Local Path Planning

CSCE Summer Intern Project for Undergraduate Students

Weekly Report

Syed Mustafa

Department of Computer Science
Texas A&M University

July 17, 2023

Contents

1 Introduction

This weekly report mainly talks about the deep reinforcement environment that I have set up. The report is structured as follows: section 2 describes the environment in terms of its Markov decision process, i.e. it goes over its state, action, and reward of the environment.section 3 goes over the python API used to implement the environment. section 4 goes over the DRL algorithm used, namely DQN. **??** describes the issues faced. Lastly, section 5 talks about the plan moving forward.

2 Markov decision process

Markov decision process (MDP) is a classic formalization of sequential decision-making. A typical Markov decision process is a mathematically idealized form of reinforcement learning [1]. An MDP can be described as a tuple of five values $(S, \pi, r(s,a), v(s), A)$ where: S is the set of all possible states of the environment, π is the policy function, i.e. given a state the function should return an action. $r(s,a)$ is the reward function, i.e. $r : S \times A \to \mathbb{R}$. A is the set of all actions.
$v(s)$ is a function that maps out the state of the environment with some real number. The function tells how good it is for the agent to be in a particular state. The value function is sometimes changed with the action value function $q(s,a)$. It shows how good it is for the agent to perform a given action in a particular state.

$$q : S \times A \to \mathbb{R}$$

A famous reinforcement learning algorithm, Q-Learning. The algorithm defines the q-table as:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t)\alpha[R + \gamma max_a Q(S_{t+1}, a) - Q(S, A)]$$

. Since in our project we are working with Deep reinforcement learning, i.e. DQN, we will mostly be concerned with the action value function $(Q(S, A))$
For the project, I modelled an MDP on the Carla environment, Figure 1 gives a high-level description of the MDP environment.

2.1 Action function

The agent can perform 4 functions, i.e:

1. Go left
2. Go Right
3. Go Forward
4. Follow the Global Path using the pure pursuit controller, that I had previously implemented.

2.2 Reward Function

The reward function would return the reward and done flag. The reward was based on the The reward function is inspired by [3]. The reward set is denoted by R such that

$$R = \{r_1, r_2, r_3, r_4, r_5\}$$

. The function is as follows:
For $r \in R$

- $r = r_1$ if the agent is not on the global path. Where $r_1 = -1$.
- $r = r_1 + r2 * N$ if the agent is on the global path where N is the number of steps since the agent was on the global path and $r_2 = 2$.
- $r = r_3$ if the agent has collided. Where $r_3 = -1000$.
- $r = r_4$ if the maximum number of time steps has been reached. Where $r_4 = -1200$.
- $r = r_4$ if the end is reached where $r_5 = 1000$.

The done flag only tells if the episode is finished. The reward function was created to encourage the agent to follow the global path but if a dynamic object is on the path then it would follow a different route

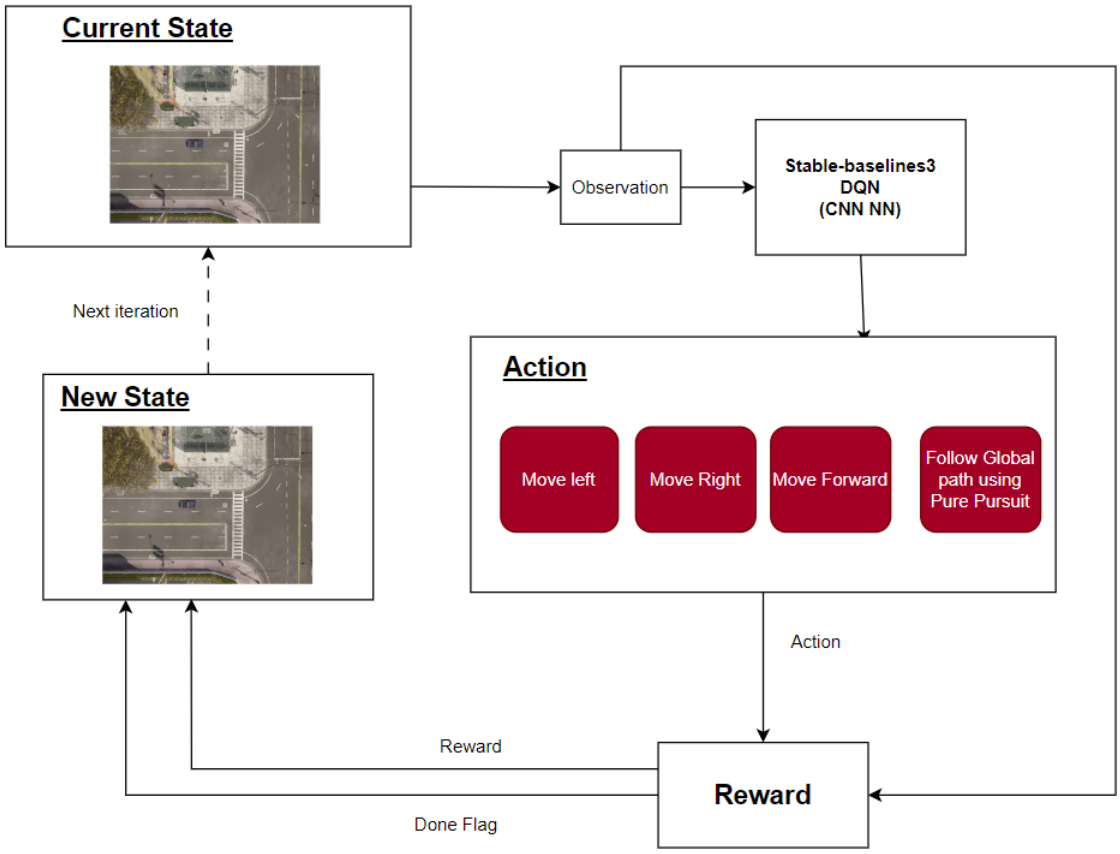

Figure 1: System Diagram

3 Gymnasium and stable-baselines3 implementation

Figure 2 is a diagram to explain the implementation of the above stated MDP. The Gymnasium framework requires the three functions: `__init__()`, `step`, `reset()`.
Initialization: The main purpose of the init() function is to define the action and observation space. We have 4 actions. They can be represented by discrete numbers/ The observation is a 480x640 image. For that gymnasium provides the space type Box. The box is essentially like an array except each element has a lower and upper bound. Thus spaces are implemented as

```
self.action_space = Discrete(4) #Left, Up, right, follow pure pursuit
self.observation_space = Box(
        low=0, high=255, shape=[IMG_HEIGHT, IMG_WIDTH, 3], dtype=np.uint8)
```

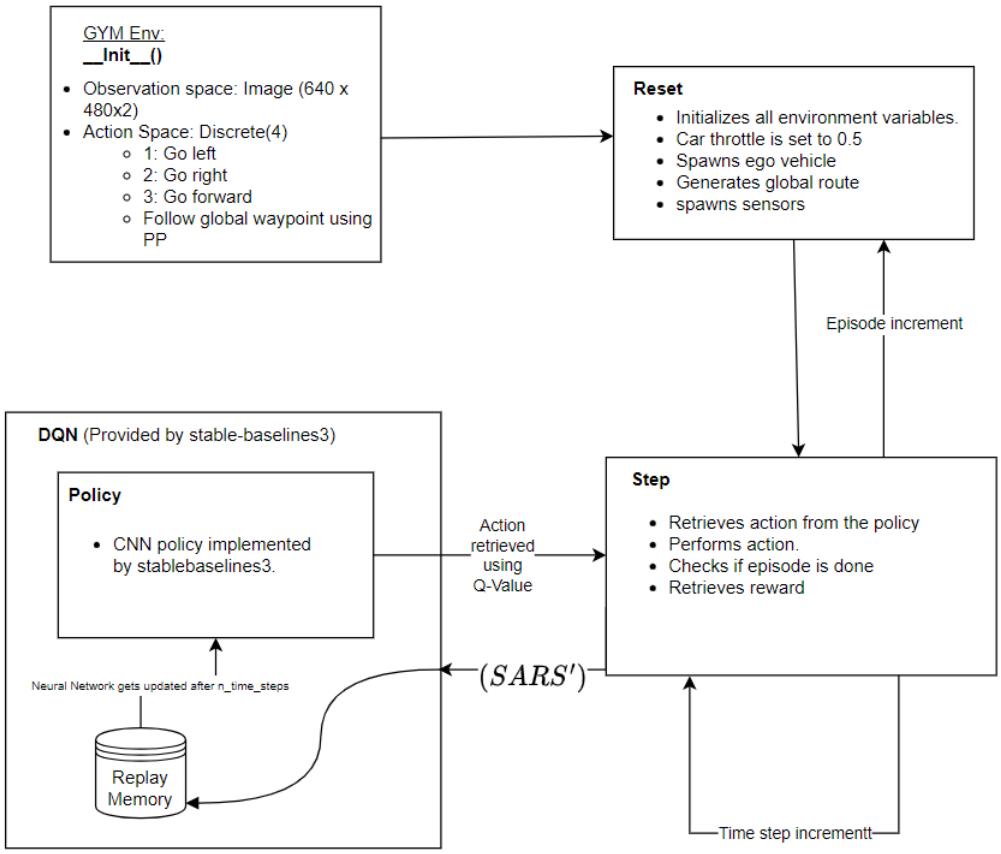

Figure 2: High-level diagram of the system

4 Deep Q Network

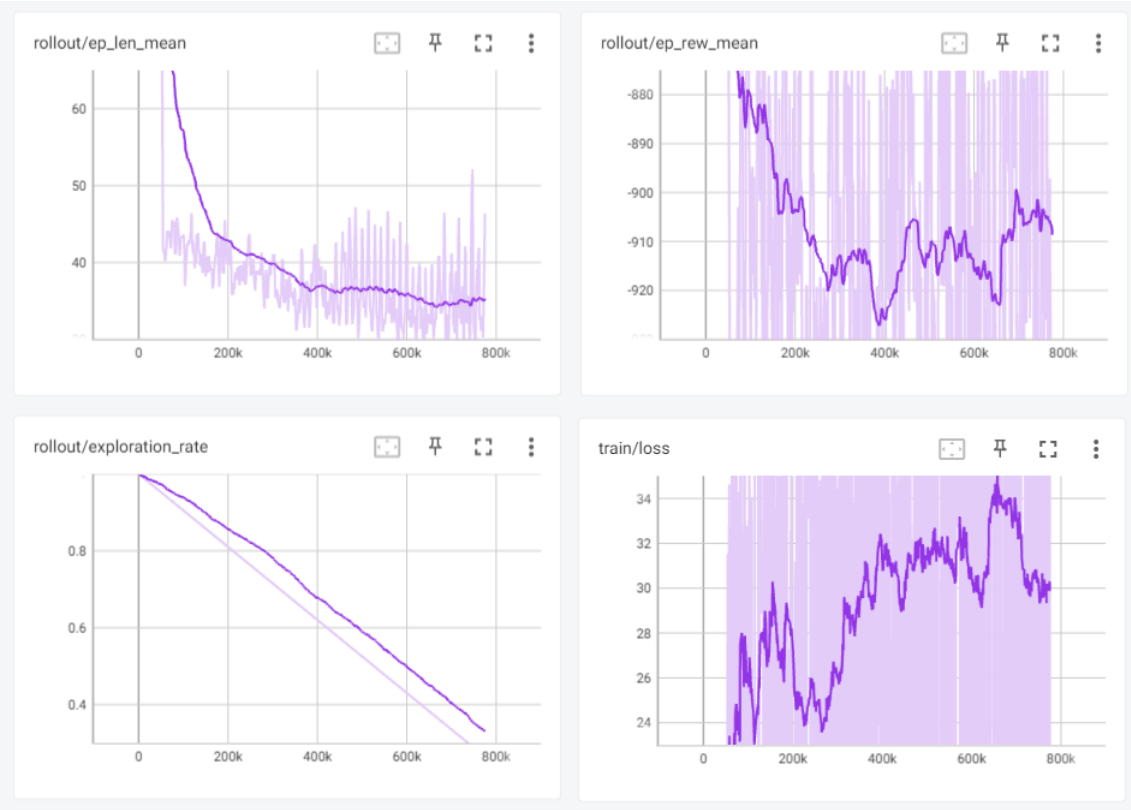

Figure 3: Tensorboard Result

Figure 3 shows poor results. It seems that the agent is actually getting worse since the loss is increasing, and the episode average reward is decreasing. This could be due to two reasons:

1. I haven't implemented the code properly. Thus the code does something different to what I want it to do.

2. The MDP used to model the environment is not good enough. For example, even though there is a high reward for getting to the endpoint there is no observation with relation to the endpoint. I.e. the end_point is not visible in the image. So this could be one of the future work.

5 Conclusion and Future Work

1. Firstly, I need to increase my knowledge of Deep reinforcement learning and the DQN implementation of stable baselines 3. This is because the model is not learning at all, i.e. it is not maximizing its reward over time.

2. I found this interesting thing in the stablebaselines3 documentation: . Policy in this context is the neural network used to approximate the Q-table ($v(s, a)$).

3. We are simply using a convolutional neural network provided by the stable-baselines3 API. The next step in the research could be incorporating a more complex neural network into the mix.

4. Currently, I am using a discrete representation of the action space. I could potentially use `gymnasium.Spaces.Box`. This would return actions as a list with each element of the array representing an action. The value of each element would be within a specified upper and lower bound. This would allow me to get precise steering values between negative one and one. Moreover, I could incorporate the acceleration/throttle values into the mix as well.

References

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[2] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

[3] Wang, Binyu, et al. "Mobile robot path planning in dynamic environments through globally guided reinforcement learning." IEEE Robotics and Automation Letters 5.4 (2020): 6932-6939.