CMPE 260 - Reinforcement Learning

Department of Computer Engineering

San Jose State University

Instructor: Jahan Ghofraniha

Driving the Future: A Deep Learning Approach to Car Racing Game

Github link: https://github.com/ChirudeepG/RL_project.git

Team Members

Name	**Student Id**
Chirudeep Gorle	**016682627**
Abdul Vahed Shaik	**016452540**

Table of Contents

1. Executive Summary

In this project, we implemented and compared Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic algorithms in the OpenAI game CarRacing-v0. Despite limited processing power and time, we undertook significant research on each method, which was followed by rigorous training sessions. While some unexpected behaviors occurred, most likely as a result of resource constraints, the overall results were satisfactory. The models performed admirably in terms of keeping the virtual car on course and maximizing rewards. The dedication, patience, and passion of our team were critical in overcoming technological

obstacles. Despite the project's failure to achieve optimal results, it provides useful insights into the actual application of reinforcement learning algorithms in complicated gaming contexts. The outcomes, while not ideal, demonstrate the team's dedication to pushing the bounds within the constraints.

2. Introduction

Our project focuses on the application and comparison of three sophisticated reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic—to the OpenAI CarRacing-v0 game. Our key goal in this difficult virtual environment was to create models that could efficiently navigate the virtual car, stay on the road, optimize rewards, and eventually achieve game success. Our journey began with a thorough examination of the theoretical underpinnings of each algorithm, providing the platform for our subsequent implementations. We persevered through an extended training period despite constraints such as inadequate processing capacity and time constraints, particularly in Google Colab.

We carefully tested with the three chosen algorithms over several days, monitoring their habits and analyzing their performances. While certain results departed from our theoretical assumptions due to computational resource and time restrictions, the overall results were regarded good. The models demonstrated impressive skill in navigating the virtual automobile within the CarRacing-v0 environment. The team's unwavering commitment, patience, and passion, all of which played critical roles in overcoming technological hurdles, can be ascribed to their achievement.

Our effort offers light on the practical applications of reinforcement learning algorithms in complicated game settings, albeit not achieving perfection due to stated constraints. The voyage demonstrates the team's commitment to pushing the limits of what is possible under certain constraints, giving vital insights for future attempts in the realms of reinforcement learning and virtual worlds.

3. Problem Statement

Implementing and comparing three distinct reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic—with the goal of training virtual agents to navigate a simulated racing environment effectively posed a multifaceted challenge for the CarRacing-v0 project. The fundamental problem was to create models capable of keeping a virtual car on the road, maximizing rewards, and ultimately succeeding in OpenAI's dynamic and sophisticated CarRacing-v0 game.

The main challenges were processing power constraints and time constraints, particularly in the Google Colab setting. These limits influenced the depth and breadth of training sessions, which could have an impact on the models' learning ability. Furthermore, unanticipated actions observed during the training process prompted concerns regarding the alignment of theoretical expectations with practical realities. Thus, the problem statement revolves around overcoming the constraints imposed by limited computational resources and time constraints, optimizing the performance of the implemented algorithms within these constraints, and reconciling any discrepancies between theoretical predictions and observed behaviors. Resolving these challenges will help not only to complete the project aims, but also to gain a better knowledge of the practical implications and limitations of reinforcement learning algorithms in complex gaming environments like CarRacing-v0.

4. Motivation

Our motivation stems from the enormous potential of reinforcement learning techniques to develop virtual agents into intelligent entities capable of performing complex tasks. The CarRacing-v0 project is a great testing ground, providing a dynamic setting in which algorithms such as Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic can be polished to handle complex scenarios. The challenge of maintaining a virtual automobile on track, maximizing rewards, and ultimately winning the game drives our desire to learn more about these algorithms' capabilities and limits.

In the pursuit of overcoming processing power and time limits, our team is motivated by a desire to push the frontiers of what is possible within given constraints. The promise of deciphering unanticipated behaviors and connecting theoretical knowledge to real outcomes boosts our determination even more. The success of the study could not only boost reinforcement learning applications, but also deepen our understanding of optimizing models in resource-constrained contexts.

By confronting these difficulties straight on, we hope to make a significant contribution to the field, fostering developments that go beyond the scope of CarRacing-v0. Our motivation originates from the notion that conquering these challenges would not only improve our understanding of reinforcement learning, but will also pave the way for more robust implementations in a variety of diverse and challenging real-world contexts.

5. Contribution

Our project stands out in the literature by providing a detailed and practical examination of three prominent reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic—that have been specifically tailored to the complexities of

OpenAI's CarRacing-v0 game. While previous literature frequently includes theoretical frameworks and abstract debates, our study goes beyond by digging into real-world implementation issues and limits, providing a new viewpoint on how these algorithms might be applied in a dynamic gaming context.

Our emphasis on practical experimentation within resource restrictions is one major differentiation. Many studies focus on theoretical issues without properly addressing real-world implementation challenges, such as limited processing power and time constraints. Our effort delivers practical insights helpful to researchers and practitioners trying to deploy reinforcement learning algorithms in situations with computing limits by facing and clearly documenting these constraints.

Our project identifies and addresses unexpected behaviors noticed during training, allowing us to gain a more sophisticated knowledge of algorithmic adaptation. While theoretical literature describes expected outcomes, real-world application frequently reveals complexities and differences. Our research looks into the gaps between theoretical assumptions and practical results, offering insight on the intricacies of algorithmic behavior in a dynamic game environment. This brings realism not only to the CarRacing-v0 setting, but also to the broader discourse on reinforcement learning.

Furthermore, our project provides a detailed assessment of the difficulties experienced due to parallel environment constraints and RAM limitations. These difficulties, which are frequently disregarded in theoretical debates, have a considerable impact on the viability and scalability of reinforcement learning systems. We provide a significant resource for researchers negotiating similar constraints by freely disclosing our experiences, allowing for a more educated and realistic approach to algorithm development.

In addition to overcoming limits, our study contributes by demonstrating each algorithm's adaptability and effectiveness in a tough and dynamic setting. DQN, PPO, and Actor-Critic are compared within the CarRacing-v0 framework to give a practical benchmark for their different strengths and limitations. This empirical assessment goes beyond theoretical talks, providing a practical reference for academics looking to select the best algorithm for specific applications.

6. Methodology

Deep Q-Network (DQN) Implementation:

1. **Initialize Parameters:** Define hyperparameters such as learning rate, discount factor, exploration strategy, and neural network architecture.
2. **Initialize Q-Network:** Create a neural network model to represent the Q-function. Initialize the target Q-network for stability.
3. **Initialize Replay Buffer:** Create a replay buffer to store experiences for random sampling during training.
4. **Training Loop:** Observe the current state from the environment. Select an action using an exploration strategy (e.g., epsilon-greedy). Execute the action and observe the next state and reward. Store the experience in the replay buffer.
5. **Sample from Replay Buffer:** Randomly sample a batch of experiences from the replay buffer.
6. **Compute Q-Values:** Compute the Q-values for the current and next states using the Q-network.
7. **Compute Loss:** Calculate the temporal difference error and define the loss function.
8. **Backpropagation:** Perform backpropagation to update the Q-network parameters.
9. **Update Target Network:** Periodically update the target Q-network for stability.

10. **Repeat:** Repeat the training loop for a specified number of iterations.

Proximal Policy Optimization (PPO) Implementation:

1. **Initialize Parameters:** Define hyperparameters for learning rate, clip range, discount factor, etc.
2. **Initialize Policy and Value Networks:** Create neural networks for the policy and value functions.
3. **Initialize Optimizer:** Choose an optimizer (e.g., Adam) for updating the networks.
4. **Training Loop:** Collect trajectories by interacting with the environment using the current policy.
5. **Compute Advantages:** Calculate advantages using the rewards and estimated values from the value network.
6. **Update Policy:** Optimize the policy network using the advantage-weighted surrogate loss with a clipping mechanism.
7. **Update Value Function:** Minimize the mean squared error between estimated and actual values.
8. **Repeat:** Repeat the training loop for a specified number of iterations.

Actor-Critic Implementation:

1. **Initialize Parameters:** Define hyperparameters including learning rate, discount factor, etc.
2. **Initialize Actor and Critic Networks:** Create neural networks for both the actor (policy) and critic (value) functions.
3. **Initialize Optimizers:** Choose optimizers for updating the actor and critic networks.
4. Training Loop: Interact with the environment to collect experiences.

5. **Compute Critic Loss:** Calculate the mean squared error between predicted and actual values from the critic network.
6. **Compute Actor Loss:** Formulate the policy gradient using advantages derived from the critic.
7. **Backpropagation:** Perform backpropagation to update both actor and critic networks.
8. **Repeat:** Repeat the training loop for a specified number of iterations.

7. Implementation & Results

Team Member	**Contribution**
Chirudeep	Worked on DQN and Actor-Critic
Vahed	Worked on PPO

DQN Experimentation

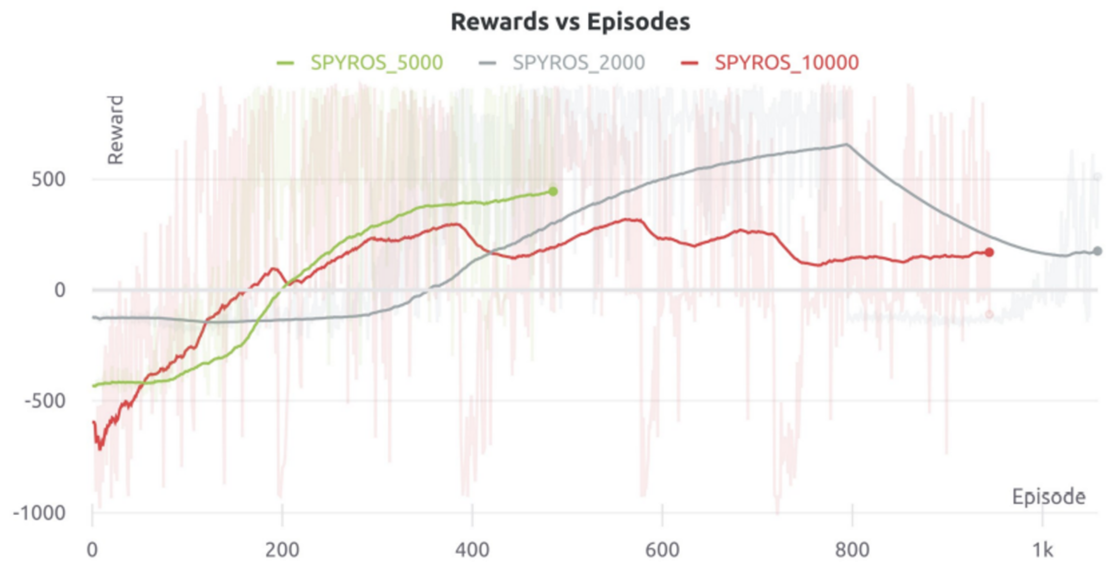
Policy	**CNNPolicy**
learning_rate	**1e-4**
learning_starts	**5.000**
batch_size	**32**
gamma	**0.99**
exploration_factor	**0.9**

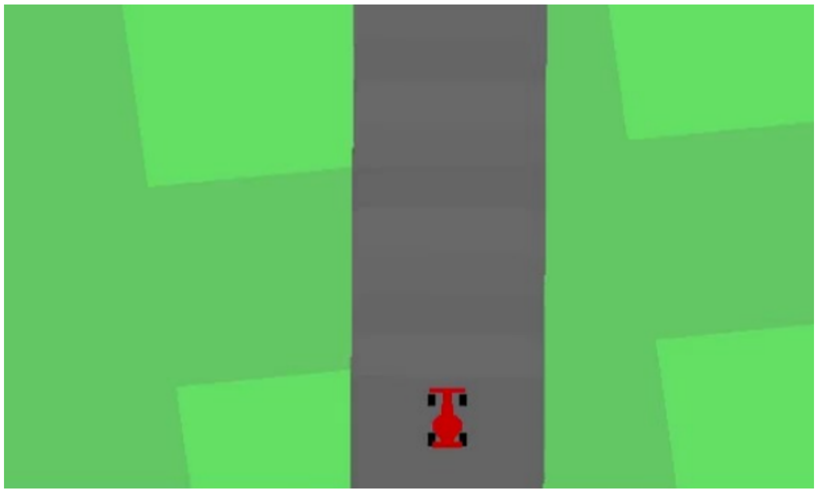
exploration_initial_eps	**1.0**
exploration_final_eps	**0.5**

Difference of Experiments

First Experiment	Second Experiment	Third Experiment
Buffer_Size: 100000	Buffer_Size: 150000	Buffer_Size: 150000
Maximum_Steps_Per_Episode: 10000	Maximum_Steps_Per_Episode: 2000	Maximum_Steps_Per_Episode: 5000
Log_Interval: 20	Log_Interval: 50	Log_Interval: 50
Episodes: 1000	Episodes: 1000	Episodes: 500

Results of DQN



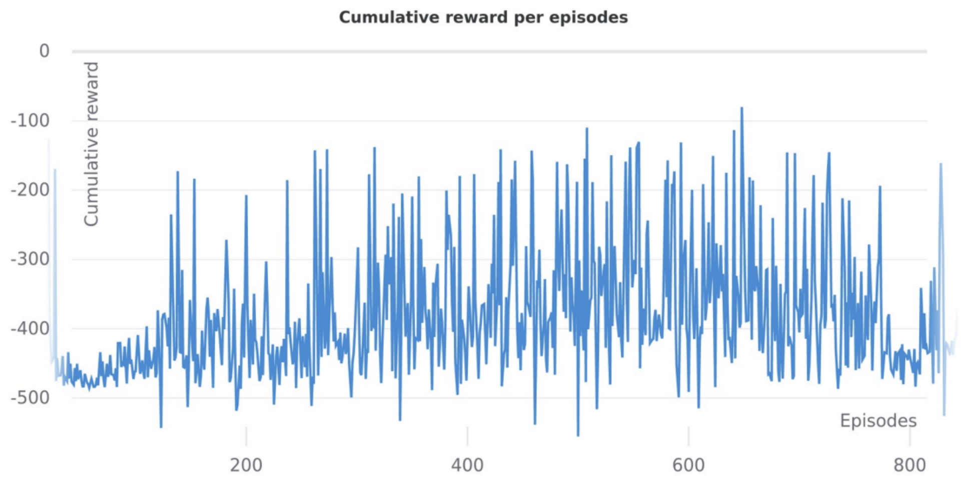


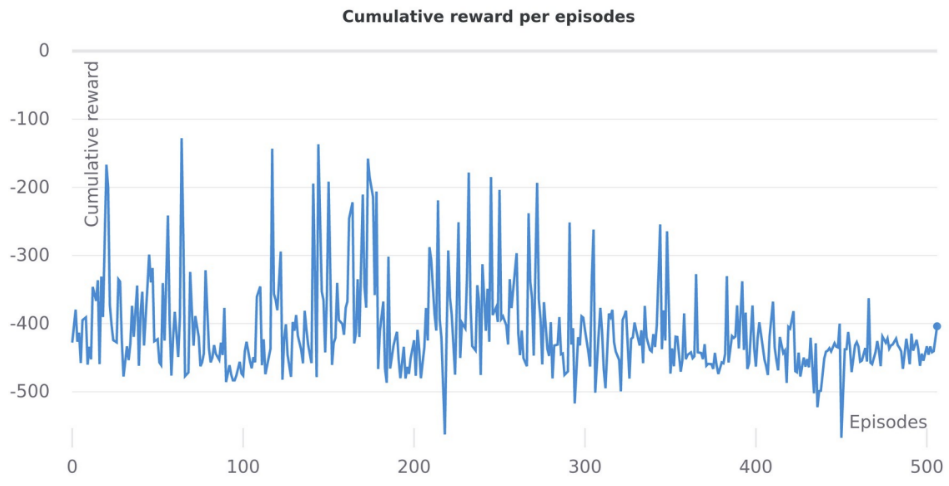
Actor-Critic Experimentation

For the experiments on the A2C algorithm we used the library of stable-baselines. We carried experiments on the A2C algorithm with different parameters. The one that contributed the most was the n step size. The n step size is the interval in which the agent will update its parameters and also its the interval when observations are made. Smaller step size may help the agent to learn better short term actions but it lacks in long term policies. Bigger step size may lead to a better policy but the agent might struggle on simple short term actions. Below there is the graph of the cumulative reward per episode of a training sequence with step size 16. The training took about 10 hours with only one environment due to hardware limitations.

As we have mentioned before, the policy gradient models show high variance. This experiment comes to verify our previous analysis. The model not only has high variance but it also finds it very difficult to converge. From another experiment that took another 10 hours with step size 32 we can see no difference.

Results

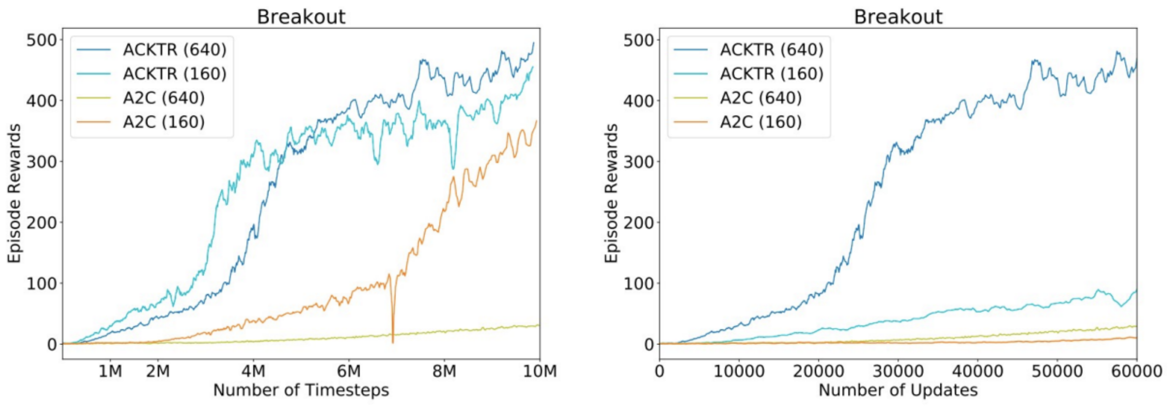




Our experiments yielded unsatisfactory results. These outcomes can be attributed to a variety of variables. Although the model itself and the physical constraints of our training appear to be the most relevant.

For starters, parallel settings enhance the A2C model. With such a high sample size, the collected data might achieve independent and identically distributed random variable status.

Our experiments are done in Google Colab, which provides a highly useful and pricey GPU for free. However, it only provides one CPU core, which is insufficient for parallelization and simultaneous execution of the tests. On the other side, we could run the experiments for a longer period of time, but this would not be realistic.



PPO Experimentation

Here are the hyperparameters that we used, while we train the model:

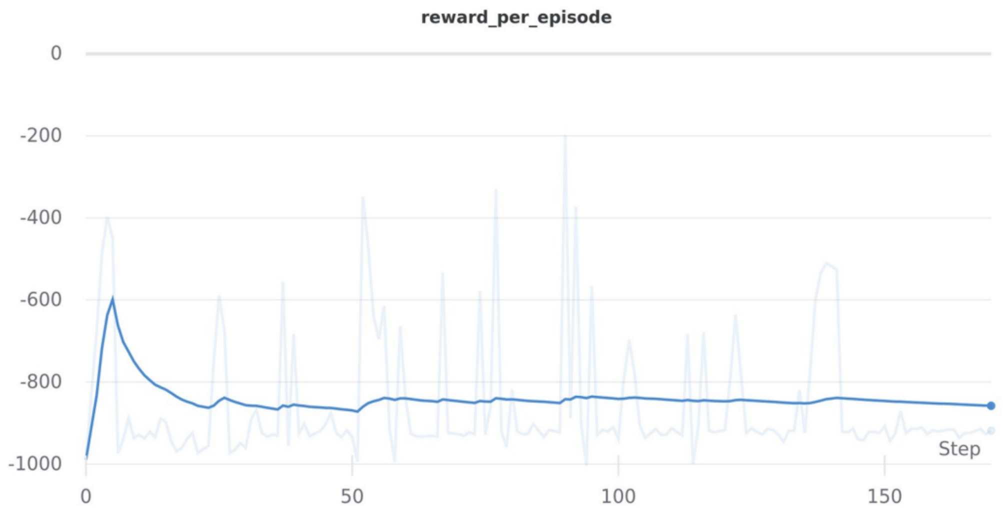
gamma	0.99
n_steps	64
ent_coeff	0.01
learning_rate	0.00025

vf_coeff	0.5
max_grad_norm	0.5
lam	0.95
nminibatches	4
noptepochs	4
cliprange	0.2

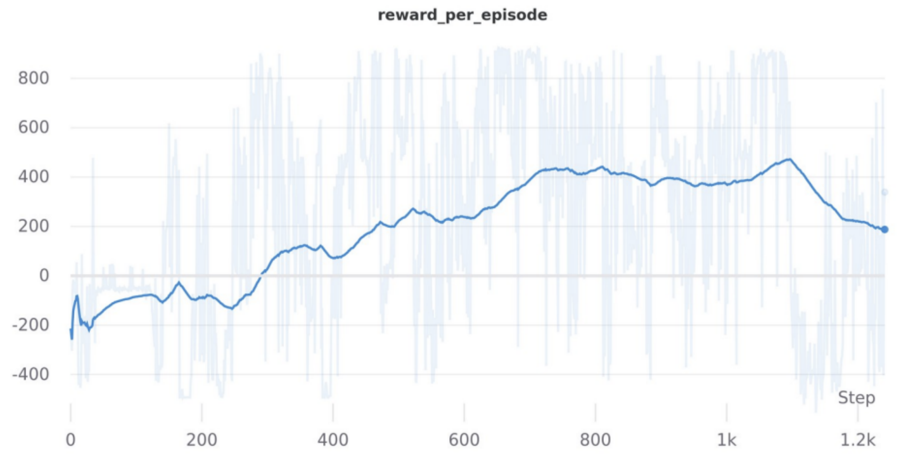
We should note that we did not experiment with the hyperparameters extensively because the default ones provide us with adequate performance. Furthermore, because of its simplicity, PPO is a simple algorithm that can be fine-tuned.

However, it is worth noting that we play about a lot with the number of timesteps and episodes. Our initial research focused on large numbers of timesteps and episodes (10000*100). We noticed that in the early footage of our Car Racing, where the car was unfamiliar with its surroundings, So we determined that it didn't require as much time to learn that grass isn't nice, and we cut the number of timesteps in half (5000*100). So now, whenever it moved into the grass portion, it didn't spend as much time there and had to restart from the beginning.

When we found the appropriate hyperparameters, we trained it for almost two days in Google Colab. Training these kind of Reinforcement Learning models is a very time consuming procedure. So we increased the number of episodes until 1.3k episodes. After almost 1100 episodes, 30 we observed that the reward per episode was decreased dramatically. So we decide to stop the training procedure after 2 days of training.



Example 1: *10.000 timesteps * 150 episodes*



Example 2: *5.000 timesteps * 1.200 episodes*

8. Conclusion

Finally, our study sought to investigate and apply three well-known reinforcement learning algorithms—Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Actor-Critic—within the demanding environment of OpenAI's CarRacing-v0 game. Despite limited processing power and time, our team took a systematic approach to understanding, implementing, and evaluating these algorithms.

While there were obstacles and deviations from theoretical expectations, the results revealed noteworthy effectiveness in training models that could efficiently navigate the virtual car. Real-world experimentation yields practical insights that add greatly to a broader knowledge of reinforcement learning algorithms in complicated gaming situations.

Not only did our research expose the subtleties of algorithmic behavior within the CarRacing-v0 framework, but it also addressed difficulties such as unanticipated behaviors and resource constraints. We hope to provide helpful information to scholars and practitioners working in comparable resource-constrained environments by offering a transparent description of our experiences.

Future Work

Our next work in combating catastrophic forgetting within the CarRacing-v0 project will focus on developing ways to improve the adaptability of our reinforcement learning models. Recognizing the intrinsic difficulty of forgetting previous experiences when confronted with new scenarios, we recommend delving deeper into approaches that dynamically allocate neural network resources to crucial parameters. We will investigate the usefulness of memory-augmented networks, which use the retention of key prior experiences to drive learning and reduce the impact of forgetting. Furthermore, we intend to incorporate transfer learning methodologies and ensemble methods that are suited to the CarRacing-v0 environment.

Our goal in integrating these tactics is to strengthen the model against catastrophic forgetting, allowing it to adapt to new tasks and situations while maintaining skill in previously acquired areas of the game. This future effort is consistent with our goal to improving reinforcement learning models for practical applications, as well as fostering adaptation and resilience in the face of dynamic difficulties in the CarRacing-v0 gaming environment.

9. References

[1] Huang, L., Ma, W., Wang, L., & An, K. (2023). Using Reinforcement Learning to Handle the Unintended Lateral Attack in the Intelligent Connected Vehicle Environment. *Journal of Advanced Transportation*, *2023*.

[2] Li, J. (2018). *The Simulation of Autonomous Racing Based on Reinforcement Learning* (Doctoral dissertation, The Ohio State University).

[3] Tammewar, A., Chaudhari, N., Saini, B., Venkatesh, D., Dharahas, G., Vora, D., ... & Alfarhood, S. (2023). Improving the Performance of Autonomous Driving through Deep Reinforcement Learning. *Sustainability*, *15*(18), 13799.

[4] Youssef, F., & Houda, B. (2019). Optimal Combination of Imitation and Reinforcement Learning for Self-driving Cars. *Revue d'Intelligence Artificielle*, *33*(4).

[5] Carton, F. (2021). *Exploration of reinforcement learning algorithms for autonomous vehicle visual perception and control* (Doctoral dissertation, Institut polytechnique de Paris).

[6] Ravichandiran, S. (2020). *Deep Reinforcement Learning with Python: Master classic RL, deep RL, distributional RL, inverse RL, and more with OpenAI Gym and TensorFlow*. Packt Publishing Ltd.