REPORT on Q1

1.a) Develop a small code snippet to load the corresponding Gym environment(s) and print out the respective state and action space. Develop a random agent to understand the reward function of the environment. Record your observations.

A. MOUNTAIN CAR

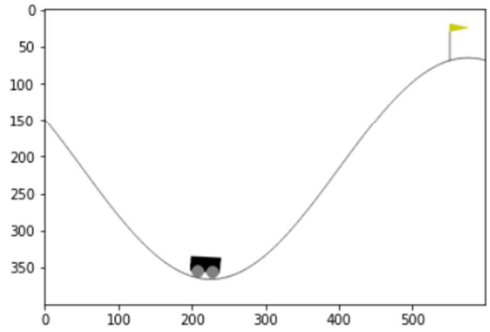
NOTEBOOK – "Q1_DDPG_Mountain_Car_OU. ipynb"

Print Gym Environment Specs

```
[ ]  env = gym.make('MountainCarContinuous-v0') #('LunarLanderContinuo
     print('Action Space-->',env.action_space)
     print('Observation Space-->',env.observation_space)
     print('reward range-->',env.reward_range)
     print('Meta data --> ',env.metadata)
     print('Specifications -->',env.spec)
     env.reset()

     prev_screen = env.render(mode='rgb_array')
     plt.imshow(prev_screen)

     for i in range(200):
         env.render(mode='rgb_array')
         action = env.action_space.sample()
         obs, reward, done, info = env.step(action)
         if done:
             env.reset()

     env.close()
```

```
Action Space--> Box(-1.0, 1.0, (1,), float32)
Observation Space--> Box([-1.2  -0.07], [0.6  0.07], (2,), float32)
reward range--> (-inf, inf)
Meta data -->  {'render_modes': ['human', 'rgb_array', 'single_rgb_array'],
'render_fps': 30}
Specifications --> EnvSpec(id='MountainCarContinuous-v0',
entry_point='gym.envs.classic_control.continuous_mountain_car:Continuous_Mo
untainCarEnv', reward_threshold=90.0, nondeterministic=False,
max_episode_steps=999, order_enforce=True, autoreset=False,
disable_env_checker=False, new_step_api=False, kwargs={}, namespace=None,
name='MountainCarContinuous', version=0)
```

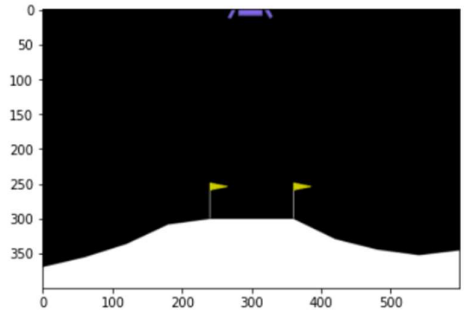
```
Action Space--> Box(-1.0, 1.0, (1,), float32)
Observation Space--> Box([-1.2  -0.07], [0.6  0.07], (2,), float32)
reward range--> (-inf, inf)
Meta data -->  {'render_modes': ['human', 'rgb_array', 'single_rgb_array'], 'render_fps': 30}
Specifications --> EnvSpec(id='MountainCarContinuous-v0', entry_point='gym.envs.classic_control.continuous_mountain_car:Cc
```



B. LUNAR LANDER

NOTEBOOK – "Q1_DDPG_LunarLander_OU. ipynb"

```
Action Space--> Box(-1.0, 1.0, (2,), float32)
Observation Space--> Box([-1.5       -1.5       -5.        -5.        -3.1415927 -5.
 -0.        -0.        ], [1.5       1.5       5.        5.        3.1415927 5.
 1.        ], (8,), float32)
reward range--> (-inf, inf)
Meta data -->  {'render_modes': ['human', 'rgb_array', 'single_rgb_array'], 'render_fps': 50}
Specifications --> EnvSpec(id='LunarLanderContinuous-v2', entry_point='gym.envs.box2d.lunar_lander:LunarLander',
```

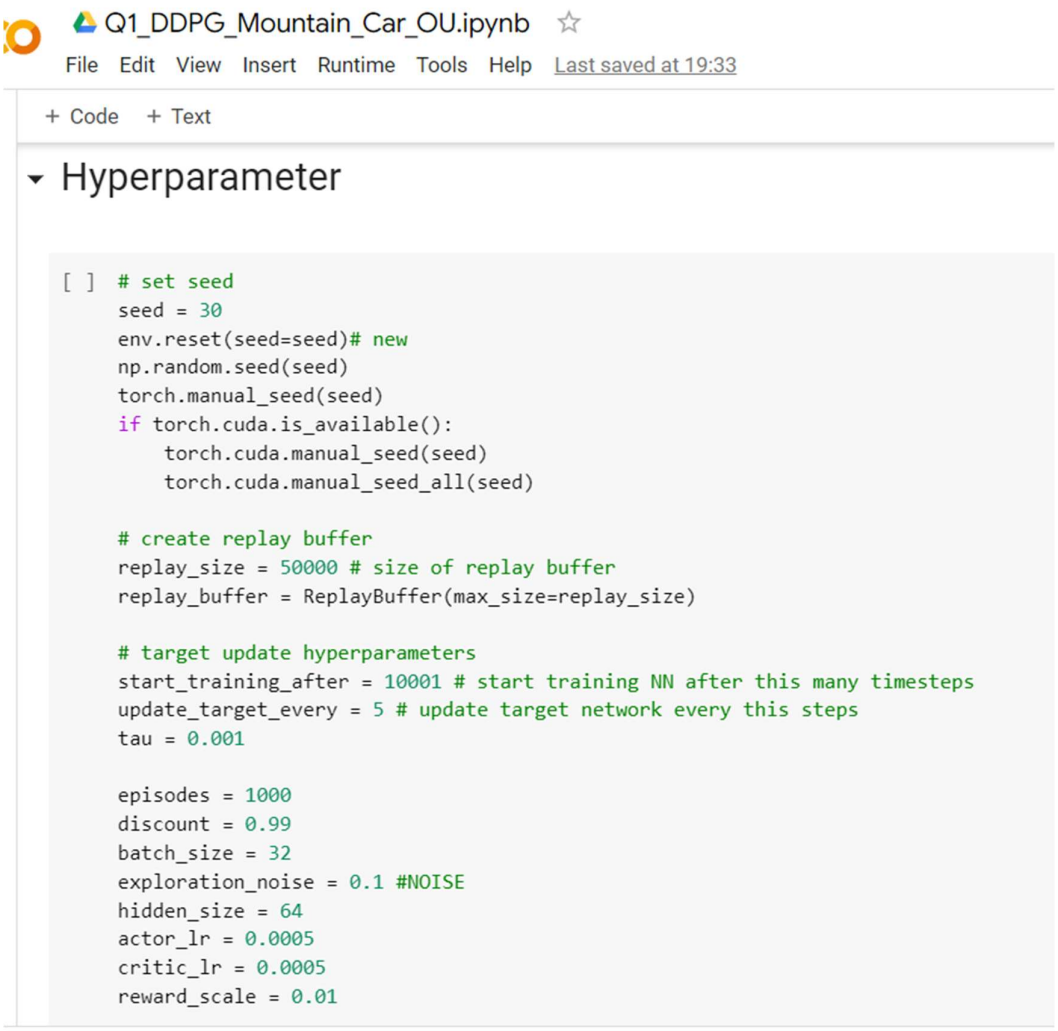


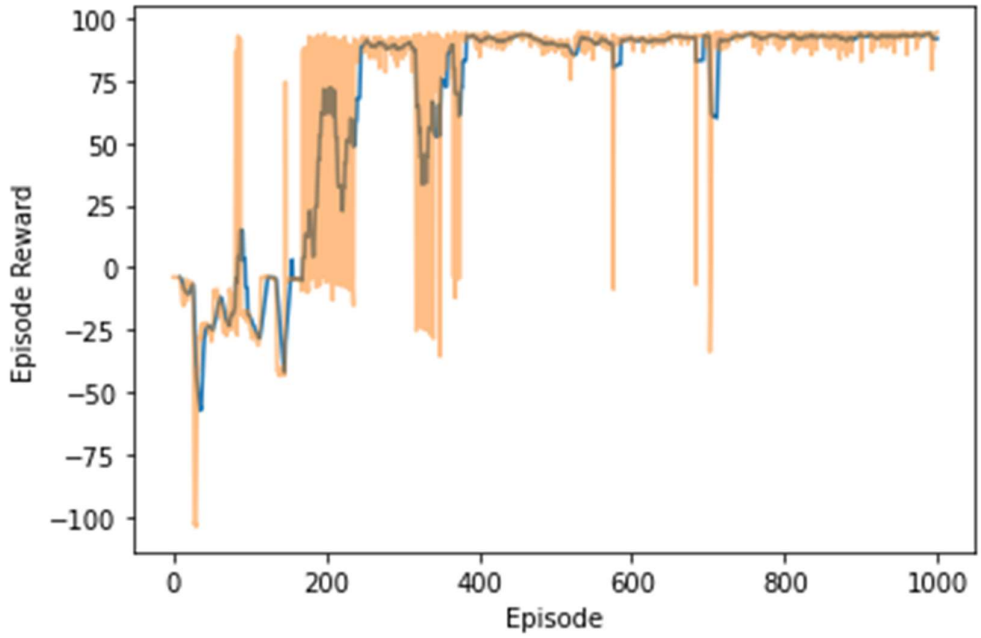

```
Specifications --> EnvSpec(id='LunarLanderContinuous-v2',
entry_point='gym.envs.box2d.lunar_lander:LunarLander',
reward_threshold=200, nondeterministic=False, max_episode_steps=1000,
order_enforce=True, autoreset=False, disable_env_checker=False,
new_step_api=False, kwargs={'continuous': True}, namespace=None,
name='LunarLanderContinuous', version=2)
```

Q1.(b) : Implement the DDPG algorithm to solve the tasks envisioned by the two environments. Provide corresponding learning graphs in your Jupyter notebooks.

A. MOUNTAIN CAR

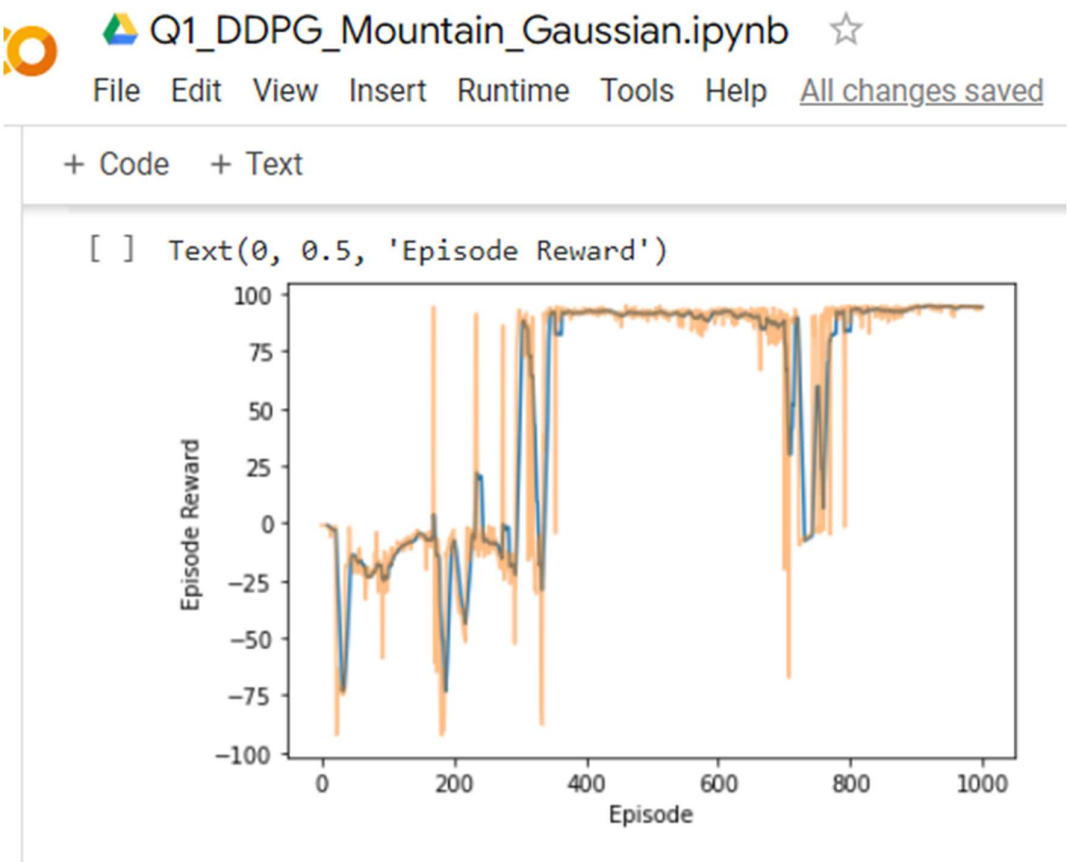
NOTEBOOK – "Q1_DDPG_Mountain_Car_OU. ipynb"





Q1.c) Check with Gaussian Noise & compare if takes similar number of episodes for Mountain Car

NOTEBOOK – "Q1_DDPG_Mountain_Gaussian. ipynb"



COMPARISON Between Gaussian & OU Noise for MOUNTAIN CAR –

1. We see that the agent converges faster for OU (at 200 + episodes) whereas it takes 300+ episodes for Gaussian noise (for the same Hyperparameter configuration)
2. For Gaussian noise, even after convergence there are sudden points with 0 scores for 10 episode mean, whereas for OU Noise it never goes below 70. Hence the variance of Mean 10 Episode reward is less for OU- ie. More stable.

Q1(b) & 1(c) for LUNAR LANDER

Q1 b) LUNAR LANDER with OU Noise

NOTEBOOK – "Q1_DDPG_LunarLander_OU. ipynb"

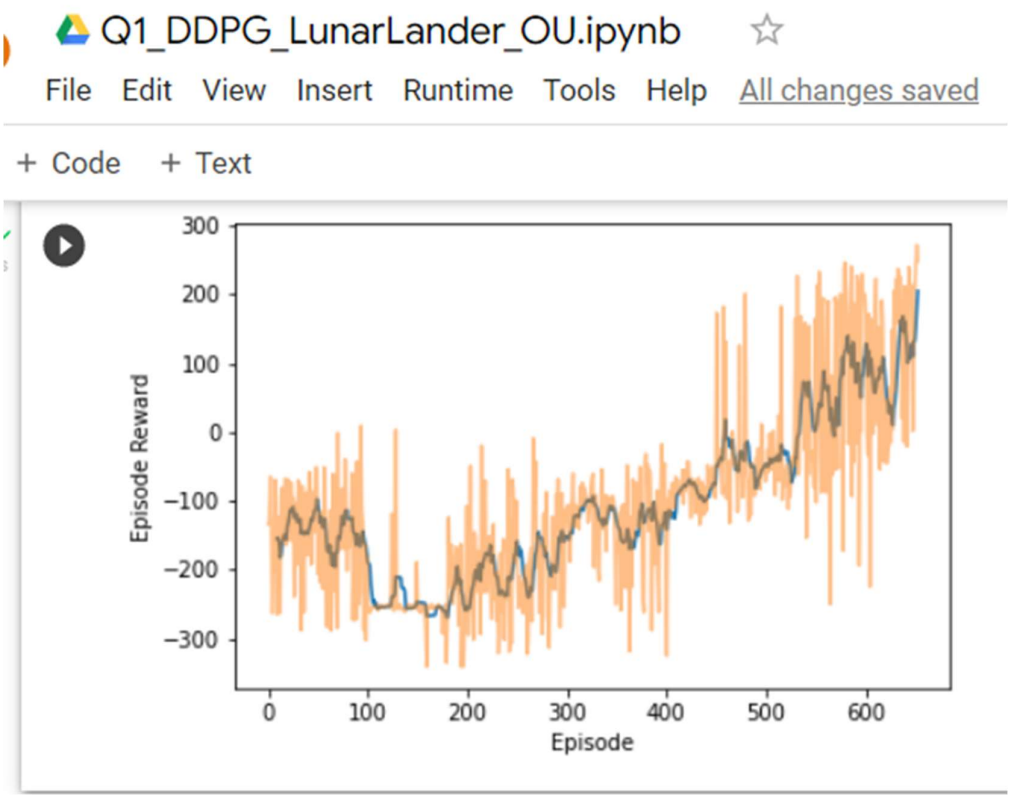
Q1_DDPG_LunarLander_OU.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Hyperparameter

```
[ ]  # set seed
     seed = 30
     env.reset(seed=seed)# new
     np.random.seed(seed)
     torch.manual_seed(seed)
     if torch.cuda.is_available():
         torch.cuda.manual_seed(seed)
         torch.cuda.manual_seed_all(seed)

     # create replay buffer
     replay_size = 50000 # size of replay buffer
     replay_buffer = ReplayBuffer(max_size=replay_size)

     # target update hyperparameters
     start_training_after = 10001 # start training NN after this many timesteps
     update_target_every = 5 # update target network every this steps
     tau = 0.001

     episodes = 1000
     discount = 0.99
     batch_size = 32
     exploration_noise = 0.1 #NOISE
     hidden_size = 64
     actor_lr = 0.0005
     critic_lr = 0.0005
     reward_scale = 0.01
```



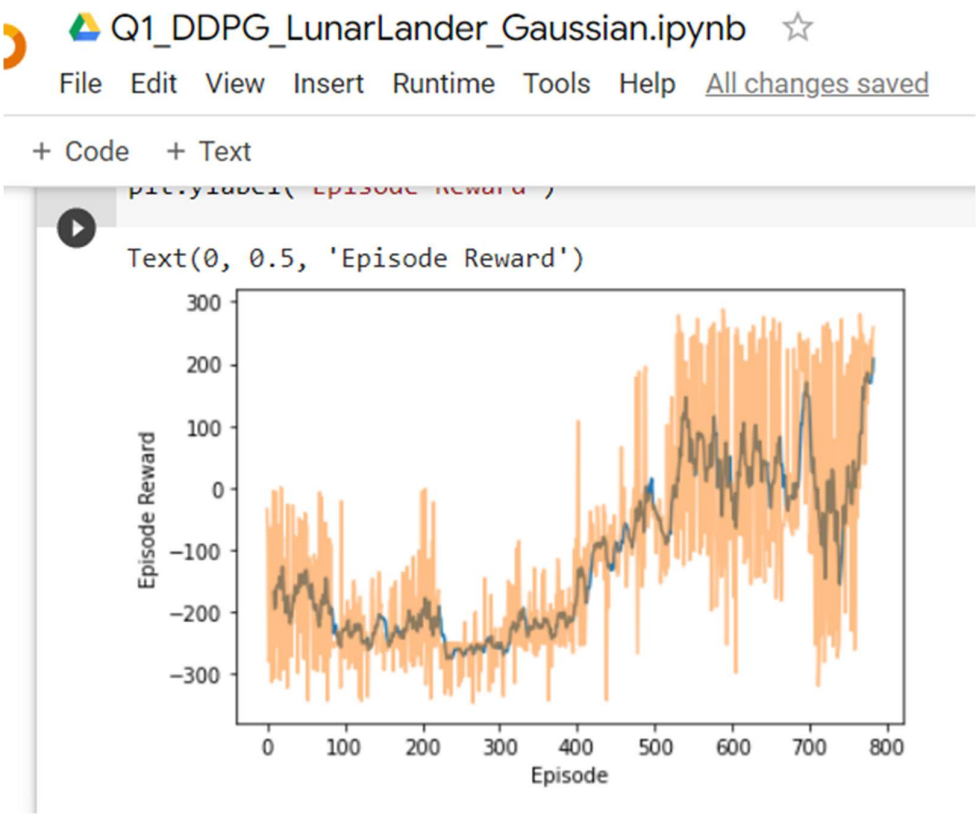
Q1.c) Check with Gaussian Noise & compare if takes similar number of episodes for Lunar Lander

NOTEBOOK – "Q1_DDPG_LunarLander_Gaussian. ipynb"

+ Code + Text

Hyperparameter

```
[ ]  # set seed
     seed = 31
     env.seed(seed)
     np.random.seed(seed)
     torch.manual_seed(seed)
     if torch.cuda.is_available():
         torch.cuda.manual_seed(seed)
         torch.cuda.manual_seed_all(seed)

     # create replay buffer
     replay_size = 50000 # size of replay buffer
     replay_buffer = ReplayBuffer(max_size=replay_size)

     # target update hyperparameters
     start_training_after = 10001 # start training NN after this many timesteps
     update_target_every = 5 # update target network every this steps
     tau = 0.001

     episodes = 5000
     discount = 0.99
     batch_size = 32
     exploration_noise = 0.1
     hidden_size = 64
     actor_lr = 0.0005
     critic_lr = 0.0005
     reward_scale = 0.01
```



COMPARISON Between Gaussian & OU Noise for LUNAR LANDER –

Note that the two codes were run on colabs of two different google accounts. So, despite setting the seeds, there would be a randomness factor involved. Keeping that in mind, what we observe is that :

Comparing the first 600 episodes, both the agents reached 100 rewards at approx. 550 episodes, but OU noise agent a lot more steady improvement, less pronounced ups and down as compared to the Gaussian noise Agent. Hence the OU noise agent is preferred.