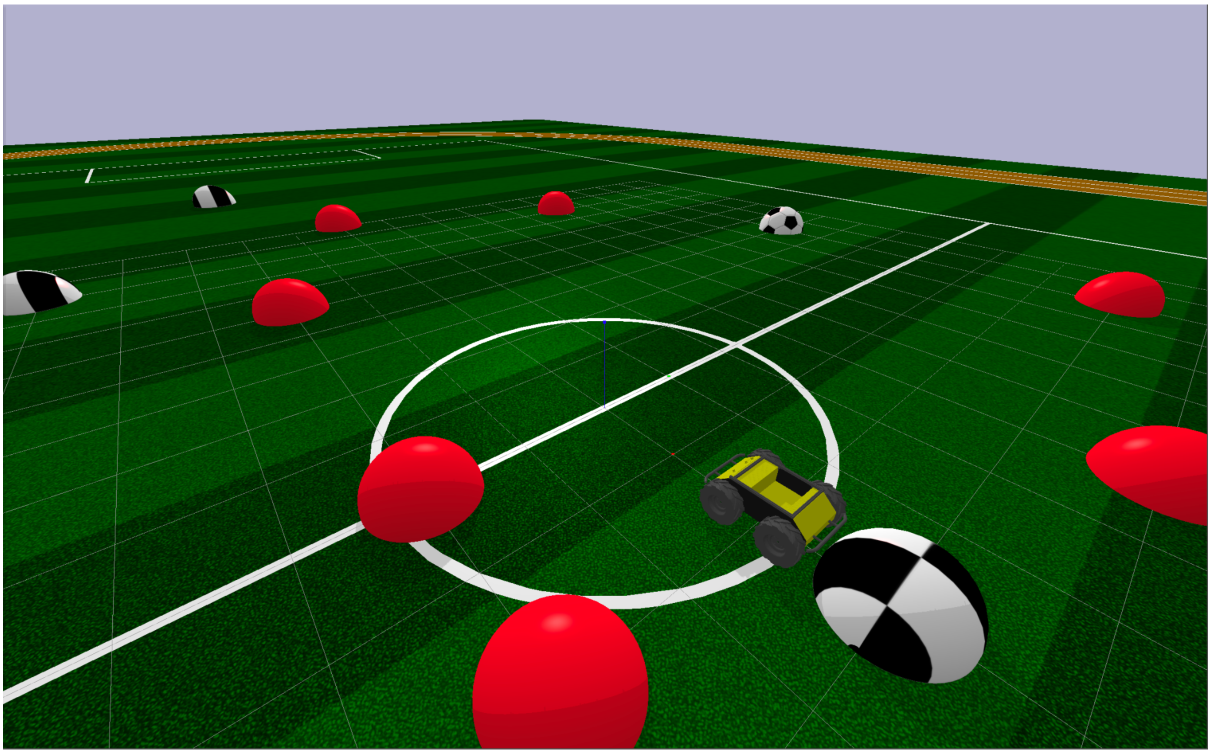
Deep Deterministic Policy Gradients in Pytorch with Simulation in PyBullet

Objective:
Use Deep Deterministic Policy Gradients to find the optimal path to the destination using an environment created using pybullet. The environment contains obstacles and a goal point(soccerball). The input states to the network are images and the total area of obstacles in the scene (found using segmentation masks).

The environment in PyBullet (Car with obstacles & Soccerball Goal point)



Theory

Why do we need DDPG?

Deep Q networks only work for low dimensional discrete action space. It relies on finding the action that maximizes the action-value function, which in the continuous-valued case requires an iterative optimization process at every step.

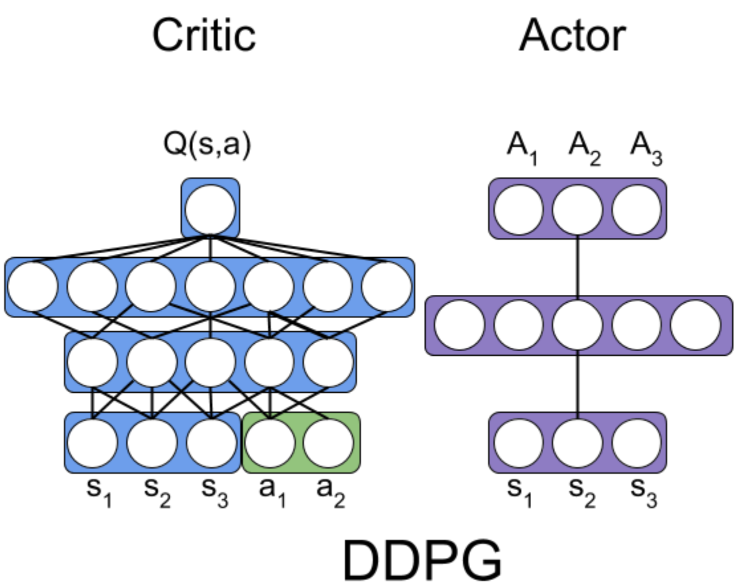
Therefore, we use Deep Deterministic Policy Gradients(DDPG) for continuous-valued high dimensional action space.

Deep Deterministic Policy Gradients(DDPG)

Deep Deterministic Policy Gradients (DDPG) algorithm uses deep networks to regress to the most optimum action in continuous high dimensional action space. DDPG combines both deep Q networks and policy gradients.
DDPG is an “**off-policy**” method. DDPG is “**deterministic**” since the actor computes the action directly instead of a probability distribution over actions.

Model

It uses an actor-critic model. The actor network is the policy network that takes states as input and outputs a continuous action value between [-1,1]. The policy is **deterministic** since it directly outputs the action. In order to promote **exploration** some Ornstein Uhlenbeck noise is added to the action determined by the policy.



The critic network is a Q-value network that takes as input both the state and the action value(from the actor) + noise, then outputs a Q-value. We try to maximize the Q-value obtained for the action decided by the actor network, and minimize the critic loss. The critic loss is computed by the TD error where target networks are used to compute Q-value for the next state.

To stabilize learning we create target networks for both critic and actor. These target networks will have soft-updates based on main networks.

Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Training Details

Hyperparameters Used:

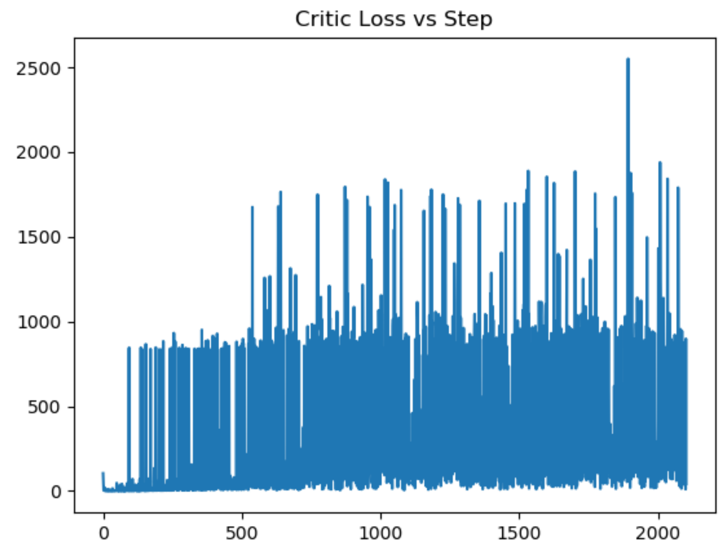
- Input State: **image captured + total area of all obstacles obtained using segmentation masks**
- Output Action - **Steering Angle [-1,1]**
- Reward
    - On reaching goal = 5000
    - On collliding with obstacles = -0.01*area of obstacles
    - Outside boundary = -100
- Threshold distance to goal = 2
- Episodes = 500
- Max_Steps = 60
- Replaymemory size = 1000
- Ornstein Uhlenbeck noise
    - Sigma = 0.2
    - Theta = 0.15
- Initialization of network weights = 3e-4
- Actor LR = 0.0001
- Critic LR = 0.001

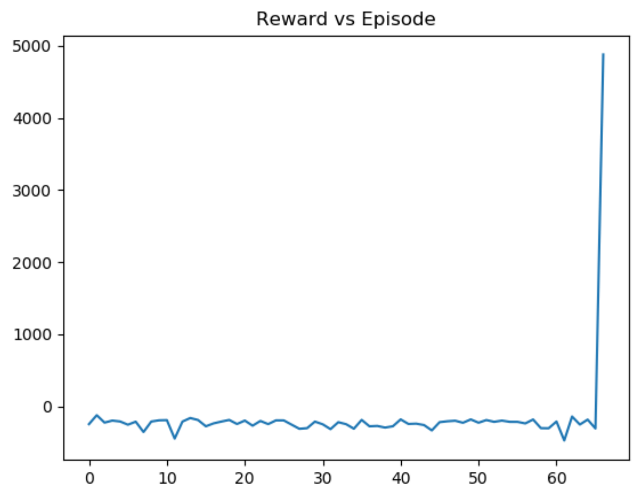
Results

Video - Car reaching the goal

https://iiitaphyd-my.sharepoint.com/:v:/g/personal/gowri_lekshmy_research_iiit_ac_in/EYyj3LAq0KpBqa7n11ePehoBTowvk_YzmY8xZSzaMvPz8Q?e=GkeHzW

Plots





Code:
https://github.com/gowrijsuria/DDPG

References

1. Lillicrap, et al. Continuous control with Deep Reinforcement Learning
2. Silver, et al. Deterministic Policy Gradients
3. https://pemami4911.github.io/blog/2016/08/21/ddpg-rl.html