REINFORCEMENT LEARNING (CS60077)

TERM-PROJECT (PHASE-2)

PRANAV NYATI

20CS30037

Controlling a Toy Car around a Continuous Grid [Version 2] (Project Code: CCV2)

(A) Problem Description:

- **Objective:** Control a car around a continuous circular track and complete the lap (starting from the start position and returning after a full lap) without going off the track.
- **Tasks:**
    - Model the state space and reward system for the problem
    - Implement the Deep RL Algorithms as follows:
        - DDQN
        - REINFORCE
        - GAE
        - A3C

(B) Environment, State Space, and Reward Formulation:

- **Environment:**
    - A circular continuous grid with a pre-defined track on which the car has to complete the lap. The Environment is simulated using the pygame interface.
- **State Space:**
    - Each state is a tuple of size **eight** the form **(sensor1_val, sensor2_val, sensor3_val, sensor4_val, sensor5_val, car_x, car_y, car_angle),** where the **(car_x, car_y)** denotes the **current position of the car in the grid,** and the **car_angle** is the orientation of the car wrt to the horizontal and can take values from **[-pi to pi].**
    - At each time step, the car can choose to either **turn a little towards left, turn a little towards right**, or **go straight.** The speed of car is constant.
    - **Note:** the top left corner of the window has the coordinates (0, 0), and the bottom-right one has the coordinates (window_size-1, window_size-1)
    - I have also created two circular regions concentric to the track, one of smaller radius (black-colored) and one of larger radius (green colored), to restrict and reflect back the car when it tries to go across the boundary of those two circles. This was done to prevent the car from roaming into

unwanted states, so as to train all the Deep RL agents involving neural networks properly. There is a negative rewards also when the car hits the boundary of those two circles so as to make it learn not to avoid those regions.

- **Reward structure:**

```
      self.reward_inside_path = 2 # reward to the agent if it stays inside the track
      self.reward_outside_path = -15 # high penalty for going out of the track
      self.score = 0
      self.time_penalty = -1  # penalty for each time step to encourage fast complete of lap using shortest path
      self.angle_change_penalty = -10  # to prevent the agent from learning a jittery policy
      self.goal_reward = 5000 # very high reward for reaching goal
      self.three_quarter_lap_reward = 1500  # reward for reaching 3/4th section of lap; this is to prevent the car from turning back after 1/2 lap
      self.no_access_hit_penalty = -200 # penalty for hitting the inner and the outer no-acess circles
      self.obstacle_sense_penalty = -5 # penalty when majority of the sensors are outside the track
```

(C) DDQN Algorithm:

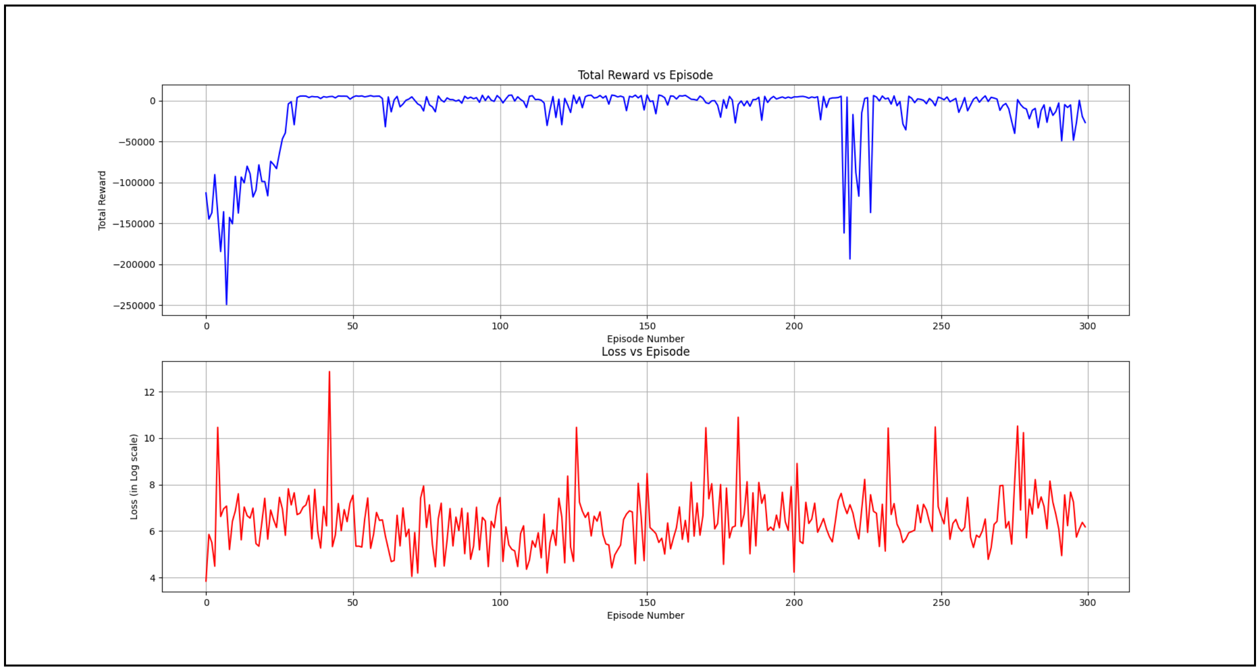
- **Configurations:**
    - Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 3.

    - **Other configurations:**

```
learning_rate = 5e-4
```

```
buffer_size = int(1e5)
batch_size = 64
discount_factor = 0.99
update_target_freq = 5
tau = 1e-3
epsilon_start = 0.2
epsilon_end = 0.0002
epsilon_decay_rate = 1e-6
hidden_dim1 = 64
hidden_dim2 = 64
max_t = 8000 # maximum number of timesteps per episode
max_episodes = 300 # maximum number of episodes to train the agent
```

- **Training curve:**



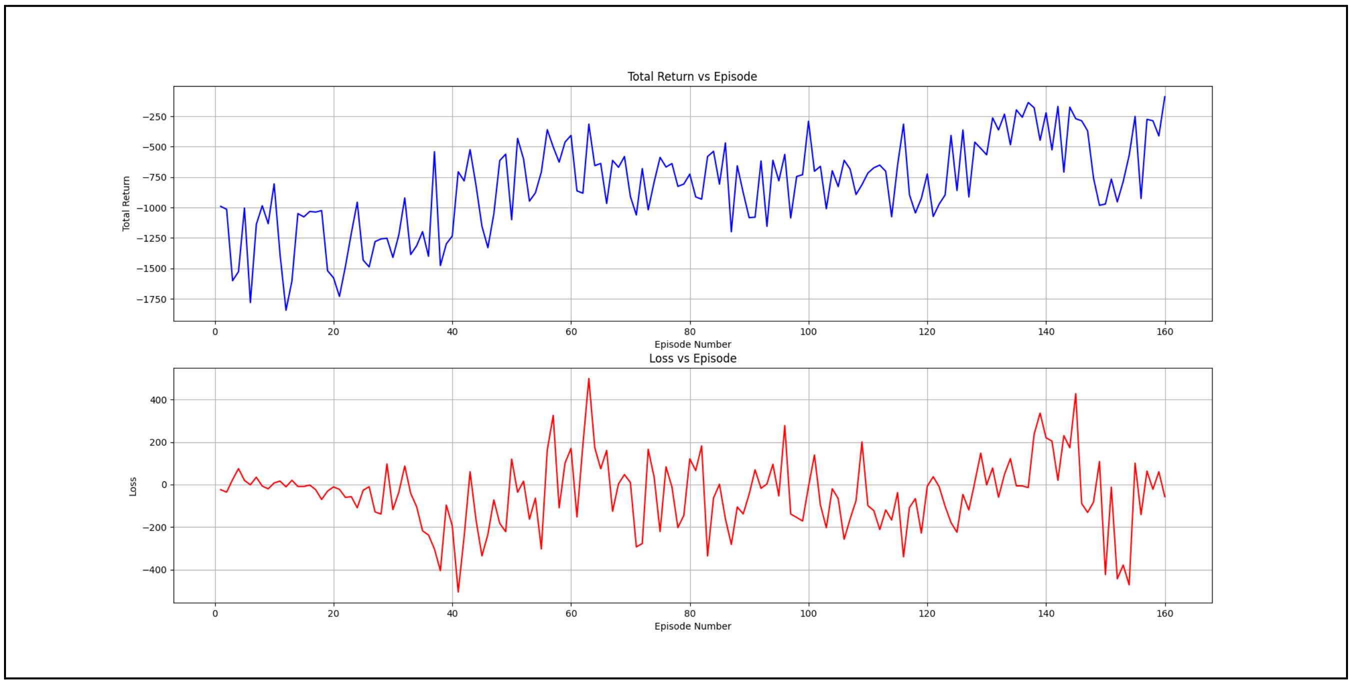
(D) REINFORCE Algorithm:

- **Configurations:**
  - Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 3, followed by softmax layer

- **Other configurations:**

```
learning_rate = 1e-3
discount_factor = 0.99
hidden_dim1 = 128
hidden_dim2 = 64
max_t = 40000 # maximum number of timesteps per episode
max_episodes = 160 # maximum number of episodes to train the agent
```

- **Training curve:**



(E) GAE Algorithm:

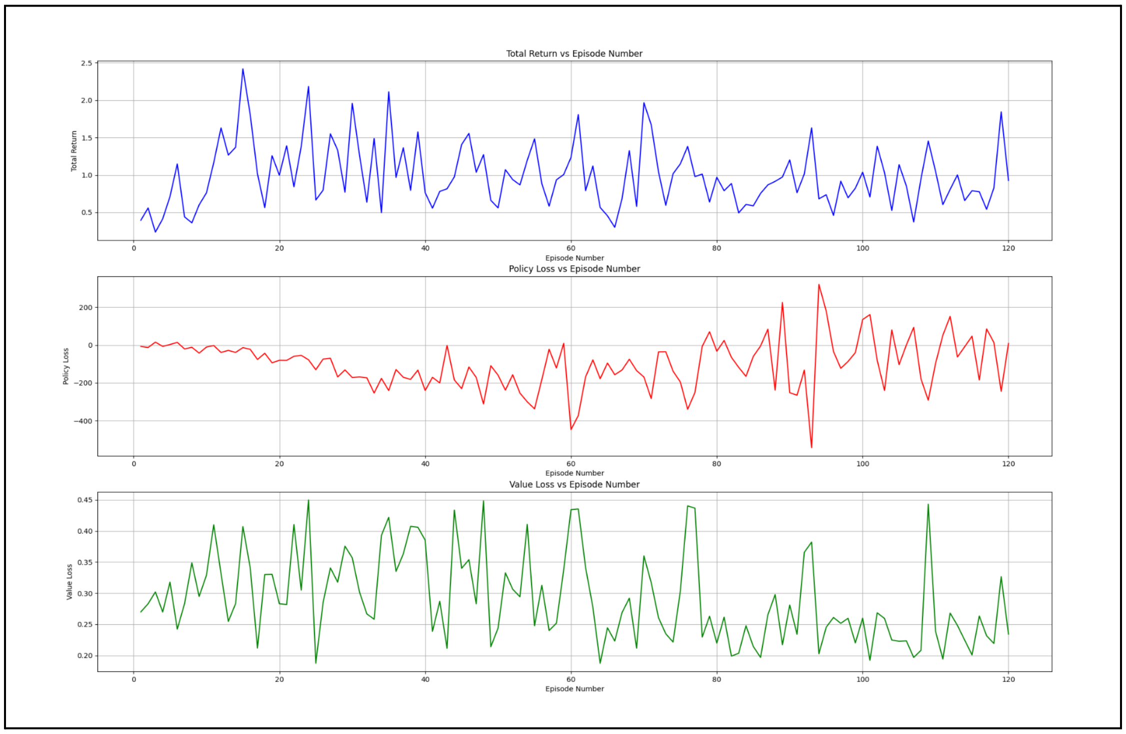
- **Configurations:**
  - Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 3, followed by a softmax layer for the **actor** component of the agent. Also used Droupout for robust learning.

- ○ Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 1, for the **critic** component of the agent. Also used Droupout for robust learning.

- ○ **Other configurations:**

```
learning_rate = 5e-4
discount_factor = 0.99
trace_decay = 0.95
hidden_dim1 = 64
hidden_dim2 = 64
dropout = 0.1
max_t = 15000 # maximum number of timesteps per episode
max_episodes = 125 # maximum number of episodes to train the agent
```

- **Training curve:**



(F) A3C Algorithm:

- **Configurations:**
  - ○ Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 3, followed by a softmax layer for

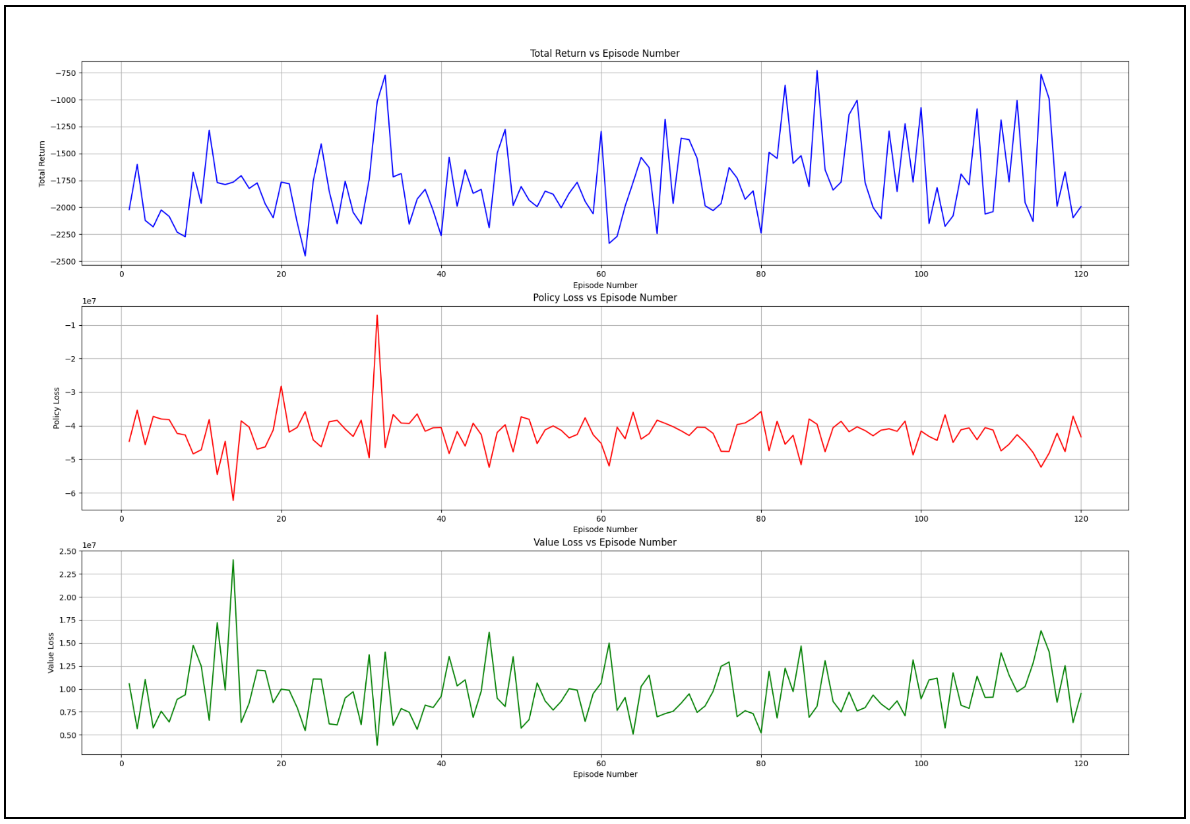
the **actor** component of the agent. Also used Droupout for robust learning.

- Used a neural network with 1 input layer of size 8, two hidden layers and 1 output layer of size 1, for the **critic** component of the agent. Also used Droupout for robust learning.

- **Other configurations:**

```
learning_rate = 5e-4
discount_factor = 0.99
trace_decay = 0.95
hidden_dim1 = 64
hidden_dim2 = 64
dropout = 0.1
max_t = 15000 # maximum number of timesteps per episode
max_episodes = 125 # maximum number of episodes to train the agent
```

- **Training curve:**


Total Return vs Episode Number
Total Return
Episode Number
Policy Loss vs Episode Number
Policy Loss
1e7
Value Loss vs Episode Number
Value Loss
1e7
Episode Number