Ninad Kale : 50478031

Question 1

Objective: Train stable baseline implementation of DQN on environment with discrete and continuous observation space

Description: DQN algorithm learns state-action value function using deep neural network which given discrete or continuous observation space gives Q value for each action.

Configuration: I used all default parameters as when I changed them I figured training was not converging. Standard model configuration is as follow

```
# Instantiate the agent
model = DQN("MlpPolicy",
            env,
            verbose=0,
            tensorboard_log="dqn_training_cartpole",
            learning_rate = 0.0001,
            buffer_size = 1000000,
            learning_starts = 50000,
            batch_size= 64,
            tau= 1,
            gamma= 0.99,
            device = "cuda",
            max_grad_norm = 10,
            exploration_initial_eps = 1,
            exploration_final_eps = 0.05,
            exploration_fraction = 0.2,
            target_update_interval = 10000,
            )
```

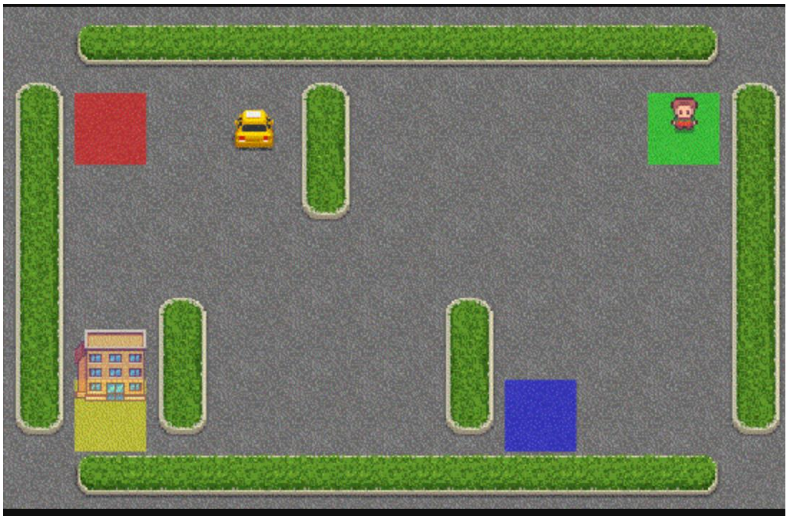
I used MlpPolicy as it expect observation in 1D array. Other options are CnnPolicy

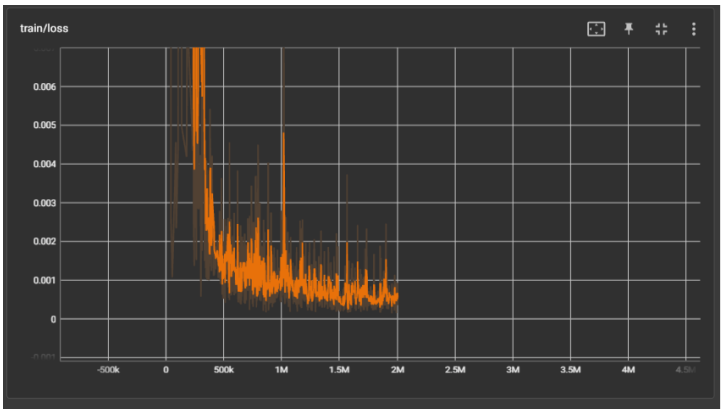
Training: I trained for 10000 timesteps for total 200 epoch. I trained it in epoch batch of 25 to make sure model is not overfitting.

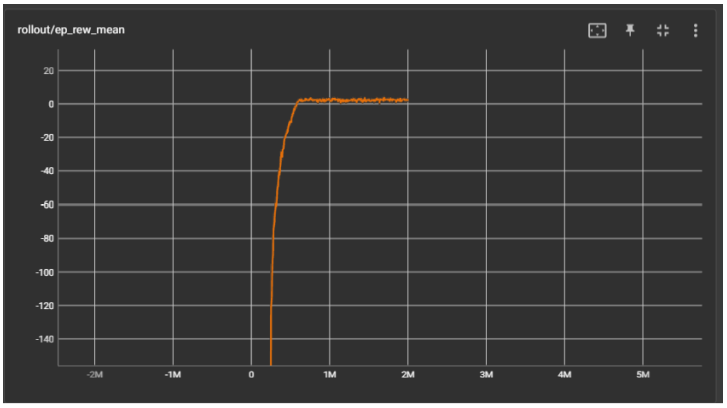
Testing: I ran trained model for 100 episode and measured total episodic reward till termination to check if problem is solved or not

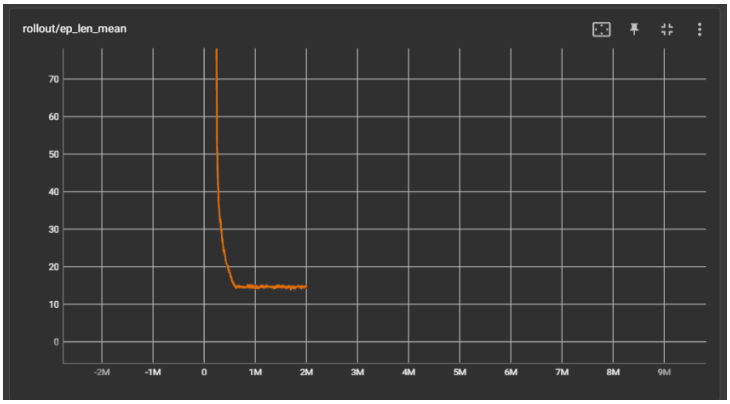
Run visualize.ipynb for visualization

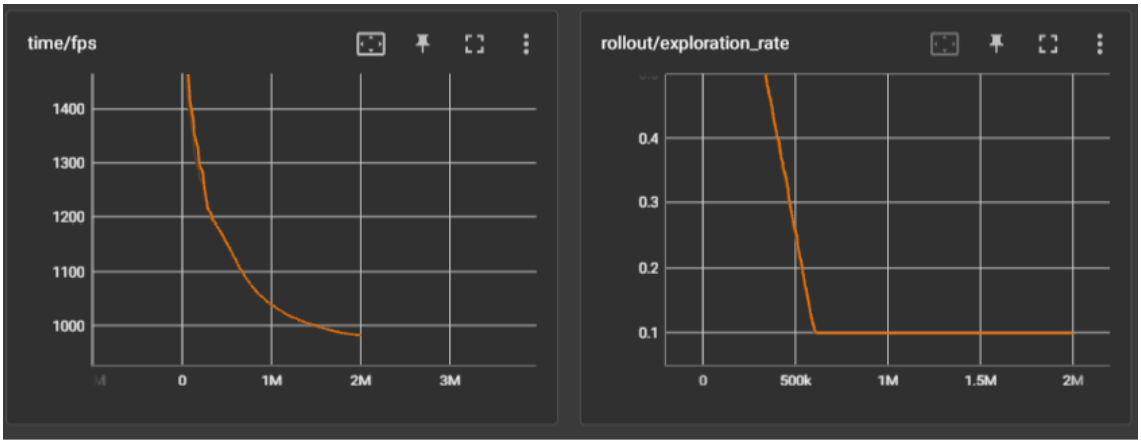
1. Taxi-v3 : Discrete observation space



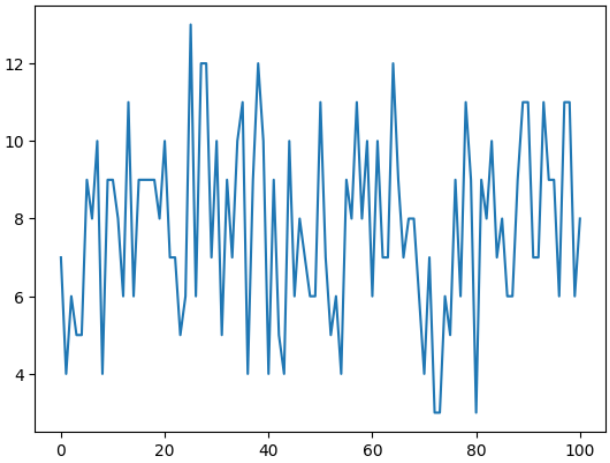






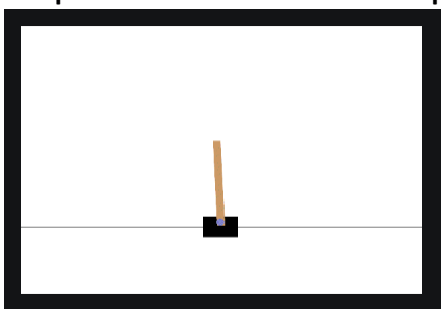


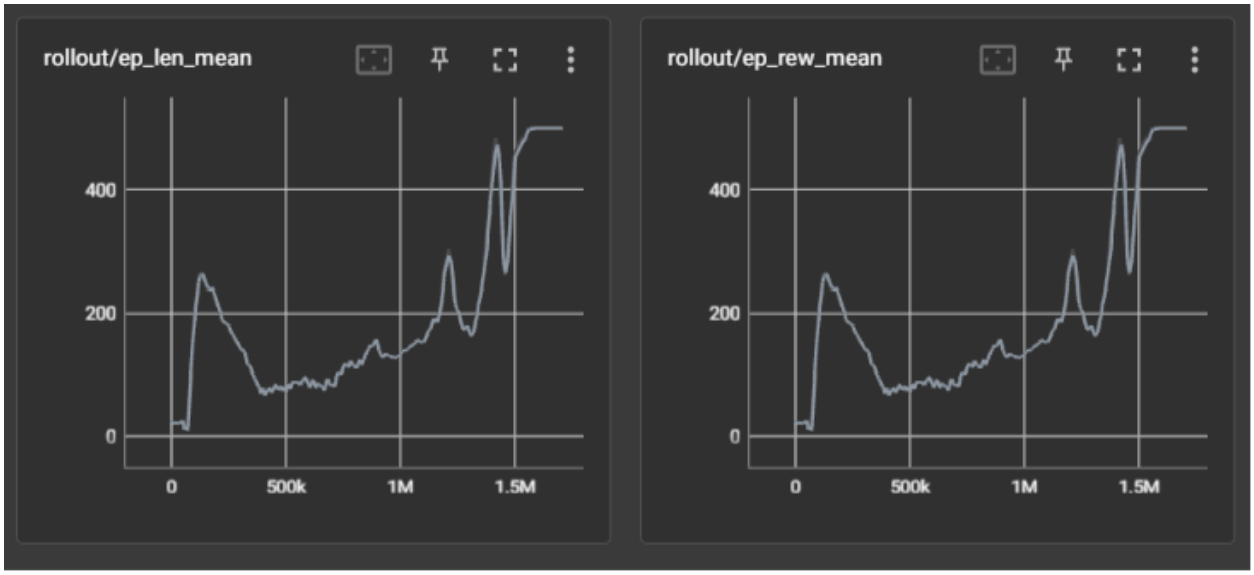
Reward per episode

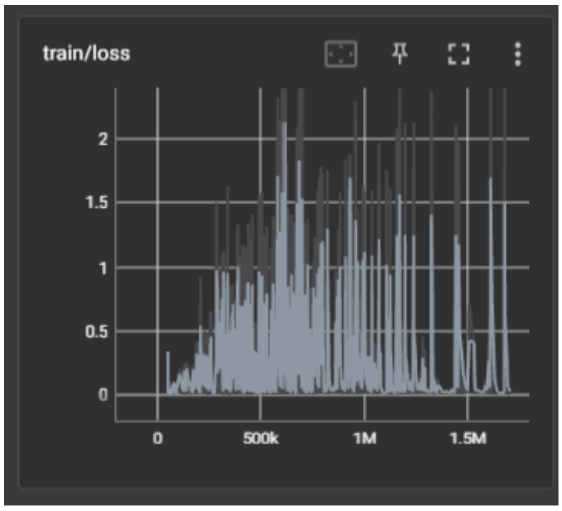


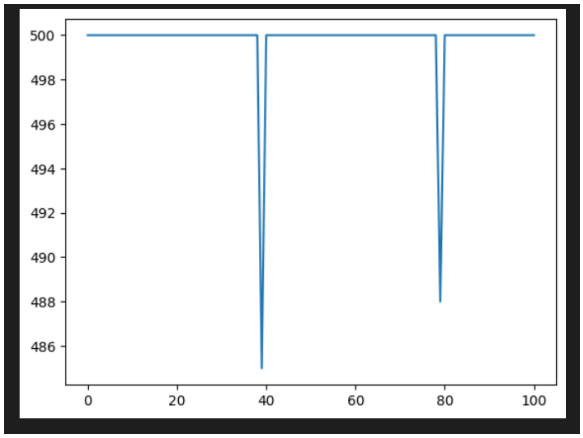
Model is trained and take least time to take passenger to destination.

2. Cartpole: Continuous observation space





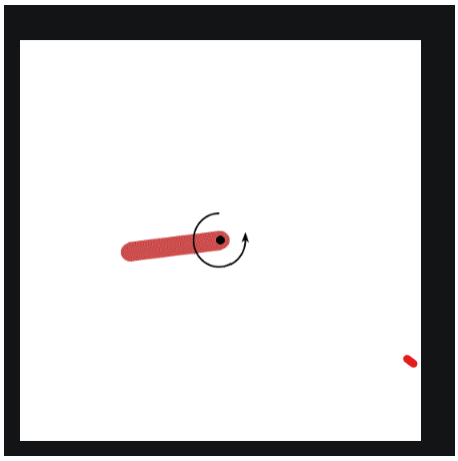




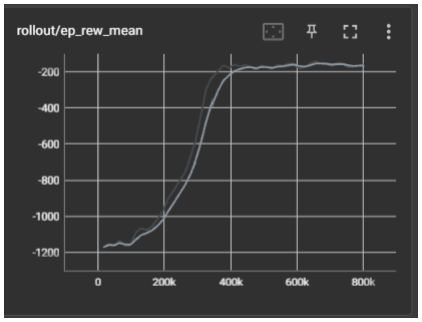
Graph on right is reward per episode. As can be seen cartpole takes 500steps mostly every time. Hence trained

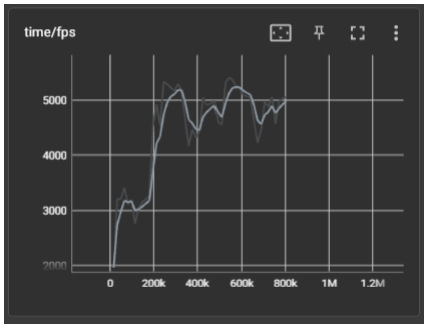
Q2. Implement PPO on gymnasium environment with continuous observation and action space. I used same configuration as mentioned in Question1 expect PPO algorithm instead of DQN.

1. Pendulum :

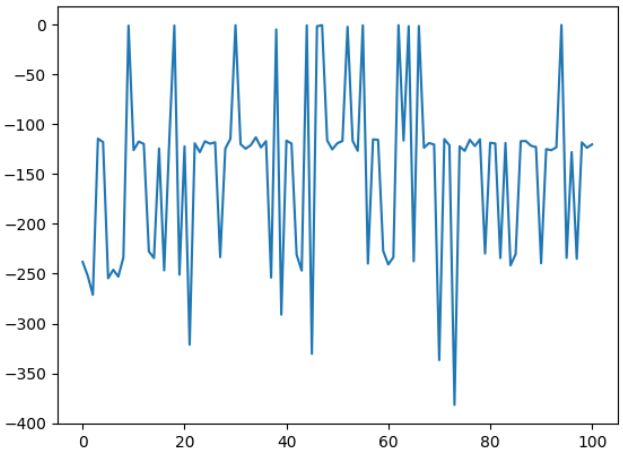


Action Space	Box(-2.0, 2.0, (1,), float32)
Observation Shape	(3,)
Observation High	[1. 1. 8.]
Observation Low	[-1. -1. -8.]
Import	`gym.make("Pendulum-v1")`



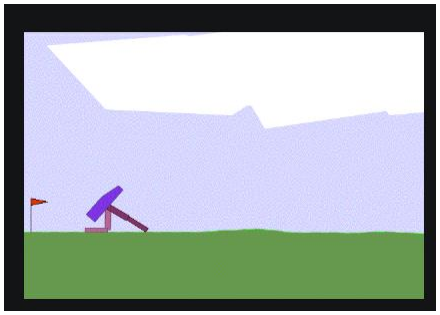


Reward per episode

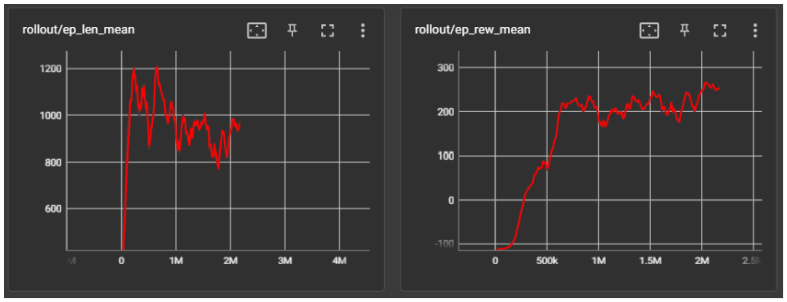


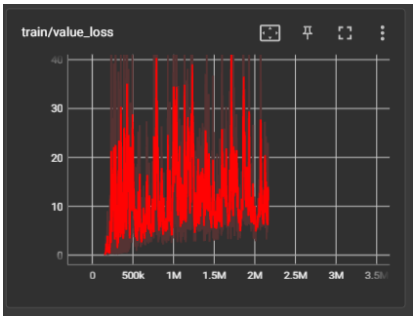
Model takes least torque to stabiles pendulumn

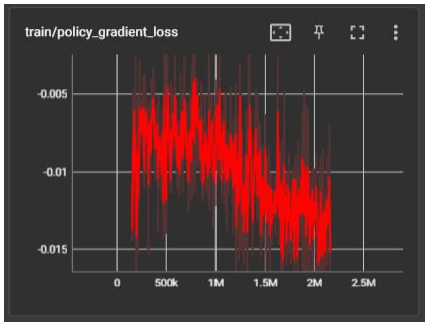
2. Bipedal Walker:

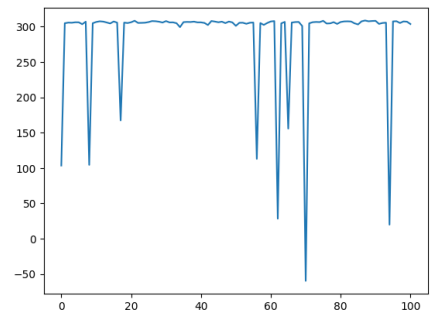


Action Space	Box(-1.0, 1.0, (4,), float32)
Observation Shape	(24,)
Observation High	[3.14 5. 5. 5. 3.14 5. 3.14 5. 5. 3.14 5. 3.14 5. 5. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Observation Low	[-3.14 -5. -5. -5. -3.14 -5. -3.14 -5. -0. -3.14 -5. -3.14 -5. -0. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Import	`gym.make("BipedalWalker-v3")`

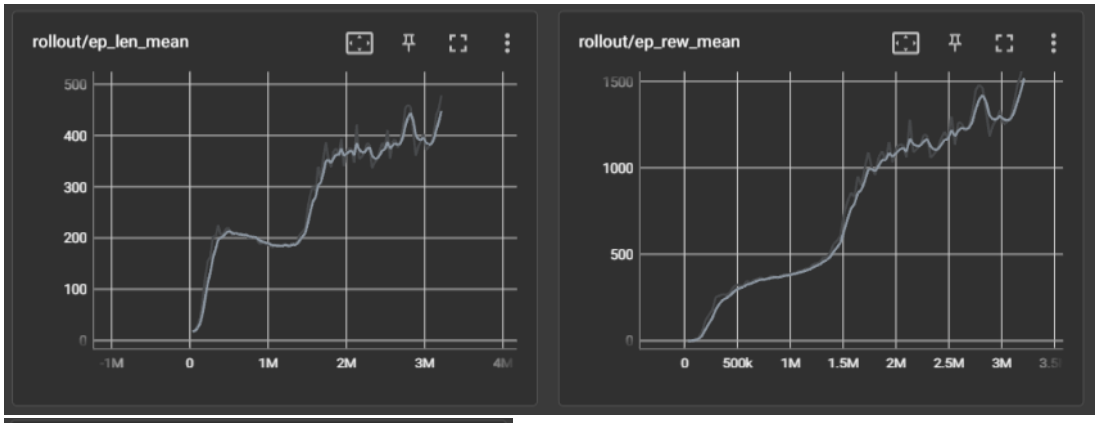


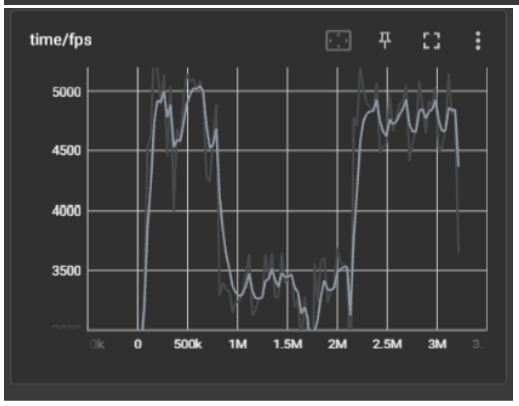


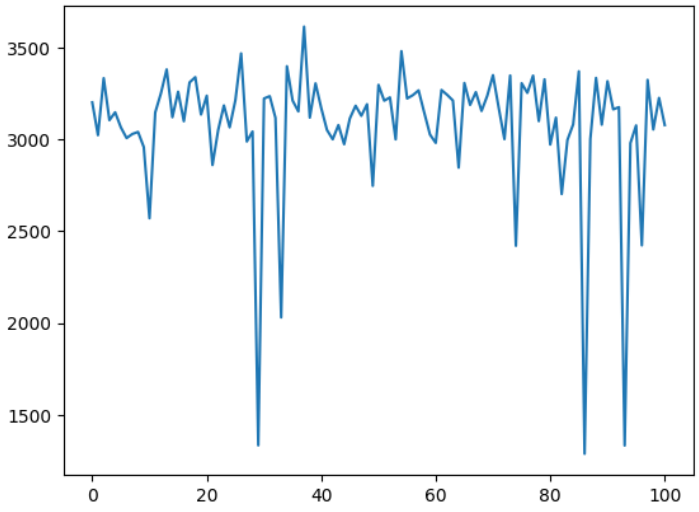




Q3: MuJoCo Walker 2D: I used PPO and able to make Walker 2D run continuously.







Above figure is reward per episode using trained model.