2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

SAC(λ): Efficient Reinforcement Learning for Sparse-Reward Autonomous Car Racing using Imperfect Demonstrations

Heeseong Lee, Sungpyo Sagong, Minhyeong Lee, Jeongmin Lee, and Dongjun Lee

***Abstract*— Recent advances in Reinforcement Learning (RL) have demonstrated promising results in autonomous car racing. However, two fundamental challenges remain: sparse rewards, which hinder efficient learning process, and the quality of demonstrations, which directly affects the effectiveness of RL from Demonstration (RLfD) approaches. To address these issues, we propose SAC(λ), a novel RLfD algorithm tailored for sparse-reward racing tasks with imperfect demonstrations. SAC(λ) introduces two key components: (1) a discriminator-augmented Q-function, which integrates prior knowledge from demonstrations into value estimation while maintaining off-policy learning benefits, and (2) a Positive-Unlabeled (PU) learning framework with adaptive prior adjustment, which enables the agent to progressively refine its understanding of positive behaviors, while mitigating the overfitting problem. Through extensive experiments in the Assetto Corsa simulator, we demonstrate that SAC(λ) significantly accelerates training, surpasses the provided demonstrations, and achieves superior lap times over existing RL and RLfD approaches. Code and videos are available at `https://heesungsung.github.io/AC-RLRacer/`.**

I. Introduction

Autonomous car racing presents formidable challenges in robotics due to several key factors. First, vehicles frequently encounter tire slip during high-speed driving and varying race conditions, leading to highly nonlinear behaviors. Second, strategic maneuvers are required to navigate around opponents and through high-curvature sections of the track. The primary goal of racing is to achieve the fastest lap times and ultimately win the race, necessitating precise and extreme control actions at the vehicle's physical limits. Recent studies have investigated autonomous racing across diverse platforms, including full-scale vehicles [1], [2], small-scale models [3], [4], and simulated environments [5], [6].

Traditional approaches [7], [8] decompose the racing task into distinct modules such as perception, planning, and control. Although these methods have achieved meaningful results, they face two major challenges: the complex nonlinear dynamics required for high-performance racing make real-time computation difficult, and inaccuracies in any module can degrade overall system performance.

This work is supported in part by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant code RS-2021-KA162182) and in part by the Technology Innovation Program (20024355 and 1415187329, Development of autonomous driving connectivity technology based on sensor-infrastructure cooperation) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea).

The authors are with the Department of Mechanical Engineering, Institute of Advanced Machines and Design (IAMD) and Institute of Engineering Research (IOER), Seoul National University, Seoul 08826, South Korea. Corresponding author: Dongjun Lee. {heesungsung, sjjh1123, minhyeong, ljmlgh, djlee}@snu.ac.kr

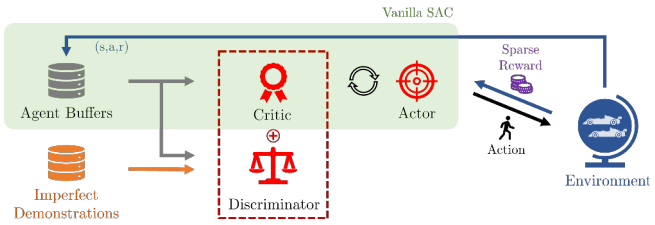


Fig. 1: Overview of SAC(λ). Based on the original SAC algorithm (green shaded box), additional value from the discriminator is augmented to the original Q-function (red dotted box). The objective of discriminator is formulated as a positive-unlabeled learning problem, and is learned using given low-quality and narrow-distributed demonstrations.

Learning-based approaches have emerged as an alternative, processing high-dimensional sensory inputs to produce low-level control commands using the powerful representational capabilities of deep neural networks. One approach is Imitation Learning (IL) [9], [10], which extracts driving expertise from high-quality demonstrations. However, this method often lacks generalization and is limited by the availability of expert data, as well as its inherent performance ceiling. Another approach is Reinforcement Learning (RL) [3], [11], which enables agents to learn from their own experience through self-exploration, leading to better generalization. However, RL faces challenges with exploration efficiency, particularly in complex autonomous racing scenarios where proper handling is crucial for success, while high-fidelity simulators such as Gran Turismo Sport and Assetto Corsa further exacerbate this issue due to their inherently slow sample collection process. While previous works attempted to address these issues through parallelized data collection on multiple machines or adoption of rich per-step rewards, these approaches either introduce significant computational overhead or bias the agent's behavior toward predefined paths [6], [12], [13].

To address these limitations, RL from Demonstration (RLfD) methods [14], [15] have been proposed to guide exploration effectively by integrating key elements from both IL and RL. While these approaches improve the sample efficiency by incorporating expert demonstrations, they still suffer in sparse-reward settings, where the effectiveness of learning is constrained by the quality and the diversity of demonstrations. Most available demonstrations in autonomous racing are suboptimal and exhibit limited variety, making it difficult for the agent to generalize beyond the given dataset. Nevertheless, when effectively leveraged, im-

perfect demonstrations offer a cost-effective and scalable way to provide structured priors for exploration, while still allowing the agent to surpass their limitations. However, existing RLfD approaches often fail to balance the contributions of priors from demonstrations and online exploration, leading to conservative policies that struggle to outperform the provided demonstrations [16].

To tackle these challenges, we propose SAC(λ), a novel RLfD algorithm that effectively incorporates imperfect demonstrations into off-policy learning, as shown in Fig. 1, thereby improving learning efficiency and enhancing the agent's performance in sparse-reward autonomous racing. Our main idea can be summarized into the following two novelties. 1) We integrate the prior knowledge from demonstrations into agent's value estimation via an augmented Q-function using a discriminator. This improves sample efficiency in our task, which is characterized by slow sample collection and sparse rewards, without requiring dense reward shaping. 2) We employ a Positive-Unlabeled (PU) learning [17] framework with a prior adjustment strategy for the discriminator. This mechanism continuously evaluates the quality of agent's experiences, refining the set of positive demonstrations over time and enabling the agent to surpass the performance of the given imperfect demonstrations. Through extensive experiments in a high-fidelity racing simulator, we demonstrate that SAC(λ) significantly accelerates training and achieves fastest lap times compared to existing RL and RLfD approaches.

II. Related Work

A. RL for Autonomous Racing

RL has demonstrated state-of-the-art performance in learning-based frameworks for autonomous racing across various agile robotic tasks [5], [18], [19]. RL enables agents to learn policies guided by reward signals, aiming to maximize the sum of future discounted rewards through interactions with the environment. Model-free RL methods [5], [11] directly optimize parameterized policies using sampled trajectories, making them well-suited for racing environments. However, sparse-reward formulations remain an open challenge, as infrequent feedback often leads to inefficient exploration and slow policy convergence [20]. While dense rewards can accelerate early-stage learning, they often bias agents toward heuristic-driven behaviors, whereas effectively utilized sparse rewards promote long-term strategic decision-making, leading to more generalized and high-performance policies. To the best of our knowledge, our approach presented in this paper is the first work to explicitly formulate autonomous car racing as a sparse-reward problem. Previous studies [6], [21] have primarily relied on dense reward shaping, often leading to agents that track predefined paths rather than allowing them to discover optimal racing strategies. Despite the inherent difficulties of sparse rewards, our approach successfully overcomes these challenges without relying on complex reward shaping.

Various simulators have been used to evaluate RL-based autonomous racing strategies. CARLA [22], commonly employed in autonomous driving research, has been used to develop controllers such as the drift controller in [23] using Soft Actor-Critic (SAC) [24]. Racing-specific simulators, such as F1TENTH [4] and DeepRacer [25] for small-scale vehicles, as well as TORCS [26] and AirSim [27] for full-scale vehicles, provide useful benchmarks but suffer from limited physics accuracy and insufficient photorealism. To address these limitations, [12] used the Gran Turismo Sport developed by Sony AI to construct a highly realistic RL environment and train learning-based controllers. However, the environment of [12] remains closed-source, restricting accessibility to the broader research community. In this paper, we use Assetto Corsa, which provides high-fidelity vehicle dynamics, realistic rendering, and open API access [28]. These features enable faithful simulation of real-world racing conditions. However, a major limitation is its slow sample collection speed, as it does not support time acceleration and parallel execution—a characteristic that closely resembles real-world scenarios. This shortcoming necessitates the development of RL algorithms that enhance learning efficiency from limited interactions.

B. Online RL with Offline Demonstrations

Leveraging expert demonstrations to initialize and guide policy learning has been extensively studied in IL, inverse RL, offline RL, and online RL in sparse-reward settings. Recent studies have shown that adversarial approaches effectively combine the principles of both IL and RL by aligning the agent's visitation distribution with expert priors [29]–[31]. In these methods, the discriminator generates reward signals based on the similarity between the agent's behavior and the expert data, which are defined as per-step rewards. Specifically, offline demonstrations are considered positive samples, while data collected by the online agent are typically treated as negative samples. Thus, these approaches heavily rely on the quality of demonstrations and often fail to discover policies that surpass the given priors. To address this limitation, [32] introduced prior-matching scores from the discriminator as auxiliary rewards, leveraging not only priors from demonstrations but also rich online interactions from the environment. However, since online learning in this method is conducted with an on-policy algorithm, it lacks sample efficiency and often fails to maintain performance in long-horizon tasks. Moreover, it still results in a conservative policy, as the fixed positive demonstrations hinder exploration throughout the entire learning process. Recent studies have explored an alternative approach by training discriminators using a PU-learning objective within IL frameworks, treating data collected by the agent as unlabeled rather than strictly negative samples [33], [34]. Inspired by this paradigm, our proposed algorithm adopts the PU-learning discriminator into an actor-critic framework. By augmenting the Q-function with discriminator outputs, our approach effectively leverages imperfect demonstrations while maintaining off-policy capabilities, thereby facilitating efficient policy exploration and refinement.

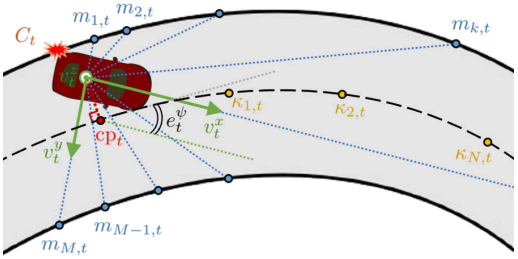


Fig. 2: A subset of observation components used as network inputs. All values are locally obtainable from onboard sensors. Prior knowledge of track boundaries and a 2D rangefinder is required. cp_t denotes the nearest centerline point at time t.

III. Problem Definition

In this section, we first define the task formulation, then specify the characteristics of the demonstrations, and finally describe the observations and actions.

A. Sparse-Reward Autonomous Racing

The primary objective of our work is to develop a real-time controller that autonomously navigates a race car while minimizing lap time. Unlike prior works [6], [12] that rely on dense rewards, which often require extensive trial and error for proper shaping, we define the task using a sparse-reward formulation. This formulation encourages the agent to discover optimal driving strategies solely based on long-term outcomes, as no intermediate heuristics or trajectory-based guidance is imposed.

Our reward function $r_t = \sum_i k_i r_{i,t}$ does not require any prior knowledge about the optimal racing line and each term is defined as:

1) **checkpoint reward**: $r_{1,t} = +1$ is given whenever the car passes through one of the pre-defined N checkpoints,
2) **lap-time reward**: $r_{2,t} = t_{\text{standard}} - t$ is given only upon lap completion, incentivizing faster lap times,
3) **off-track penalty**: $r_{3,t} = -1$ is given whenever more than two tires leave the track boundaries,

where $k_1 = 3$, $k_2 = 1$, $k_3 = 5$, $N = 10$, and t_{standard} is a predefined reference lap time used for reward scaling.

B. Imperfect Demonstrations

To mitigate the challenges of sparse rewards, we incorporate demonstrations as an auxiliary learning resource. Instead of relying on high-quality expert demonstrations, we collect imperfect demonstrations using a simple Model Predictive Control (MPC) policy based on a kinematic bicycle model. This controller is designed to track a predefined reference trajectory rather than optimize lap times, inherently limiting the diversity of collected data. As a result, the dataset exhibits a narrow distribution, primarily reflecting the characteristics of the reference trajectory, and fails to sufficiently capture advanced racing strategies such as optimal braking points, throttle modulation, and aggressive cornering techniques. Each dataset is collected in about 30 minutes per track.

These imperfect demonstrations provide a practical alternative to expert demonstrations by significantly reducing cost while enabling scalable data collection. Unlike human-collected demonstrations, which require extensive driving expertise and substantial resources, our method uses a rule-based controller to automate the data generation process. This approach not only reduces the cost but also facilitates the efficient collection across various tracks and initial conditions.

C. Observations and Actions

Observations. The observations at time step t are represented as $\mathbf{o}_t = [\mathbf{v}_t, \dot{\mathbf{v}}_t, e_t^\psi, C_t, \mathbf{m}_t, \boldsymbol{\kappa}_t, \delta_{t-1}]$, where $\mathbf{v}_t \in \mathbb{R}^3$ and $\dot{\mathbf{v}}_t \in \mathbb{R}^3$ denote the vehicle's linear velocity and acceleration along the body axes, $e_t^\psi \in (-\pi, \pi]$ denotes the yaw error relative to the tangent vector of the nearest centerline waypoint, $C_t \in \{0, 1\}$ indicates wall contact, $\mathbf{m}_t \in \mathbb{R}^M$ denotes distance measurements from M rangefinder rays, $\boldsymbol{\kappa}_t \in \mathbb{R}^N$ denotes the curvature values sampled from N waypoints ahead on the track, and $\delta_{t-1} \in [-1, 1]$ denotes the steering command from the previous step. A subset of the observations is shown in Fig. 2 and all components represent locally available information that can be collected using onboard sensors.

Actions. The action at time step t is represented as $\mathbf{a}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \boldsymbol{\xi}, \boldsymbol{\xi} \sim \mathcal{N}(0, I)$, where $\boldsymbol{\mu}_t = [\mu_t^\tau, \mu_t^\delta] \in \mathbb{R}^2$ and $\boldsymbol{\sigma}_t = [\sigma_t^\tau, \sigma_t^\delta] \in \mathbb{R}^2$, with τ and δ denote throttle-brake and steering command respectively, both ranging from -1 to 1. The reparameterization trick is employed, enabling the model to learn both deterministic and stochastic action distributions. Sampled actions $\hat{\mathbf{a}}_t$ promote exploration during the training stage, and only the $\boldsymbol{\mu}_t$ values are used during inference.

IV. SAC(λ) Algorithm

SAC has demonstrated strong performance in autonomous racing tasks due to its ability to handle continuous action spaces and its off-policy capabilities, which significantly improves sample efficiency [35]. However, despite these advantages, SAC still struggles with efficient exploration in long-horizon and sparse-reward tasks due to the lack of structured guidance, making it difficult to discover optimal racing strategies solely through reward signals.

To address these limitations, we propose SAC(λ), an RLfD algorithm that effectively incorporates imperfect demonstrations into the online learning scheme. Our approach introduces two key components: (1) a discriminator-augmented Q-function and (2) a Positive-Unlabeled (PU) learning framework with adaptive prior adjustment. The following subsections describe the two components in detail.

A. Discriminator-Augmented Q-Function

SAC(λ) is built on SAC, aiming to effectively integrate knowledge from both online interactions and offline demonstrations, with λ controlling the balance between these two sources of information, while maintaining the off-policy learning benefits. The algorithm employs a discriminator to generate additional value estimates, which are then used to

form augmented Q-functions. We assume access to imperfect demonstrations $\mathcal{D} = \{\tau_i\}$, where each trajectory τ_i consists of transition tuples $(\mathbf{o}, \mathbf{a}, r, \mathbf{o}', d)$, without direct access to the demonstration policy π_E. Here, d and $\mathbf{o}'$ represent the termination signal and next observations, respectively.

We begin with the SAC minimization objective, incorporating an auxiliary policy guidance term to encourage alignment between the agent policy π_θ and the demonstration policy π_E, measured by the Jensen-Shannon (JS) divergence:

$$\begin{aligned}\mathbb{E}_{\mathbf{o}\sim\mathcal{B}}\left[\alpha \log \pi_\theta(\hat{\mathbf{a}}_\theta(\mathbf{o})|\mathbf{o}) - \min_i Q_{\phi_i}(\mathbf{o}, \hat{\mathbf{a}}_\theta(\mathbf{o}))\right] \\ + \lambda \mathbb{D}_{\text{JS}}(\pi_E, \pi_\theta)\end{aligned} \tag{1}$$

where $\mathcal{B}$ is the agent's replay buffer, α is the temperature parameter, Q_{ϕ_i} is the value network, and $\hat{\mathbf{a}}_\theta(\mathbf{o}) = \tanh(\boldsymbol{\mu}_\theta(\mathbf{o}) + \boldsymbol{\sigma}_\theta(\mathbf{o}) \odot \boldsymbol{\xi})$, $\boldsymbol{\xi} \sim \mathcal{N}(0, I)$. For brevity, we denote $\hat{\mathbf{a}}_\theta(\mathbf{o})$ as $\hat{\mathbf{a}}_\theta$ in the following discussion.

Since π_E is unknown, (1) is intractable. Instead, we enforce policy guidance through observation-action occupancy measure matching objective, leveraging the important property that the occupancy measure uniquely determines the policy [36]:

$$\begin{aligned}\mathbb{E}_{\mathbf{o}\sim\mathcal{B}}\left[\alpha \log \pi_\theta(\hat{\mathbf{a}}_\theta|\mathbf{o}) - \min_i Q_{\phi_i}(\mathbf{o}, \hat{\mathbf{a}}_\theta)\right] \\ + \lambda \mathbb{D}_{\text{JS}}\left(\rho_{\pi_E}(\mathbf{o}, \mathbf{a}), \rho_{\pi_\theta}(\mathbf{o}, \mathbf{a})\right)\end{aligned} \tag{2}$$

where $\rho_\pi(\mathbf{o}) = \sum_{t=0}^{\infty} \gamma^t \Pr(\mathbf{o}_t = \mathbf{o}|\pi)$ is the observation occupancy measure, and $\rho_\pi(\mathbf{o}, \mathbf{a}) = \rho_\pi(\mathbf{o})\pi(\mathbf{a}|\mathbf{o})$ is the observation-action occupancy measure.

Rather than directly optimizing the Jensen–Shannon (JS) divergence involving the unnormalized distribution ρ_π, we instead optimize its variational lower bound, given by:

$$\begin{aligned}\mathbb{D}_{\text{JS}}(\rho_{\pi_E}, \rho_{\pi_\theta}) \geq \sup_D \big(\mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{D}}\left[\log(D(\mathbf{o}, \mathbf{a}))\right] \\ + \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{B}}\left[\log(1 - D(\mathbf{o}, \mathbf{a}))\right]\big)\end{aligned} \tag{3}$$

where $U(\mathbf{o}, \mathbf{a}) : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ and $D(\mathbf{o}, \mathbf{a}) = \frac{1}{1+e^{-U(\mathbf{o},\mathbf{a})}} : \mathcal{O} \times \mathcal{A} \to (0, 1)$ are arbitrary mapping functions (*Theorem 2* of [32]). A suitable choice for D is an optimal binary classifier, i.e., a discriminator, which differentiates the agent policy from the expert policy based on observation-action pairs sampled from the replay buffer $\mathcal{B}$ and $\mathcal{D}$, respectively.

Substituting (3) into (2) with the discriminator network D parameterized by w, we obtain the policy optimization objective of SAC(λ):

$$\begin{aligned}\min_\theta \max_w \mathcal{L}_\pi =& \mathbb{E}_{\mathbf{o}\sim\mathcal{B}}\left[\alpha \log \pi_\theta(\hat{\mathbf{a}}_\theta|\mathbf{o}) - \min_i \tilde{Q}_{\phi_i}(\mathbf{o}, \hat{\mathbf{a}}_\theta)\right] \\ &+ \lambda \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{D}}\left[\log(D_w(\mathbf{o}, \mathbf{a}))\right]\end{aligned} \tag{4}$$

where the augmented Q-function is defined as $\tilde{Q}_{\phi_i}(\mathbf{o}, \hat{\mathbf{a}}_\theta) = Q_{\phi_i}(\mathbf{o}, \hat{\mathbf{a}}_\theta) - \lambda \log(1 - D_w(\mathbf{o}, \hat{\mathbf{a}}_\theta))$, with the objective of discriminator being slightly modified to evaluate agent's actions with the current policy. A noteworthy aspect of this formulation is its effectiveness in capturing the latent structure of the positive dataset by the discriminator, which extracts informative representations that encode expert behaviors [37]. Moreover, the auxiliary policy guidance term is now embedded within the augmented Q-function $\tilde{Q}_{\phi_i}$, allowing demonstration information to be seamlessly integrated into the environment reward while dynamically adjusting the contribution of prior knowledge via λ. A well-calibrated λ ensures that the agent effectively leverages demonstration knowledge while maintaining sufficient exploration to refine its policy through RL.

B. PU Learning with Adaptive Prior

A key challenge in leveraging imperfect demonstrations is ensuring that the agent does not become overly constrained by suboptimal behaviors present in the initial dataset. Traditional discriminators are trained with a Positive-Negative (PN) classification objective, where demonstrations are identified as positive samples and the agent's experiences as negative samples [29]. However, this approach inherently limits the agent's performance ceiling to the quality of the initial dataset, as it lacks a mechanism to incorporate better experiences over time.

To overcome this limitation, we leverage the PU learning framework [17], allowing the discriminator to progressively refine its understanding of the positive dataset. The agent's experiences are now treated as unlabeled samples, framing the problem as a semi-supervised learning and enabling a continuous expansion of the positive demonstrations. The refined discriminator maximization objective is formulated as:

$$\begin{aligned}\mathcal{L}_D = &\eta \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{D}}\left[\log(D_w(\mathbf{o}, \mathbf{a}, \log \pi_\theta(\mathbf{a}|\mathbf{o})))\right] \\ &+ \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{B}}\left[\log(1 - D_w(\mathbf{o}, \mathbf{a}, \log \pi_\theta(\mathbf{a}|\mathbf{o})))\right] \\ &- \eta \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{D}}\left[\log(1 - D_w(\mathbf{o}, \mathbf{a}, \log \pi_\theta(\mathbf{a}|\mathbf{o})))\right]\end{aligned} \tag{5}$$

where η is the positive class prior, which is assumed to be known, and the term $\log \pi_\theta$ is included as an input to the discriminator to facilitate better class separation [38].

Since the agent continually interacts with the environment, its collected data contain not only negative samples but also potentially positive experiences that exceed the quality of the initial demonstrations. To incorporate these high-quality experiences, we introduce an adaptive prior adjustment, where η is continuously updated based on the agent's performance. Specifically, we temporarily defer episode replays and assign labels based on lap times. The labeled data are then evaluated against predefined thresholds to quantitatively determine their positiveness. This process allows the positive experience set to evolve over time, preventing the agent from being constrained by the suboptimality of the initial demonstrations. By progressively refining the discriminator through this adaptive PU learning process, our method mitigates overfitting to suboptimal demonstrations, enabling the agent to surpass the performance of the given dataset.

C. Practical Implementation

For practical implementation, the training loop of SAC(λ) sequentially repeats the following three key steps:

1) **Discriminator learning**: Perform a gradient ascent step on w using (5) with ϕ_i and θ fixed.

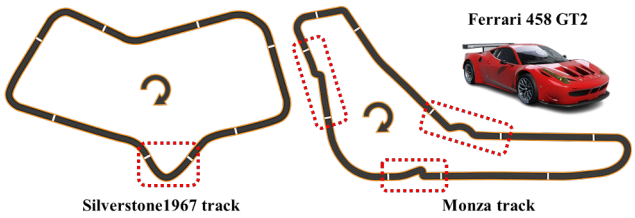


Fig. 3: We use the Ferrari 458 GT2 as our racing car and evaluate on two tracks: Silverstone1967 (S67) and Monza (MNZ). The red dotted boxes indicate sections requiring advanced driving strategies.

2) **Critic learning**: Perform a gradient descent step on ϕ_i using following loss with w and θ fixed:

$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{B}}\left[\tilde{Q}_{\phi_i}(\mathbf{o},\mathbf{a}) - \tilde{y}(r,\mathbf{o}',d)\right]^2.$$

Here, the target function is defined as $\tilde{y}(r,\mathbf{o}',d) = r + \gamma(1-d)[\min_{i=1,2}\tilde{Q}_{\bar{\phi}_i}(\mathbf{o}',\hat{\mathbf{a}}'_\theta) - \alpha\log\pi_\theta(\hat{\mathbf{a}}'|\mathbf{o}')]$, where $\hat{\mathbf{a}}'_\theta = \hat{\mathbf{a}}_\theta(\mathbf{o}')$ and ϕ_i denote parameters of target Q-networks.

3) **Actor learning**: Perform a gradient descent step on θ using following loss with w and ϕ_i fixed:

$$\mathcal{L}_\pi = \mathbb{E}_{\mathcal{B}}\left[\alpha\log\pi_\theta(\hat{\mathbf{a}}_\theta|\mathbf{o}) - \min_{i=1,2}\tilde{Q}_{\phi_i}(\mathbf{o},\hat{\mathbf{a}}_\theta)\right]$$

See Appendix B for the pseudocode of our algorithm.

V. Experiment

In this section, we describe the experimental setup and present the results to address the following questions:

- **Training Efficiency**: How effectively does SAC(λ) accelerate the early stages of learning?
- **Final Performance**: Does SAC(λ) achieve superior performance after sufficient training steps?
- **Learned Behavior**: Is the behavior learned by the agent qualitatively comparable to that of a human expert driver?
- **Effect of** λ: How does the choice of λ affect the entire learning process?

A. Setup

We develop our own interfacing platform based on Assetto Corsa, adopting a similar architecture to [6], but extending it by incorporating a random initial spawn function to improve robustness. Control inputs are given at 25Hz. Each demonstration consists of 20 episodes, collected using the random spawn function.

All experiments are conducted in single-player mode under rolling start conditions, using the Ferrari 458 GT2 model on two tracks: Silverstone1967 (S67) and Monza (MNZ). As shown in Fig. 3, S67 consists mainl y of simple corners, while MNZ features chicanes and esses, which demand advanced driving strategies. All experiments are run on a single desktop equipped with an Intel i9 CPU and a NVIDIA RTX 4090 GPU.

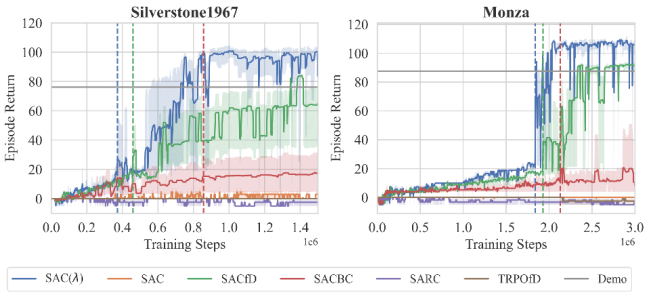
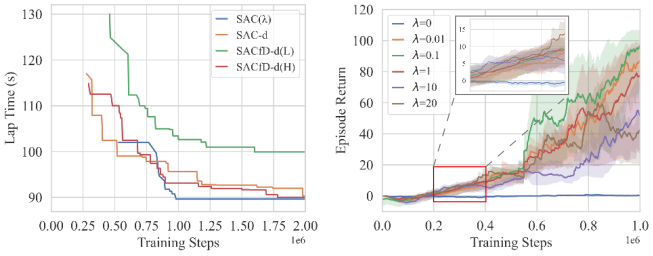


Fig. 4: Evaluation results of training efficiency for each algorithm on the S67 (left) and MNZ (right) tracks. The graphs show episode returns over training steps during the early training stages. Vertical dotted lines indicate the step at which the first lap is completed, and the horizontal gray line represents the average return of the demonstrations.



(a) (b)

Fig. 5: Evaluation results on the S67 track: (a) comparison of best lap times with algorithms trained using dense rewards, and (b) ablation study on the effect of different λ values on the learning performance of SAC(λ).

B. Baselines

We compare SAC(λ) against the following baselines:

- **SAC** [24]: A well-known model-free, off-policy RL algorithm. We use a basic implementation of SAC, where the suffix "-fD" indicates buffer initialization with offline datasets, and "-d" indicates dense rewards.
- **SACBC**: SAC augmented with an auxiliary Behavior Cloning (BC) loss, implemented as part of the objective in (2).
- **SARC** [31]: A discriminator-aided Actor-Critic RL algorithm where rewards are densely generated by evaluating the similarity between the online agent's policy and offline datasets.
- **TRPOfD**: A modified version of the Trust Region Policy Optimization (TRPO) algorithm, inspired by [32]. This variant leverages reshaped reward function, which augments the environment reward with demonstration information.

For dense rewards, we define $r_t = \|v_t\| \cdot (1 - \beta e_t^y)$, where e_t^y is L2 distance error from the reference trajectory, and β is a constant coefficient that controls the trade-off between tracking accuracy and deviation (higher β allows the agent to deviate more) [6].

TABLE I: Evaluation results comparing final performance with criteria of lap times (s).

Track	Demo	SAC(λ)	SAC	SACfD	SACBC	SARC	TRPOfD	SAC-d	SACfD-d(L)	SACfD-d(H)
S67	117.03±3.18	**90.47±0.82**	fail	99.17±1.69	125.10±21.43	fail	fail	91.40±0.77	100.85±0.94	90.72±0.71
MNZ	148.28±5.82	**116.20±1.29**	fail	125.14±2.89	145.13±15.26	fail	fail	121.72±1.13	125.41±1.24	117.57±0.87

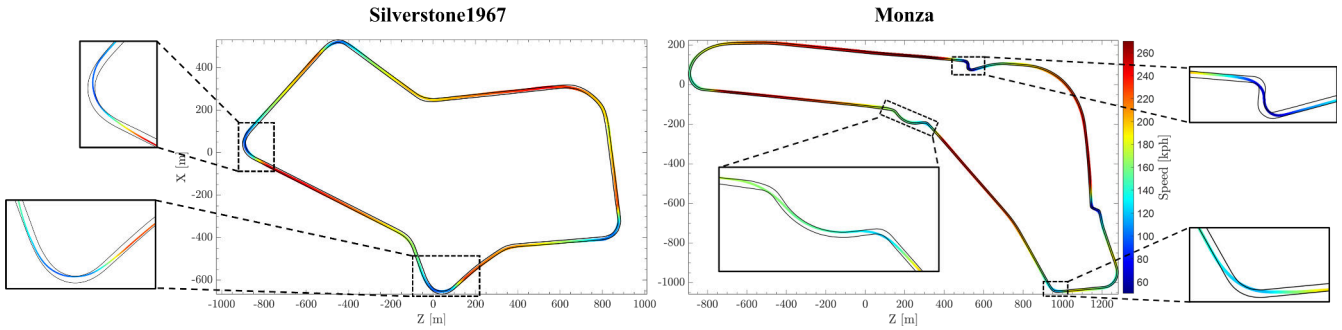


Fig. 6: Speed profile of the SAC(λ) agent along the tracks. Challenging sections are highlighted to provide a detailed visualization of the trajectory, where the agent employs specific strategies to achieve the minimum lap time.

C. Results

1) Training Efficiency: Fig. 4 shows the episode returns during the initial stages of training. Training efficiency is evaluated based on two criteria: (1) the number of training steps required for the agent to complete its first lap and (2) the lap time at that episode. Vertical dotted lines in the figure represent the average number of steps required for the first lap completion.

Among all algorithms, SAC(λ) demonstrates the fastest learning speed, completing the first lap in fewer steps and achieving the highest returns, indicating faster lap times. SACfD and SACBC also successfully complete a lap but fall short of SAC(λ) in both episode returns and lap times. In contrast, the naive SAC algorithm fails to learn in a sparse-reward environment, highlighting its inability to handle such challenging scenarios. Similarly, SARC, an IL-based algorithm, struggles to deal with out-of-distribution (OOD) data as it does not leverage online interaction information. TRPOfD also fails to complete a lap, becoming trapped in a local minimum due to its lack of a replay buffer, a limitation that leads to policy collapse [39], as demonstrated in the videos on our project page. These results collectively highlight that SAC(λ) outperforms other algorithms in both learning speed and lap time performance in the initial stages.

2) Final Performance: Table I shows a comparison of lap time statistics for the best-performing model of each algorithm on the S67 and MNZ tracks. Each model was evaluated over 20 laps to assess its performance. For demonstrations, the table lists the average lap time across the provided dataset.

SAC(λ) achieves the fastest lap times, surpassing the performance of the given demonstrations by maintaining higher speeds across all track sections. Specifically, it records 89.65s on the S67 track and 114.91s on the MNZ track. This success stems from its ability to effectively leverage online interaction rewards and its generalization to OOD state-action pairs. In contrast, SACfD and SACBC show degraded performance due to their inability to effectively integrate online and offline information. Notably, SACBC is hindered by the BC loss on the MNZ track, which limits the agent's ability to improve in areas requiring skillful driving.

Compared to algorithms trained with dense rewards, SAC(λ) still demonstrates superior performance, as shown in Fig. 5a. The SACfD-d algorithm is evaluated using two types of demonstrations: a low-quality version (L), corresponding to the imperfect demonstrations used in our main experiments, and a high-quality version (H), collected from an expert-level policy. While dense reward-based algorithms, SAC-d and SACfD-d(H), achieve relatively competitive lap times, their performance is constrained by their reliance on a path-tracking behavior. This method assumes that the provided reference path is perfectly optimal for both the car and the track environment, which is often not the case. As a result, strict adherence to these suboptimal reference paths limits long-term planning capabilities. Furthermore, SACfD-d(L) exhibits slower convergence and shows consistently worse performances in the later stages of training.

3) Learned Behavior: Fig. 6 shows the agent's trajectory for achieving the minimum lap time, with challenging sections of varying curvatures selected for detailed visualization. The agent effectively uses the full width of the track to minimize the curvature of its path and maximize speed, following an "out-in-out" trajectory—a technique commonly used by expert human drivers. By learning this optimal strategy, the agent attains higher speeds through these sections, resulting in improved overall performance. Visit our project page for more detailed videos.

4) Effect of λ: We analyze the impact of different λ values on the overall learning performance, as shown in Fig. 5b. When $\lambda = 0$, equivalent to naive SAC, the algorithm fails to learn racing strategies. As λ increases, the algorithm begins to leverage offline demonstrations more effectively, reaching peak performance at $\lambda = 0.1$. Beyond this point, while

the initial learning speed improves, performance starts to decline in the later stages due to the stronger influence of IL, which limits the agent's ability to explore and adapt. At $\lambda = 20$, excessive reliance on demonstrations hinders the agent's learning, causing it to become trapped in a local minimum.

VI. Conclusion

To address the limitations of existing RLfD methods for sparse-reward autonomous racing, we proposed SAC(λ), a novel RLfD algorithm that enhances policy learning. By augmenting the Q-function with auxiliary value estimates generated by a discriminator, SAC(λ) incorporates structured prior knowledge from given demonstrations. Furthermore, our adaptive PU learning framework continuously refines the positive demonstration set, allowing the use of low-quality and narrowly distributed demonstrations by effectively utilizing progressively collected online experiences from the agent. Experimental results demonstrate that our method outperforms existing RL and RLfD approaches in both training efficiency and final lap time performance, even surpassing the provided demonstrations. SAC(λ) provides an efficient solution for tasks characterized by sparse rewards, slow sample generation, and limited demonstration quality, making it broadly applicable beyond autonomous racing.

Future work can extend SAC(λ) in several directions. First, adaptive scheduling of λ during training could optimize the balance between demonstration reliance and autonomous exploration. Second, incorporating multimodal sensor inputs, such as vision-based observations, could improve generalization beyond simulated racing. Third, applying this framework to multi-agent racing would enable more sophisticated decision-making in competitive environments. Finally, evaluating the framework on real-world autonomous racing platforms would be an effective way to demonstrate its strength in learning efficiently from limited interactions.

Appendix

A. Implementation Details

Observations and Rewards We adopt a 2D rangefinder with a maximum measurement range of 100m. In the vehicle's body frame, the sensor is positioned on the xy-plane, and its Region of Interest (ROI) is defined by angles measured with respect to the z-axis, spanning from -100deg to 100deg. A total of 21 rays are uniformly distributed within this range. For preview curvatures, 30 points are sampled along the centerline ahead of the vehicle at a constant interval of 2m. Regarding the reward structure, N is set to 10, and t_{standard} is set to 160s and 200s for the S67 and MNZ tracks, respectively.

Network Specifications We use Multi-Layer Perceptron (MLP) models to construct the entire network, including the actor, critics, and discriminator. The hidden dimensions for the actor and critics are $[2048, 2048, 1024, 512]$, while the discriminator has hidden dimensions of $[1024, 512, 128, 64]$. ReLU is used as the activation function, with Tanh applied only to the last layer. This setup is shared across all baselines.

Training Hyperparameters All hyperparameters are listed in Table II. We use the same parameters for SAC(λ) and other baselines whenever possible. Parameters specific to the on-policy TRPOfD algorithm are highlighted in gray. The replay buffer is initialized with offline demonstrations and gradually overwritten as it reaches capacity.

TABLE II: List of hyperparameters used for the training.

Parameter	Value
Replay buffer size	1e6
Batch size	1,024
Rollout length	4,000
Polyak	0.995
α	0.01
γ	1.00
λ	0.1
Damping coefficient	0.01
Number of iterations of conjugate gradient	50
Max. number of steps allowed in backtracking	10
Coefficient for backtracking search	0.8
KL divergence limit	0.03
Coefficient for GAE	0.97
Learning rate	1e-3 (Actor) 1e-3 (Critic) 1e-4 (Discriminator) 1e-4 (α)
Update frequency (steps)	2,000
Number of updates	300 (Actor) 300 (Critic) 100 (Discriminator)

B. Pseudocode of SAC(λ)

Algorithm 1: SAC(λ)

Initialize parameters w, θ, ϕ_i, $\bar{\phi}_i$, η.
Initialize empty replay buffer $\mathcal{B}$, $\tilde{\mathcal{B}}$.
Initialize demonstrations $\mathcal{D}$.
repeat
  Observe o and execute $a \sim \pi_\theta(\cdot|o)$.
  $\tilde{\mathcal{B}} \leftarrow \tilde{\mathcal{B}} \cup (o, a, r, o', d)$
  if o' is terminal **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup (\tilde{\mathcal{B}}, \text{lap time})$
    $\tilde{\mathcal{B}} \leftarrow \varnothing$
    Reset environment.
  end
  if it's time to update **then**
    for each gradient step **do**
      Random sample a batch from $\mathcal{B}$ and $\mathcal{D}$.
      $w \leftarrow w + \nu_d \nabla_w \mathcal{L}_D$
      $\phi_i \leftarrow \phi_i - \nu_Q \nabla_{\phi_i} \mathcal{L}_Q$
      $\theta \leftarrow \theta - \nu_\pi \nabla_\theta \mathcal{L}_\pi$
      $\bar{\phi}_i \leftarrow \rho \bar{\phi}_i + (1-\rho)\phi_i$
      Calculate and update η.
    end
  end
until convergence;

References

[1] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[2] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, *et al.*, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[3] A. Brunnbauer, L. Berducci, A. Brandstätter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Latent imagination facilitates zero-shot transfer in autonomous racing," in *2022 international conference on robotics and automation (ICRA)*. IEEE, 2022, pp. 7513–7520.

[4] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.

[5] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[6] A. Remonda, N. Hansen, A. Raji, N. Musiu, M. Bertogna, E. Veas, and X. Wang, "A simulation benchmark for autonomous racing with large-scale human data," *arXiv preprint arXiv:2407.16680*, 2024.

[7] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 2397–2403.

[8] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[9] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020.

[10] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[11] E. Chisari, A. Liniger, A. Rupenyan, L. Van Gool, and J. Lygeros, "Learning from simulation, racing in reality," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8046–8052.

[12] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.

[13] B. D. Evans, H. A. Engelbrecht, and H. W. Jordaan, "High-speed autonomous racing using trajectory-aided deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2023.

[14] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7262–7269, 2021.

[15] C. V. Samak, T. V. Samak, and S. Kandhasamy, "Autonomous racing using a hybrid imitation-reinforcement learning architecture," *arXiv preprint arXiv:2110.05437*, 2021.

[16] H. Yang, C. Yu, S. Chen, *et al.*, "Hybrid policy optimization from imperfect demonstrations," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[17] D. Xu and M. Denil, "Positive-unlabeled reward learning," in *Conference on Robot Learning*. PMLR, 2021, pp. 205–219.

[18] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[19] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[21] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2070–2075.

[22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[23] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1247–1254, 2020.

[24] 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*. PMLR, 2018, pp. 1861–1870.

[25] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, *et al.*, "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 2746–2754.

[26] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[27] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[28] S. H. de Frutos and M. Castro, "Assessing sim racing software for low-cost driving simulator to road geometric research," *Transportation research procedia*, vol. 58, pp. 575–582, 2021.

[29] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[30] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Hk4fpoA5Km

[31] A. Deka, C. Liu, and K. P. Sycara, "Arc-actor residual critic for adversarial imitation learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1446–1456.

[32] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International conference on machine learning*. PMLR, 2018, pp. 2469–2478.

[33] Q. Wang, R. McCarthy, D. C. Bulens, K. McGuinness, N. E. O'Connor, F. R. Sanchez, N. Gürtler, F. Widmaier, and S. J. Redmond, "Improving behavioural cloning with positive unlabeled learning," in *Conference on robot learning*. PMLR, 2023, pp. 3851–3869.

[34] Y. Wang, B. Du, and C. Xu, "Unlabeled imperfect demonstrations in adversarial imitation learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 8, 2023, pp. 10 262–10 270.

[35] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 9403–9409.

[36] U. Syed, M. Bowling, and R. E. Schapire, "Apprenticeship learning using linear programming," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1032–1039.

[37] X. Mao, Z. Su, P. S. Tan, J. K. Chow, and Y.-H. Wang, "Is discriminator a good feature extractor?" *arXiv preprint arXiv:1912.00789*, 2019.

[38] H. Xu, X. Zhan, H. Yin, and H. Qin, "Discriminator-weighted offline imitation learning from suboptimal demonstrations," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24 725–24 742.

[39] S. Dohare, Q. Lan, and A. R. Mahmood, "Overcoming policy collapse in deep reinforcement learning," in *Sixteenth European Workshop on Reinforcement Learning*, 2023.