soft constraint during the early stages of policy training and as an additional source of training samples when populating the replay buffer. We demonstrate this approach in a multi-lane highway trap escape overtaking scenario with challenging traffic patterns that create strong exploration barriers, presenting results that consistently outperform a number of standard RL algorithms.

17. Key Words	18. Distribution Statement
Reinforcement Learning, Automated Driving System, Intelligent Vehicle Control, Complex Traffic Scenarios, Decision Making, Optimization	No restrictions.

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No Pages	22. Price
Unclassified	Unclassified	43 pages	n/a

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Abstract

Automated vehicle control using reinforcement learning (RL) has attracted significant attention due to its potential to learn driving policies through environment interaction. However, RL agents often face training challenges in sample efficiency and effective exploration, especially in complex driving scenarios with delayed rewards, making it difficult to discover an optimal driving strategy that can escape myopic behaviors. To address these issues, we propose guiding the training of the RL driving agent with a demonstration policy that need not be a highly optimized or expert-level controller. Specifically, we integrate a rule-based lane-change controller, which embeds human-like driving heuristics, with the Soft Actor-Critic (SAC) algorithm to enhance exploration and learning efficiency. The suboptimal controller is employed both as a soft constraint during the early stages of policy training and as an additional source of training samples when populating the replay buffer. We demonstrate this approach in a multi-lane highway trap escape overtaking scenario with challenging traffic patterns that create strong exploration barriers, presenting results that consistently outperform a number of standard RL algorithms.

Keywords: Reinforcement Learning, Automated Driving System, Vehicle Control, Complex Traffic Scenarios, Optimization

The results presented in this report have been accepted for publication in the *IEEE Open Journal of Intelligent Transportation Systems* [1], a peer-reviewed archival technical journal.

The source code for the work presented in this report and in [1] is available from:
https://github.com/Zhang1998-06/suboptimal.

Contents

List of Figures **2**

List of Tables **3**

1 Introduction **4**
1.1 Related Work . . . 6

2 Problem Formulation **8**
2.1 Critical Driving Scenarios . . . 8
2.2 State . . . 9
2.3 Action . . . 9
2.4 Reward . . . 10
2.5 Traffic Vehicle Control . . . 12
2.6 Reinforcement Learning Foundations . . . 13
2.6.1 Online Reinforcement Learning . . . 13
2.6.2 Offline Reinforcement Learning . . . 14
2.7 Behavior Cloning Methods Formulation . . . 16

3 Proposed RL Bootstrapping Framework **18**
3.1 Rule-based Heuristic Suboptimal Controller . . . 18
3.2 Low-level Motion Planner . . . 19
3.3 Bootstrapping RL Methodology . . . 20
3.3.1 Soft Constraint through KL Divergence . . . 20
3.3.2 Integration of Suboptimal Demonstrations . . . 21

4 Experimental Evaluation **23**
4.1 Rule-Based Suboptimal Demonstrator Policy . . . 24
4.2 Imitation Learning Baselines . . . 24
4.3 Offline Reinforcement Learning Baselines . . . 25
4.4 Demonstration-Augmented Reinforcement Learning . . . 27
4.5 Component-Wise Ablation . . . 29
4.6 Effect of Entropy Regularization and Exploration Scheduling . . . 32
4.7 Generalization to Unseen Scenarios . . . 33
4.8 Ablation on Hyperparameters . . . 34
4.9 Continuous Action-Space Extension . . . 36

5 Conclusion **38**

Bibliography **39**

List of Figures

1.1 A pipeline of vehicle control process. 4
1.2 RL agent interacts with the environment and learn from the environment. 5
1.3 Bootstrapping framework using a soft constraint and demonstration replay. 7

2.1 A highway "Trap" situation is represented by two vehicles moving slower than the typical traffic, forcing the ego vehicle to reduce speed to avoid a collision. The agent's objective is to maneuver out of this "Trap" formed by slow-moving traffic and subsequently accelerate to a desired speed. D_1 and D_2 indicated the initialization distances of the trap vehicles relative to the ego vehicle. 8
2.2 Action space for Single-level DRL controller. 10
2.3 Speed reward function. 10
2.4 Lane Centering Reward: This reward mechanism is designed to ensure that the vehicle maintains proximity to the lane's center while cruising. 11

3.1 Speed and steering motion planner 20

4.1 Training comparison with offline RL baselines. 26
4.2 Training comparison with demonstration-augmented RL baselines. 28
4.3 Comparison of marginal loss and reward augmentation. 30
4.4 Component-wise training ablation. 31
4.5 Comparison of exploration dynamics using (a) policy entropy; (b) effective action count during training. 32
4.6 Our method is trained under the trap initialization shown in Scenario 3. To evaluate generalization to unseen conditions, we test on Scenarios 1, 2, and 4 with different trap initializations. 33
4.7 Ablation study of hyperparameters. 35

List of Tables

2.1 Simulation settings 9
2.2 Experimental parameters 12
2.3 Traffic vehicle parameters 13
2.4 SAC hyper-parameters used in this study. 14
2.5 CQL hyper-parameters used in this study. 15

4.1 Testing comparison with rule-based methods, imitation learning, offline RL, and demonstration-augmented RL baselines (reported as mean ± stddev) 24
4.2 Baseline-specific hyperparameters 28
4.3 Testing comparison results for different methods of integrating suboptimal demonstrations 29
4.4 Simulation settings for four scenarios. D_i denotes the initial longitudinal distance between the ego vehicle and trap vehicle i. 33
4.5 Testing performance across four scenarios 34
4.6 Hyperparameter settings for the continuous action space and 5-Hz policy update experiments. 37
4.7 Testing performance under continuous action space and 5-Hz decision policy updates. 37

Chapter 1

Introduction

The emergence of autonomous vehicles has introduced an innovative trajectory in the narrative of highway driving. Autonomous highway driving, as studied in this project, pertains to the decision-making and vehicle control processes through which autonomous vehicles navigate and interact within highway environments. A typical high-level pipeline for the vehicle sensing and control architecture is shown in Figure 1.1. It represents the convergence of cutting edge technologies and involves the integration of advanced artificial intelligence and control algorithms, sensors and sensor fusion, and real-time data processing to enable vehicles to make informed choices, such as lane changes, merging, overtaking, and responding to intricate and dynamic traffic conditions [2,3], promising enhancements in traffic management, safety, and overall efficiency [4]. The goal is to ensure safe, efficient, and reliable autonomous travel on highways by enabling vehicles to analyze complex scenarios, anticipate potential hazards, and execute actions that align with traffic rules and user preferences [4–7]. Yet, despite remarkable strides, the task of crafting autonomous vehicles with the capability to make discerning decisions remains a formidable obstacle. This complexity emanates from the complicated amalgamation of diverse disciplines and the intricacies of real-world scenarios, necessitating profound innovation and interdisciplinary collaboration [4, 7–9].

In the rapidly evolving field of automated driving systems, three fundamental methodologies have emerged as essential strategies for addressing the complex challenge of decision-making [7, 9].The first approach, characterized as conventional, embraces a modular framework that governs the decision-making process within highway scenarios [3,10–15]. Within this paradigm, distinct modules, often rule-based or model-based, are meticulously crafted, and each is responsible for addressing specific aspects of decision-making. These modules encompass specific behaviors such as lane changes, merging, overtaking, and other maneuvers crucial for highway navigation. Ultimately, the amalgamation of these modules along with a behavior planning and selection algorithm culmi-

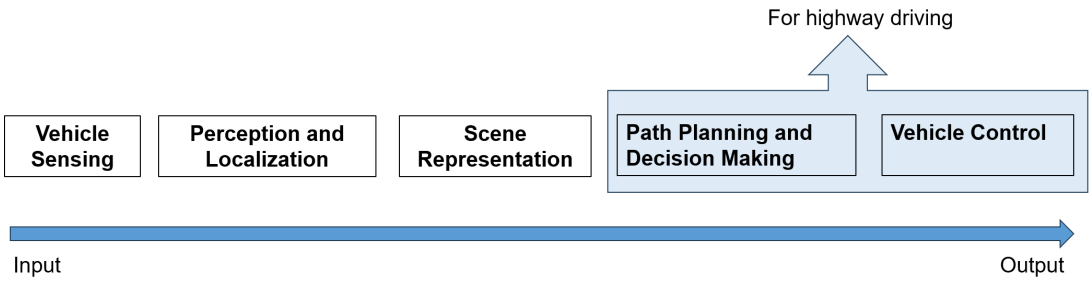


Figure 1.1: A pipeline of vehicle control process.

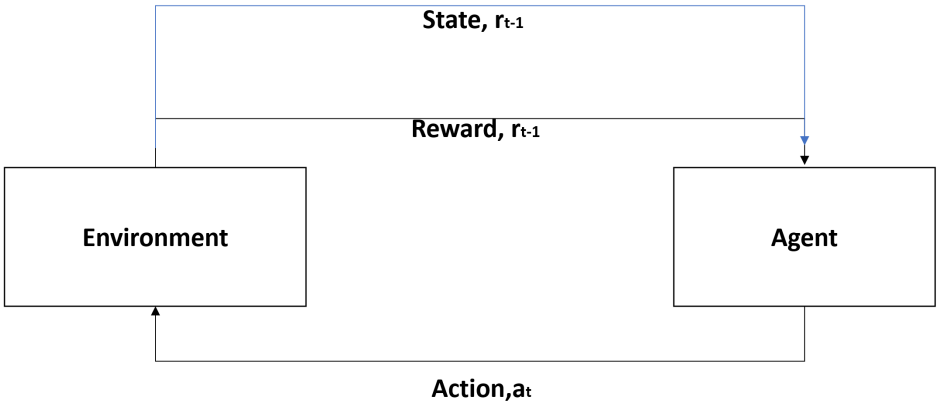


Figure 1.2: RL agent interacts with the environment and learn from the environment.

nates in a cohesive decision-making system, which then interfaces with vehicle control mechanisms. This approach, while promoting transparency and interpretability in decision processes, demands the meticulous integration of diverse driving factors. It necessitates a comprehensive understanding of adapting to an array of driving scenarios while accounting for diverse safety considerations.

Conversely, data-driven methodologies, exemplified by supervised learning, take a different path towards decision-making refinement [16–20]. This approach simplifies the process by using machine learning methods to replicate expert driving behavior. Developing such a controller entails concurrently gathering and correlating sensor data with the respective steering, brake, and throttle actuator actions performed by human drivers. However, a notable caveat of this method is its demand for extensive data collection.

Finally, reinforcement learning emerges as another compelling paradigm, aiming to equip autonomous systems with decision-making capabilities [2,21,22]. Through simulated interactions with the environment, the self-driving agents accumulate data and experience, facilitating the acquisition of driving strategies as illustrated in Figure 1.2. However, DRL controllers often rely on exploring the environment in the early stages of their training. Insufficient exploration might hinder the DRL controller from discovering better decision-making behaviors, which are essential for coping with more complex traffic environments [10]. Moreover, existing studies gravitate towards simpler traffic scenarios, restricting the generalizability of their findings when facing more intricate and unpredictable real-world conditions [23, 24].

One of the primary challenges in RL is achieving sufficient exploration of the environment while maintaining sample efficiency. Despite methods such as maximum entropy DRL algorithms that promote continuous exploration during training, agents still struggle to identify optimal policies in complex environments that require long-term planning and effective handling of delayed rewards [25, 26].

In our previous study, we found that traditional online RL driving agents can encounter dense traffic bottleneck situations where multiple steps of undesirable rewards discourage exploration and prevent the discovery of long-term beneficial solutions [27]. Consequently, this limited exploration can cause the agents to converge on suboptimal policies and overly conservative driving strategies such as cruising behind slow vehicles. To address this issue, we propose a novel approach that incorporates a demonstration controller to guide RL agents. This auxiliary demonstration controller need not be carefully designed to ensure optimal performance, but it can help RL agents to overcome exploration barriers and converge on an optimal driving policy by providing plausible and human-

like behavior that is less obvious and more difficult to discover. Thus, an effective suboptimal demonstration policy should encourage the learning of less obvious behaviors without being overly complex or comprehensive.

Our contributions can be summarized as follows:

1. We present a driving-oriented bootstrapping paradigm for RL-based driving agents. Our proposed approach systematically integrates imperfect demonstrations to break exploration barriers and steer the learned policy toward higher optimality.

2. We introduce a multi-lane highway trap escape overtaking scenario as a case study to stress-test RL driving agents under strong exploration barriers. This scenario also demonstrates how our bootstrapping paradigm can be used to accelerate learning in challenging driving settings.

1.1 Related Work

Prior research on RL-based autonomous driving primarily focused on a driving environment with sparse traffic and predictable traffic patterns to manage complexity [7, 23, 24, 28–30]. Within these simplified domains, RL agents effectively learn driving policies by repeated interactions with a simulated environment. Although these approaches have demonstrated success in basic tasks, such as lane-keeping and straightforward car-following scenarios, they often encounter significant challenges related to exploration efficiency and sample inefficiency in some critical driving scenarios. Agents frequently settle into suboptimal behaviors, particularly in complex and safety-critical scenarios involving delayed reward feedback and long-horizon decision-making.

To address these challenges and enhance sample efficiency, researchers have incorporated prior domain knowledge into RL frameworks for driving. This prior knowledge commonly originates from rule-based or model-based methods, as well as from human expert demonstrations.

Rule-based and model-based methods have been widely explored for providing structured guidance to RL training. One strategy uses rule-based or model-based methods as soft and hard constraints throughout the entire training process [21, 31]. Although these methods provide structured guidance and accelerate learning, they often lack flexibility in assessing dynamic traffic conditions and vehicle dynamics. Consequently, RL agents can become overly dependent on fixed rules or waypoint trajectories, severely limiting their ability to adapt to real-time uncertainties or rapidly changing driving environments. Such dependence can cause failures in scenarios that require immediate deviation from planned trajectories due to sudden obstacles or unexpected events. Alternatively, some integration methods embed RL within a rule-based or model-based driving framework, using RL to optimize the framework's tunable parameters [32–34] or adopt hierarchical designs in which RL is applied to high-level decision-making modules [35–37]. These hybrid strategies offer a balanced trade-off between interoperability and adaptability. However, their effectiveness still heavily depends on the accuracy and reliability of the underlying rule-based or model-based components. Overly restrictive rules might also hinder the discovery of an optimal driving strategy.

Human expert demonstrations represent another critical source of prior knowledge for online RL training. These demonstrations leverage the expertise of experienced human drivers, either through real-time human-in-the-loop interactions or through offline expert datasets [38]. In interactive RL scenarios, a human expert actively provides immediate corrective feedback during training, improving learning efficiency and minimizing undesirable exploratory behavior. However, this approach requires substantial human effort and continuous online supervision, making it labor-intensive and potentially difficult to scale. In contrast, offline learning methods, including imitation learning or

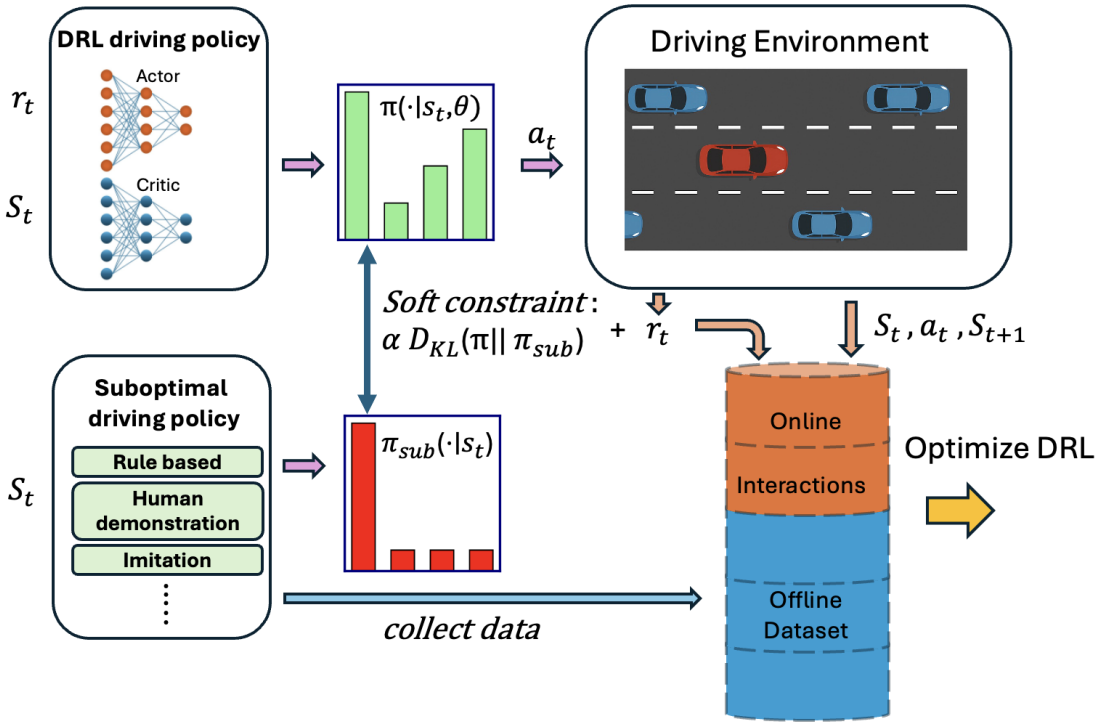


Figure 1.3: We leverage the suboptimal policy in two ways: (1) as a soft constraint on the RL policy during initial training stages, and (2) as a source for populating the replay buffer with additional training samples.

offline RL, enable agents to efficiently bootstrap policy training by mimicking expert behaviors extracted from high-quality driving datasets [16–20]. Although this approach is more scalable in principle, collecting comprehensive, high-quality datasets can be costly.

We propose a DRL-based autonomous driving framework specifically designed to address critical driving scenarios. Unlike previous approaches that rely heavily on expert demonstrations or optimal control methods, our framework utilizes a suboptimal controller as guidance [22, 38–40]. This suboptimal controller, while not optimal in performance, encapsulates human-like driving heuristics, effectively guiding the RL exploration toward practical driving behaviors. Compared to optimal control designs, developing a heuristic controller significantly reduces design complexity and costs while simultaneously enhancing sample efficiency aligned with human driving strategies. As shown in Figure 1.3, we integrate driving heuristics into the RL framework as a soft constraint and as an additional source of training samples. The incorporation of heuristic guidance balances structured insights from human expertise with the necessary flexibility, enabling the RL agent to efficiently explore and ultimately identify better driving policies.

Chapter 2

Problem Formulation

We formulate the highway driving problem as a Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma) \tag{2.1}$$

where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $\mathcal{P}$ represents the state transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. Our goal is to learn a policy $\pi(a \mid s)$ that maximizes the expected cumulative discounted reward

$$\max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \, r\big(s(t),\, a(t)\big)\Big], \quad a(t) \sim \pi(\cdot \mid s(t)). \tag{2.2}$$

2.1 Critical Driving Scenarios

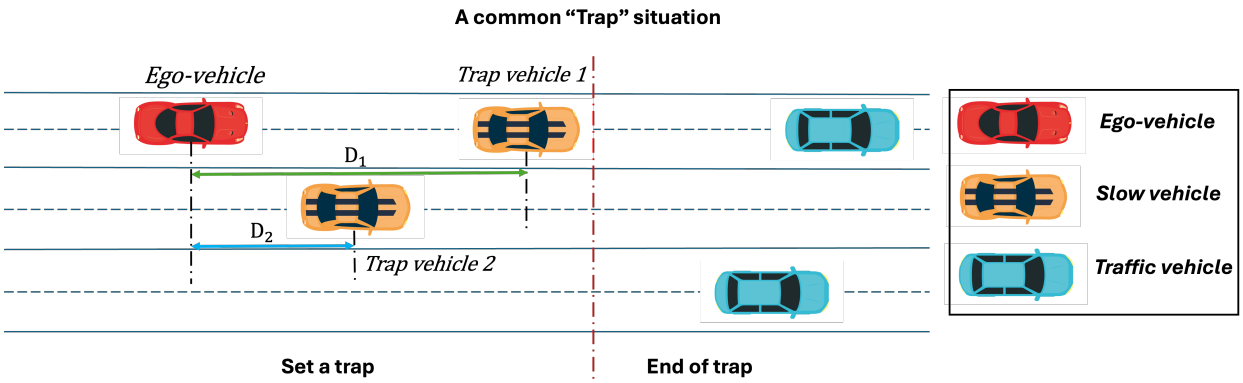


Figure 2.1: A highway "Trap" situation is represented by two vehicles moving slower than the typical traffic, forcing the ego vehicle to reduce speed to avoid a collision. The agent's objective is to maneuver out of this "Trap" formed by slow-moving traffic and subsequently accelerate to a desired speed. D_1 and D_2 indicated the initialization distances of the trap vehicles relative to the ego vehicle.

We utilize a critical driving scenario for which finding an effective driving strategy is notably challenging for standard DRL agents. As illustrated in Figure 2.1, upon entering the main road, the ego vehicle encounters two[1] slower vehicles ahead, maintaining relatively short headway distances

[1]See Section 4.7 for additional scenario variants with more surrounding vehicles.

Table 2.1: Simulation settings

Description	Value
Vehicle length	$5m$
Vehicle width	$2m$
Road width	$4m$
D_1: Initial longitudinal distance to trap veh 1	$[14.80, 16.44]m$
D_2: Initial longitudinal distance to trap veh 2	$[4.06, 7.43]m$

D_1 and D_2. These slower vehicles create a traffic "trap" by driving significantly slower than the surrounding traffic and below the ego vehicle's preferred cruising speed. To successfully overtake the trap vehicles, the ego vehicle must exploit the backward gap. This strategy is intuitive for human drivers but challenging for RL agents because the delayed rewards obtained after escaping the trap must outweigh the short-term rewards of remaining trapped. The higher benefit is not evident until the maneuver is completed, which creates an exploration barrier.

Our simulated traffic scenario consists of trap vehicles intentionally driving at lower speeds and regular traffic vehicles controlled by the Intelligent Driver Model (IDM) [14] and Minimizing Overall Braking Induced by Lane Changes (MOBIL) [15]. Additional dynamic traffic vehicles prevent overfitting to a fixed pattern and require the agent to combine overtaking with adaptive maneuvering. The simulation settings used for training and testing are identical and are listed in Table 2.1. Detailed traffic-vehicle configurations follow our previous work [27].

2.2 State

The observation contains 26 features describing the ego vehicle and four surrounding vehicles. Each state $s \in \mathcal{S}$ encodes both the ego vehicle's motion and local traffic context as defined by

$$\begin{aligned} s = \Big(& p^e,\ x^e,\ y^e,\ v^e_{\text{lon}},\ v^e_{\text{lat}},\ d_e, \\ & \{p^n,\ \Delta x^n,\ \Delta y^n,\ \Delta v^n_{\text{lon}},\ \Delta v^n_{\text{lat}}\}_{n=1}^{4} \Big) \end{aligned} \tag{2.3}$$

where p^e denotes the presence of the ego vehicle, x^e, y^e is the ego vehicle's longitudinal and lateral position, $v^e_{\text{lon}}, v^e_{\text{lat}}$ is the ego vehicle's longitudinal and lateral velocity, d^e is its lateral offset from the lane center, $p^n \in \{0, 1\}$ indicates the presence of the n-th traffic vehicle, and Δx^n, Δy^n, Δv^n_{lon}, and Δv^n_{lat} denote the relative position and velocity of the n-th surrounding vehicle with respect to the ego vehicle in both the longitudinal and lateral directions.

2.3 Action

We employ a discrete action space consisting of 9 potential actions. Each action A consists of an acceleration and steering pair (a, θ):

$$a \in \{-1, 0, 1\}\,\text{m/s}^2 \tag{2.4}$$

$$\theta \in \{-\pi/50, 0, \pi/50\}\,\text{rad}. \tag{2.5}$$

Thus, the automated driving agent operates with nine discrete actions as shown Figure 2.2.

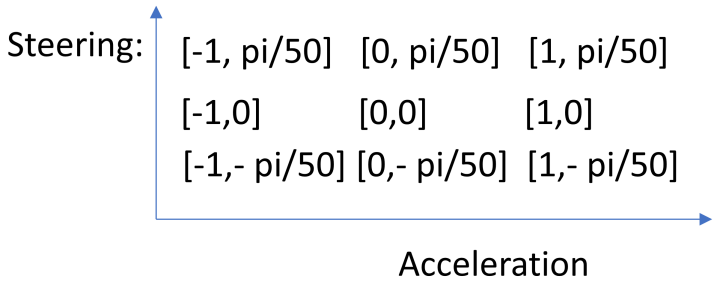


Figure 2.2: Action space for Single-level DRL controller.

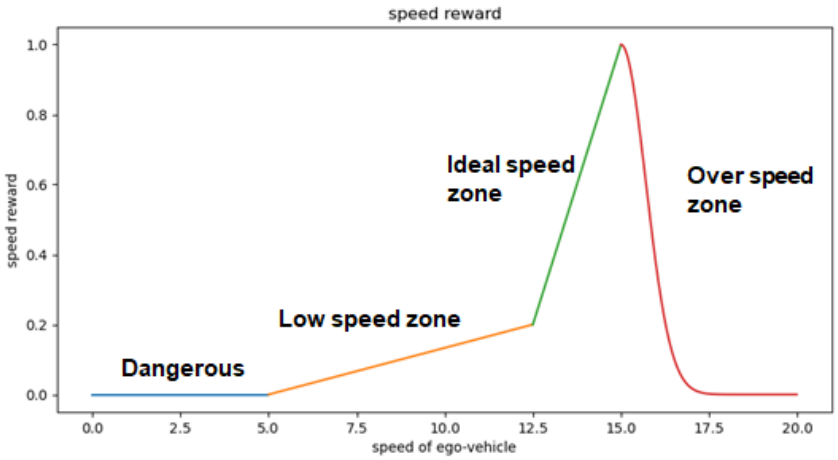


Figure 2.3: Speed Reward Function in a Highway Environment: The vehicle's speed is categorized into four distinct zones for effective management. These are: 1) Dangerous Low Speed Zone, for speeds significantly below the optimal range; 2) Low Speed Zone, for speeds moderately below the ideal; 3) Ideal Speed Zone, representing the optimal speed range for safety and efficiency; and 4) Over Speed Zone, for speeds exceeding the safe upper limit.

2.4 Reward

We define four reward components: speed, lane centering, steering, and a penalty for critical failures. A critical failure occurs when the ego vehicle stops dangerously on the highway, leaves the drivable road surface, or collides with another vehicle. An episode terminates upon a critical failure or when the maximum number of time steps is reached.

Speed reward

$$r_v = \begin{cases} e^{-(v-15)^2} & , v > 15 \\ \frac{8}{25}v - \frac{19}{5} & , 12.5 < v \leq 15 \\ \frac{2}{75}v - \frac{2}{15} & , 5 < v \leq 12.5 \\ 0 & , v \leq 5 \end{cases} \tag{2.6}$$

As shown in Figure 2.3, the speed reward is structured around four distinct zones: the Dangerous Speed Zone ($v \in [0, 5]\,\mathrm{m/s}$), the Low Speed Zone ($v \in (5, 12.5]\,\mathrm{m/s}$), the Ideal Speed Zone ($v \in (12.5, 15]\,\mathrm{m/s}$), and the Over Speed Zone ($v \in (15, \infty)\,\mathrm{m/s}$). The Dangerous Speed Zone is

characterized by speeds too low for highway safety, posing a risk of traffic accidents. The Low Speed Zone, while not hazardous, represents suboptimal vehicle speeds. Our target lies within the Ideal Speed Zone, with a specified speed of 15 m/s. If the ego vehicle's speed drops below this threshold, the controller encourages acceleration back to 15 m/s. Conversely, the Over Speed Zone denotes speeds exceeding the limit, which are discouraged.

Lane-centering reward

$$r_y = e^{-1.5\Delta d^2} \tag{2.7}$$

As shown in Figure 2.4, the lane-centering reward encourages the vehicle to remain near the lane center, which is important for safe and efficient driving, especially on curves [41]. Here, Δd is the lateral distance to the center of the current lane.

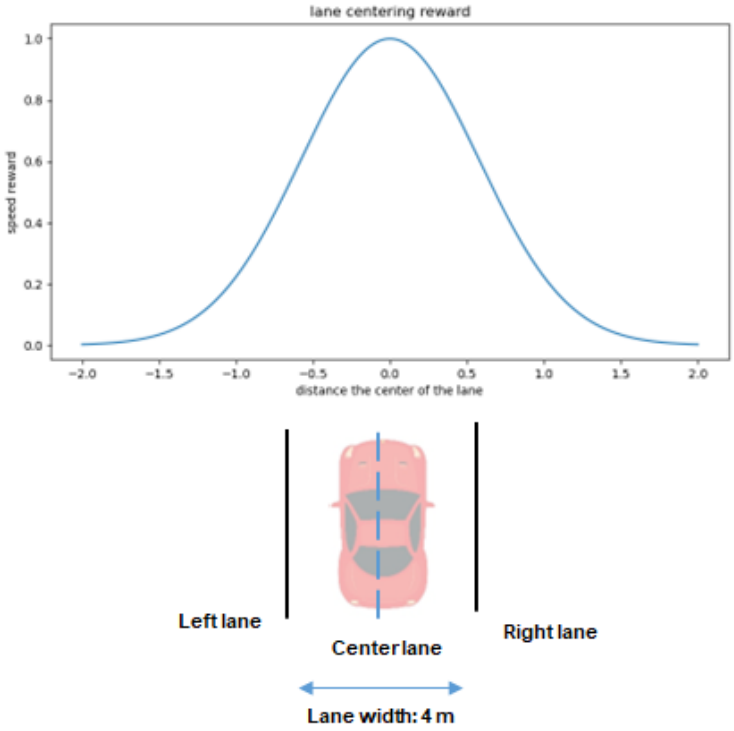


Figure 2.4: Lane Centering Reward: This reward mechanism is designed to ensure that the vehicle maintains proximity to the lane's center while cruising.

Steering reward

$$r_\theta = -|\sin(\theta)| \tag{2.8}$$

The steering reward aims to prevent excessive steering, promote smoother driving, and reduce the risk of overcorrection.

Penalty for accident

$$r_{\text{total}} = \begin{cases} \frac{w_v r_v + w_\theta r_\theta + w_y r_y}{w_v + w_\theta + w_y} & \text{if no critical failure} \\ -10 & \text{if a critical failure occurs} \end{cases} \tag{2.9}$$

Successful Completion

We define successful completion in terms of the ego vehicle's ability to overtake the leading slow-moving traffic, referred to as "Trap Vehicle 1" in Fig. 2.1. Specifically, a trial is considered successful if the ego vehicle overtakes "Trap Vehicle 1" and remains ahead of it for the remainder of the episode. If the ego vehicle has an accident or departs from the lane after overtaking "Trap Vehicle 1", the episode is still marked as a successful escape. However, such critical failures are accounted for separately under the collision rate metric in the safety evaluation.

Each of these reward terms is assigned a specific weight factor, which helps to balance their influence on the vehicle's learning process. Additionally, the model imposes a significant penalty of -10 for critical failures such as crashes, driving out of the lane, or stopping in the lane. Specific weight values can be found in Table 2.2.

Table 2.2: Experimental parameters

Parameters	value
speed reward weight w_v	1.5
steering reward weight w_θ	0.05
lane centering reward weight w_y	0.05
Training episodes	3000
Training episode duration	150 s

2.5 Traffic Vehicle Control

We employ the IDM [14] for longitudinal car-following dynamics and the MOBIL model [15] for lane-change decisions. IDM calculates the desired longitudinal acceleration by considering the current velocity of the vehicle, the distance to the lead vehicle, and the relative speed. Specifically, given a gap s, a speed v, and a speed difference Δv with respect to the vehicle ahead, the IDM acceleration a is defined as

$$a = a_{\max} \left[1 - \left(\frac{v}{v_{\text{desired}}} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right], \tag{2.10}$$

where $a_{\max}$ is the maximum acceleration, v_{desired} is the desired cruising speed, and δ is an exponent value. The desired gap s^* captures the minimum distance required to maintain safety and is computed as

$$s^*(v, \Delta v) = s_0 + T\, v + \frac{v\, \Delta v}{2\sqrt{a_{\max}\, b}}, \tag{2.11}$$

where s_0 is the minimum standstill gap, T is the desired time headway, and b is the comfortable deceleration. This formulation balances free-flow acceleration and collision avoidance within traffic streams.

For lateral movements, the MOBIL model triggers lane changes when the overall benefit, accounting for both the ego vehicle and its neighbors, exceeds a threshold. The incentive criterion is expressed as

$$\Delta a_{\text{ego}} + p\,(\Delta a_{\text{rear}} + \Delta a_{\text{front}}) \;>\; a_{\text{th}}, \tag{2.12}$$

where Δa_{ego} is the ego vehicle's acceleration gain, Δa_{rear} and Δa_{front} denote changes in acceleration of the affected trailing and leading vehicles in the target lane, p is a politeness factor, and a_{th} is the acceleration threshold. A safety criterion further ensures no vehicle is forced to brake beyond a comfortable limit. By combining IDM and MOBIL, we capture realistic traffic interactions: vehicles maintain sensible headways and make lane-change decisions that balance individual and collective benefits. Traffic vehicle parameters can be found in Table 2.3.

Table 2.3: Traffic vehicle parameters

Symbol	Description	Value
$a_{\max}$	Maximum acceleration	$0.5\,\mathrm{m/s^2}$
v_{desired}	Desired cruising speed	$12.5\,\mathrm{m/s}$
s_0	Minimum standstill gap	$10\,\mathrm{m}$
T	Desired time headway	$1.5\,\mathrm{s}$
b	Comfortable deceleration	$0.5\,\mathrm{m/s^2}$
a_{th}	Acceleration threshold	$0.2\,\mathrm{m/s^2}$
p	Politeness factor	0.5
δ	Exponent value	4

2.6 Reinforcement Learning Foundations

We use the MDP formulation introduced at the beginning of this chapter and summarize the online and offline RL methods used in the experimental comparisons.

We adopt two complementary deep-RL algorithms to learn highway-driving policies: Soft Actor–Critic (SAC) for online interaction and Conservative Q-Learning (CQL) for offline learning [25, 42]. Below we summarize their objectives and optimization mechanisms.

2.6.1 Online Reinforcement Learning

SAC optimizes a stochastic actor policy π_ϕ by maximizing the discounted return augmented with an entropy bonus:

$$J_{\text{SAC}}(\pi_\phi) = \mathbb{E}_{\pi_\phi}\Big[\textstyle\sum_{t=0}^{\infty} \gamma^t \big(r(s_t, a_t) + \alpha\, \mathcal{H}\big[\pi_\phi(\cdot \mid s_t)\big]\big)\Big], \tag{2.13}$$

where the temperature $\alpha > 0$ trades off task reward against the Shannon entropy $\mathcal{H}\big[\pi_\phi(\cdot \mid s_t)\big]$ of the actor [25]. Entropy regularization encourages broad exploration in early stages and guards against premature convergence to deterministic, sub-optimal strategies. Two action-value critic functions Q_{θ_1} and Q_{θ_2} are trained with the soft Bellman backup operator. Taking their minimum mitigates the positive bias caused by overestimating action values.

Rather than hand-tuning the temperature parameter α, SAC adjusts it online by solving the dual problem

$$\alpha^\star = \arg\min_{\alpha>0}\; \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}\big[\alpha\big(-\log \pi_\phi(a_t \mid s_t) - \bar{\mathcal{H}}\big)\big], \tag{2.14}$$

where $\mathcal{B}$ is the replay buffer and $\bar{\mathcal{H}}$ is the target entropy, set to -1 in this study. This adaptive mechanism is especially useful in discrete action spaces whose utilities can vary sharply across states, as is the case in highway navigation.

Because SAC is off-policy, each transition stored in the replay buffer can be reused for many gradient updates. The algorithm therefore achieves high sample efficiency, an essential property when interacting with a simulation environment that enables continuous data collection without real-world risk. The pseudocode and training hyperparameters are presented in Algorithm 1 and Table 2.4, respectively.

Table 2.4: SAC hyper-parameters used in this study.

Symbol	Meaning	Value
γ	Discount factor	0.8
η_π	Learning rate (actor network)	5×10^{-4}
η_Q	Learning rate (critic network)	1×10^{-3}
B	Batch size	64
$\lvert\mathcal{D}\rvert$	Replay buffer size	50,000
β	Initial offline ratio	0.6
$\bar{\mathcal{H}}$	Target entropy	-1
α_{init}	Initial entropy coefficient	0.01
η_α	Entropy coefficient learning rate	1×10^{-3}

Algorithm 1 Soft Actor–Critic for Discrete Action Spaces

Require: Environment $\mathcal{E}$, replay buffer $\mathcal{D}$
Require: Learning rates $\eta_Q, \eta_\pi, \eta_\alpha$, discount γ, target-mix τ
Initialise actor π_ϕ, critics $Q_{\theta_1}, Q_{\theta_2}$
Initialise target critics $\bar{\theta}_i \leftarrow \theta_i$ $(i = 1, 2)$
while training not converged **do**
  Observe state s and sample $a \sim \pi_\phi(\cdot \mid s)$
  Execute a in $\mathcal{E}$, receive (r, s', d)
  Store (s, a, r, s', d) in $\mathcal{D}$
  for all gradient steps **do**
    Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
    $V_{\bar{\theta}}(s') \leftarrow \sum_{a'} \pi_\phi(a'|s') \big[\min_i Q_{\bar{\theta}_i}(s', a') - \alpha \log \pi_\phi(a'|s')\big]$
    $y \leftarrow r + \gamma(1 - d)\, V_{\bar{\theta}}(s')$
    $\theta_i \leftarrow \theta_i - \eta_Q \nabla_{\theta_i} \frac{1}{2} \big(Q_{\theta_i}(s, a) - y\big)^2 \quad (i = 1, 2)$
    $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \sum_a \pi_\phi(a|s) \big[\alpha \log \pi_\phi(a|s) - \min_i Q_{\theta_i}(s, a)\big]$
    $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \sum_a \pi_\phi(a|s) \big[-\alpha(\log \pi_\phi(a|s) + \bar{\mathcal{H}})\big]$
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i \quad (i = 1, 2)$
  end for
end while

2.6.2 Offline Reinforcement Learning

Offline RL forbids further interaction with the environment and must learn solely from a fixed dataset $\mathcal{D} = \{(s, a, r, s', d)\}$ collected under unknown behavior policies, where d is the terminal indicator. Standard Q-learning tends to overestimate unseen state and action pairs, causing ex-

trapolation error and catastrophic policy degradation when deployed. CQL [42] combats this failure mode by penalizing Q-values that are large for actions not observed in $\mathcal{D}$, thereby inducing a pessimistic critic that stays within the support of the dataset.

Let Q_θ denote the parametric critic and let $\mathcal{L}_{\mathrm{TD}}$ be the usual temporal-difference loss with a target network $\bar{\theta}$. CQL augments this loss with a conservative regularizer that lowers any Q-value which the dataset does not support:

$$\mathcal{L}_{\mathrm{CQL}}(\theta) = \mathcal{L}_{\mathrm{TD}}(\theta) + \lambda\, \mathbb{E}_{s\sim\mathcal{D}} \left[\log \sum_{a\in\mathcal{A}} \exp Q_\theta(s,a) - \mathbb{E}_{a\sim\mathcal{D}(\cdot|s)}[Q_\theta(s,a)] \right], \tag{2.15}$$

where the log sum expectation term upper-bounds the maximum Q over all actions and the second term anchors the estimate to actions actually present in $\mathcal{D}$. The trade-off coefficient $\lambda > 0$ controls conservatism, with larger values imposing stronger pessimism on unsupported actions.

Once Q_θ is updated, a stochastic actor π_ϕ is improved by minimizing the Kullback–Leibler divergence between $\pi_\phi(\cdot \mid s)$ and the Boltzmann distribution induced by the conservative Q-values:

$$\min_{\phi}\ \mathbb{E}_{s\sim\mathcal{D}} \Big[\mathrm{KL}\big(\pi_\phi(\cdot \mid s) \,\|\, \mathrm{softmax}\big(Q_\theta(s,\cdot)/\alpha\big)\big) \Big], \tag{2.16}$$

where α is the SAC entropy temperature. The resulting actor is discouraged from selecting unsupported actions because its update is shaped by the conservative critic.

CQL is well suited to highway-navigation logs containing safety-critical maneuvers. Its conservative value estimates reduce optimistic extrapolation for actions that are poorly represented in the dataset, although they do not by themselves guarantee safe behavior. The pseudocode and training hyperparameters are presented in Algorithm 2 and Table 2.5, respectively.

Table 2.5: CQL hyper-parameters used in this study.

Symbol	Meaning	Value
γ	Discount factor	0.99
η_π	Actor learning rate	1×10^{-5}
η_Q	Critic learning rate	5×10^{-5}
η_α	Entropy-coefficient learning rate	5×10^{-5}
τ	Target-network soft update	0.005
$\bar{\mathcal{H}}$	Target entropy	-1
B	Batch size	64
$\lvert\mathcal{D}\rvert$	Fixed dataset size	200,000
T	Gradient-update steps	2000

Algorithm 2 Discrete CQL built on SAC

Require: Fixed dataset $\mathcal{D}$
Require: Actor π_ϕ, critics $Q_{\theta_1}, Q_{\theta_2}$
Require: Hyperparams: $\gamma, \tau, \eta_Q, \eta_\pi, \eta_\alpha$
Require: CQL weight λ, target entropy $\bar{\mathcal{H}}$
Initialise target critics $\bar{\theta}_i \leftarrow \theta_i$
while training not converged **do**
  for all gradient steps **do**
    Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
    $V_{\bar{\theta}}(s') \leftarrow \sum_{a'} \pi_\phi(a'|s') \big[\min_i Q_{\bar{\theta}_i}(s', a') - \alpha \log \pi_\phi(a'|s')\big]$
    $y \leftarrow r + \gamma(1-d)\, V_{\bar{\theta}}(s')$
    MSE $\leftarrow \frac{1}{2}(Q_{\theta_i}(s,a) - y)^2$ for i=1, 2
    CQLTerm $\leftarrow \lambda \left[\log \sum_{a'} \exp Q_{\theta_i}(s,a') - Q_{\theta_i}(s,a)\right]$
    $\theta_i \leftarrow \theta_i - \eta_Q \nabla_{\theta_i}$[MSE + CQLTerm]
    $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \sum_a \pi_\phi(a|s) \big[\alpha \log \pi_\phi(a|s) - \min_i Q_{\theta_i}(s,a)\big]$
    $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \sum_a \pi_\phi(a|s) \big[-\alpha(\log \pi_\phi(a|s) + \bar{\mathcal{H}})\big]$
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$
  end for
end while

2.7 Behavior Cloning Methods Formulation

BC treats imitation learning as supervised learning: given a demonstration log $\mathcal{D} = \{(s, a)\}$ gathered from a demonstrator, a parametric policy π_ϕ is fitted by minimizing the negative log-likelihood of demonstrated actions,

$$\min_\phi \; \mathcal{L}_{\mathrm{BC}}(\pi_\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[-\log \pi_\phi(a \mid s)\big]. \tag{2.17}$$

Although BC attains low prediction error on the dataset, it is prone to covariate shift: small test-time deviations from expert trajectories compound over time, leading to off-distribution states that the policy cannot handle. We therefore complement BC with a distribution-matching method that retains the ability to reason about unseen states.

Generative Adversarial Imitation Learning (GAIL) casts imitation as a two-player minimax game between a policy π_ϕ (generator) and a discriminator D_ψ that distinguishes expert state–action pairs from those produced by the policy [43]:

$$\min_\phi \max_\psi \; \mathbb{E}_{(s,a)\sim\pi_E}\big[\log D_\psi(s,a)\big] + \mathbb{E}_{(s,a)\sim\pi_\phi}\big[\log\big(1 - D_\psi(s,a)\big)\big] - \lambda \mathcal{H}(\pi_\phi). \tag{2.18}$$

Here, π_E denotes the expert state–action distribution and λ is the entropy-regularization weight. The optimal discriminator delivers a learned reward signal $r_\psi(s,a) = -\log(1 - D_\psi(s,a))$ that is large for expert-like behavior and small otherwise. Policy improvement is achieved with any RL optimizer.

At each iteration we (i) generate roll-outs with the current policy, (ii) update D_ψ via stochastic gradient ascent on (2.18), and (iii) update π_ϕ by maximizing the expected, shaped reward $r_\psi(s,a)$ under a trust-region constraint. Because D_ψ is trained on both expert and policy data, the reward automatically adapts to the policy's current occupancy measure, alleviating the covariate-shift pathology of plain BC.

Highway feature multi-modal decisions (e.g. yield, merge, or overtake) and sparse, hard-coded rewards are difficult to design. GAIL circumvents manual reward shaping and instead derives a dense, informative signal from expert demonstrations. The adversarial objective encourages the learned occupancy measure to approach that of the demonstrations, but it does not by itself guarantee safe behavior outside the demonstrated state distribution.

Algorithm 3 Generative Adversarial Imitation Learning

Require: Expert dataset $\mathcal{D}_E$, initial policy π_{θ_0}, discriminator D_ϕ, entropy weight λ

while not converged **do**
  (Generator) roll out π_θ to collect trajectories τ_π
  (Discriminator) update ϕ via $\nabla_\phi \Big[\mathbb{E}_{\tau_E}\big[\log D_\phi\big] + \mathbb{E}_{\tau_\pi}\big[\log(1 - D_\phi)\big] \Big]$
  Compute rewards $r_t \leftarrow -\log\big(1 - D_\phi(s_t, a_t)\big)$
  (Policy update) perform a TRPO step on π_θ with reward r_t and entropy bonus $\lambda\mathcal{H}$
  $\theta \leftarrow \mathrm{TRPO}(\theta, \nabla_\theta J_{\mathrm{RL}})$
end while

Chapter 3

Proposed RL Bootstrapping Framework

We integrate a suboptimal rule-based controller into the RL framework to enhance learning efficiency and guide exploration. Specifically, we leverage the suboptimal policy in two ways: (1) as a soft constraint on the RL policy during initial training stages, and (2) as a source for populating the replay buffer with additional training samples.

3.1 Rule-based Heuristic Suboptimal Controller

We introduce a rule-based suboptimal controller $\mu^{\text{rule}}(s)$ designed specifically for overtaking scenarios, as illustrated in Algorithm 4. The controller utilizes an overtaking decision flag O_{dec}, which indicates whether an overtaking maneuver is permitted and is safe. Specifically, the flag is set to true only when the following safety conditions are satisfied:

- The ego vehicle will not rear-end a slower vehicle in the target lane after the lane change.
- The ego vehicle will not be rear-ended by a faster vehicle already in the target lane.

In our current scenario, there are no following vehicles in either the current or target lane. Traffic vehicles are spawned 25–32 m ahead of the frontmost trap vehicle and are never slower than the trap vehicles, so the trap-traffic gap does not shrink and the ego can overtake without being blocked by front traffic. Within this setting, the flag can be safely set to true to allow the overtaking behavior. If overtaking is enabled ($O_{\text{dec}} = \text{true}$), the controller may change lane to the target lane index l_{target}, otherwise the ego vehicle remains in its current lane l_{current}.

To ensure safe lane-changing decisions, we employ a unified safety-distance metric, denoted as d_s, which represents the minimum required separation distance to the preceding vehicle to perform overtaking maneuvers to the target lane. This safety distance is calculated as follows:

$$\begin{aligned} d_s = \big(v^e_{\text{lon}} - \min\{\, v_{\text{l\_current}},\, v_{\text{l\_target}} \,\}\big)\, t_{\text{safe}} \\ + \Delta x_{\text{l\_current}} - \Delta x_{\text{l\_target}} \end{aligned} \tag{3.1}$$

where v^e_{lon} represents the speed of the ego vehicle, $v_{\text{l\_current}}$ and $v_{\text{l\_target}}$ denote the speeds of the current preceding vehicles in the current lane and the speed of the new preceding vehicle in the target lane, respectively, $\Delta x_{\text{l\_current}}$ and $\Delta x_{\text{l\_target}}$ denote the longitudinal distance of the current preceding vehicle and the new preceding vehicle with respect to the ego vehicle, $\Delta x_{\text{l\_current}} - \Delta x_{\text{l\_target}}$ indicates

Algorithm 4 Suboptimal heuristic overtaking algorithm

Require: Environment state $s(t)$, overtaking decision flag O_{dec}, target lane index l_{target}
while $O_{\text{dec}} = \text{true}$ **do**
  if $(d_s - \Delta x_{\text{l\_current}}) > 0$ **then**
    $(v, l) \leftarrow \left(v_{\text{target}},\, l_{\text{target}}\right)$
  else
    $(v, l) \leftarrow \left(v_{\text{target}},\, l_{\text{current}}\right)$
  end if
end while
Output: Target speed and lane index (v, l)

the longitudinal distance between the current preceding vehicle and the new preceding vehicle, and t_{safe} is the safety time distance to the preceding vehicle.

The target speed for the ego vehicle, denoted by v_{target}, is selected as the maximum feasible speed from the predefined set

$$V = \left\{ v_{\min} + \frac{i-1}{n-1}(v_{\max} - v_{\min}) \;\middle|\; i = 1, \ldots, n \right\}. \tag{3.2}$$

Specifically,

$$v_{\text{target}} = \max\left\{ v_i \in V \mid v_i < \min\left(v_{\text{l\_current}}, v_{\text{l\_target}}\right) \right\}, \tag{3.3}$$

where $v_{\min} = 6$ m/s, $v_{\max} = 15$ m/s, and $n = 5$. This selection ensures that a safe distance is maintained from the trap vehicles during the overtaking maneuver.

In conclusion, the controller initiates the overtaking maneuver by identifying the new preceding vehicle in the target lane and maintaining a safe overtaking distance from the current preceding vehicle. The actual lane change is then executed by tracking the designated target speed and lane index, which are subsequently handled by the lower-level rule-based controller [27].

3.2 Low-level Motion Planner

The target speed and lane index (v, l) generated by Algorithm 4 are converted into the low-level acceleration and steering commands $(\dot{v}, \theta)$. The acceleration command is obtained using proportional speed tracking,

$$\dot{v} = \text{clip}[K_v\left(v - v_{\text{ego}}\right), -1, 1], \tag{3.4}$$

where $K_v = 0.6$ is the longitudinal control gain.

For steering control, the target lane index l identifies the corresponding target-lane centerline. Let e_l denote the signed lateral displacement of the ego vehicle from this centerline, and let ψ_l denote the heading of the target-lane centerline at a look-ahead position. The steering command is generated by

$$\begin{aligned}
\psi_{\text{ref}} &= \psi_l + \text{clip}\left[\sin^{-1}\left(\frac{-K_l e_l}{v_{\text{ego}}}\right), -\frac{\pi}{4}, \frac{\pi}{4}\right], \\
\dot{\psi}_{\text{ref}} &= K_\psi \ \text{wrap}_{[-\pi,\pi]}(\psi_{\text{ref}} - \psi_{\text{ego}}), \\
\beta &= \sin^{-1}\left(\text{clip}\left[\frac{L\dot{\psi}_{\text{ref}}}{2v_{\text{ego}}}, -1, 1\right]\right), \\
\theta &= \text{clip}\Big[\tan^{-1}(2\tan\beta), -\frac{\pi}{50}, \frac{\pi}{50}\Big],
\end{aligned} \tag{3.5}$$

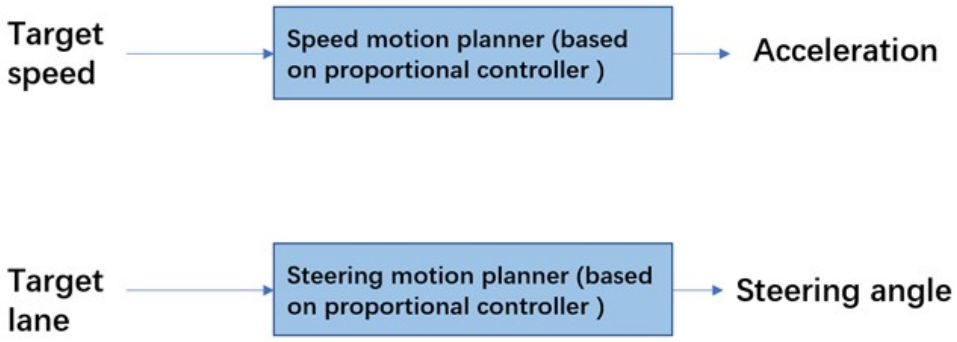


Figure 3.1: Speed and steering motion planner

where $K_l = 0.3$ and $K_\psi = 2.0$ are the lateral-position and heading-control gains, respectively; ψ_{ref} is the desired reference heading; ψ_{ego} is the current ego-vehicle heading; v_{ego} is the current ego-vehicle speed; $\dot{\psi}_{\text{ref}}$ is the desired yaw rate; β is the intermediate slip angle; L is the vehicle length; $\text{clip}[x, a, b]$ bounds x to $[a, b]$; and $\text{wrap}_{[-\pi,\pi]}(\cdot)$ maps an angular difference to its equivalent angle in $[-\pi, \pi]$.

3.3 Bootstrapping RL Methodology

3.3.1 Soft Constraint through KL Divergence

Instead of directly imitating the rule-based policy, we apply a Kullback-Leibler (KL) penalty to nudge the learned policy π toward the suboptimal strategy during initial training [39]. Specifically, we modify the entropy-regularized RL objective as

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(r(s_t,a_t) \;+\; \alpha\,\mathcal{H}\big(\pi(\cdot\mid s_t)\big) \\ -\;\xi\,D_{\text{KL}}\Big(\pi(\cdot\mid s_t)\,\|\,\tilde{\pi}^{\text{rule}}(\cdot\mid s_t)\Big)\Big)\Bigg] \tag{3.6}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denotes a trajectory generated by π, $\alpha > 0$ trades off reward maximization against policy entropy $\mathcal{H}(\pi)$, and $\xi \geq 0$ controls the strength of the KL term. Here, $\tilde{\pi}^{\text{rule}}$ denotes a *stochastic* version of the rule-based policy as defined below. During early training, a higher ξ ensures the agent's policy stays near the suboptimal policy, but over time this value will be relaxed so that π surpasses the demonstrator's performance. Specifically, we parameterize ξ as a function of the entropy coefficient α, setting $\xi = 200 \cdot \alpha$ to maintain a balance between imitation and exploration throughout training.

Our rule-based policy $\mu^{\text{rule}}(s)$ is inherently deterministic, outputting a single best action from a discrete set of $|\mathcal{A}| = 9$. To introduce uncertainty, we construct $\tilde{\pi}^{\text{rule}}(\cdot \mid s)$ by assigning a high probability to the deterministic output and small probabilities to other actions. Specifically, if a_{rule} is the chosen action, we let

$$\tilde{\pi}^{\text{rule}}(a \mid s) = \begin{cases} p, & \text{if } a = \mu^{\text{rule}}(s), \\ \frac{1-p}{|\mathcal{A}|-1}, & \text{otherwise}, \end{cases} \tag{3.7}$$

As shown in Figure 4.7d, we set $p = 0.9$ to maximize the average episodic return.

Introducing stochasticity serves two purposes. First, it aligns action distributions: since SAC policies are inherently stochastic, introducing stochasticity into the suboptimal controller yields a more compatible distribution. This alignment makes KL-based regularization both meaningful and numerically stable. Second, it mitigates over-reliance on the suboptimal controller and promotes robust exploration. By incorporating stochasticity, the agent avoids overfitting to the demonstration policy and is exposed to a broader range of action options. This exposure enables the agent to generalize more effectively and discover improved policies beyond the limitations of the initial suboptimal demonstrations.

3.3.2 Integration of Suboptimal Demonstrations

To guide the agent toward heuristic driving strategies and provide safe reference actions during the early stages of learning, we enrich the replay buffer with demonstration data generated by the suboptimal rule-based controller $\tilde{\pi}^{\text{rule}}$. We collected 200 episodes of demonstration transitions $(s, a_{\text{rule}}, r(s, a_{\text{rule}}), s')$ by running $\tilde{\pi}^{\text{rule}}$ in the simulated environment. As illustrated in Figure 1.3, during training both offline demonstrations and online experience are sampled according to an annealed ratio β, which decays from 0.6 to 0 over the first 1000 episodes. This ensures early guidance without constraining policy improvement in the long term.

We evaluate two strategies for integrating these demonstrations into learning. The comparison results are presented in Section 4.5.

Q-Value Regularization via Action Margins

This strategy explicitly regularizes the Q-values by enforcing a margin between the demonstrator's action and all others [44–46]. The margin supervised loss for each demonstration (s, a_E) is

$$J_E(Q) = \max_{a \in \mathcal{A}} [Q(s, a) + l(a_E, a)] - Q(s, a_E), \tag{3.8}$$

where $l(a_E, a)$ is a margin function defined as

$$l(a_E, a) = \begin{cases} 0, & \text{if } a = a_E, \\ 1, & \text{otherwise.} \end{cases} \tag{3.9}$$

Let a_π denote the action selected by the current policy. We apply a Q-filter to prevent the agent from strictly imitating suboptimal actions using

$$J_E^Q(Q) = \begin{cases} J_E(Q), & \text{if } Q(s, a_E) \geq Q(s, a_\pi), \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

This filter ensures that supervision is only applied when the demonstrator's action is at least as good as the agent's current estimate.

Demonstration-Guided Reward Augmentation

As an alternative to explicit supervision, we implement reward shaping by adding a bonus to demonstration actions [47]

$$r'(s, a_E) = r(s, a_E) + c \tag{3.11}$$

where $c = 2$ is a constant value. This method implicitly increases the Q-values of demonstrated actions and produces effects similar to large-margin regularization. Empirical study shows that reward shaping yields better long-term performance and generalization, especially when the goal is to surpass or recover from imperfect demonstrations [47].

Chapter 4

Experimental Evaluation

As described in Section 2.1, the simulation is initialized with trap vehicles, followed by traffic vehicles governed by IDM and MOBIL models. The environment is implemented using the `highway-env` platform [48]. The simulator runs at 15 Hz, while the discrete policy selects an action at 2 Hz. All experiments are conducted on an Ubuntu system with an NVIDIA GeForce RTX-3070 Ti GPU.

We report the mean and standard deviation over five initialization seeds (1, 2, 3, 4, 5) for cross-scenario generalization, and we use the same protocol for all testing analyses and training figures. The only exception is the hyperparameter ablation in Section 4.5, where we report results over three seeds (1, 2, 3) due to the substantially higher computational cost of the sweep.

For each test episode $k = 1, \ldots, N_{\text{test}}$, the episode lasts for 150 time steps, and the performance metrics are accumulated at each policy update. In addition to standard performance metrics, including episode reward, average speed, and distance traveled, we report safety and task-specific metrics to better characterize policy behavior.

Accident rate: Let $C^{(k)} \in \{0, 1\}$ indicate whether the ego vehicle collides with any traffic vehicle in episode k (geometric overlap), and let $O^{(k)} \in \{0, 1\}$ indicate whether the ego vehicle leaves the drivable road surface in episode k (road excursion), i.e., its footprint exits the union of all lanes in the road network. Either event terminates the episode. We define $A^{(k)} = \max(C^{(k)}, O^{(k)})$ and report the accident rate as $\text{Accident} = \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} A^{(k)}$.

Road excursion rate: We report the road excursion rate separately as

$$\text{RoadExc} = \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} O^{(k)}, \tag{4.1}$$

which isolates off-road events from collisions with other vehicles.

Overtaking success rate: An episode is counted as a successful trap escape overtake if the ego vehicle overtakes the foremost trap vehicle and remains ahead of it until the end of the episode. Let $E^{(k)} \in \{0, 1\}$ denote this event for test episode k, and let N_{test} be the number of test episodes. We report $\text{Overtake} = \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} E^{(k)}$.

Minimum headway: Let x_t^{ego} and x_t^{lead} be the longitudinal positions of the ego vehicle and the lead vehicle in the ego lane at time t. The headway is $h_t = x_t^{\text{lead}} - x_t^{\text{ego}}$ with $h_t > 0$. The per-episode minimum headway is $h_{\min}^{(k)} = \min_t h_t^{(k)}$ and we report its mean and standard deviation over test episodes.

Minimum TTC: Using the same lead vehicle, let v_t^{ego} and v_t^{lead} be the longitudinal velocities and define $v_t^{\text{rel}} = v_t^{\text{ego}} - v_t^{\text{lead}}$. When $h_t > 0$ and $v_t^{\text{rel}} > 0$, $\text{TTC}_t = h_t / v_t^{\text{rel}}$. The per-episode minimum TTC is $\text{TTC}_{\min}^{(k)} = \min_{t:\, v_t^{\text{rel}} > 0} \text{TTC}_t^{(k)}$ and we report its mean and standard deviation over test episodes.

Table 4.1: Testing comparison with rule-based methods, imitation learning, offline RL, and demonstration-augmented RL baselines (reported as mean ± stddev)

Category	Method	Reward	Velocity [m/s]	Distance [m]	Accident [%]	Overtake [%]	Road Excursion [%]	Min Headway [m]	Min TTC [s]
Rule-based	Suboptimal	25.0 ± 0.5	11.5 ± 0.1	1731.8 ± 9.1	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.1 ± 1.0	13.3 ± 1.4
	IDM+MOBIL	34.2 ± 29.9	12.2 ± 0.9	1554.5 ± 491.0	38.0 ± 49.0	0.0 ± 0.0	0.0 ± 0.0	15.6 ± 19.2	203.3 ± 306.6
Imitation	BC	-9.0 ± 1.4	10.2 ± 0.6	211.0 ± 145.0	100.0 ± 0.0	0.0 ± 0.0	42.3 ± 18.1	12.5 ± 4.5	97.4 ± 221.1
	GAIL	22.0 ± 4.0	10.0 ± 0.0	1478.6 ± 89.7	6.5 ± 24.7	0.0 ± 0.0	3.3 ± 12.3	16.1 ± 0.5	74.0 ± 4.2
Offline RL	BCQ	41.0 ± 24.0	12.1 ± 0.6	1684.5 ± 305.0	36.6 ± 48.2	91.4 ± 28.0	16.5 ± 23.5	14.7 ± 2.5	6.4 ± 1.9
	IQL	22.2 ± 2.1	11.4 ± 0.1	1628.0 ± 65.2	8.0 ± 5.6	65.6 ± 12.9	0.1 ± 0.2	11.7 ± 0.6	13.3 ± 0.8
	AWAC	20.7 ± 3.2	11.3 ± 0.1	1549.4 ± 141.1	14.8 ± 13.3	60.2 ± 14.5	0.5 ± 0.8	11.3 ± 1.1	33.2 ± 26.9
	CQL	-9.4 ± 1.1	10.0 ± 0.5	140.7 ± 111.0	100.0 ± 0.0	0.0 ± 0.0	39.8 ± 20.2	12.9 ± 4.5	67.7 ± 103.2
	TD3+BC	-7.6 ± 4.7	10.3 ± 0.6	224.5 ± 361.0	100.0 ± 0.0	0.0 ± 0.0	44.2 ± 10.6	14.1 ± 3.4	17.3 ± 10.7
DARL	DDPGfD	50.6 ± 14.3	12.1 ± 1.0	1791.8 ± 147.8	4.8 ± 8.6	80.0 ± 40.0	2.4 ± 4.3	14.3 ± 1.4	13.7 ± 0.9
	DAPG	33.9 ± 3.9	12.0 ± 0.1	1783.8 ± 25.1	1.2 ± 1.0	96.0 ± 2.8	0.2 ± 0.4	12.8 ± 0.2	13.7 ± 0.5
	SQIL	14.1 ± 2.5	11.1 ± 0.7	1356.7 ± 35.5	79.2 ± 12.0	58.0 ± 43.7	34.0 ± 12.7	13.0 ± 1.0	18.6 ± 7.6
	Ours	104.8 ± 4.3	13.7 ± 0.1	2056.5 ± 20.5	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.9 ± 0.9	9.5 ± 3.0

For episodes with $C^{(k)} = 1$, the physical headway and TTC at impact are zero. Since evaluation is discrete-time, we compute headway and TTC from pre-impact states, while collision occurrence is captured by the accident indicator.

4.1 Rule-Based Suboptimal Demonstrator Policy

We first evaluate the rule-based suboptimal controller in the trap scenario and summarize the testing results in Table 4.1. This demonstrator is intentionally simple to implement and provides a suboptimal strategy that an RL agent is unlikely to discover from scratch. It serves as a practical and easily implemented source of imperfect demonstrations while encoding longer-horizon, human-like driving heuristics. Because the rule-based controller does not explicitly address corner cases, its task completion can degrade in more complex overtaking scenarios, as shown in Section 4.7. Starting from these demonstrations, the RL agent further optimizes the driving policy through online interaction with the simulated environment and learns to handle common driving scenarios.

Table 4.1 confirms that the suboptimal controller is substantially below the learned policy. It achieves a much lower episode reward, lower average speed, and shorter distance traveled. We also include IDM+MOBIL as a classical rule-based baseline. For a fair comparison, IDM+MOBIL controls all traffic vehicles, and the ego vehicle follows the same policy with a higher desired speed of 15 m/s. We use the highway-env IDM+MOBIL implementation and set its steering and acceleration ranges to match our discrete action space.

Overall, this comparison motivates the use of suboptimal rule-based demonstrations as a bootstrapping signal. They provide safe and structured behavior for early-stage learning, but their limited performance and strategy coverage make them insufficient as a standalone solution.

4.2 Imitation Learning Baselines

Table 4.1 compares our method with two imitation-learning baselines, Behavior Cloning (BC) and Generative Adversarial Imitation Learning (GAIL). Both baselines are trained on the same

demonstration dataset collected using the rule-based suboptimal controller. All demonstration trajectories in this dataset successfully overtake the frontmost trap vehicle, so BC and GAIL have access to successful demonstrations during training.

BC: BC [49] is a supervised learning baseline that directly fits a policy to demonstrator actions using state–action pairs from the continuous-control version of the demonstration dataset. It is implemented as an MLP that predicts acceleration and steering by minimizing a regression loss.

GAIL: GAIL [43] leverages adversarial training to learn policies that mimic expert behaviors from demonstrations without explicitly defining reward functions. GAIL utilizes a discriminator to distinguish between expert and learned behaviors, guiding the policy toward human-like actions.

BC performs the worst, with negative rewards and high collision rates. GAIL achieves a higher reward than BC and reduces accident frequency on average. However, it fails to overtake the frontmost trap vehicle in every trial and yields low speeds. Overall, Table 4.1 indicates that imitation alone is not sufficient for the trap scenario, even when the demonstrations are consistently successful. This motivates integrating demonstrations with online interaction and reinforcement learning updates to correct compounding errors and to improve performance outside the demonstration distribution.

4.3 Offline Reinforcement Learning Baselines

This subsection evaluates whether offline reinforcement learning can solve the trap scenario when training uses the same pre-collected demonstration dataset. We compare several representative offline RL baselines during training (Figure 4.1) and at test time (Table 4.1).

TD3 with Behavior Cloning (TD3+BC): TD3+BC [50, 51] is an offline variant of TD3 that adds a behavior cloning regularizer to the actor objective. The RL objective encourages high-return actions, while the BC term constrains the policy toward actions observed in the dataset, improving stability in the low-coverage offline setting.

Batch-Constrained Q-Learning (BCQ): BCQ [52] is an offline RL algorithm that restricts the learned policy to remain close to the behavior policy by sampling candidate actions from a learned generative model. By constraining actions to the support of the dataset, BCQ reduces extrapolation error when evaluating out-of-distribution actions.

Conservative Q-Learning (CQL): CQL [42] is an offline RL algorithm designed for learning Q-functions from pre-collected datasets without online interaction. Its conservative regularizer mitigates overestimation errors common in offline RL settings.

Implicit Q-Learning (IQL): IQL [53] is an offline RL method that avoids explicit policy improvement over out-of-distribution actions by learning value functions using expectile regression. It derives a policy through advantage-weighted regression, which biases the policy toward actions with higher advantages while remaining stable under purely offline training.

Advantage Weighted Actor Critic (AWAC): AWAC [54] learns a value function and then updates the policy using advantage-weighted supervised regression on actions in the dataset. Actions with larger estimated advantages receive higher weights, enabling the policy to leverage suboptimal demonstrations while still improving beyond the behavior policy.

Figure 4.1 highlights clear differences in learning dynamics across offline RL baselines. In Figure 4.1a, some methods rarely learn trap escape overtaking behavior during training, while others achieve higher escape frequency but exhibit substantial instability across seeds. The reward, speed, and distance traveled trends in Figure 4.1 show a similar pattern: during training, several offline baselines plateau early at low performance, indicating that offline optimization struggles to discover and sustain an effective escape strategy under the trap configuration.

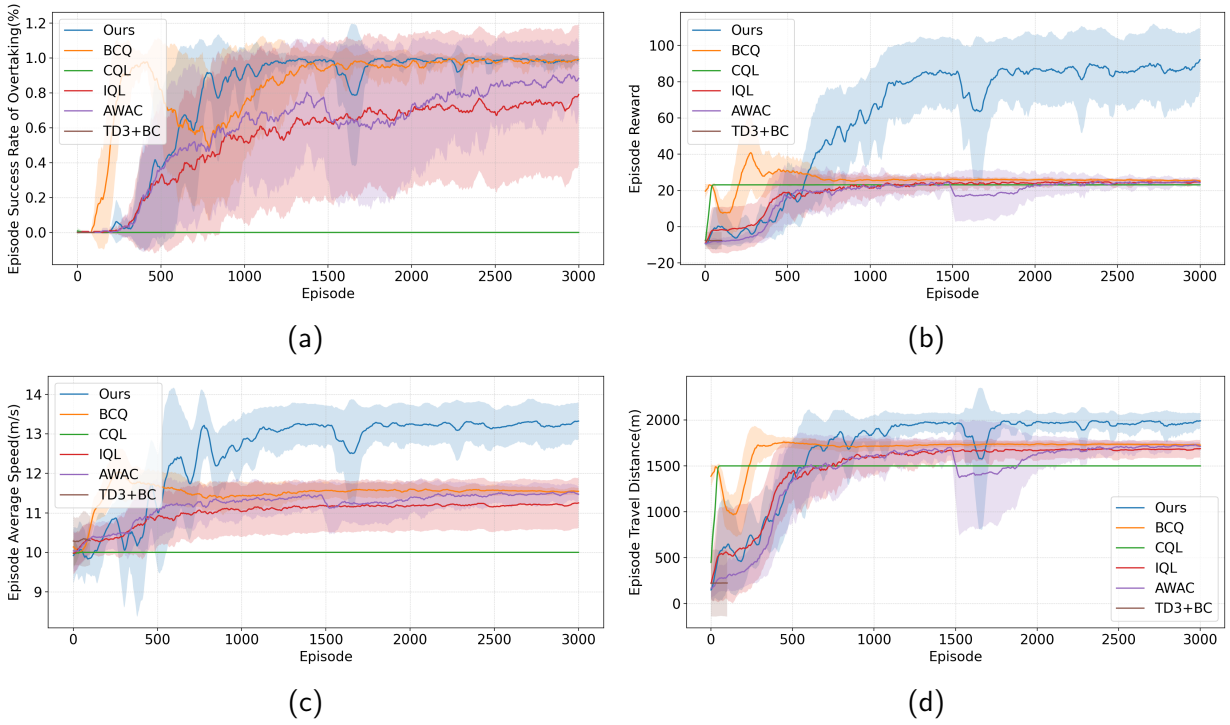


Figure 4.1: Training performance comparison with other offline RL baseline methods: (a) Average success rate in overtaking low-speed traffic; (b) Average reward per episode; (c) Average speed per episode; (d) Average distance traveled per episode.

The testing results in Table 4.1 provide a consistent conclusion. Among the offline RL baselines, BCQ achieves the strongest task completion signal and the highest reward on average. However, this gain comes with severe safety degradation, including a high accident rate and a non-trivial road excursion rate, indicating poor robustness and high sensitivity to distribution shift. This behavior is consistent with the fact that BCQ relies on an approximate generative model and candidate action selection; small modeling errors can lead to unsafe action choices even when the nominal task objective improves.

IQL and AWAC exhibit more conservative performance profiles. They obtain moderate rewards and moderate overtaking success rates, but both still incur non-negligible accident rates. This suggests that advantage-weighted style updates can leverage the offline data to some extent, but the resulting policies remain limited by dataset coverage and cannot reliably maintain safe overtaking behavior under evaluation-time state distribution shift. Finally, CQL and TD3+BC collapse in this setting. Both methods yield negative rewards, fail to overtake, and generate accidents in every episode. They also show strikingly high road excursion rates, indicating severe control instability. These outcomes reflect a core difficulty of offline RL in this problem: when the learned policy deviates from the dataset support, value estimation and policy updates can become unreliable, and the lack of online correction prevents recovery from compounding errors.

Overall, Figure 4.1 and Table 4.1 show that directly optimizing on suboptimal, low-variance demonstrations in a purely offline manner is insufficient for reliable overtaking with strong safety performance. Some offline methods partially improve task success but suffer from large safety costs, while others fail catastrophically. Compared with imitation learning, offline RL still exhibits a clear advantage in this setting. Unlike BC and GAIL, which fail to overtake in all trials, several

offline RL baselines achieve non-trivial overtaking success and higher reward. This improvement suggests that offline RL can exploit the same demonstration data more effectively by optimizing for long-horizon return through value learning, rather than only matching actions locally. However, the gains are often accompanied by reduced safety margins or high variance, indicating that purely offline optimization remains vulnerable to distribution shift and value extrapolation errors.

Learning a robust neural policy from a fixed dataset remains challenging in this trapped-vehicle scenario. The task depends on a small number of temporally critical decisions, such as when to slow down, change lanes, and accelerate to escape. These decisive actions occupy only a small fraction of each trajectory, while most samples correspond to ordinary lane-keeping or car-following behavior. Therefore, imitation learning and offline RL baselines may reproduce reasonable local driving behavior but still fail to complete the full closed-loop escape maneuver.

This difficulty is further increased by the threshold-sensitive nature of overtaking behavior. A learned policy trained only on a fixed dataset may slightly deviate from the demonstrated trajectory and enter states that are poorly covered by the data. In such off-distribution states, BC and GAIL do not have an explicit recovery mechanism, while conservative offline RL methods such as CQL, IQL, AWAC, and BCQ may prefer common low-risk actions, such as lane keeping or car following, instead of the decisive maneuver required for escape.

Therefore, the weaker performance of the baselines reflect the difficulty of applying static imitation or offline RL methods to this closed-loop, threshold-sensitive trapped-vehicle escape task. The proposed Suboptimal Bootstrap method improves performance because it uses the suboptimal controller as structured training-time guidance, repeatedly exposing the learner to states near successful escape behaviors instead of relying only on fixed demonstration fitting.

These results motivate integrating demonstrations with online interaction, so that the agent can correct dataset bias, recover from distribution shift, and refine the escape strategy beyond what is represented in the offline data.

4.4 Demonstration-Augmented Reinforcement Learning

This subsection compares our approach with representative demonstration-augmented reinforcement learning (DARL) baselines. These methods leverage demonstrations during online RL via imitation-style regularization, modified replay strategies, or reward shaping. Figure 4.2 and Table 4.1 report training curves and test results across multiple evaluation metrics.

Demo-Augmented Policy Gradient (DAPG): DAPG [55] augments the policy-gradient update with an additional behavior-cloning log-likelihood term on demonstration state–action pairs, pulling the policy toward demonstrated actions during early learning. The demonstration loss stabilizes early learning and encourages the policy to follow demonstrated behaviors while still optimizing return through online interaction.

DDPG from Demonstrations (DDPGfD): DDPGfD [56, 57] extends off-policy actor-critic learning by injecting demonstrations into the replay buffer and biasing updates toward demonstrated transitions. This design aims to accelerate learning in sparse or hard-exploration tasks while retaining the ability to improve online.

Soft Q Imitation Learning (SQIL): SQIL [47] reduces imitation learning to standard off-policy RL by assigning a constant positive reward to demonstration transitions and zero reward to non-demonstration transitions. This reward relabeling encourages long-horizon imitation by driving the policy back toward demonstrated states, and it can be implemented on top of common Q-learning or off-policy actor–critic backbones.

Demonstration-augmented RL baselines leverage demonstration data to guide learning and can

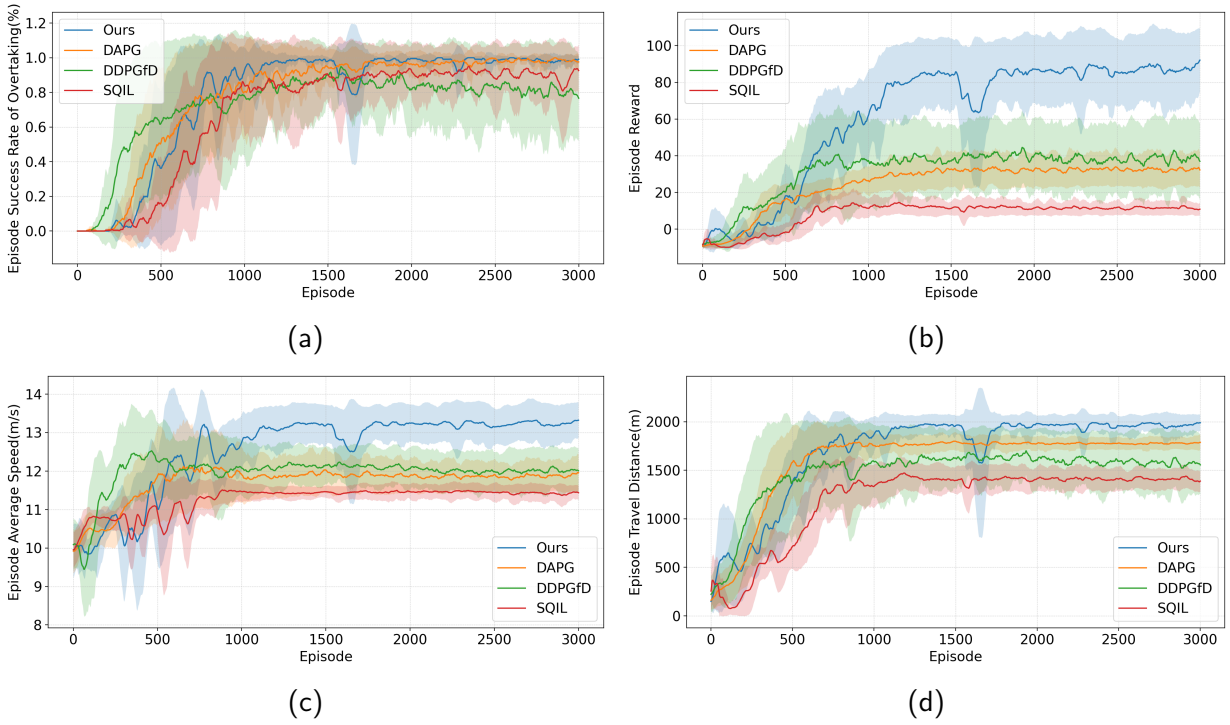


Figure 4.2: Training performance comparison with other demonstration-augmented RL baseline methods: (a) Average success rate in escaping low-speed traffic; (b) Average reward per episode; (c) Average speed per episode; (d) Average distance traveled per episode.

Table 4.2: Baseline-specific hyperparameters

Algorithm	Hyperparameter	Value
SQIL	Demonstration replay ratio	0.6
DAPG	Imitation coefficient	2.0
DAPG	Demonstration replay ratio	0.6
DDPGfD	Imitation coefficient	1.0
DDPGfD	Exploration rate	0.1
TD3+BC	Policy noise	0.2
TD3+BC	Noise clipping bound	0.5
TD3+BC	Delayed update period	2
TD3+BC	BC coefficient	2.5
CQL	Conservative penalty coefficient	1.0
AWAC	Advantage temperature	1.0
AWAC	Maximum advantage weight	20.0
IQL	Expectile parameter	0.7
IQL	Policy temperature	3.0
IQL	Maximum policy weight	100.0
BCQ	Action filter threshold	0.3

outperform pure imitation or offline imitation-style baselines in our setting. Figure 4.2 shows that several representative methods eventually acquire non-trivial overtaking behavior during training, indicating that demonstrations can help the agent reach exploration regimes that are difficult to achieve from scratch. However, many baselines plateau early and do not consistently converge to high-return solutions.

These limitations are also reflected at test time. Table 4.1 shows that baseline performance varies substantially across reward and efficiency metrics, and several methods incur non-negligible accident rates. Overall, Figure 4.2 and Table 4.1 suggest that while demonstration-guided baselines provide meaningful gains, their performance remains sensitive to the specific scenario and often saturates at suboptimal behavior in our driving scenario.

Table 4.2 lists only the additional baseline-specific hyperparameters that are not already provided in Table 2.4.

4.5 Component-Wise Ablation

As previously introduced, we integrate the suboptimal driving policy by applying soft constraints and integrating a demonstration dataset. Both methods are built on vanilla SAC with a discrete action space.

SAC: SAC [25] is an off-policy actor-critic algorithm that optimizes a stochastic policy while maximizing both expected returns and policy entropy. In this project, SAC serves as an off-policy baseline to illustrate the benefits of bootstrapping online RL with prior demonstration data.

To determine the better strategy for integrating suboptimal demonstration data into RL training, we conduct an ablation study comparing the two approaches. As shown in Figure 4.3, using reward augmentation leads to better learning outcomes across all metrics compared with using margin loss. These results suggest that reward augmentation offers more effective guidance for suboptimal demonstration integration.

Figure 4.4 and Table 4.3 detail an ablation study examining each approach during training and testing. They demonstrate that adding a reward-augmented offline replay buffer improves the success rate in overtaking the trap vehicles, especially during initial training phases, likely due to the diversity in action distributions introduced early on. However, using the reward-augmented offline replay buffer alone introduces training instability and increases the risk of potential collisions. Employing SAC with only soft constraints results in slower learning and reduced performance compared to Backbone SAC. This outcome may occur because soft constraints limit SAC's actions to those recommended by the suboptimal policy, reducing exploration potential and adaptability.

The combination of soft constraints with an offline replay buffer appears to better balance the exploration and exploitation trade-off. Soft constraints guide SAC toward safer behaviors initially, while the offline replay buffer provides diverse experiences, thus enabling more robust policy improvement and stable convergence to optimal performance.

Table 4.3: Testing comparison results for different methods of integrating suboptimal demonstrations

Method	Overtake [%]	Reward	Velocity [m/s]	Distance [m]	Accident [%]
SAC	0.0 ± 0.0	25.7 ± 9.4	10.5 ± 0.3	1536.4 ± 125.2	12.0 ± 32.8
SAC + soft constraint	0.0 ± 0.0	26.8 ± 0.9	10.9 ± 0.0	1631.5 ± 3.4	0.0 ± 0.0
SAC + offline replay buffer	100.0 ± 0.0	36.7 ± 9.7	12.1 ± 0.4	1804.4 ± 55.8	5.0 ± 29.7
Ours	100.0 ± 0.0	106.4 ± 2.6	13.7 ± 0.1	2060.5 ± 18.7	0.0 ± 0.0

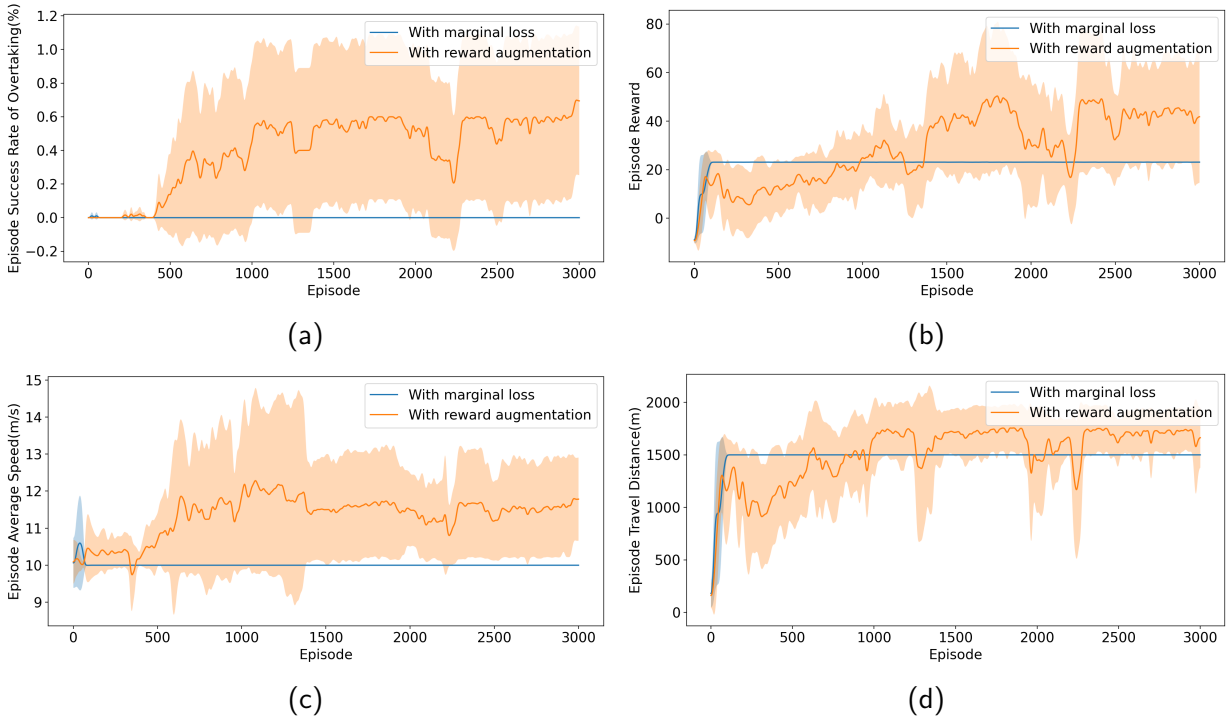


Figure 4.3: Training performance comparison for integrating demonstration data via marginal loss or reward augmentation: (a) Average success rate in escaping low-speed traffic; (b) Average reward per episode; (c) Average speed per episode; (d) Average distance traveled per episode.

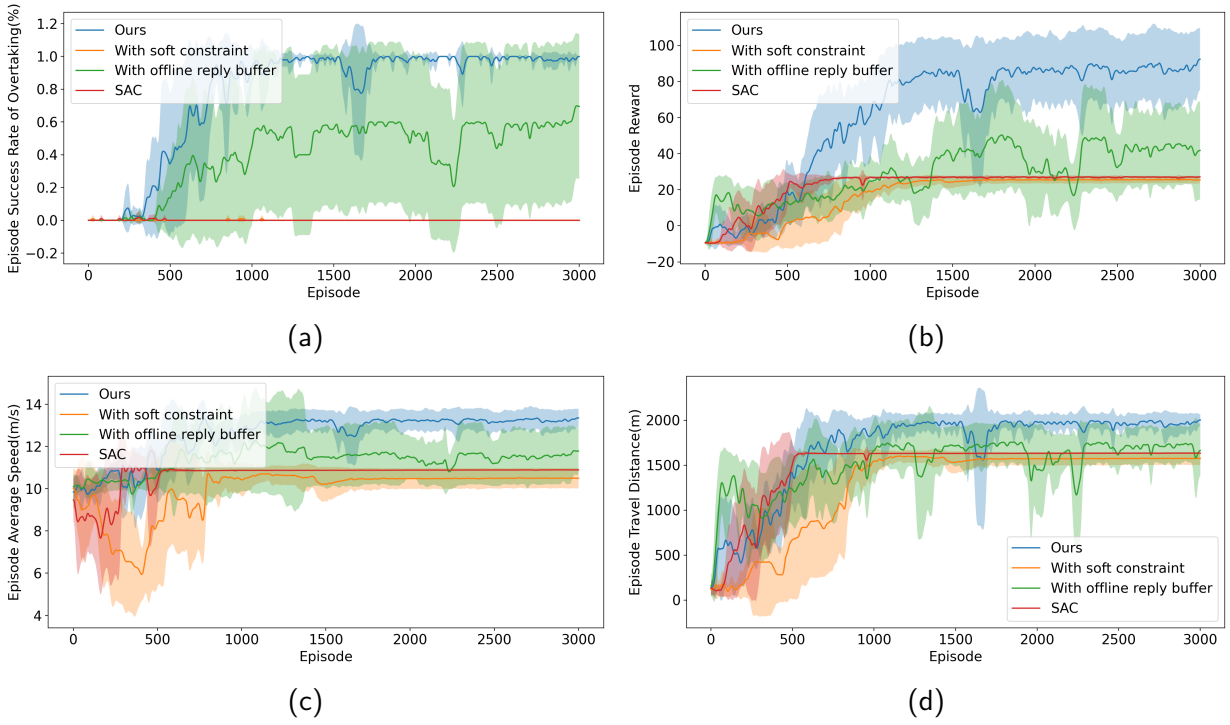


Figure 4.4: Performance during training episodes for SAC with soft constraint, SAC with offline replay buffer, and our method: (a) Average success rate in escaping low-speed traffic; (b) Average reward per episode; (c) Average speed per episode; (d) Average distance traveled per episode.

4.6 Effect of Entropy Regularization and Exploration Scheduling

We compare the exploration performance of our proposed method against standard SAC by measuring the average policy entropy and the average effective action count over the course of training. Policy entropy H_{avg} quantifies the randomness or diversity in the policy's action distribution and is computed from

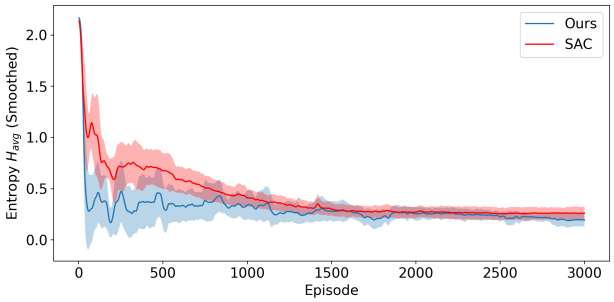
$$H_{\text{avg}} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}}\pi_t(a)\log\pi_t(a), \tag{4.2}$$

where T is the episode length and $\pi_t(a)$ represents the actor's stochastic distribution over the discrete action set $\mathcal{A}$. Under a uniform policy $\pi_t(a) = 1/|\mathcal{A}|$, entropy attains its maximum, indicating maximal exploration.

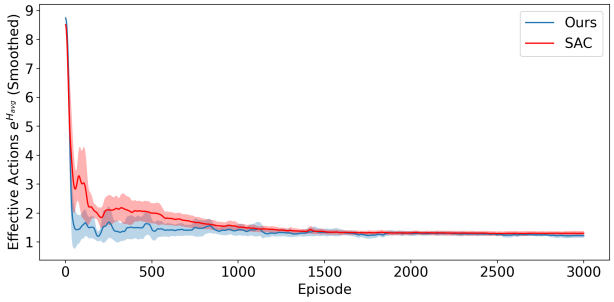
Figure 4.5 displays the exploration dynamics of the proposed method and vanilla SAC. Both algorithms begin with broad action distributions. SAC maintains higher entropy and a larger effective action count for longer, whereas the proposed method reduces entropy more rapidly as demonstration guidance biases its action distribution toward a smaller subset of actions.

To provide a more intuitive measure of exploration, we calculate the effective number of actions explored per episode by exponentiating the entropy:

$$\text{Effective actions} = e^{H_{\text{avg}}}. \tag{4.3}$$



(a)



(b)

Figure 4.5: Comparison of exploration dynamics using (a) policy entropy; (b) effective action count during training.

4.7 Generalization to Unseen Scenarios

We evaluate cross-scenario generalization by testing the learned controller under four trap configurations. Figure4.6 and Table 4.4 summarize the simulation settings that differentiate these scenarios. Scenarios with larger indices typically require the ego vehicle to manage larger longitudinal spacing before committing to an escape maneuver, due to the presence and placement of additional trap vehicles. Scenario 3 corresponds to the training configuration. Scenarios 1, 2, and 4 are unseen test configurations and therefore serve as out-of-distribution evaluations.

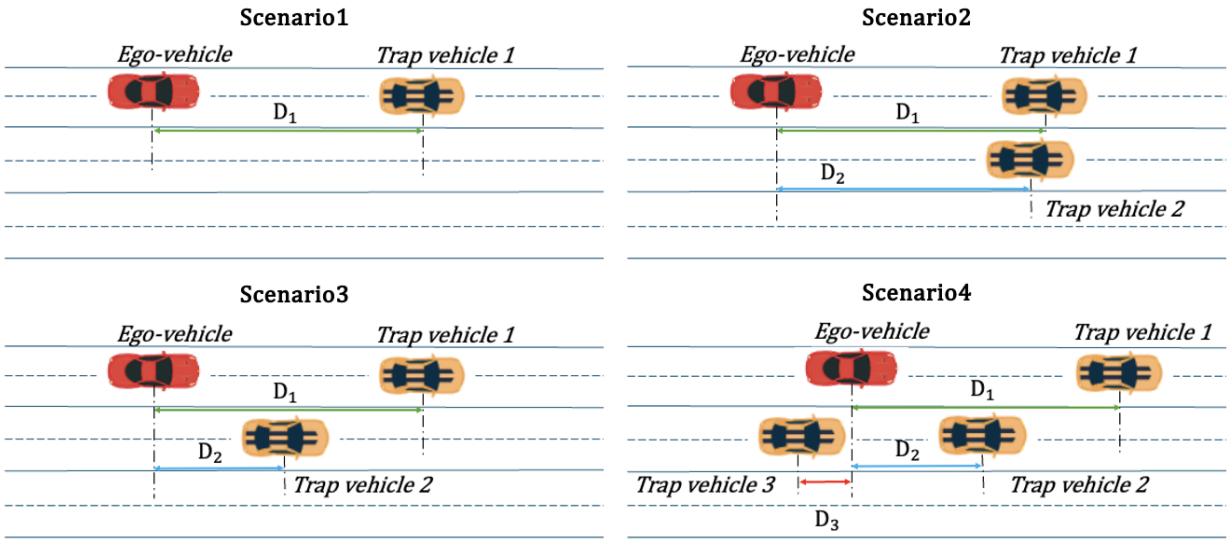


Figure 4.6: Our method is trained under the trap initialization shown in Scenario 3. To evaluate generalization to unseen conditions, we test on Scenarios 1, 2, and 4 with different trap initializations.

Table 4.5 shows that our method transfers well to unseen scenarios. In Scenario 1 and Scenario 2, our method achieves 100% overtaking success with zero accidents, while maintaining high reward and high average speed. These results indicate that the learned escape strategy is not overfitted to the training configuration and can generalize to different trap geometries. Scenario 4 is the most complicated unseen configuration in this suite in the sense that the ego vehicle typically needs larger safety buffers and more deliberate spacing management to escape. In this scenario, our method exhibits a reduced overtaking success rate. Despite the drop in completion rate, it still achieves the highest reward and distance traveled among all compared methods in Scenario 4. This suggests that the primary degradation manifests as incomplete overtaking rather than unsafe behavior.

Table 4.4: Simulation settings for four scenarios. D_i denotes the initial longitudinal distance between the ego vehicle and trap vehicle i.

Scenario	D_1 (m)	D_2 (m)	D_3 (m)
1	[14.80, 16.44]	–	–
2	[14.80, 16.44]	[14.65, 16.32]	–
3	[14.80, 16.44]	[1.13, 4.43]	–
4	[14.80, 16.44]	[1.13, 4.43]	[9.89, 13.88]

The baselines further clarify the generalization gap. The suboptimal demonstrator succeeds in the training configuration and the simpler unseen settings (Scenarios 1–3). Its limitation becomes evident in Scenario 4, where the overtaking success rate drops sharply and accidents emerge. IDM+MOBIL performs poorly as an ego policy across all scenarios, with 0% overtaking success rate and high accident rates, which is consistent with a controller designed for nominal traffic flow rather than adversarial trap-overtaking. SAC shows unstable behavior across scenarios and fails to achieve consistent overtaking success in the unseen configurations, with elevated accident or road excursion in the more restrictive settings.

Overall, Table 4.5 demonstrates that our method provides the best combination of task completion and safety across both seen and unseen trap configurations. The performance gap becomes most pronounced in unseen configurations that require explicit spacing management and longer-horizon overtaking planning, supporting the conclusion that bootstrapping from suboptimal demonstrations improves robustness to trap geometries beyond the training setup.

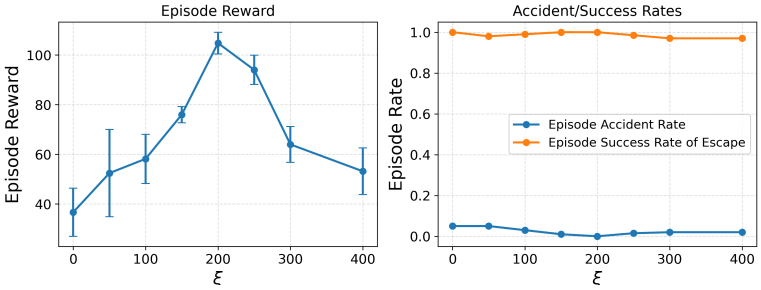
Table 4.5: Testing performance across four scenarios

Scen. index	Method	Reward	Velocity [m/s]	Distance [m]	Accident [%]	Overtake [%]	Road Excursion [%]	Min headway [m]	Min TTC [s]
1	Ours	102.3 ± 14.6	14.3 ± 0.4	2151.4 ± 64.7	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	15.1 ± 3.4	11.1 ± 4.5
	IDM+MOBIL	41.1 ± 31.8	12.4 ± 0.9	1682.6 ± 391.2	34.0 ± 47.9	0.0 ± 0.0	0.0 ± 0.0	23.5 ± 32.5	220.6 ± 629.2
	Suboptimal	28.1 ± 0.1	12.0 ± 0.0	1795.6 ± 0.6	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	15.7 ± 0.4	16.0 ± 0.5
	SAC	24.4 ± 15.5	11.4 ± 0.4	1378.7 ± 395.0	54.0 ± 50.4	42.0 ± 49.9	0.0 ± 0.0	12.6 ± 4.5	16.7 ± 17.6
2	Ours	109.0 ± 6.6	14.2 ± 0.2	2125.3 ± 22.5	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	16.0 ± 0.7	12.2 ± 2.1
	IDM+MOBIL	30.9 ± 30.2	12.2 ± 0.9	1563.9 ± 417.9	52.0 ± 50.5	0.0 ± 0.0	0.0 ± 0.0	13.3 ± 15.1	428.9 ± 1324.2
	Suboptimal	26.5 ± 0.6	11.9 ± 0.1	1788.1 ± 7.5	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.1 ± 0.8	13.2 ± 1.4
	SAC	29.9 ± 6.4	10.6 ± 0.3	1577.8 ± 68.6	2.0 ± 14.1	0.0 ± 0.0	0.0 ± 0.0	15.2 ± 2.8	10.0 ± 3.7
3	Ours	106.4 ± 2.6	13.7 ± 0.1	2060.5 ± 18.7	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.4 ± 1.6	8.5 ± 1.6
	IDM+MOBIL	34.2 ± 29.9	12.2 ± 0.9	1554.5 ± 491.0	38.0 ± 49.0	0.0 ± 0.0	0.0 ± 0.0	15.6 ± 19.2	203.3 ± 306.6
	Suboptimal	25.0 ± 0.5	11.5 ± 0.1	1731.8 ± 9.1	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.1 ± 1.0	13.3 ± 1.4
	SAC	25.7 ± 9.4	10.5 ± 0.3	1536.4 ± 125.2	12.0 ± 32.8	0.0 ± 0.0	2.0 ± 9.9	13.8 ± 3.3	7.2 ± 2.6
4	Ours	62.3 ± 19.2	12.0 ± 0.7	1805.9 ± 110.0	0.0 ± 0.0	66.7 ± 50.0	0.0 ± 0.0	15.1 ± 0.7	27.2 ± 35.0
	IDM+MOBIL	34.5 ± 33.1	12.2 ± 0.9	1570.3 ± 446.8	42.0 ± 49.9	0.0 ± 0.0	0.0 ± 0.0	20.6 ± 33.4	1079.6 ± 5275.5
	Suboptimal	21.4 ± 8.5	11.1 ± 0.3	1555.5 ± 355.5	10.0 ± 30.3	12.0 ± 32.8	0.0 ± 0.0	11.6 ± 2.6	11.8 ± 5.5
	SAC	21.2 ± 10.5	9.9 ± 0.6	1395.1 ± 342.9	18.0 ± 38.8	0.0 ± 0.0	4.0 ± 13.7	15.2 ± 2.5	7.9 ± 2.8

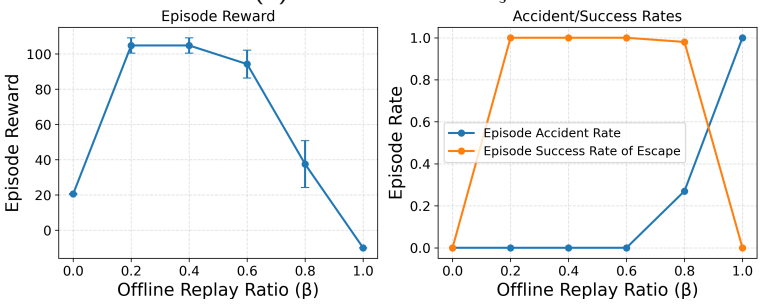
4.8 Ablation on Hyperparameters

As shown in Figure 4.7, we conduct an ablation study on several key hyperparameters in our bootstrapping framework. We keep the training and evaluation setup unchanged and report the mean and standard deviation over three random initialization seeds (1, 2, 3). These seeds affect both network initialization and environment randomness.

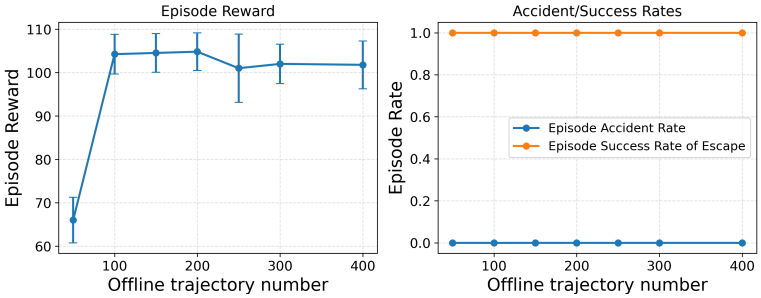
As shown in Figure 4.7a, we ablate the magnitude of the KL-divergence coefficient used for training initialization, ξ. Varying ξ changes the average testing reward return, but it has only a minor effect on the overtaking success rate and the accident rate. This observation is consistent with the component-wise ablation in Section 4.5. The offline replay buffer has a stronger influence on safety-related metrics, while the KL-based soft constraint mainly improves driving efficiency and yields a higher return.



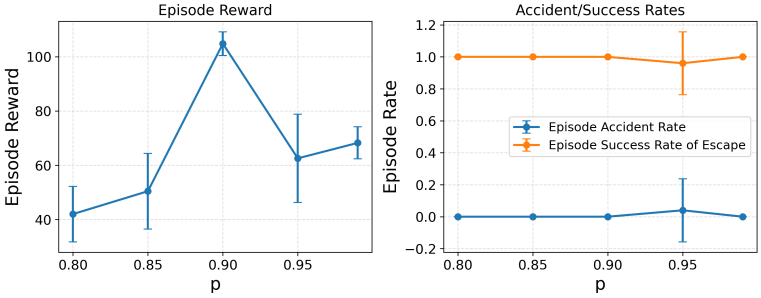
(a) Ablation across ξ



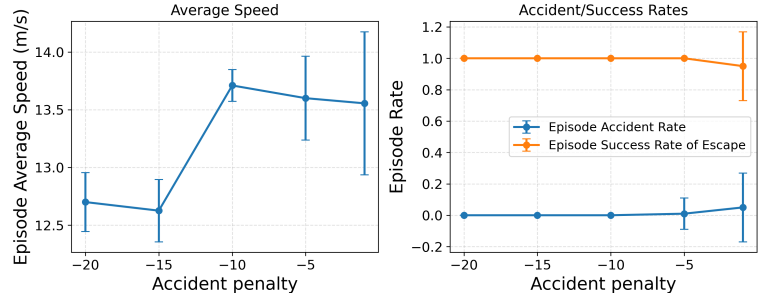
(b) Ablation across offline replay ratio β



(c) Ablation across number of offline trajectories



(d) Ablation across p



(e) Ablation across accident penalty value

Figure 4.7: Ablation study of hyperparameters.

As shown in Figure 4.7b, we ablate the initial offline replay ratio β, which controls the fraction of samples drawn from the offline buffer versus the online buffer during early training before the offline portion is gradually annealed. The results indicate that β should not be extreme. If β is too small, the method loses the intended bootstrapping effect and behaves similarly to vanilla online RL, which yields low reward and poor overtaking performance. If β is too large, learning is dominated by suboptimal demonstrations and robustness degrades, which is reflected by worse reward and sharply worse success and accident statistics. Intermediate β values achieve consistently strong rewards with high overtaking success rate and low accident rates.

As shown in Figure 4.7c, we vary the number of offline trajectories and observe that performance improves quickly when increasing the dataset from very small to moderate. This improvement occurs because more trajectories increase coverage over randomized initial traffic configurations and provide more reliable early guidance. After a moderate dataset size, the episode reward saturates and the safety metrics remain stable. This trend indicates diminishing returns once the offline data already provides sufficient behavioral diversity for bootstrapping.

As shown in Figure 4.7d, we study the stochasticity parameter p in the demonstrator distribution (Eq. (3.7)). In this construction, probability mass p is assigned to the rule-based action and the remaining mass is distributed over the other actions. The plot shows that a moderate value around $p = 0.9$ performs best. If p is too small, the demonstrator distribution becomes too diffuse and the KL-based guidance is weakened in the initial learning phase. If p is too large, the demonstrator becomes nearly deterministic and can over-constrain exploration or amplify suboptimal biases.

As shown in Figure 4.7e, we vary the accident penalty used in the terminal reward design. The plot shows a clear safety-efficiency trade-off. A more negative penalty encourages safer behavior but reduces average speed. A mild penalty makes the policy less risk-sensitive and increases the accident rate. A moderate penalty magnitude provides the best balance in our setting, because it preserves a high overtaking success rate with near-zero accidents while sustaining higher cruising speed. Overall, the best results occur when the demonstrator is confident but not deterministic.

4.9 Continuous Action-Space Extension

Our main experiments use a discrete 9-action interface with a 2-Hz policy decision rate. To address whether our approach can also support finer continuous control, we adapt the proposed Suboptimal Bootstrap framework to a continuous 5-Hz policy with a 15-Hz simulation rate. This experiment is not introduced as a new algorithm, rather it maintains the same core idea of using a suboptimal driving strategy as structured guidance, but changes the action interface, policy distribution, critic input, and KL regularization to match continuous SAC.

To construct this stochastic rule-based policy, the continuous rule-based action is mapped to the pre-squash action space by

$$\mu^{\text{rule,pre}}(s_t) = \operatorname{atanh}\left(\mu^{\text{rule}}(s_t)\right). \tag{4.4}$$

The continuous stochastic rule-based policy is then defined as

$$\tilde{\pi}_{\text{rule}}(a \mid s_t) = \mathcal{N}\left(\mu^{\text{rule,pre}}(s_t), \sigma^2_{\text{rule}} I\right), \tag{4.5}$$

where $\sigma_{\text{rule}} = 0.10$. This is the continuous counterpart of the discrete stochastic rule-based policy in Eq. (3.7), and has the same role of converting the deterministic suboptimal controller $\mu^{\text{rule}}(s)$ into a stochastic policy distribution for KL-based soft guidance. Here, $\operatorname{atanh}(\cdot)$ is applied elementwise

Table 4.6: Hyperparameter settings for the continuous action space and 5-Hz policy update experiments.

Symbol	Description	Value
γ	Discount factor	0.95
ξ_r	Action-deviation reward-penalty weight	0.05
σ_{rule}	Rule-based policy standard deviation	0.10

to map the bounded rule-based action to the pre-squash space, a denotes a pre-squash continuous action, and I is the identity matrix.

For the continuous 5-Hz experiment, we also apply an auxiliary reward-level action-deviation penalty,

$$\tilde{r}_t = r_t - \xi_r \frac{\left\| a_t^{\text{pre}} - \mu^{\text{rule,pre}}(s_t) \right\|_2^2}{2\sigma_{\text{rule}}^2}, \tag{4.6}$$

where a_t^{pre} is the pre-squash action sampled by the learned policy, and $\xi_r = 0.05$. This penalty is only used for the continuous 5-Hz experiment. This auxiliary penalty is introduced to stabilize the continuous-control training. In the 2-Hz discrete-action setting, the policy selects from a small finite action set, so exploration is strongly constrained. In the continuous-control setting, acceleration and steering are real-valued commands, which substantially enlarges the action search space and may lead to overly aggressive actions during early SAC training. The Euclidean action-deviation penalty provides a fine-grained regularization signal by measuring how far the learned action deviates from the suboptimal guiding action. It softly discourages large deviations and keeps early exploration within a physically reasonable region.

The hyperparameters that differ from the discrete RL setting in Table 2.4 are summarized in Table 4.6.

Table 4.7 shows that the adapted continuous 5-Hz Suboptimal Bootstrap controller retains strong performance. Compared with the original discrete 2-Hz setting, the continuous 5-Hz implementation shows more decisive escape behavior while maintaining safety. It achieves a higher average speed, a longer travel distance, and a larger minimum headway, while still maintaining 0.0% accidents, 100.0% overtaking, and 0.0% road excursions. The lower minimum TTC (5.9 s vs. 9.5 s) is therefore not an indication of collision-prone behavior; rather, together with the zero-accident and successful-overtaking results, it suggests a more active and decisive escape maneuver during the critical interaction window. These results support the extensibility of the bootstrapping principle to a higher-frequency continuous-action setting and show that the proposed bootstrapping principle is not limited to the discrete 2-Hz action abstraction. The continuous result should be interpreted as an adapted version of the same framework rather than a direct one-to-one comparison with the discrete experiments.

Table 4.7: Testing performance under continuous action space and 5-Hz decision policy updates.

Method	Reward	Velocity [m/s]	Distance [m]	Accident [%]	Overtake [%]	Road Excursion [%]	Min Headway [m]	Min TTC [s]
Ours (5 Hz, Continuous action space)	304.2 ± 17.1	14.2 ± 0.1	2134.9 ± 13.4	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	15.5 ± 0.7	5.9 ± 0.7
Ours (2 Hz, Discrete action space)	104.8 ± 4.3	13.7 ± 0.1	2056.5 ± 20.5	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	13.9 ± 0.9	9.5 ± 3.0
Backbone SAC (5 Hz, Continuous action space)	60.9 ± 9.3	10.9 ± 0.0	1634.1 ± 3.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	15.7 ± 0.4	34.8 ± 7.7

Chapter 5

Conclusion

In this report, we introduced a novel DRL-based autonomous driving framework specifically tailored to address a critical driving scenario. As opposed to traditional methods that rely on optimal control or large-scale expert demonstrations, our approach integrates a suboptimal controller that embeds human-like driving heuristics, effectively guiding the policy toward practical and efficient driving behaviors. The suboptimal controller is employed both as a soft constraint during the early stages of policy training and as an additional source of training samples. Experimental results consistently demonstrated that our framework outperforms standard RL algorithms. These findings highlight the practical benefits of incorporating structured, heuristic-based guidance to accelerate policy training, improve sample efficiency, and enhance autonomous driving performance.

Bibliography

[1] Z. Zhang, C. Peng, E. Yurtsever, and K. A. Redmill, "Bootstrapping reinforcement learning with sub-optimal policies for autonomous driving," *IEEE Open Journal of Intelligent Transportation Systems*, 2026. Accepted for publication.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[4] R. Schubert, "Evaluating the utility of driving: Toward automated decision making under uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 354–364, 2011.

[5] P. Varaiya and S. E. Shladover, "Sketch of an ivhs systems architecture," in *Vehicle Navigation and Information Systems Conference, 1991*, vol. 2, pp. 909–922, IEEE, 1991.

[6] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 509–522, 2011.

[7] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3884–3897, 2018.

[8] J. Rosenzweig and M. Bartl, "A review and analysis of literature on autonomous driving," *E-Journal Making-of Innovation*, pp. 1–57, 2015.

[9] J. Peng, S. Zhang, Y. Zhou, and Z. Li, "An integrated model for autonomous speed and lane change decision-making based on deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21848–21860, 2022.

[10] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, *et al.*, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 105–120, 2016.

[11] C. Hatipoglu, U. Ozguner, and K. A. Redmill, "Automated lane change controller design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 1, pp. 13–22, 2003.

[12] R. E. Chandler, R. Herman, and E. W. Montroll, "Traffic dynamics: studies in car following," *Operations Research*, vol. 6, no. 2, pp. 165–184, 1958.

[13] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[14] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[15] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[16] T. Han, J. Jing, and Ü. Özgüner, "Driving intention recognition and lane change prediction on the highway," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 957–962, IEEE, 2019.

[17] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[18] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

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

[20] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, *et al.*, "Urban driving with conditional imitation learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 251–257, IEEE, 2020.

[21] E. Yurtsever, L. Capito, K. Redmill, and U. Ozgune, "Integrating deep reinforcement learning with model-based path planners for automated driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1311–1316, IEEE, 2020.

[22] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254, IEEE, 2019.

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

[24] E. Sonu, Z. Sunberg, and M. J. Kochenderfer, "Exploiting hierarchy for scalable decision making in autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2203–2208, IEEE, 2018.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, pp. 1861–1870, Pmlr, 2018.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[27] Z. Zhang, E. Yurtsever, and K. A. Redmill, "Extensive exploration in highway overtaking scenarios using hierarchical reinforcement learning," *arXiv preprint arXiv:2501.14992*, 2025.

[28] J. Wang, Y. Wang, D. Zhang, Y. Yang, and R. Xiong, "Learning hierarchical behavior and motion planning for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2235–2242, IEEE, 2020.

[29] M. Masmoudi, H. Friji, H. Ghazzai, and Y. Massoud, "A reinforcement learning framework for video frame-based autonomous car-following," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 2, pp. 111–127, 2021.

[30] R. Valiente, B. Toghi, R. Pedarsani, and Y. P. Fallah, "Robustness and adaptability of reinforcement learning-based cooperative autonomous driving in mixed-autonomy traffic," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 397–410, 2022.

[31] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane change decision-making through deep reinforcement learning with rule-based constraints," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2019.

[32] A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano, "Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving," *Robotics and Autonomous Systems*, vol. 131, p. 103568, 2020.

[33] L. Wang, S. Yang, K. Yuan, Y. Huang, and H. Chen, "A combined reinforcement learning and model predictive control for car-following maneuver of autonomous vehicles," *Chinese Journal of Mechanical Engineering*, vol. 36, no. 1, p. 80, 2023.

[34] X. Zhang, J. Sun, Y. Wang, and J. Sun, "A hierarchical framework for multi-lane autonomous driving based on reinforcement learning," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 626–638, 2023.

[35] J. Lubars, H. Gupta, S. Chinchali, L. Li, A. Raja, R. Srikant, and X. Wu, "Combining reinforcement learning with model predictive control for on-ramp merging," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 942–947, IEEE, 2021.

[36] M. Moghadam and G. H. Elkaim, "A hierarchical architecture for sequential decision-making in autonomous driving using deep reinforcement learning," *arXiv preprint arXiv:1906.08464*, 2019.

[37] T. Shi, P. Wang, X. Cheng, C.-Y. Chan, and D. Huang, "Driving decision and control for automated lane change behavior based on deep reinforcement learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2895–2900, IEEE, 2019.

[38] J. Wu, Z. Huang, W. Huang, and C. Lv, "Prioritized experience-based reinforcement learning with human guidance for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 855–869, 2022.

[39] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7391–7403, 2022.

[40] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, *et al.*, "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7553–7560, IEEE, 2023.

[41] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 2326–2331, IEEE, 2019.

[42] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[43] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[44] B. Piot, M. Geist, and O. Pietquin, "Boosted bellman residual minimization handling expert demonstrations," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 549–564, Springer, 2014.

[45] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[46] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[47] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," in *International Conference on Learning Representations*, 2020.

[48] E. Leurent, "An environment for autonomous driving decision-making." `https://github.com/eleurent/highway-env`, 2018.

[49] M. Bain and C. Sammut, "A framework for behavioural cloning.," in *Machine Intelligence 15*, pp. 103–129, 1995.

[50] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, (Red Hook, NY, USA), Curran Associates Inc., 2021.

[51] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[52] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, pp. 2052–2062, 2019.

[53] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

[54] A. Nair, M. Dalal, A. Gupta, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," in *International Conference on Learning Representations*, 2021.

[55] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Robotics: Science and Systems XIV*, 2018.

[56] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2016.