# Efficient Reinforcement Learning for Autonomous Car Racing with Imperfect Demonstrations

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

***Abstract*— Recent advances in Reinforcement Learning (RL) have shown that end-to-end controllers can achieve promising results in autonomous racing, effectively handling highly nonlinear dynamics and extreme action constraints. However, these systems often suffer from exploration inefficiencies, particularly in sparse reward settings such as lap completion tasks. Learning from Demonstrations (LfD) methods have emerged as a potential solution, leveraging expert demonstrations to guide the learning process. In this work, we propose a class of Discriminator-Augmented Q-function (DAQ) RL algorithms that effectively utilize imperfect demonstrations to guide exploration by enforcing occupancy measure matching. We conduct experiments using Assetto Corsa, a widely recognized simulator for its realistic modeling of car dynamics. The evaluation results show that DAQ aided Soft Actor-Critic (SAC) accelerates learning, achieves better final lap times than existing methods, and even outperforms the given demonstrations.**

## I. INTRODUCTION

Autonomous car racing presents a significant challenge in robotics due to three key factors. First, vehicles often experience slip and over/under-steer during high-speed maneuvers, resulting in complex nonlinear dynamics. Second, vehicle behavior changes rapidly in response to variables such as tire wear, fuel consumption, and varying race conditions. Third, strategic maneuvers are required to navigate around opponents and through high-curvature sections of the track. The primary objective in racing is to achieve the fastest lap times and ultimately win the race, necessitating precise and extreme control actions at the vehicle's dynamic limits. Recent research has explored autonomous racing in various contexts, 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. These methods face two major challenges: the complex nonlinear dynamics required for high-performance racing make real-time computations difficult, and inaccuracies in any module can degrade the overall system performance.

End-to-end learning has emerged as an alternative, processing high-dimensional sensory inputs to produce low-level control commands using the powerful representational capabilities of deep neural networks. In the early stages, Imitation Learning (IL) algorithms [9], [10] are employed to extract driving expertise from high-quality demonstrations.

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, djlee}@snu.ac.kr

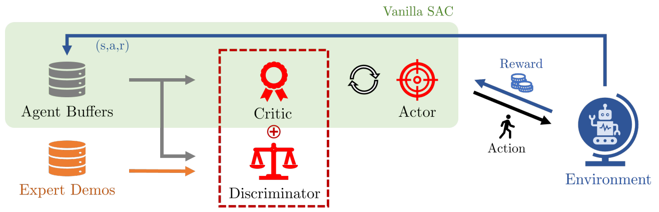

Fig. 1. Overview of DAQ-SAC. Based on the original SAC algorithm, additional values from the discriminator is augmented to the Q-function. Discriminator is learned through a positive-unlabeled setup, and an experience tagging technique is employed to efficiently handle positive replays.

However, this method often lacks generalization and is limited by the availability of expert data, as well as its inherent performance ceiling.

To overcome the limitations of IL, Reinforcement Learning (RL) [3], [11] allows agents to learn from their own experience through self-exploration, leading to better generalization. In contrast, RL faces challenges with exploration efficiency, especially in racing tasks characterized by a vast observation space and infrequent complex scenarios. Reinforcement Learning from Demonstration (RLfD) [12], [13] has been proposed to address these challenges by using demonstrations to guide exploration more effectively, integrating key elements from both IL and RL.

In this work, we address autonomous car racing in Assetto Corsa (AC), a simulator known for its realistic vehicle dynamics. Although AC provides valuable insights into extreme vehicle control, it has limitations, such as the inability to fast-forward or replicate environments, which results in low sample efficiency. Additionally, the racing task requires long and precise action sequences to complete a lap, making it challenging for the agent to find an optimal solution. To overcome these challenges, we propose a novel RLfD algorithm called Discriminator-Augmented Q-value aided Soft Actor-Critic (DAQ-SAC). This algorithm augments the Q-function with signals from a discriminator, as shown in Fig. 1, preserving off-policy capabilities and allowing learning with low-quality demonstrations. Our experimental results demonstrate that DAQ-SAC not only achieves superior training efficiency in sparse reward settings but also outperforms existing methods in terms of final lap time performance, even surpassing the provided demonstrations.

## II. RELATED WORK

Previous end-to-end frameworks for car racing primarily fall into two categories: Reinforcement Learning (RL) and Imitation Learning (IL). RL learns policies guided by rewards, aiming to maximize the sum of future discounted rewards through interactions with the environment. Nevertheless, successful training requires extensive exploration, which can be problematic in real-world scenarios due to sample inefficiency and the risk of damaging physical agents when applying suboptimal policies. As a result, RL is typically restricted to simulations, especially in safety-critical tasks such as autonomous driving.

Model-free RL methods, as seen in [11], [14], learn parameterized policies directly from sampled trajectories. These studies demonstrate the viability of model-free RL in realistic racing games. However, the inherent stochastic exploration process can limit its effectiveness in transferring to real-world environments. Alternatively, model-based RL methods [3] reduce sample complexity by learning a world model to inform decision-making. Despite this advantage, model-based approaches require an additional model learning phase and the planning performance heavily depends on the accuracy of the learned model.

IL learns policies by directly mapping observations to control actions using labeled data from expert demonstrations or classical frameworks. Behavior Cloning (BC) [15] is trained to maximize the likelihood of the expert's actions in a supervised manner but suffers from error accumulation. Generative Adversarial Imitation Learning (GAIL) [16] introduces a more sophisticated approach by training two adversarial models: a discriminator and a generator. Studies such as [10] and [17] have adopted IL algorithms to develop end-to-end self-driving controllers. Although distilling expert knowledge into a general policy can speed up training, it tends to lack generalization and its performance is inherently limited by the quality and size of the dataset.

RLfD methods combine the strengths of both RL and IL. These approaches assume access to both reward signals ($r$) and demonstrations ($\mathcal{D}$), using demonstrations to improve sample efficiency and facilitate exploration in environments with sparse rewards. One common approach is to initialize the agent's replay buffer with demonstrations [18], [19], ensuring that demonstrations are not under-exploited when they outperform the agent's own experiences. A second approach involves optimizing a combination of RL and IL objectives by introducing an auxiliary imitation loss term [20], [21]. A third approach is Adversarial Imitation Learning (AIL) aided RL [22], where the agent learns from rewards evaluated by a discriminator, effectively transferring expert features to the agent. However, these methods rely on mixed losses or rewards, which can hinder learning in later stages and usually necessitate on-policy algorithms.

Our algorithm aims to integrate AIL-based learning methods into an off-policy framework to ensure sample efficiency and stability, while extracting the benefits of all three RLfD approaches. First, we prioritize positive experiences by initializing the replay buffer with demonstrations. Second, our method is grounded in a mixture of the original Soft Actor-Critic (SAC) algorithm [23] and occupancy matching objectives. Unlike conventional approaches, DAQ-SAC is not hindered by the imitation objective because the positive experience set improves as learning progresses. Finally, we leverage environmental rewards by augmenting the Q-function with signals from the discriminator, ensuring that the agent benefits from both expert demonstrations and interaction with the environment.

## III. METHODOLOGY

Our main objective is to design a controller that can autonomously navigate a race car without prior knowledge of the dynamics while minimizing lap time. The racing task presents following key challenges: 1) it requires a long and precise sequence of actions to successfully complete an entire lap, and 2) rewards are given sparsely, typically using a terminal reward structure where feedback is provided only at the end of each episode. Additionally, our environment cannot be replicated or fast-forwarded, resulting in a low sample collection speed.

In this section, we first define the environment as a Markov Decision Process (MDP) and then introduce our novel algorithm, Discriminator-Augmented Q-value aided Soft Actor-Critic (DAQ-SAC).

### *A. Reward Function*

Previous studies [5], [14] use a per-step progress reward defined as $r_t = (cp_t - cp_{t-1})$, where $cp_t$ represents the normalized track progress along the centerline at step $t$. This dense reward provides direct guidance to the agent's trajectory, which can potentially limit the ability for long-term planning. Despite its limitations, this structure is commonly used because agents typically struggle to learn how to race with sparse rewards and may fail to complete the task.

In contrast, our algorithm can effectively handle sparse rewards, so we design a semi-sparse reward to avoid the formation of sub-goals and the risk of overfitting, denoted as $r_t = \sum_{i=1}^{5} \lambda_i r_{t,i}$. Each term is defined as follows:

1) **Track progress reward** : $r_{t,1} = +1$ whenever the car passes through one of the predefined $N$ checkpoints.
2) **Time penalty** : $r_{t,2} = -1$ for every step, encouraging the agent to complete the lap as quickly as possible.
3) **Tire-off-track penalty** :
$$r_{t,3} = \begin{cases} -10 & \text{if numTiresOffTrack} > 2, \\ -1 & \text{elseif numTiresOffTrack} > 0. \end{cases}$$
This penalty ensures the agent to stay within track boundaries. Generally, up to two tires are allowed to be outside the boundaries; exceeding this results in a disqualification.
4) **Collision penalty** : $r_{t,4} = -1$ whenever a collision occurs.
5) **Under-pace penalty** : $r_{t,5} = -1$ whenever the speed falls below a certain threshold. This penalty prevents low-speed driving, especially during the initial stages of learning.

In practical applications, the program does not run at the exact specified frequency, resulting in slight variations in step intervals. The per-step progress rewards are sensitive to these variations, which can lead to inconsistent rewards for the same observation-action pairs. However, our semi-sparse

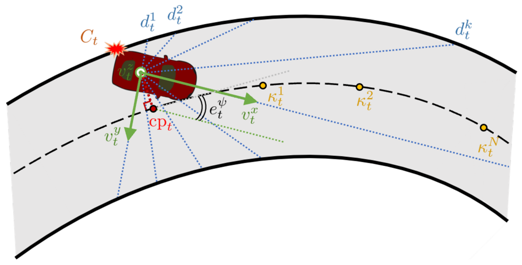

Fig. 2. A subset of the observations fed to the networks. All components are selected to ensure they are locally collected, enabling sim-to-real transfer. Prior knowledge of track boundaries and adoption of 2D rangefinder are required.

reward is given only when passing checkpoints with fixed progress, resulting in more consistent and stable learning. Furthermore, previous approaches aim to minimize lap time by maximizing track progress rewards, which is heavily dependent on a discount factor, whereas our approach minimizes the time penalty and provides a fixed positive reward upon lap completion, offering a more intuitive solution.

## B. Network Specifications

DAQ-SAC utilizes four deep neural networks, consisting of two Q-functions, one policy function, and one discriminator. In the following, observation and action fed to the networks are explained in detail.

**Observation** The observation at time step $t$ is represented as $\mathbf{o}_t = [\mathbf{v}_t, \dot{\mathbf{v}}_t, e_t^\psi, C_t, \mathbf{d}_t, \boldsymbol{\kappa}_t, \delta_{t-1}]$. A subset of the observations is shown in Fig. 2. Specifically, $\mathbf{v}_t \in \mathbb{R}^3$ 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 nearest centerline waypoint, $C_t \in \{0, 1\}$ indicates wall contact, $\mathbf{d}_t \in \mathbb{R}^M$ measures the distance for each $M$ rangefinder rays, $\boldsymbol{\kappa}_t \in \mathbb{R}^N$ denotes the curvature values sampled from $N$ waypoints ahead on the track, and $\delta_{t-1} \in [-1, 1]$ records the steering command from the previous step. All components of the observation are local information, which can be collected via sensors, facilitating sim-to-real transfer.

**Action** The action at time step $t$ is represented as $\mathbf{a}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \cdot \epsilon, \epsilon \sim \mathcal{N}(0, 1)$. The reparameterization trick is used, allowing the learning of both deterministic and stochastic elements. Specifically, $\boldsymbol{\mu}_t = [\mu_t^\tau, \mu_t^\delta] \in \mathbb{R}^2$ and $\boldsymbol{\sigma}_t = [\sigma_t^\tau, \sigma_t^\delta] \in \mathbb{R}^2$, where $\tau$ and $\delta$ denote throttle-brake and steering command respectively, both ranging from -1 to 1. Sampled actions promote exploration during the training stage, and only the $\mu$ values used during inference.

## C. DAQ-SAC Algorithm

It has been proven that SAC achieves the best performance in our problem settings [24]. Based on this, we design DAQ aided SAC with two key ideas. First, the discriminator is adapted to generate additional values, which are then used to augment the Q-function. Second, the discriminator is trained through a positive-unlabeled learning setup, enabling the continuous improvement of positive experience sets.

**Algorithm 1:** DAQ-SAC

```
Initialize parameters $w$, $\theta$, $\phi_i$, $\bar{\phi}_i$, $\eta$.
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 $\mathcal{D}$.
            $w \leftarrow w + \nu_d \hat{\nabla}_w \mathcal{L}_d$
            $\phi_i \leftarrow \phi_i - \nu_Q \hat{\nabla}_{\phi_i} \mathcal{L}_Q$
            $\theta \leftarrow \theta - \nu_\pi \hat{\nabla}_\theta \mathcal{L}_\pi$
            $\bar{\phi}_i \leftarrow \rho \bar{\phi}_i + (1 - \rho)\phi_i$
            Calculate and update $\eta$.
        end
    end
until convergence;
```

We begin the derivation with the learning objective of the policy in SAC, augmented with an auxiliary policy guidance term. The Jensen-Shannon (JS) divergence is selected as the discrepancy measure between the expert's policy and the agent's policy,

$$\mathbb{E}_{\pi_\theta}\left[\alpha \log \pi_\theta(\tilde{a}_\theta|o) - \min_{i=1,2} Q_{\phi_i}(o, \tilde{a}_\theta)\right] + \lambda_1 \mathbb{D}_{\text{JS}}(\pi_E, \pi_\theta). \tag{1}$$

It is assumed that only imperfect expert demonstrations $\mathcal{D} = \{\tau_i\}$ are available, where each trajectory $\tau_i$ consists of transition pairs $(o, a, r, o', d)$, without access to the expert policy $\pi_E$. Based on this setting, (1) cannot be used in practice.

Alternatively, we adopt an approach that treats policy guidance as enforcing occupancy measure matching between the current policy and the demonstrations, where the observation occupancy measure $\rho_\pi(o) : \mathcal{S} \to \mathbb{R}$ is defined as $\rho_\pi(o) = \sum_{t=0}^{\infty} \gamma^t P(o_t = o|\pi)$. The observation-action occupancy measure is then written as $\rho_\pi(o, a) = \rho_\pi(o)\pi(a|o)$ and the expected reward over policy $\pi$ is expressed as $\mathbb{E}_\pi[r(o, a)] = \sum_{o,a} \rho_\pi(o, a) r(o, a)$. An important property here is that the occupancy measure uniquely specifies the policy [25]. The original objective in (1) is now rewritten as

$$\mathbb{E}_{\rho_\theta}\left[\alpha \log \pi_\theta(\tilde{a}_\theta|o) - \min_{i=1,2} Q_{\phi_i}(o, \tilde{a}_\theta)\right] + \lambda_1 \mathbb{D}_{\text{JS}}(\rho_E, \rho_\theta). \tag{2}$$

Instead of performing difficult direct optimization, we consider the lower bound of the JS divergence, given by

$$\sup_{D \in (0,1)} \left(\mathbb{E}_{\rho_E}\left[\log(D(o, a))\right] + \mathbb{E}_{\rho_\theta}\left[\log(1 - D(o, a))\right]\right), \tag{3}$$

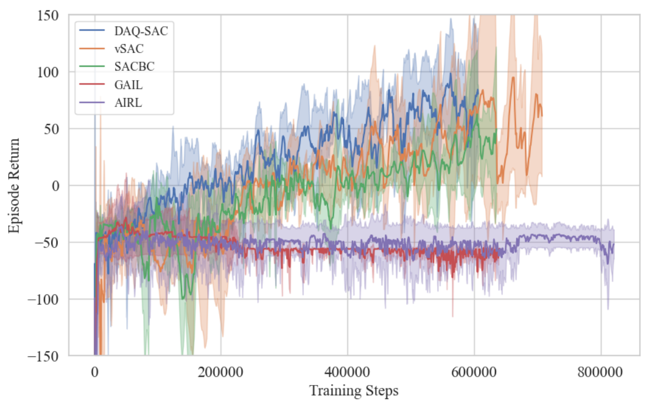

Fig. 3. Evaluation results comparing training efficiency. The graph shows the episode return over training steps until the first lap completion. DAQ-SAC demonstrates the best performance in terms of the fastest learning speed and highest return.

where $D(o,a) = \frac{1}{1+e^{-U(o,a)}} : \mathcal{O} \times \mathcal{A} \rightarrow (0,1)$ is an arbitrary mapping function. It can be treated as an optimal binary classifier which distinguishes the current agent's policy from the expert policy with respect to the observation-action pairs sampled from each buffer $\mathcal{B}$ and $\mathcal{D}$, respectively. The objective of the policy is then formulated by substituting (3) into (2), with the discriminator parameterized by $w$,

$$\begin{aligned}\min_{\theta}\max_{w}\mathcal{L}_{\pi} =& \mathbb{E}_{\mathcal{B}}\left[\alpha \log \pi_{\theta}(\tilde{a}_{\theta}|o) - \min_{i=1,2} Q'_{\phi_i}(o,\tilde{a}_{\theta})\right] \\ &+ \lambda_1 \mathbb{E}_{\mathcal{D}}\left[\log(D_w(o,\tilde{a}_{\theta}))\right]. \end{aligned} \tag{4}$$

Here, $Q'_{\phi_i}(o,\tilde{a}_{\theta}) = Q_{\phi_i}(o,\tilde{a}_{\theta}) - \lambda_1 \log(1 - D_w(o,\tilde{a}_{\theta}))$. The first key idea stems from this formulation, combining IL and RL in an adversarial manner. A noteworthy aspect here is that the auxiliary policy guidance term is now viewed as an augmented $Q$-function, $Q'_{\phi_i}$, which incorporates demonstration information into the environment reward. Then, target function is computed as $y'(r,o',d) = r + \gamma(1-d)[\min_{i=1,2} Q'_{\bar{\phi}_i}(o',\tilde{a}') - \alpha \log \pi_{\theta}(\tilde{a}'|o')]$.

The second key idea is to incorporate positive-unlabeled (PU) learning into the discriminator's training process. At the initial stage, demonstrations are treated as positive replays, while the agent's experiences are treated as unlabeled. The objective is to identify negative samples among the unlabeled experiences, framed as a semi-supervised problem. By leveraging this approach, the better trajectories in the agent's buffer can be treated as expert demonstrations, leading to an improvement of the demonstration set. As a result, we can solve the overfitting problem that GAIL suffers from, and the agent can outperform the given imperfect expert's ability. The refined discriminator objective is given as follows:

$$\begin{aligned}\max_{w} \eta \mathbb{E}_{\mathcal{D}}&\left[\log(D_w(o,a,\log\pi_{\theta}(a|o)))\right] \\ &+ \mathbb{E}_{\mathcal{B}}\left[\log(1 - D_w(o,a,\log\pi_{\theta}(a|o)))\right] \\ &- \eta \mathbb{E}_{\mathcal{D}}\left[\log(1 - D_w(s,a,\log\pi_{\theta}(a|o)))\right].\end{aligned}$$

Here, the signal $\log \pi_{\theta}$ is added into the input of the discriminator, to mitigate the difficulty of distinguishing between classes by only relying on the observation-action pairs. Unlike standard PU learning, the positive class prior $\eta$ changes dynamically as learning progresses. We compute $\eta$ directly by employing an experience tagging technique: during each episode, the agent's replays are temporarily suspended instead of being stored in the buffer. At the end of the episode, all replays are tagged with the lap time, then stored in the buffer in a single batch.

Finally, our algorithm, DAQ-SAC is proposed by combining the aforementioned key ideas into SAC. The objective (4) is divided into three steps for the practical implementation :

1) **Discriminator learning**: Perform a gradient ascent step on $w$, with following loss with $\phi_i$ and $\theta$ fixed.
$$\begin{aligned}\mathcal{L}_d =& \eta \mathbb{E}_{\mathcal{D}}\left[\log(D_w(o,a,\log\pi_{\theta}(a|o)))\right] \\ &+ \mathbb{E}_{\mathcal{B}}\left[\log(1 - D_w(o,a,\log\pi_{\theta}(a|o)))\right] \\ &- \eta \mathbb{E}_{\mathcal{D}}\left[\log(1 - D_w(o,a,\log\pi_{\theta}(a|o)))\right]\end{aligned}$$
2) **Critic learning**: Perform a gradient descent step on $\phi_i$ with following loss with $w$ and $\theta$ fixed.
$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{B}}\left[Q'_{\phi_i}(o,a) - y'(r,o',d)\right]^2$$
3) **Actor learning**: Perform a gradient descent step on $\theta$ with following loss with $w$ and $\phi_i$ fixed.
$$\mathcal{L}_{\pi} = \mathbb{E}_{\mathcal{B}}\left[\alpha \log \pi_{\theta}(\tilde{a}(o)|o) - \min_{i=1,2} Q'_{\phi_i}(o,\tilde{a}(o))\right]$$

TABLE I

EVALUATION RESULTS OF BEST PERFORMANCE.

| | **Demostration** | **DAQ-SAC** | **vSAC** | **SACBC** |
|---|---|---|---|---|
| **Lap time** | 1:37:330 | **1:29:767** | 1:39:624 | 1:38:539 |

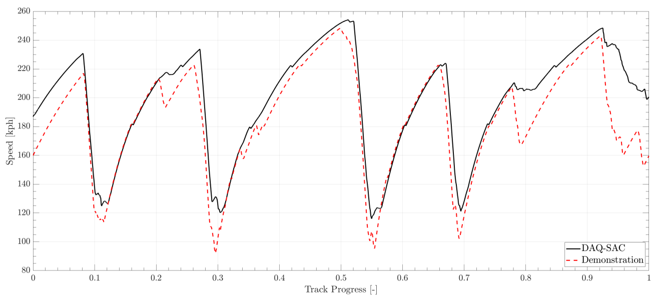

Fig. 4. Speed comparison between the best profile of the DAQ-SAC agent and the best profile of demonstrations. The race is conducted under rolling start conditions.

## IV. EXPERIMENT

In this section, we describe the structure of our interfacing framework used to create the virtual RL environment. Then, we explain the experimental setup. Finally, we present the experimental results to address the following questions:

- **Training Efficiency**: How much does DAQ-SAC boost the initial learning stage?
- **Final Performance**: Does DAQ-SAC achieve the best performance after sufficient training steps?
- **Learned Behavior**: Is the agent's learned racing behavior comparable to that of a expert human driver?

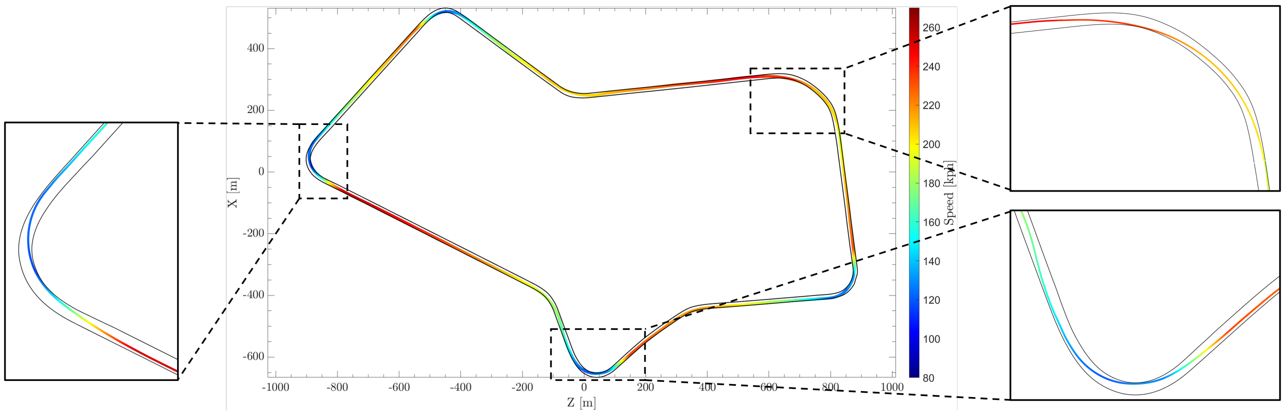

Fig. 5. Speed profile of the DAQ-SAC agent along the track. Three corners with different curvatures are selected to visualize the trajectory where the agent employs specific out-in-out strategies to achieve the minimum lap time.

### A. Experimental Setup

**Settings** The experiment is conducted in rolling start, single-player mode on the Silverstone 1967 track using a full-scale Ferrari 458 GT2 model. In-game settings include automatic gear shifting, clutch, traction control, stability control, and automatic braking system are turned on.

**Overview** We conduct two types of experiments. First, we compare the learning speed, which indicates training efficiency, with the metric being the number of training steps taken until the first successful lap completion. Second, we compare the racing performance, with the metric being the best lap time achieved within a fixed and sufficiently large number of training steps. Demonstrations are collected using Model Predictive Control (MPC) with a simple kinematic bicycle model, designed to follow a precomputed reference path that accounts for both the racing line and the vehicle's actuator capabilities. These consist of sub-optimal trajectories and even include a few failure cases. All experiments are run on a desktop equipped with an Intel i9-12900K CPU and n NVIDIA RTX 3080Ti GPU.

### B. Training Efficiency Comparison

We conducted the experiment on five algorithms: DAQ-SAC, vanilla SAC, SAC+BC, GAIL, and AIRL [26]. All the algorithms share the same hyperparameters, such as learning rate and discount factor, except for specific parameters like the rollout length for on-policy algorithms and $\lambda$ for DAQ-SAC.

Training efficiency is compared with the standard of the required training steps until the agent completely finish a lap, and the result it shown in Fig. 3. Our approach exhibits the best performance in two aspects: required training steps and the episode return at that steps. This indicates that the DAQ-SAC effectively utilizes the given demonstrations to accelerate the learning process. Notably, the IL-based algorithms fail to learn how to drive, highlighting the difficulty of generalization without the agent's online exploration.

### C. Performance Comparison

Table I shows the comparison of the best lap times achieved achieved after 500,000 training steps. The lap time for demonstrations indicates the average lap time among the existing data. Our approach achieves the fastest lap time of 1:29.867 and maintains higher speeds in all sections compared to the best performance of the demonstrations, as shown in Fig. 4.

Vanilla SAC performs the worst due to its slow learning speed in the sparse reward setting. Although SACBC performs better, it fails to surpass the demonstration's lap times. The BC term guides the agent to closely follow the trajectories provided by the demonstrations, which helps the agent perform well in the initial stage. However, even when the agent discovers more optimal actions, the BC term discourages this improvement, causing the agent to revert to its previous performance. Our algorithm overcomes this limitation, achieving superior performance in both training efficiency and lap time. Additionally, both GAIL and AIRL algorithms never succeed in completing a full lap.

### D. Learned Driving Behaviors

This subsection qualitatively analyzes the driving behaviors learned by the agent. Fig. 5 shows the agent's trajectory for achieving the minimum lap time, with three corners 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 curve sections, resulting in improved overall performance.

## V. CONCLUSIONS

Motivated by the challenges of applying existing LfD methods to racing tasks, we proposed DAQ-SAC, which integrates Adversarial Imitation Learning (AIL) and Soft Actor-Critic (SAC) through an augmented Q-function. This DAQ-aided approach can be incorporated into any value-based off-policy algorithm.

While this paper presents preliminary results, ongoing experiments on more complex tracks show that DAQ-SAC exhibits a greater performance gap as the complexity increases. Additionally, we plan to conduct ablation studies, examining factors such as the size of the demonstration set and variations in hyperparameters.

## ACKNOWLEDGMENT

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant code RS-2021-KA162182). This work is supported 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).

## 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. Brandstátter, 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] 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.

[6] J. Herman, J. Francis, S. Ganju, B. Chen, A. Koul, A. Gupta, A. Skabelkin, I. Zhukov, M. Kumskoy, and E. Nyberg, "Learn-to-race: A multimodal control environment for autonomous racing," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9793–9802.

[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] 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.

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

[14] 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.

[15] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

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

[17] M. Bojarski, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[18] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.

[19] 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.

[20] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[21] R. Zhao, U. Topcu, S. Chinchali, and M. Phielipp, "Learning sparse control tasks from pixels by latent nearest-neighbor-guided explorations," *arXiv preprint arXiv:2302.14242*, 2023.

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

[23] 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.

[24] 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.

[25] 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.

[26] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.