CentraleSupélec

Reinforcement Learning Project
Report

Highway Env

Students :
Abdelillah Bourchad : Part 2
Mohamed Aziz Triki : Part 3
Lotfi Kacha: Part 1
Ibrahim Ramdane: Part 1

Teacher :
Hédi Hadiji

GitHub

April 27, 2024

Contents

1 Part 1 : Discrete Actions : Highway

The "highway-v0" environment is a simulation where an agent controls a vehicle on a busy highway. The primary goal is to manage the vehicle's position and speed to navigate safely and efficiently through traffic without collisions. The environment presents complex scenarios involving multiple vehicles, requiring the agent to adapt its strategies based on the dynamic conditions of the highway. The agent's challenge is to balance the task of maintaining high speeds for efficiency while ensuring safety and lane discipline.

1.1 Configuration

We configured the simulation to closely mimic real-world driving conditions on a highway with multiple lanes and varying traffic patterns. This configuration is critical for training the agent to handle real-life driving tasks such as overtaking slower vehicles and avoiding collisions. We integrated a reward system that incentivizes the agent for keeping a safe distance, staying in the right lane unless overtaking, and maintaining a steady high speed within legal limits. Here are the details of the most important parameters:

Category	**Parameters**	**Values/Description**
Action Space	Type of Action	Discrete, [Lane Change, Speed Adjust]
State Space	Observation Grid	$[[-100, 100], [-100, 100]]$, [5, 5]
Reward Space	Safety, Efficiency, Lane	-1, 0.1, 0.5

Table 1: Summary of key spaces in the Highway environment configuration

1.2 Algorithmic Approach for Policy Optimization

Our approach utilized a Deep Q-Network (DQN) algorithm, designed to maximize cumulative rewards through strategic decision-making in a highway driving context. The neural network, representing the agent's decision-making policy, denoted as $Q(s, a)$, where s represents the state of the environment, and a represents the chosen action. In each training episode, the agent interacted with the environment by selecting actions based on its current policy. The outcomes of these actions provided rewards that were used to assess and update the agent's decision-making strategy.

The policy update involved solving the optimization problem to maximize the expected return, with the DQN loss function defined as follows:

$$L(\theta) = E_t \left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)^2 \right]$$

Where γ is the discount factor, r_t is the reward at time t, s_t is the state at time t, and a_t is the action taken at time t.

1.3 Results

During the training sessions, we observed a gradual but slow improvement in the rewards per episode. The agent was trained over 100 episodes, starting with maximum score around

25. Over time, and particularly towards the end of the training period, the average reward per episode improved a little bit and achieved sometimes values like 30-40.

The reward improve very slowly, maybe it needs more time to converge. But with only 100 episodes it looks like the results are not satisfactory

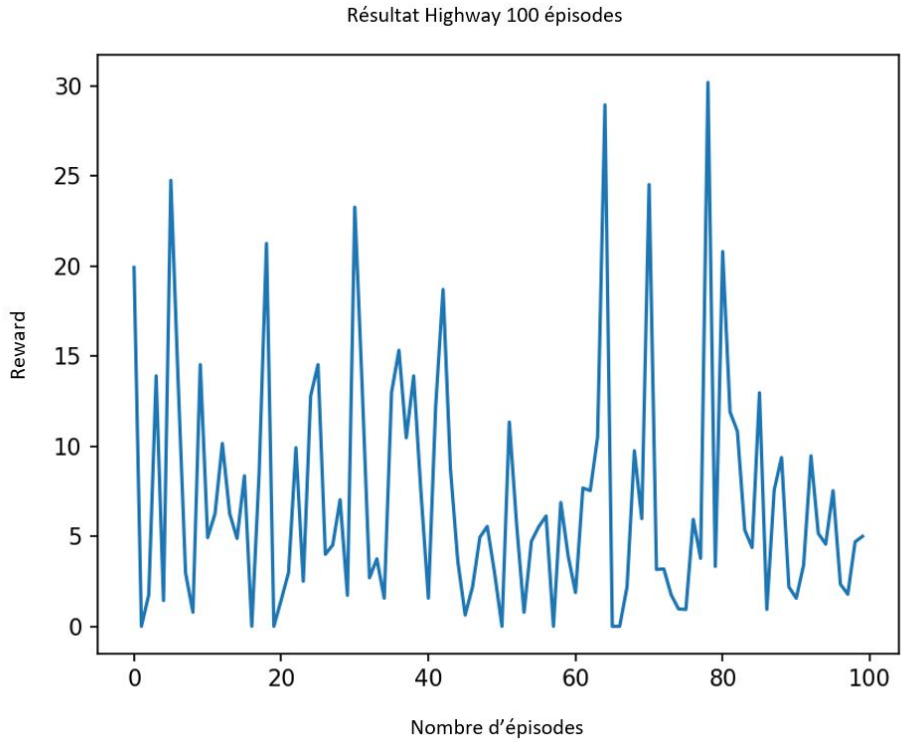


Figure 1: Descriptive caption of the image.

2 Part 2 : Continuous Actions : Racetrack

The "racetrack-v0" environment we explored is a driving simulation where an agent controls a vehicle on a race track. The goal of this environment is to teach the agent to navigate the circuit as quickly and efficiently as possible, while avoiding going off the road. The track is represented visually, allowing the agent to receive visual information from which it can learn to perform maneuvers. The challenges are taking tight turns and making decisions to keep the vehicle on the track. The agent receives rewards for desirable behaviors such as staying on the road, while penalties are applied for undesirable actions, such as leaving the road or hitting other vehicles.

2.1 Configuration

We started by defining specific parameters to guide actions and perception of the environment. This setup is essential to help the officer navigate through challenges like sharp turns and make the right decisions to avoid going off the road. We have integrated a reward system to encourage the agent to keep the vehicle on the track and penalties to

dissuade him from risky behavior defined previously. Below are the details of the most important parameters:

Category	**Parameters**	**Values/Description**
Action Space	Type of Action, Target Speeds	Continuous, [0, 5, 10]
State Space	Grid Size and Step of Grid	([-18, 18], [-18, 18]), [3, 3]
Reward Space	Action, Lane Centering, Collision	-0.2, 1, -1

Table 2: Summary of key spaces in the Racetrack environment configuration

2.2 Algorithmic approach for policy optimization

Our algorithmic approach, inspired by Proximal Policy Optimization (PPO), is based on the maximization of cumulative rewards by the agent in a racing environment. To do this, we use a neural network representing the agent's policy, denoted $\pi_\theta(a|s)$, where θ are the weights of the network, s represents l 'state of the environment, and a is the chosen action.

In each training episode, the agent interacts with the environment, choosing actions according to its current policy. The rewards obtained are used to evaluate the agent's performance. We then calculate the benefit $A(s, a)$, measuring how much better than average each action was in a given state, using the rewards obtained.

The policy update is performed by solving the following optimization problem:

$$\max_\theta E_t[L_t(\theta)]$$

$Where \mathrm{L}_t(\theta)$ is the PPO loss function defined by:

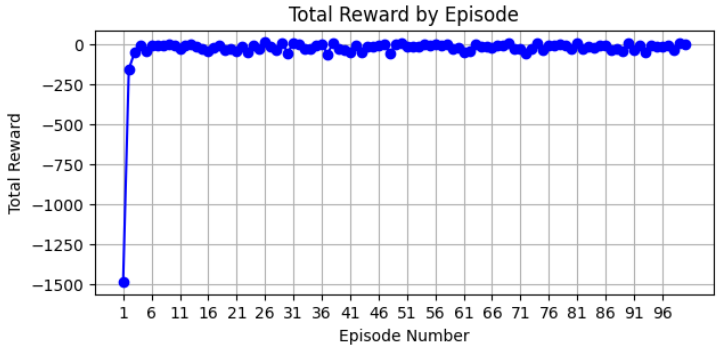
$$L_t(\theta) = E_t\left[\min\left(\pi_\theta(a_t|s_t)\pi_{\theta_{old}}(a_t|s_t)A_t, \mathrm{clip}\left(\pi_\theta(a_t|s_t)\pi_{\ theta_{old}}(a_t|s_t), 1-\epsilon, 1+\epsilon\right)A_t\right)\right]$$

Where ϵ is a clip hyperparameter, s_t is the state at time t, a_t is the chosen action at time t, and $\pi_{\ theta_{old}}(a_t|s_t)$ is the previous policy.

This approach allows the agent to gradually learn to optimize its actions to maximize future rewards, while taking past performance into account. Just to clarify, we did not use the loss function directly $L_t(\theta)$ of Proximal Policy Optimization (PPO). However, we implemented a similar form of policy optimization using a Mean Squared Error (MSE)-based loss function to update the policy model weights."

2.3 Results

Finally, we trained our model over 100 episodes to maximize the agent's performance in the racing environment. During these episodes, the rewards obtained by the agent varied between -100 and +13.54, thus demonstrating a non-progressive variation in his success. By tuning the hyperparameters of our policy, such as learning rate, gamma reduction factor, and epsilon exploration rate, we successfully maximized the agent's cumulative rewards. The final validation of this method is done by a visual representation which confirms our solution.



3 Part 3

The environment is called "intersection". The objective is for the car to be able to turn left on the intersection without colliding with the other cars.
The agent has no control over the trajectory. There are three available actions: accelerate, decelerate or stay idle. The observations are the absolute positions of all other 15 vehicles. Note that the other vehicles may collide, since the implemented system is simple and may fail. We hope that the agent would simply stop and wait for the episode to end if this happens (relatively rare event).
There are 3 rewards. One negative reward when the vehicle collides with another car (which also marks the end of an episode), one positive reward when reaching the end point of the left turn and another positive reward when the vehicle has a high speed (to encourage the vehicle to complete the turn as fast as possible).

The aim of this part is to train an agent on this environment using the StableBaselines library. The library is very easy to use and the environment is easily integrable into the workflow. We will use three different learning algorithms to train our agents: A2C, PPO and DQN.
Given that the observations are positions of other cars, which are represented as numerical data, the Multi-Layer Perceptiron Policy should work well. We also start with a high discount factor $\gamma = 0.99$.
The rewards are kept at their default value: $collision = -5, arrival = 1, high\_speed = 1$.

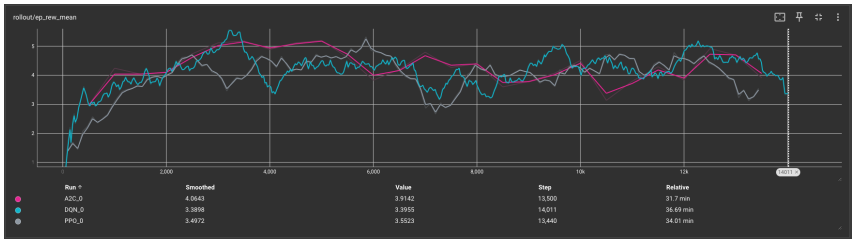


Figure 2: Mean reward $\gamma = 0.99, collision = -5, arrival = 1, high\_speed = 1$

No steady improvement of the mean reward. The episode length has a very similar shape. The best performing models never slow down even if it leads to a guaranteed collision.
In order to change this behaviour, we try to making the collision and arrival extreme, while keeping a high discount factor.

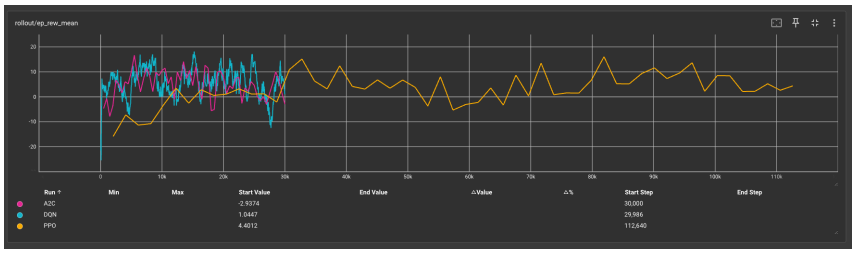


Figure 3: Mean reward $\gamma = 0.99, collision = -50, arrival = 50, high\_speed = 1$

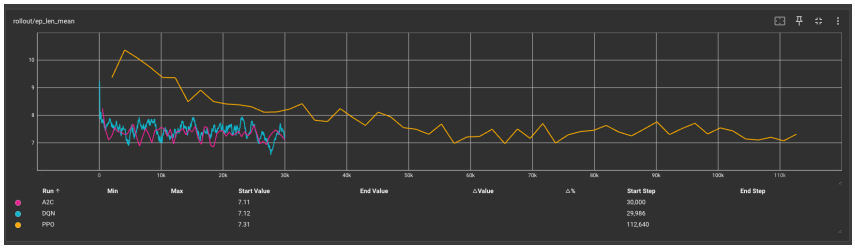


Figure 4: Mean episode length $\gamma = 0.99, collision = -5, arrival = 1, high\_speed = 1$

This time, we train much longer (especially for the PPO model because it seemed more stable than the other models). The results are disappointing: The agents speed up even more than before (never slow down). This is because on many occasions, this strategy leads to arrival. So the only way for the model to obtain the high reward is to go as fast as possible and potentially dodge all the incoming traffic before it arrives. The episodes are as a result much shorter (when crashing or succeeding), which can be clearly seen on the mean episode length curve. Setting the reward $high\_speed = 0$ gives similar results.

Next we try setting $\gamma = 0.6$, with the default rewards, to let the agent think more "medium-term" and avoid collisions. The results are still bad. However, the agent sometimes decelerates but still cannot avoid collisions.

We tried many different combinations of the mentionned settings, but none seemed to give good results. We believe that what makes the learning so difficult is the high frequency of episodes where it is possible to succeed while only accelerating. This is because no cars are initially within the intersection. We attempted modifying the environment to have some cars initially spawn inside the intersection but were unsuccesful. One other major difficulty we faced is the time that it takes to train the agents.
This task shows the difficulties of succesfully training an agent, despite the simplicity of the task at hand (3 discrete actions). The next steps would be to try other policies (use LSTMs per example) and varying other hyperparameters to ensure the mean reward increases over time.