Autonomous RL Racing with Strategic Decision Making

Souren Pashangpour
Department of Mechanical
And Industrial Engineering
University of Toronto
Toronto, ON M5S 1A1
souren.pashangpour@mail.utoronto.ca

Gibran A. Rajput
Department of Mechanical
And Industrial Engineering
University of Toronto
Toronto, ON M5S 1A1
gibran.rajput@mail.utoronto.ca

Zicheng Wang
Department of Mechanical
And Industrial Engineering
University of Toronto
Toronto, ON M5S 1A1
zchengw.wang@mail.utoronto.ca

Abstract

This paper introduces a hierarchical reinforcement learning (HRL) framework for autonomous racing that integrates macro-action planning with resource management, focusing on tire wear and fuel consumption. Our model utilizes a two-layer architecture: a high-layer for strategic decisions such as pit stops, and a low-layer for direct vehicle control. By incorporating typically omitted physical constraints, our framework improves the realism and strategic depth of race simulations. The model's effectiveness is demonstrated through improved race performance and resource optimization in simulated environments, surpassing traditional methods and human benchmarks. Code is available at: https://github.com/ghssx19/RL-Project-Gym?tab=readme-ov-file

1 Introduction

Autonomous racing and driving is a field where the majority of efforts are concentrated at the intersection of deep learning and automobiles. More specifically Autonomous racing has led to community projects such as F1Tenth, A2RL, AWS DeepRacer, and the Nvidia JetRacer Project. The solutions proposed by competitors in the aforementioned projects and challenges are especially useful in game design, pre-race motorsport simulations, and motorsport real-time decision-making support systems. However, previous work has overtly omitted factors such as tyre wear, energy consumption, and physical restraints of a vehicle such as battery or breaking system temperatures. The omission of these factors which we will refer to as physical constraints leads to simplified decisions making and omits strategic decision-making. Therefore, the developed works are not well-suited for decision support systems or accurate motorsport simulations. Moreover, research such as [1], which has addressed the aforementioned constraint to some degree is difficult to reproduce due to confidentiality constraints making it difficult for the research community to progress upon.

The choice of model to address the problem of autonomous racing has been primarily reinforcement learning (RL), and Deep RL (DRL), where an agent in control of the car learns directly from experiences collected in a simulation environment. The simulation environments used to train and deploy these models are often proprietary such as AWS DeepRacer, or packages such as Gazebo, OpenAI Gym, or Ansys. The omission of the physical constraints can be justified thus far as simulation platforms lacked parallelization capabilities until the introduction of IsaacLab and therefore it would lead to increased computational overhead to include the physical constraints, while the problem of autonomous racing itself was relatively unexplored at the time.

Thus far only one paper has incorporated macro actions which are temporally extended actions which themselves are constituted of a series of primitive actions each last one single timestep[1]. Simply put macro actions refer to strategic decisions such as deciding to take a pit stop, maintain current position in race, or overtake in the race an agent can take to mitigate the effects of physical constraints to grant itself an age over other competitors, during race time, where each of these actions consists of low-level agent actions such as directional movement. This additional aspect of making high-level strategic decisions is often referred to as macro action planners and have been applied in Wearhouse delivery [2].

In this paper, we introduce a Hierarchal RL-based multi-agent racing approach that accounts for physical constraints, namely 1) tire wear, and 2) fuel consumption. Our unique contributions is the development of a multi-agent Hierarchical RL project which can be expanded on by other researchers with the objective of advancing macro action planning in the area of autonomous racing called Hierarchical -RL Racing Model.

2 Related work

Existing autonomous racing and driving methods can be categorized as: 1) Hierarchical control [3], [4], [5], [6], [7], [8], [9], or 2) single model control [10], [11], [12], [13], [14].

2.1 Hierarchical RL for autonomous driving and racing

Hierarchical RL methods for autonomous driving and racing separate the macro and micro actions that an agent can take and delegate it to its respective deep learning model or algorithm. More specifically hierarchical methods either use a higher level model to 1) generate way points for the lower level model to drive to [4], [6], [7], 2) is used to identify the current state of the environment and delegates the agent's actions to the expert model or algorithm developed for the agents current state [5], [8], [9].

Furthermore, the method by which the high level RL model is embedded with the low level model is either 1) interconnected, where the outputs from the higher level model are used as an output to the lower level model which allows both components to learn simultaneously and increase the adaptability of the framework [3], [5]Furthermore, the method by which the high level RL model is embedded with the low level model is either 1) interconnected, where the outputs from the higher level model are used as an output to the lower level model which allows both components to learn simultaneously and increase the adaptability of the framework [3], [5], [7] . Or 2) where the inputs to the lower level model are strictly from the environment observation space and there is no communication between the two models [4], [6], [8], [9]. In the latter case a series of if/else statements are used to select the correct expert model for the current state of the environment based on the high level models output.

2.2 Single model control

Single model RL methods used in autonomous racing are generally applied to the problem of optimizing the racing line. In single model RL architectures the reward functions are carefully hand crafted to extract as much unique information from the environment as possible and address a specific issue, for example minimizing time around the track given physical constraints such as friction, maximum speed, maximum turning radius and etc… The single model control architectures are often then adopted in hierarchical RL approaches to serve as the expert models for the different cases possible in the environment. Some of the common models trained and tuned with hand crafted reward function for autonomous driving and racing include PPO [10], Soft

Actor Critic [10], twin delayed deep determinist policy gradient [11], Deep Deterministic Policy Gradient [12], Deep Q-Network [13], and Probabilistic Inference for Learning Control [14]

2.2 Summary of limitation

In summary, there has been much research done in applying models such as LIST THE MODELS in applying single model control to different aspects of autonomous driving such as optimizing the path taken around a given track, or in road situations or maneuvers such as changing lanes or merging onto traffic. As regards to hierarchical approaches the majority of the work done is based around either 1) generating waypoints or areas which the lower level model should navigate to or 2) selecting an expert model best suited for a given environment's conditions, such as if the objective is to make a left turn the higher level model will apply the model trained for the task of making a left turn.

To the authors' knowledge hierarchical control for making strategic decisions has only been done once in a non-reproducible manner in [3], our aim with this project is to allow the community to have a maintained and working code base which they can use to address the problems introduced in the next section.

3 Hierarchal-RL racing methodology

In this section, we define the autonomous racing problem and present our Hierarchical DRL-based Racing model architecture. Figure 1 shows the overall model architecture.

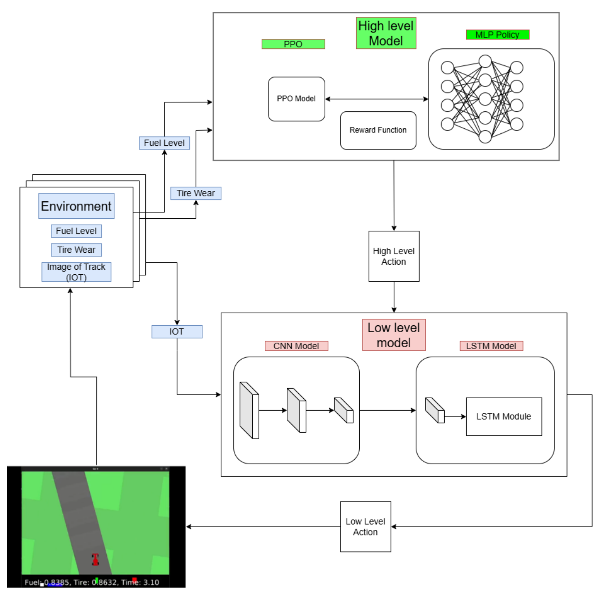

Figure 1: Model Architecture

3.1 Problem definition

The autonomous racing problem involve a vehicle V navigating a race track while considering key resource constraints: gas volume (v_g) and tire wear condition (t_w). The racetrack is represented as a continuous path within an environment characterized by sharp turns. From the simulation we can observe gas volume and tire wear per time step. The rate of consumption for these resources are dependent on factors such as periods of acceleration, deacceleration, and the turning radius.

The objective is to minimize 1) the total amount of supplies (gas, tire thread) used, and 2) the lap time shows in Eq. (1). Therefore we need to maximize the speed of the car at any given time around the track. The aforementioned recursion and the fact that a pitstop can only be taken at a brief section of the track lead to the complexity creating a model for predicting pitstops. In this paper we aim to start the preliminary works which can serve as a platform for future researchers to address the defined problem. Moreover, we will also focus on minimizing the frequency of pit stops, contributing to the **overarching goal of** optimizing race performance.

$$\min T_{total} = \min \left[T_d + \sum^{n_{pit}} T_{pit} \right] \min T_{total} \tag{1}$$

where T_{total} denotes total race time, T_d denotes total time spent driving on the track and T_{pit} denotes time spent in the pit stop, lastly n_{pit} denotes total number of pit stops. Specifically, this paper focuses on minimizing the frequency of pit stops, contributing to the overarching goal of optimizing race time.

3.2 Autonomous racing architecture

The proposed hierarchical RL racing architecture consists of two distinct layers: (1) a high-layer RL that handles strategic decision-making, and (2) a low-layer RL responsible for low-level vehicle control on the racetrack. This separation allows the architecture to efficiently divide complex tasks into manageable sub-tasks, where the high-layer RL focuses on task-specific optimization (Macro actions) [4], [15], and the low-layer RL specializes in directly interacting with the environment and producing control actions (Micro actions) [31]. The following sections describe the architecture's key components in detail.

3.2.1 Pit stop model

The Pit Stop RL module forms the high-layer RL and is responsible for strategic decision-making regarding whether the vehicle should continue driving or take a pit stop for refueling or tire replacement. This decision is critical in optimizing the race performance while adhering to resource constraints, such as tire wear ($\boldsymbol{t_w}$) and fuel volume ($\boldsymbol{v_g}$). The module is implemented using Proximal Policy Optimization (PPO) [16], [17], [18]] with policy.

PPO is an advanced gradient method designed to improve the stability and efficiency of policy updates during training. It addresses the limitations of earlier methods such as Trust Region Policy Optimization (TRPO) [19]by using a clipped objective function to constrain policy changes within a trust region, thus preventing overly large updates that may destabilize training.

The clipped surrogate objective in PPO is defined in Eq. (2).

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{2}$$

where: $r_t(\theta)$ is the ratio of new to old policy probabilities, A_t is the estimated advantage function, ε is a clipping parameter (e.g., 0.2), which limits the extent of policy updates.

This clipped objective ensures that updates are neither too small (inefficient learning) nor too large (destabilizing), leading to a more robust and sample-efficient training process.

The advantage function A_t is computed in Eq. (3).

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots \tag{3}$$

Where:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

r_t is the reward at time t. γ is the discount factor, controlling how far into the future rewards are considered. λ is the GAE (Generalized Advantage Estimation) decay parameter, balancing bias and variance in the estimation. $V(s_t)$ is the value function estimate for state s_t. The advantage function quantifies the relative quality of the chosen action a_t in state s_t, aiding the policy in learning which actions are advantageous.

The MLP policy is trained to maximize this clipped objective. The MLP policy consists of: 1) Input Layer: Two state variables (v_g, t_w). 2) Hidden Layers: Fully connected layers with non-linear activation functions (ReLU) [20], designed to capture complex state-action relationships such as trade-off between tire wear and speed. 3) Output Layer: A softmax [21] layer that outputs the probabilities of each high-level action (a_h). If $a_n = 1$, then pit stop is needed. If $a_n = 0$, then continue driving. The policy is trained to maximize cumulative rewards, where the reward function reflects race performance metrics, pit stop efficiency, and resource utilization.

3.2.2 Vehicle control model

The low-layer RL module is responsible for low-level vehicle control, ensuring the vehicle navigates the racetrack effectively and adheres to the high-layer RL's strategic decisions. Unlike the high-layer RL, which determines macro-level strategies, the low-layer RL focuses on continuous control tasks such as steering, throttle, and braking to maintain the vehicle's optimal trajectory and speed. To improve efficiency and reduce training overhead, the low-layer RL module leverages a pre-trained model. This model, based on PPO combined with a Long Short-Term Memory (LSTM) architecture [22], is specifically designed for controlling vehicles in continuous action spaces. The architecture combines the advantages of PPO's robust policy optimization with LSTM's ability to capture temporal dependencies, making it well-suited for tasks requiring sequential decision-making in dynamic environments.

The pre-trained model is composed of several key components. The first component is the Observation Encoder, which processes the image-based observation space provided by the environment. The input consists of 64×64×3 RGB frames that capture a bird's-eye view of the racetrack and the vehicle's surroundings. A convolutional neural network (CNN) [23] is used to encode these images into compact, high-level feature representations, allowing the model to extract meaningful spatial and contextual information essential for understanding the racetrack layout and obstacles.

The encoded features are then passed through a Recurrent Layer implemented as a LSTM network. This layer is designed to maintain a memory of past states, enabling the model to account for temporal dependencies that are critical in vehicle control tasks. For instance, the LSTM allows the model to anticipate upcoming turns or adjust speed in response to approaching obstacles. By maintaining hidden states over time, the LSTM ensures that decisions are informed by both current observations and historical context, improving the model's ability to navigate dynamic environments.

The Policy Network translates the output of the LSTM layer into actionable control commands for the vehicle. Specifically, it maps the processed features to a continuous action space comprising three control parameters: steering angle, throttle, and braking. The steering angle determines the direction of the vehicle, while the throttle and braking control acceleration and deceleration, respectively. These outputs are represented as probability distributions, which allow for stochastic exploration during training and deterministic execution during deployment, balancing learning efficiency and real-time performance.

Finally, the model includes a Value Function network, which estimates the expected return from a given state. This component is essential for the PPO algorithm used in the pre-trained model. By providing an estimate of future rewards, the value function reduces the variance in the reward signal and stabilizes the policy-gradient optimization process. Together, these components enable the pre-trained model to perform robust and efficient control in the continuous action space of autonomous racing.

3.2.3 Reward function

The reward function is designed to guide the vehicle's decision-making process, with the objective of minimizing pit stop frequency while ensuring optimal performance based on the vehicle's resource conditions, specifically fuel volume (v_g) and tire wear (t_w). The rewards and penalties are carefully assigned to incentivize appropriate actions under different scenarios, as described in Eq. (5).

$$r = \begin{cases} 50, & a_n = 1, v_g > 0.1, t_w > 0.1 \\ -1000, & a_n = 1, v_g \leq 0.1, t_w \leq 0.1 \\ 200, & a_n = 0, v_g \leq 0.1, t_w \leq 0.1 \\ -40, & a_n = 0, v_g > 0.5, t_w > 0.5 \end{cases} \tag{5}$$

$a_n = 1$ means pit stop action. $a_n = 0$ means continue driving action. For the pit stop action, the model is penalized with a reward of -40 when resources are in a healthy state (v_g>0.5 and t_w>0.5), discouraging unnecessary pit stops. Conversely, a reward of 200 is assigned if the pit stop action is taken under critical resource conditions (v_g ≤0.1 and t_w≤0.1), encouraging the agent to take corrective measures when resources are nearly depleted.

For the continued driving action, the model receives a reward of 50 when resources are sufficient, promoting uninterrupted progress on the track. However, a substantial penalty of -1000 is applied if the model continues driving under critical conditions (v_g≤0.1 and t_w≤0.1), strongly discouraging risky behavior that could lead to failure.

```
Algorithm 1: Multi-Car Racing Simulation Algorithm
Input: env: MultiCarRacing environment,
low_level_model: Pre-trained low-level driving model,
pit_model: Pre-trained high-level pit decision model,
fuel_rate: Rate of fuel consumption,
tire_rate: Rate of tire wear,
max_steps: Maximum number of simulation steps,
FPS: Frames per second.
Output: total_rewards: Total rewards accumulated by the car

Initialization:
Set fuel_level ← 1.0, tire_level ← 1.0, total_rewards ← 0.0,
 done ← False, step_counter ← 0 ;
Reset the environment: obs ← env.reset() ;
while done = False and step_counter < max_steps do
    Predict low-level action: action_low ← low_level_model.predict(obs)
     ;
    Update fuel and tire levels:
        fuel_level ← max(fuel_level − fuel_rate/FPS, 0.0) ;
        tire_level ← max(tire_level − tire_rate/FPS, 0.0) ;
    Formulate high-level observation: obs_high ← [fuel_level, tire_level]
     ;
    Predict high-level action:
     action_high ← pit_model.predict(obs_high) ;
    if action_high = PIT then
        Refill fuel and tire levels: fuel_level, tire_level ← 1.0 ;
        Step the environment:
         obs, rewards, done ← env.step(action_low) ;
        total_rewards ← total_rewards + rewards ;
    else
        Step the environment without pitting:
         obs, rewards, done ← env.step(action_low) ;
        total_rewards ← total_rewards + rewards ;
    Increment step counter: step_counter ← step_counter + 1 ;
return total_rewards
```

Figure 2: Pseudo code of the Hierarchical Model

4 Training

The Hierarchal-RL Racing model was trained in simulated racing environments using the PPO algorithm. Each training environment represents a racetrack with dynamic elements such as varying friction and obstacles. To simulate realistic race conditions, the environment incorporates both sharp turns and straight sections, requiring the agent to learn diverse driving strategies. The training framework uses a time-step-based episodic structure, where each episode ends when the vehicle completes a lap or exhausts its resources.

The PPO model was configured with the following hyperparameters: a learning rate of 7×10^{-5}, $n_{steps} = 2048$, batch size of 64, and $n_{epochs} = 40$. A discount factor (γ) of 0.99 was used to prioritize long-term rewards, while the Generalized Advantage Estimation (GAE) parameter (λ) was set to 0.95 to reduce variance in the advantage function. The clipping range for PPO's surrogate objective was 0.2, and the entropy coefficient was set to 0.01 to encourage policy exploration. The value function coefficient was set to 0.5, and the gradient clipping threshold was fixed at 0.5 to ensure stable updates during training.

The policy network utilized a MLP architecture with customized policy keyword arguments designed for efficient feature extraction. The MLP policy employed leaky ReLU as the activation function for all layers except the output layer, ensuring smooth gradient propagation. The training was performed over 250,000 episodes with an experience batch size of 16. The PPO agent was trained using the cumulative reward as feedback, balancing strategic decision-making in the high-layer RL with low-layer control optimization.

Training was executed over approximately 28 hours. The pseudo code for the model simulation and training is in Figure 2. Figure 3,4 is the plot showing total reward per testing epoch, while figure 4 is the accumulation of the rewards during training.

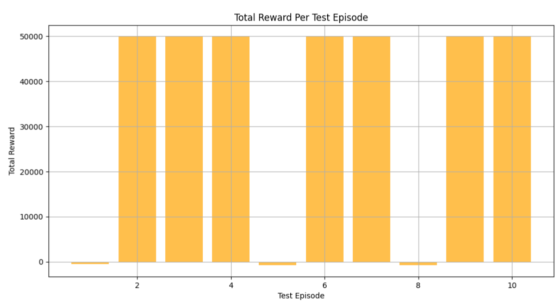

Figure 3: Reward per testing epoch

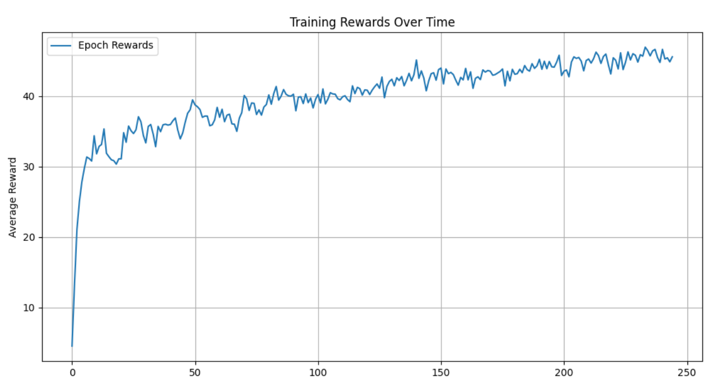

Figure 4: Accumulation of rewards during training

5 Simulated experiments

The simulated experiments include: 1) a comparison study to evaluate the performance of Hierarchal-RL Racing model with human controlling

5.1 Comparison study

We evaluated the performance of the Hierarchical RL racing model against human drivers based on total race time (TRT). The goal of this comparison was to assess how effectively the proposed model balances resource management and driving efficiency relative to human performance.

5.1.1 Race environment

The evaluation was conducted in a simulated racetrack environment designed to mimic real-world racing conditions. The track included sharp turns, and straights, requiring adaptive strategies for both speed optimization and resource management. The model and human participants evaluated under identical initial resource states (v_g =1.0, t_w=1.0) and environmental settings. These settings are the following: friction, driving backward, driving forward, on grass, steering, braking, and speed. These setting are sent to the low level model as an Image of track (IOT)

5.1.2 Human participants

One human participant controlled the vehicle using a standardized interface with keyboard inputs, with feedback on fuel and tire conditions displayed in real time. Humans were instructed to prioritize minimizing total race time while managing resources to complete the race successfully.

5.1.3 Procedure

Once the higher level model was trained, it was imported into a separate file with the lower level model with an instance of an environment. The seed level was fixed so the generated track would be the same. Another instance of the environment was created for Human with he same seed value for the same track to be generated. Trials were run by them model and by the human for 10 laps.

5.1.4 Results

Table 1 presents a comparison of tabulated results between two TRT's: the TRT of the model and the TRT of the human performance.

Table 1: TRT comparison

Lap number	Model TRT (second)	Human TRT (second)
1st Lap	27.61	54.87
2nd Lap	59.36	50.34
3rd Lap	28.87	48.89
4th Lap	18,95	45.76
5th Lap	25.67	43.32
6th Lap	27.02	40.76
7th Lap	26.54	38.56

8th Lap	29.30	35.45
9th Lap	30.08	36.61
10th Lap	24.04	34.76

6 Discussions

This work introduces a novel hierarchical reinforcement learning architecture for autonomous racing, comprising a high-layer RL module for strategic decision-making and a low-layer RL module for precise vehicle control. The framework advances the state-of-the-art by incorporating macro-action planning and efficient resource management, addressing critical factors such as fuel consumption and tire wear that are often overlooked in prior research. The comparison study demonstrates the model's ability to outperform human drivers in resource optimization and total race time, highlighting its effectiveness in minimizing unnecessary pit stops while maintaining race efficiency.

While the model does not yet achieve the overarching objective of fully optimizing all aspects of autonomous racing due to time constraints, the hierarchical architecture represents a significant step toward that goal. The ability to reduce pit stop frequency through strategic decision-making validates the potential of this approach in addressing real-world racing challenges.

Future work will focus on improving the adaptability of the high-layer RL module to handle a broader range of conditions and developing specialized low-layer RL modules tailored to dynamic scenarios. Efforts will also aim to bridge the interaction gap between the high-layer and low-layer RL modules, creating a more integrated learning framework where both layers adapt based on shared inputs and reciprocal feedback. By establishing a two-way communication between the top-level and low-level models, mutual learning can be enhanced, leading to greater control over the system's overall performance. For example, directly linking race time to both layers allows the low-level model to refine its driving efficiency, while the high-level model concurrently learns to optimize pit stop strategies based on low-level driving behaviors. These enhancements will further evolve the architecture, contributing to a more comprehensive and robust solution for autonomous racing challenges.

Acknowledgments

We extend our heartfelt gratitude to the individuals and communities whose contributions were pivotal to this project. Special thanks to igilitschenski & tseyde [24], for their foundational work on the environment, and to sb3 for developing the core model that guided our research. We also acknowledge the dedicated developers and maintainers of the libraries, frameworks, and tools that empowered our project.

References

[1] C. Amato, G. D. Konidaris, and L. P. Kaelbling, "Planning with Macro-Actions in Decentralized POMDPs".

[2] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep Reinforcement Learning for Decentralized Multi-Robot Exploration With Macro Actions," Feb. 26, 2024, *arXiv*: arXiv:2110.02181. doi: 10.48550/arXiv.2110.02181.

[3] X. Liu, A. Fotouhi, and D. Auger, "Formula-E Multi-Car Race Strategy Development—A Novel Approach Using Reinforcement Learning," *IEEE Trans. Intell. Transport. Syst.*, vol. 25, no. 8, pp. 9524–9534, Aug. 2024, doi: 10.1109/TITS.2024.3389155.

[4] R. S. Thakkar, A. S. Samyal, D. Fridovich-Keil, Z. Xu, and U. Topcu, "Hierarchical Control for Head-to-Head Autonomous Racing," *FR*, vol. 4, no. 1, pp. 46–69, Jan. 2024, doi: 10.55417/fr.2024002.

[5] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider, and J. M. Dolan, "Hierarchical Reinforcement Learning Method for Autonomous Vehicle Behavior Planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 6084–6089. doi: 10.1109/IROS45743.2020.9341496.

[6] C. Chung, H. Seong, and D. H. Shim, "Learning from Demonstration with Hierarchical Policy Abstractions Toward High-Performance and Courteous Autonomous Racing," Nov. 07, 2024, *arXiv*: arXiv:2411.04735. doi: 10.48550/arXiv.2411.04735.

[7] D. Kalaria, Q. Lin, and J. M. Dolan, "Towards Optimal Head-to-head Autonomous Racing with Curriculum Reinforcement Learning," Aug. 25, 2023, *arXiv*: arXiv:2308.13491. doi: 10.48550/arXiv.2308.13491.

[8] J. Duan, S. Eben Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intelligent Trans Sys*, vol. 14, no. 5, pp. 297–305, May 2020, doi: 10.1049/iet-its.2019.0317.

[9] B. Gangopadhyay, H. Soora, and P. Dasgupta, "Hierarchical Program-Triggered Reinforcement Learning Agents For Automated Driving," *IEEE Trans. Intell. Transport. Syst.*, vol. 23, no. 8, pp. 10902–10911, Aug. 2022, doi: 10.1109/TITS.2021.3096998.

[10] B. Petryshyn, S. Postupaiev, S. Ben Bari, and A. Ostreika, "Deep Reinforcement Learning for Autonomous Driving in Amazon Web Services DeepRacer," *Information*, vol. 15, no. 2, p. 113, Feb. 2024, doi: 10.3390/info15020113.

[11] Z. Lu, C. Zhang, H. Zhang, Z. Wang, C. Huang, and Y. Ji, "Deep Reinforcement Learning Based Autonomous Racing Car Control With Priori Knowledge," in *2021 China Automation Congress (CAC)*, Beijing, China: IEEE, Oct. 2021, pp. 2241–2246. doi: 10.1109/CAC53003.2021.9728289.

[12] A. Remonda, S. Krebs, E. Veas, G. Luzhnica, and R. Kern, "Formula RL: Deep Reinforcement Learning for Autonomous Racing using Telemetry Data," Jun. 13, 2022, *arXiv*: arXiv:2104.11106. doi: 10.48550/arXiv.2104.11106.

[13] P. Aldape and S. Sowell, "Reinforcement Learning for a Simple Racing Game".

[14] Y. Zhu and D. Zhao, "Vision-based control in the open racing car simulator with deep and reinforcement learning," *J Ambient Intell Human Comput*, vol. 14, no. 12, pp. 15673–15685, Dec. 2023, doi: 10.1007/s12652-019-01503-y.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 28, 2017, *arXiv*: arXiv:1707.06347. doi: 10.48550/arXiv.1707.06347.

[17] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs".

[18] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[19] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," Apr. 20, 2017, *arXiv*: arXiv:1502.05477. doi: 10.48550/arXiv.1502.05477.

[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Jun. 2011, pp. 315–323. Accessed: Nov. 24, 2024. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[21] I. Kouretas and V. Paliouras, "Simplified Hardware Implementation of the Softmax Activation Function," in *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, May 2019, pp. 1–4. doi: 10.1109/MOCAST.2019.8741677.

[22] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997, doi: 10.1162/neco.1997.9.8.1735.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," 1998.

[24] W. Schwarting *et al.*, “Deep Latent Competition: Learning to Race Using Visual Control Policies in Latent Space,” Feb. 19, 2021, *arXiv*: arXiv:2102.09812. doi: 10.48550/arXiv.2102.09812.