Using Reinforcement Learning for solving the Car Racing environment

1st Ichim Cosmin-Stefan
Graduate Student
Faculty of Computer Science
Iasi, Romania

***Abstract*—This paper explores the application of reinforcement learning to solve the "CarRacing-v2" environment from Farama Foundation, utilizing a modified Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. The approach integrates a pre-trained ResNet18 encoder for feature extraction and employs a combination of epsilon-greedy exploration with Ornstein-Uhlenbeck noise for efficient environment discovery. Despite showing promising strategies during training, the model's performance in testing underscores the need for extended training and further refinement.**

I. Introduction

This report delves into the development and implementation of a reinforcement learning agent tailored to solving, or at least try to solve, the Farama Foundation's "CarRacing-v2" environment.
The CarRacing-v2 environment presents a unique challenge: an agent must successfully navigate a car around a track, controlling its steering, acceleration, and braking. The objective is not only to complete laps but to do so with optimal efficiency and speed.
The proposed solution for this problem is a twist on the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, an actor-critic method that's effective in environments with continuous action spaces.
Even though the final results obtained within the training time available were poor (-35±5 for episodes of length 500) It's highly possible that with more training it would perform a lot better (500 episodes of length 500 were ran for the final form of the solution).

II. Literature Review

One of the best, if not the best results obtained while trying to solve the Car Racing environment is the one using the concept of "World Models," as explored in the paper by Ha and Schmidhuber (2018). The main idea behind their solution is besides using an encoder for obtaining the most important parts of a particular state, and a controller part that returns the action to be done, is the addition of a memory component. Compared to typical Reinforcement Learning solution, their memory is not solely used for storing state-action pairs, it is also as a generative model for predicting the future states starting from a particular state. In this way, the model has a greater context onto what's happening while also being able to plan it's moves ahead. This fact is attested by it's performance, that is better compared to other solutions.

Method	Average Score
DQN (Prieur, 2017)	343 ± 18
A3C (Continuous) (Jang et al., 2017)	591 ± 45
A3C (Discrete) (Khan & Elibol, 2016)	652 ± 10
CEOBillionaire (Gym Leaderboard)	838 ± 11
V Model	632 ± 251
V Model with Hidden Layer	788 ± 141
Full World Model	906 ± 21

TABLE I
CarRacing-v0 scores achieved using various methods

Taking this into consideration, the proposed solution is a lot closer in performance to a typical Deep Q Network solution, rather than the World Model.

III. The chosen approach

A. Describing the solution

The proposed solution for solving the challenges presented by the Farama Foundation's "CarRacing-v2" environment is a variant of the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. The TD3 algorithm, a variant of the actor-critic method, is particularly effective in environments characterized by continuous action spaces, making it well-suited for proposed environment.
The core of this solution is the same as for any actor-critic method, meaning it is based on two types of classes, actor ones and critic ones. The actor network is responsible for returning the actual input expected by the environment (steering, acceleration and breaking) based on the current state and in contrast, the critic network is responsible for taking state-action pairs and predicting the reward that is to be obtained in that context.
The main problem with the base form of the actor-critic architecture is the fact that the critic network is prone to becoming optimistic, thus making it's reward predictions unreliable. The Deep Deterministic Policy Gradient (DDPG) algorithm comes with a possible solution for this problem in the form of using two such critic members, whose initializations and training processes are separate, and the reward prediction for a given state-action pair is given to be the minimum prediction between these two. In this way the potential of becoming optimistic is greatly decreased.
However, another idea present in this solution comes from

TD3 (Twin Delayed Deep Deterministic Policy Gradient) Learning method, meaning that, in order to make sure that the actor and critic networks are as stable as possible, copies for each of them are created. The main idea is that, for the actor network, one actor network will be kept as an online copy and one as a target copy and during training the predictions will be done by the target network, but the backpropagation of the error will be done on the online network. The target network being updated to the parameter values of the online network once every couple of steps. This is all done, in theory, for keeping the solution as stable as possible.
Another point that can be either potentially good or potentially problematic is the use of a pre-trained encoder (ResNet18). This decision was made in order to minimize training time, not needing to focus on building and training a separate encoder but only allowing just some of the layers to be used in backpropagation. It was thought to be a good idea in the spirit of saving training time because the general knowledge of the encoder can be kept, while making sure to leave some place for adjusting the the environment at hand. In hindsight it would have been better to create a custom encoder for this environment, similar to the World Models paper, because it would have allowed for further temporal predicting power, not just the current and the next state.
Another important point of the solution is the way in which the agent discovers the environment and the balance between exploration and exploitation. This was achieved by using both epsilon-greedy exploration and Ornstein-Uhlenbeck noise. The epsilon-greedy exploration is used for choosing random actions with a given probability and Ornstein-Uhlenbeck noise is being added when an action is chosen by using the actor model, in order to further add to the exploration while not using simply random actions.
Another part thought to improve the agent's decision making is the recomputing of the rewards. It was observed that the agent would tend to get stuck or to simply chose actions with negative rewards. To combat this, a method was developed where consecutive negative rewards progressively decrease, while consecutive positive rewards progressively increase. This approach aims to discourage the tendency to persist in taking actions that result in negative rewards, while simultaneously encouraging actions that yield positive outcomes as much as possible.

B. Benchmark Performance

The main problem of the current solution is the fact that the training time for it's final form has not been enough. Even though during training it has achieved scores up to 60, in testing the final form either does not move or spins in circles achieving negative scores (-35$\pm$5 for episodes of length 500). This solution would benefit from a lot more training because, by adding a very small amount of noise to the action chosen by the agent it is able to "stay inside the lane" while also showing ability to turn with a curve in the road, even though it's placement is not necessarily on the road.

C. Ablation Study

In trying to solve this problem the solution started off from it's simplest form, that being a Deep Q learning solution. It's results were poor even after a lot of training and it didn't show any signs of "smartness" and because of that the solution pivoted towards the improvement of Deep Q learning, Double Q learning, while ignoring the fact that these types of agents perform well in discrete action spaces. As expected, the Double Q learning solution performed as well as the Deep Q learning solution, meaning really poorly. All the cumulative rewards during training were negative even after 1500 episodes of length 500.
Seeing that result another solution architecture was being searched for, one that would perform well in continuous action spaces and the TD3 algorithm was found, it being an improvement on the basic actor-critic architecture, while also being more stable than the DDPG (deep deterministic policy gradient) algorithm. This can be attributed to the fact that it employs, similarly to the Double Q learning algorithm, the concept of online and target networks.

IV. Improvement steps

Future improvements to the proposed solution can be explored in several areas to enhance its performance and efficiency:
1. Extended Training: Given the observation that the model shows potential in training but lacks sufficient performance in testing, an extended period of training could be beneficial. More training episodes might allow the model to refine its strategies and adapt better to the environment.
2. Custom Encoder Design: While the pre-trained ResNet18 encoder offers a good starting point, designing a custom encoder tailored specifically for the "CarRacing-v2" environment could yield better results. This encoder could focus on extracting more relevant features for the task, potentially improving decision-making and understanding of the environment of the agent.
3. Architectural Enhancements: Exploring architectural modifications, such as adding more layers or altering the network structure, could provide improvements. Additionally, integrating techniques from successful models like the World Models approach could introduce beneficial predictive capabilities.

V. Conclusions

This study presented a unique approach to solving the "CarRacing-v2" environment using a modified TD3 algorithm. It highlights the challenges faced in reinforcement learning tasks with continuous action spaces and underscores the importance of algorithmic and architectural choices in designing effective solutions. Future work, as outlined in the improvement steps, will focus on enhancing the model to achieve better performance and efficiency.