AUTONOMOUS RACING WITH RICH SEMANTICS & ASYMMETRIC SOFT ACTOR-CRITIC

Armeet S. Jatyani, Ardra Charath, Ashug Gurijala, Arnauld Martinez, Jacob W. Schuster
Mentors: Gaunzhi Wang, Sorina Lupu
California Institute of Technology

ABSTRACT

High-speed autonomous racing presents a unique challenge. Cars must be adaptable and resilient to changing conditions on the track while considering hundreds of environmental observations every second. In this paper, we present an asymmetric soft actor-critic (ASAC) model for high-speed autonomous racing, where the actor relies solely on vision inputs, while the critic uses both vision and state information. This approach eliminates the actor's dependency on state information, a critical drawback in previous methods. We also leverage foundational vision models like DINO, which provide rich features such as depth and object semantics. Our method replaces traditional GPS-based state information with field-of-view images, improving reliability by eliminating dependence on satellite communication. We train all models on the Trackmania racing game using the `TMRL` real-time gym. We compare our approach with a symmetric baseline, where both the actor and critic have access to vision and state information, and show that our model achieves comparable rewards with a collision penalty. Key contributions include the introduction of an asymmetric soft actor-critic for autonomous racing tasks and the use of advanced foundational models to enhance vision-based decision making with higher-level features.

1 INTRODUCTION

High-speed autonomous racing presents a unique set of challenges. Unlike conventional autonomous driving, which operates at relatively low speeds, high-speed scenarios demand exceptional adaptability and rapid decision-making. Autonomous systems must process many environmental observations and generate complex motion plans multiple times per second. At these speeds, even minor miscalculations can have significant consequences.

Traditional autonomous vehicles often rely on GPS for state-based information, including acceleration, velocity, and position. However, GPS-based methods become unreliable and inconsistent at high speeds, particularly in highly dynamic environments like those faced by Formula 1-style racing. Signal delays, communication interruptions, and inaccuracies can cause critical failures. These limitations necessitate a shift towards sensing and decision-making approaches that are not only robust and reliable but also fast enough to handle the split-second demands of high-speed racing.

To address these challenges, we propose a vision-based asymmetric soft actor-critic (SAC) model that processes real-time field of view images from cameras mounted on the vehicle. As in Haarnoja et al. (2018) and Geles et al. (2024), we provide the critic with privileged state information, whereas the actor is limited to vision. By eliminating the actor's dependency on externally-retrieved state-information and relying solely on vision, our approach significantly enhances reliability and adaptability under high-speed conditions. This asymmetric implementation is trained *end-to-end* so the actor can drive effectively using only visual input.

To the best of our knowledge, we are the first to apply vision-based asymmetric SAC and leverage DINO for high-speed racing.

2 BACKGROUND

2.1 REINFORCEMENT LEARNING (RL)

Reinforcement learning (RL) is a machine learning approach where an agent learns to make decisions by interacting with its environment. The agent receives rewards for favorable actions and penalties for unfavorable ones, gradually refining its behavior to maximize cumulative rewards over time. In essence, RL learns through trial and error to achieve the best possible outcome in a given situation (AlMahamid & Grolinger (2021)).

A simple example of reinforcement learning is teaching a dog a trick. The teacher gives a dog a treat (reward) if it does the trick correctly and verbal correction (penalty) otherwise. Overtime, the dog will learn the trick to get more treats.

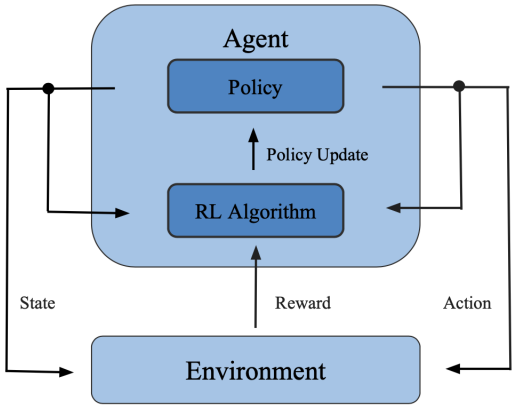

Figure 1: Reinforcement Learning (RL) Structure

The basic flow and structure of reinforcement learning is illustrated in fig. 1. The agent is the model's "decision maker." It interacts with the environment to learn an optimal behavior or policy. The agent is made of the policy and RL algorithm. The policy is the strategy that the agent uses to determine what action to take based on the current observation or state. The RL algorithm updates the policy based on the received reward, which improves decision making over time. The environment is the external system or world with which the agent interacts. It accepts actions from the agent, provides observations or states to the agent, and delivers a reward to the agent to evaluate the quality of its actions.

The goal of RL is to maximize the following objective function:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[\underbrace{r(s_t, a_t)}_{\text{reward}} \right] \tag{1}$$

In eq. (1), $J(\pi)$ is the "goal" of the agent, which is to maximize the total reward the agent collects while interacting with the environment. The reward received for taking action a_t at state s_t is represented as $r(s_t, a_t)$. ρ_π is the "behavioral strategy," or how likely the agent is to visit certain states and take specific actions under its current policy π.

2.2 SOFT ACTOR-CRITIC (SAC)

Traditional reinforcement learning methods often struggle with efficient exploration, especially in complex environments where finding optimal solutions requires a large number of interactions with the environment. Soft actor-critic (Haarnoja et al. (2018)) is a RL algorithm that addresses this by encouraging exploration and preventing premature convergence to suboptimal actions by rewarding

the agent not only for high rewards but also for taking diverse actions. This approach makes SAC particularly effective for tasks with complex dynamics such as autonomous racing.

The goal of SAC is to maximize the following objective function:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[\underbrace{r(s_t, a_t)}_{\text{reward}} + \alpha \underbrace{\mathcal{H}(\pi(\cdot \mid s_t))}_{\text{entropy}} \right]. \tag{2}$$

The equation (eq. (2)) follows the structure of eq. (1), except for the addition of the entropy term $\mathcal{H}(\pi(\cdot \mid s_t))$. The entropy term encourages the agent to act randomly, this allows the agent to explore a broader range of actions, leading to a more resilient and adaptable policy. It also reduces sensitivity to hyperparameters. The term is weighted by the temperature parameter α, which controls the balance between exploration and exploitation.

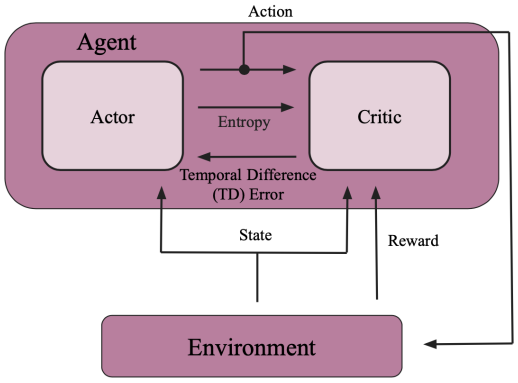

Figure 2: Soft Actor-Critic (SAC) Structure

The structure of SAC is shown in fig. 2. SAC maintains the agent-environment structure of reinforcement learning. However, the agent is now split into the actor and critic, which work together to decide the best actions and evaluate their outcome. The actor generates actions based on the current state. It also calculates the entropy term. The critic evaluates the action taken by the actor by estimating the value function, which predicts the expected cumulate future reward for a given state-action pair. Based on this, the critic computes the temporal difference (TD) error, which measures the difference between the predicted value and the actual reward combined with the estimated future reward. The critic uses the TD error to update and improve its own value function estimates and to provide feedback to the actor to help it refine its policy, enabling the actor to learn which actions are better in each state.

For a simple example in an autonomous driving scenario, the actor may choose an action such as steering or acceleration based on speed, location, and obstacles, and the critic evaluates these choices to optimize future decisions.

SAC is an off-policy algorithm. This means that it can reuse past experiences which are stored in a replay buffer. This allows it to achieve stable performance while using previous data, which is a significant advantage in scenarios where data collection can be costly.

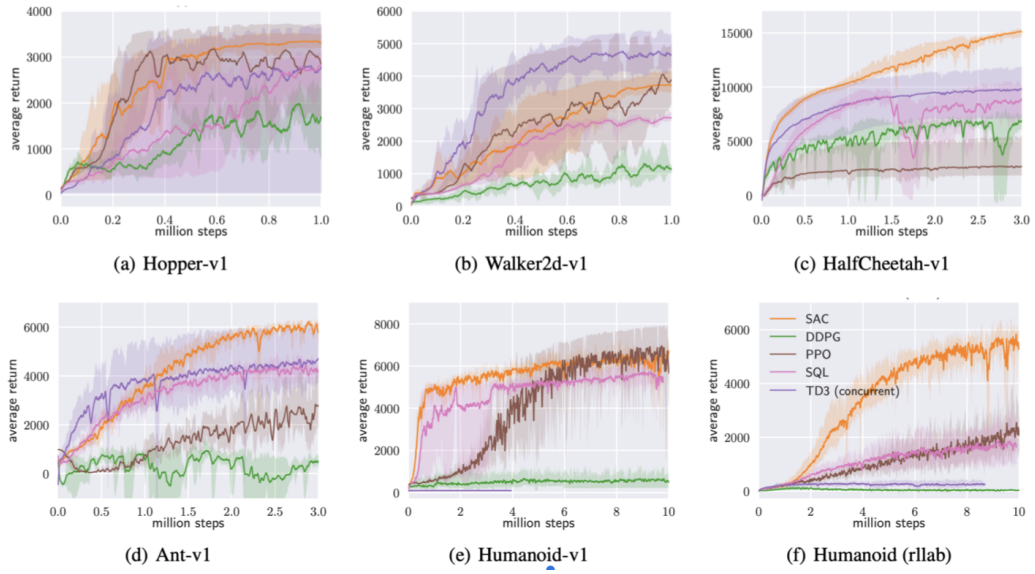

Figure 3: Training Curves of Various Reinforcement Learning Algorithms on Six Continuous Control Tasks. From Haarnoja et al. (2018)

Soft actor-critic (SAC) was tested against various reinforcement learning algorithms on six continuous control tasks (Hopper-v1, Walker2d-v1, HalfCheetah-v1, Ant-v1, Humanoid-v1, and Humanoid (rllab)) as shown in fig. 3. These tasks are used to determine how well different algorithms learn control policies in simulated environments. The graphs show how the average return (cumulative reward) changes over time as the models learn and improve their performance.

In the graphs, the orange line is soft actor-critic, the primary algorithm being evaluated. The green line is Deep Deterministic Policy Gradient (DDPG) Lillicrap et al. (2019), which is an off-policy algorithm that is sensitive to hyperparameters. The brown line is Proximal Policy Optimization (PPO) Schulman et al. (2017), a popular stable on-policy reinforcement learning method. The pink line is Soft Q-Learning (SQL) Haarnoja et al. (2017), another off-policy maximum entropy reinforcement learning method. The last purple curve is Twin Delayed Deep Deterministic Policy Gradient (TD3 (concurrent)) Fujimoto et al. (2018), which is a concurrent method that improves upon DDPG.

SAC consistently outperforms both on-policy (PPO) and off-policy (DDPG, TD3, SQL) algorithms across all tasks, especially in more complex environments such as Ant and Humanoid. It also shows better sample efficiency and stability by achieving high returns with fewer steps and less variability in performance. However, an exception to this is Walker2d-v1, where it is worse than TD3 and often PPO. Walker2d-v1 has a complex, high-dimensional action space with six continuous action dimensions. High dimensional actions increase the chance of instability and makes it harder for SAC to converge to a reliable policy. In addition, the environment's reward structure is based on forward movement and the stability of the agent's "legs." SAC may prioritize exploration over stability in early training which can delay learning a steady walking pattern. This shows that some environments may be less optimal for SAC's learning dynamics. DDPG serves as a strong contrast to SAC. It tends to struggle on all tasks due to its instability and sensitivity to hyperparameters. Part of SAC's success can be accredited to its stability and imperviousness to hyperparameters.

2.3 Asymmetric Soft actor-critic (ASAC)

Recent advancements in reinforcement learning have explored asymmetric actor-critic models (Pinto et al. (2017)), which extend the standard SAC framework by incorporating an asymmetric architecture between the actor and critic networks. This addresses scenarios where there is asymmetric access to information between training and deployment phases. This is particularly helpful for racing scenarios when GPS-based methods become unreliable and inconsistent at high speeds.

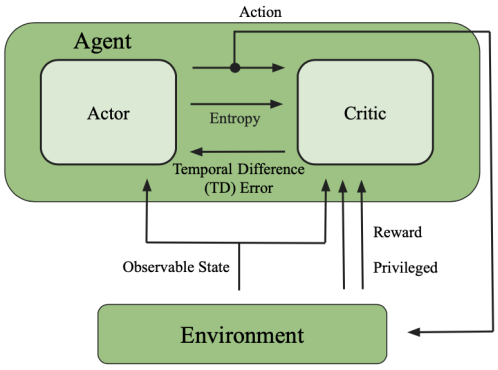

Figure 4: Structure of Asymmetric Soft Actor-Critic (ASAC)

The structure of ASAC is shown in fig. 4. In this setup, the actor operates solely on limited or noisy observations (e.g., field-of-view racing images), while the critic has access to the actor's information as well as additional state information unavailable to the actor, such as full system dynamics. This asymmetric design enables the critic to provide more accurate value estimates during training, leading to improved policy learning for the actor without requiring additional information during inference.

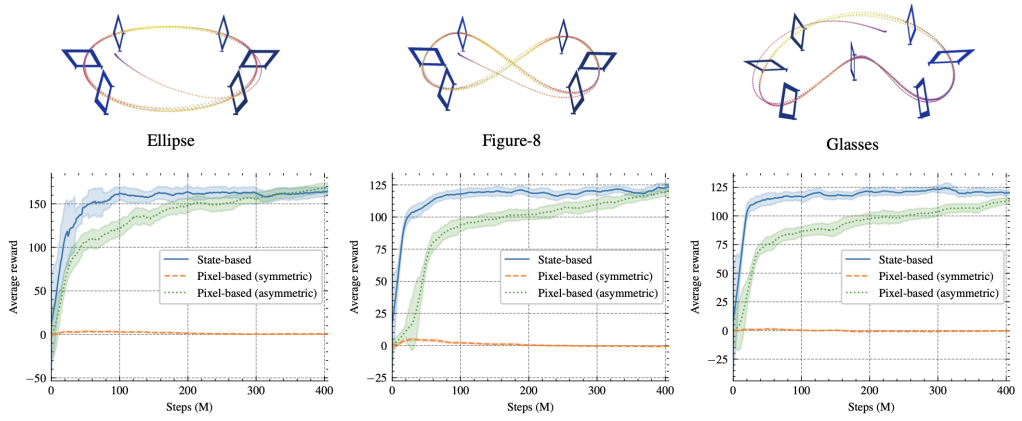

Figure 5: Average Reward Progress During Training of Vision-Based Autonomous Navigation Drone on Three Tracks. From Geles et al. (2024)

A comparison of the average reward of SAC and ASAC in an autonomous drone navigation scenario is demonstrated in fig. 5. The average reward of the two methods converge over time, which demonstrates that this framework achieves performance comparable to fully state-based policies (Geles et al. (2024)). By leveraging the critic's privileged access during training, the asymmetric SAC model enables the actor to learn robust policies that infer the environment's underlying structure despite operating with partial or noisy information.

3 Methods and Results

3.1 Overview

To demonstrate the learning capabilities of the soft actor-critic model, we first implemented and tested it in a simple gymnasium environment (see Section 3.2). In this setup, the actor was trained

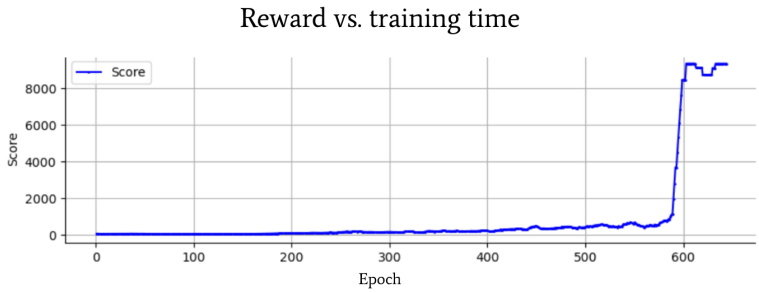

Figure 6: Reward during training for a soft actor-critic model on an inverted double pendulum.

to optimize a policy for balancing an inverted double pendulum. This environment was selected because the dynamics of an inverted double pendulum are well-defined in a small observation vector. However, the motion of the pendulum is chaotic and cannot be efficiently modeled using deterministic simulations. As a result, learning an efficient but optimal policy for the pendulum's dynamics lends itself well to the soft actor-critic framework.

After successfully building and verifying the SAC model in this toy environment, we adapted it to the complex, high-speed racing environment of Trackmania. Trackmania was selected due to its high popularity, accessibility, and plugin support. We started with a baseline SAC implementation in which both the actor and critic had access to the same vision and state information. Building on this foundation, we developed an asymmetric SAC model where the critic has access to privileged state information while the actor relies solely on vision-based inputs. The actor utilizes a convolutional neural network (CNN) to extract important features from the visual input, such as distance to nearby barriers or cars.

We further experiment with the asymmetric case by using vision transformers and segmentation models such as DINO and YOLO. These models allow the actor to extract higher-level state information only available to large foundational models. In particular, we analyze how they impact the model's ability to learn complex policies, the rate of model convergence, and inference time. These qualities are critical in a high-speed racing environment.

3.2 SAC in OpenAI Gymnasium

To familiarize ourselves with the SAC architecture, each member of our team created a model to balance an inverted double pendulum in OpenAI's MuJoCo gymnasium. In this virtual gymnasium, an actor must learn the optimal cart position based on the state of the pendulum. The action space is a 1-element vector representing the signed force applied to the cart, and the observation space is a 9-element vector serializing the angular position and velocity of each pole.

In our implementation, the actor-network is responsible for selecting an action based on the current state of the pendulum. The network learns to adjust the mean and variance of the action space, which allows it to model stochastic actions (controlled by the entropy term). Moreover, the critic-network is responsible for evaluating the quality / expected reward of a state action pair. The network relies on a replay buffer which stores state-transitions and allows the agent to learn based on all previous experiences, not just the current observation.

After training for 650 epochs, all models converged to the maximum possible reward allotted by the gymnasium: 9,350 points (6). In this simple setting, we observed that increasing entropy also increases the time to reward convergence. Additionally, the models' reward curves are not continuous. As a consequence of the actor-critic architecture, once the model discovers a successful path, it greedily exploits its new knowledge to maximize the reward.

3.3 Baseline SAC Model

For our high-speed racing environment, Trackmania was utilized due to its high popularity and large plugin support. To interact with the game, we utilized the `tmrl` Python library: a reinforcement-

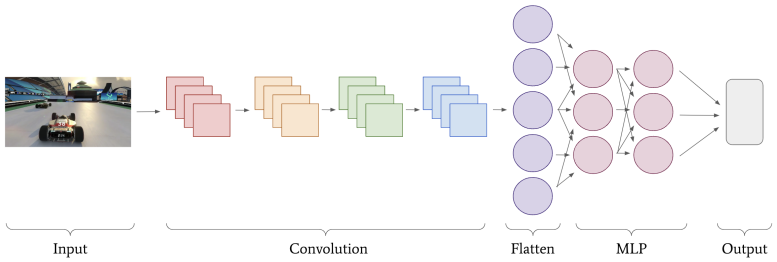

Figure 7: Architecture of the CNN model used to parse vision history

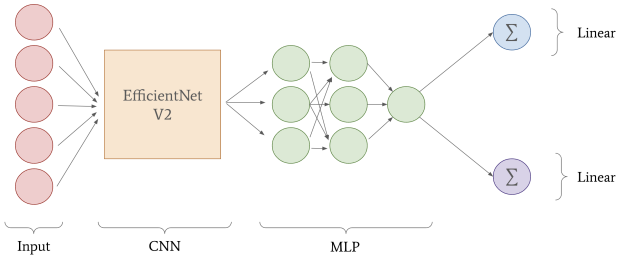

Figure 8: Architecture of the actor module in baseline SAC and asymmetric SAC model

learning package used for many real-time applications. TMRL encodes the vehicle state and a 4-screenshot history of the vehicle at any given time. The vehicle state is a 3-vector describing the angle the car is driving, the velocity of the car, and the position of the car on the track.

In our baseline SAC implementation, the reward function is used to quantify how optimal an action-observation pair is. The reward function is calculated by the user before the model is run. Here, the user must manually drive the car and stay close to the center of the track. During this run, a series of checkpoints are placed along the car's path and serialized for later use. During training, the reward function is calculated as the number of checkpoints that were passed since the previous time-step. This ensures the model is rewarded for covering a large portion of the track in a short amount of time. Naively rewarding vehicle speed does not constitute the best policy, as the optimal trajectory for many turns is to slow down at the apex of the curve and accelerate at the next straight. We also add a penalty for collisions, encouraging the model to avoid barriers and other cars.

Moreover, the actor and the critic both have access to the vehicle state and the screenshot history at any moment in time. This information is encoded in a single observation vector. To eliminate bias from lighting conditions, the screenshots are taken in grayscale and passed through a convolutional neural network. This CNN is a single module shared by the actor and the critic, ensuring they interpret the vision history the same way. The architecture of the CNN was experimentally optimized, and the final version is shown in (7).

The resulting action space is a 4-vector encoding vehicle acceleration and turning angle. A plot of the reward vs. epoch for the baseline graph is shown in (10).

3.4 Asymmetric SAC Model

Building on the baseline model, we modified the SAC architecture to give the critic access to privileged state information. In this asymmetric approach, the critic would have access to both the vehicle state and the screenshot history, while the actor would only have access to the screenshot history. Storing a screenshot history in the replay buffer is critical since it allows the CNN shown in (7) to derive the vehicle's velocity, acceleration, turning angle, and other critical state information that can be verified in the critic. This model was of particular interest since the sensors used in race cars typically become unreliable at high speeds (Magazine (2024)).

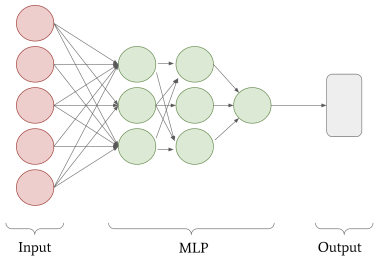

Figure 9: Architecture of the critic module in the baseline SAC, asymmetric SAC, and asymmetric SAC + DINO model.

Figure 10: Reward vs. training time for CNN baseline SAC, CNN ASAC, and DINO ASAC

In our asymmetric implementation, the architecture for the actor and critic is shown in (8 9). The reward function and the CNN used to parse the screenshot history remains the same as in the baseline model (see 7).

The reward as a function of time for the asymmetric SAC model is shown in (10).

3.5 DINO Foundational Model

For high-speed driving applications, rapid inference of semantic track or vehicle information is crucial. In Caron et al. (2021), the authors develop a novel vision transformer called DINO to learn deep vector embeddings of static images. In our high-speed setting, DINO demonstrates the capability to infer track depth, vehicle motion, and sector segmentation from the screenshot history.

Our implementation builds upon the asymmetric SAC framework, integrating a pre-trained DINO model with frozen weights. Patch embeddings are extracted from the DINO output using a PyTorch hook, serializing both spatial and semantic features from the screenshot history. This vision model replaces the CNN architecture in (7). The actor module is identical to the asymmetric case (9). However, the actor module has the updated architecture (11)

The reward as a function of time for the asymmetric SAC with DINO model is shown in (10).

4 Discussion

Our results demonstrate the success of the asymmetric soft actor-critic (ASAC) model, which achieves comparable performance to the baseline SAC despite relying solely on image-based input (see blue and red curves 10). This represents a significant advancement, as it eliminates the dependency on state information, a critical step for real-world deployment in physical vehicles where

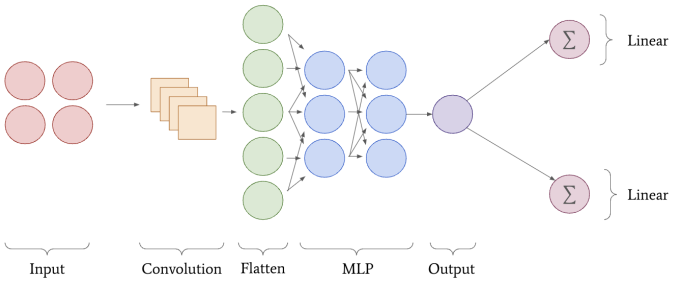

Figure 11: Architecture of the actor module in the asymmetric SAC + DINO model

precise state data may be unavailable. In early training stages, ASAC's reliance on image-based input results in slower progress compared to baseline SAC, which can leverage state information to accelerate initial learning. However, ASAC demonstrates robust adaptability as training progresses. Integrating DINO into the ASAC framework further enriches the model by expanding upon the raw image features. While this results in a longer initial learning phase (approximately 200 epochs), once the model effectively decodes the DINO features, it exhibits rapid improvement, surpassing both baseline SAC and CNN-based ASAC (see teal curve 10). Despite slight instability, these results highlight the potential of DINO-enhanced ASAC to outperform state-dependent models, underscoring its promise for robust, vision-based reinforcement learning in high-speed autonomous racing.

5 Conclusion

In conclusion, we successfully developed an improved racing model that outperforms baseline models while relying on less input information. This approach eliminates the dependency on unreliable state data, such as GPS in a physical F1 car, by utilizing first-person view images. By adopting a more intuitive method akin to how human drivers operate, our incorporation of the DINO model enables the car to extract complex features from images, allowing it to perform even better than if it had access to state information. Additionally, by optimizing reward functions—penalizing wall collisions and encouraging efficient checkpoint navigation—we significantly enhance the safety and performance of the car. Overall, our models demonstrate strong potential for deployment on physical vehicles. With further refinement and testing, our approach has the potential to not only enhance performance but also reduce susceptibility to common external communication errors.

In the future, we plan to visualize the DINO attention maps and CNN feature maps to gain insights into the features being extracted. This analysis will inform further improvements to our models, helping us stabilize and refine the DINO-enhanced ASAC model to maintain maximum rewards consistently. Transitioning to and fine-tuning our models in realistic simulations like Assetto Corsa is another key focus. While Trackmania provided an effective platform for initial learning, more realistic driving simulations will better prepare our models for deployment on physical vehicles. Additionally, Assetto Corsa offers a significant advantage in training speed, as it does not require real-time operation like Trackmania, allowing us to train models in a fraction of the time. Furthermore, one limitation of our approach is Trackmania does not allow multiplayer racing. This means that our model does not know how to respond to other cars, which will be necessary for deployment on a realistic F1 scenario.

We also aim to test our models against experienced human players in real-time races, evaluating their ability to compete with skilled drivers. Before deploying on a physical F1 car, we will first deploy and extensively test our models on a physical RC car. This will allow us to evaluate how well our simulation-trained models transfer to real-life racing, identifying areas for improvement and ensuring the model's robustness in a controlled, scaled-down environment. This critical step will help us refine our approach before advancing to full-scale F1 car deployment, pushing the boundaries of autonomous racing.

REFERENCES

Fadi AlMahamid and Katarina Grolinger. Reinforcement learning algorithms: An overview and classification. In *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–7. IEEE, September 2021. doi: 10.1109/ccece53047.2021.9569056. URL http://dx.doi.org/10.1109/CCECE53047.2021.9569056.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. URL https://arxiv.org/abs/2104.14294.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477.

Ismail Geles, Leonard Bauersfeld, Angel Romero, Jiaxu Xing, and Davide Scaramuzza. Demonstrating agile flight from pixels without state estimation, 2024. URL https://arxiv.org/abs/2406.12505.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies, 2017. URL https://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

Gears Magazine. Sensing problems: A look at common sensor failures, 2024. URL https://gearsmagazine.com/magazine/sensing-problems-a-look-at-common-sensor-failures/.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning, 2017. URL https://arxiv.org/abs/1710.06542.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.