Implementing Actor Critic Methods and Solving RL Environments

Indra teja Pidathala, 50478945

November 2022

1 Introduction

In our Last Assignments, we have used Value based Methods(Q-Learning, SARSA, DQN, DDQN) to solve RL Environments. In this one, we are trying to solve RL Environemnts using Actor Critic Methods. our goal is to solve certain environments(described in Section-3) from openAI-Gym using Q-Actor Critic.

2 Actor-Critic Methods

2.1 what is it about ?

Actor-Critic is a Temporal Difference(TD) version of Policy gradient[3]. It has two networks: Actor and Critic. The actor decided which action should be taken and critic inform the actor how good was the action and how it should adjust. The learning of the actor is based on policy gradient approach, it learns the optimal policy π_*.İn comparison, critic evaluates the action produced by the actor by computing the value functions $V(s)/Q(s,a)$ etc.

Flow diagram of the Algorithm is shown below in Fig-1.

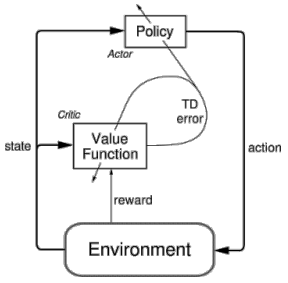

Figure 1: Actor Critic Architecture

Pseudo-code of the Q-Actor Critic Algorithm is shown below in Fig-2.

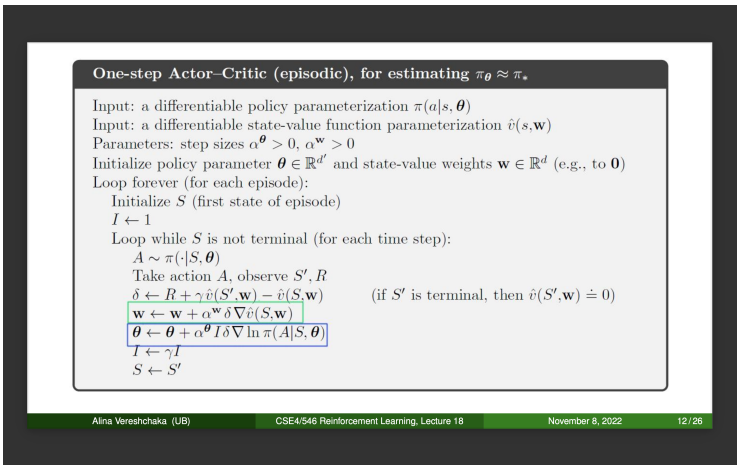

Figure 2: Q-Actor-Critic Algorithm

2.2 How does it differ from Value based methods?

The key differences between Actor Critic(AC) Methods and Value based(VB) methods are

- in AC, we estimate value function or Q-function of the current policy, use it to improve the policy whereas in VB,we estimate value function or Q-function of the current policy (no explicit policy)
- Convergence of AC methods are relatively faster than VB methods
- Method of learning in AC is Innate Exploration whereas in VB it's explicit Exploration
- Example Include for AC : Q-AC , λ-AC, Adv-AC. for VB include : Q-Learning, DQN, DDQN etc.

3 Environments

Policy for all envs are learnt through the Actor-Critic algorithm, random initially).

3.1 CartPole

The description of the CartPole-v1 as given on the OpenAI gym website is given below

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

Our Cartpole Environment is shown below.(Fig 3).

$States \rightarrow S : [(0:1),(0:1),(0:1),(0:1)]4 - continuous - values.$

$Actions \rightarrow A : \{LEFT : 0, RIGHT : 1\}.$

$Rewards \rightarrow R : \{1 : 480\}.$

$Policy \rightarrow \pi_d(s) : a.$

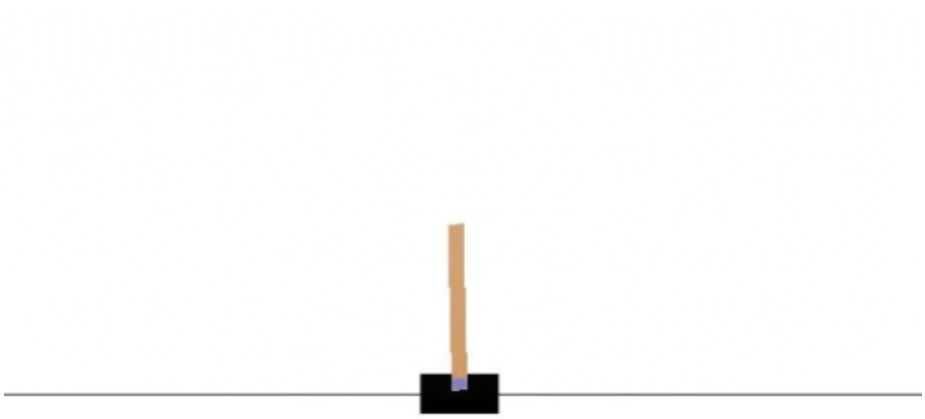

Figure 3: CartPole

3.2 Mountain Car

The description of the Mountain car as given on the OpenAI gym website is given below

The Mountain Car MDP is a deterministic MDP that consists of a car placed stochastically at the bottom of a sinusoidal valley, with the only possible actions being the accelerations that can be applied to the car in either direction. The goal of the MDP is to strategically accelerate the car to reach the goal state on top of the right hill. There are two versions of the mountain car domain in gym: one with discrete actions and one with continuous. This version is the one with discrete actions

Our MountainCar Environment is shown below.(Fig 4).

$States \rightarrow S : [(0:1),(0:1)]2 - continuous - values.$

$Actions \rightarrow A : \{Acceleate - LEFT : 0, No - Acceleration : 1, Accelerate - RIGHT : 1\}$.

$Rewards \rightarrow R : \{-inf : +inf\}$.

$Policy - Deterministic \rightarrow \pi_d(s) : a$.

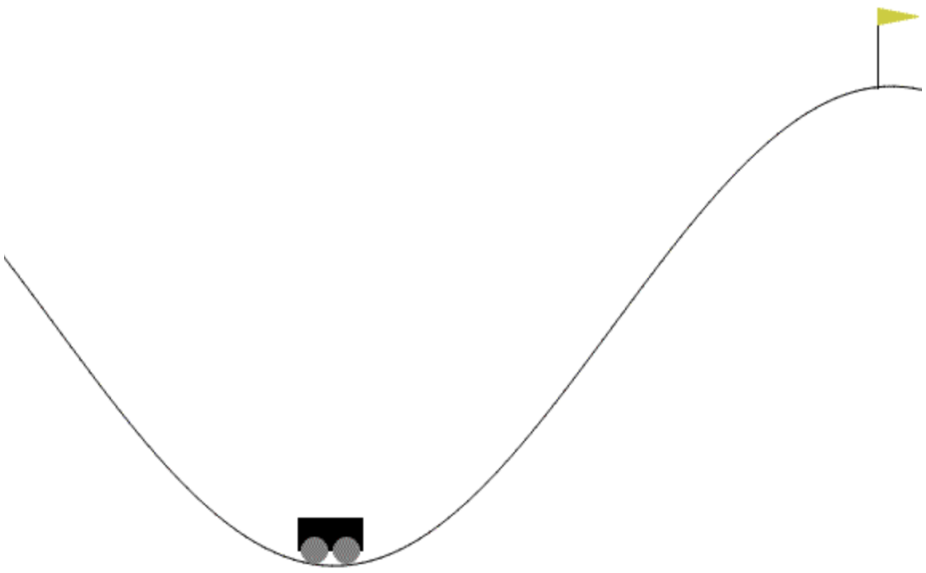

Figure 4: MountainCar

3.3 LunarLander

Our LunarLander Environment is shown below below(Fig 5).

$States \rightarrow S : [(0:1),(0:1),(0:1),(0:1),(0:1),(0:1),(0:1),(0:1)]$. 8-continuous-values

$Actions \rightarrow A : \{0, 1, 2, 3\}$. [do-nothing,fire-left-engine, fire-right-engine, fire-main-engine]

$Rewards \rightarrow R : \{-1 : 100\}$.

$Policy - Deterministic \rightarrow \pi_d(s) : a$.

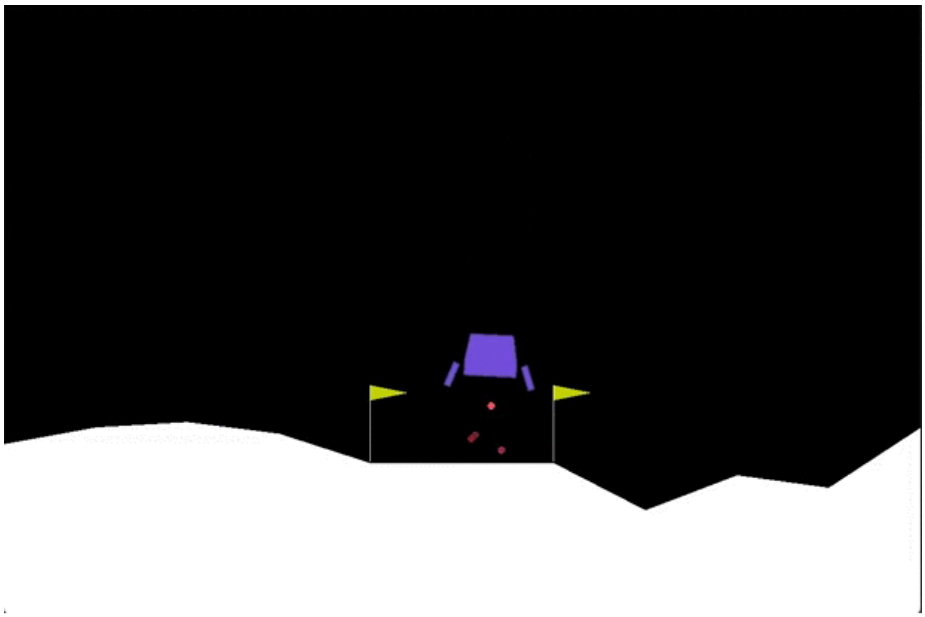

Figure 5: Lunar Lander

4 Training Results

We have used one-episode AC method for solving the envs. the architecture for Actor and Critic Networks can be found in the code.

our code for each env will be described by the file names itself. Cartpole.ipynb,MountainCar.ipynb, LunarLander.ipynb,Atari Breakout.ipynb

4.1 Cartpole

for Cartpole Env , we have ran our AC Algorithm for 1000 episodes as shown below in fig-6. but the reward never converged and achieved only a maximum if 175. this may be due to less no of episodes.

to tackle this we have tried to

4.2 MountainCar

the training for Mountaincar is very slow, so we had to change the reward function for fast training and convergence.To reach the peak from the valley, the car needs to gain mechanical energy so the optimal strategy would be one in which the car gains mechanical energy (Potential energy + Kinetic energy) at every time step. So a good reward function would be the increase in mechanical energy at every time step.

for this we have changed our reward functions to

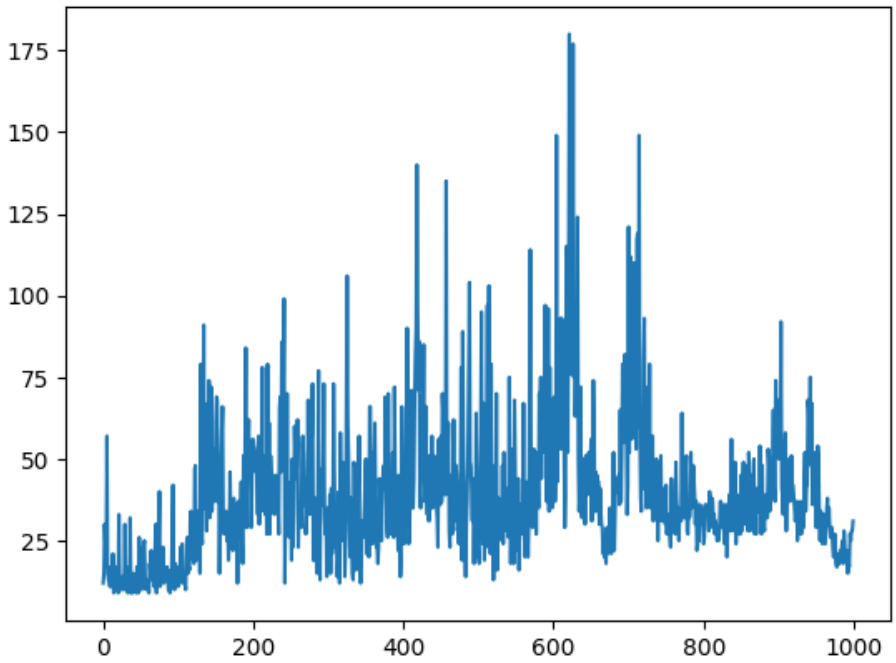

Figure 6: Cartpole Training Reward Plot

$reward = 100 * 100 * ((sin(3 * next_state[0]) * 0.0025 + 0.5 * next_state[1] * next_state[1]) - (sin(3 * state[0]) * 0.0025 + 0.5 * state[1] * state[1]))$.

this will get our rewards in range(-1,1) where 1 is goal achieved.

for MountainCar Env , we have ran our AC Algorithm for 100 episodes as shown below in fig-7. but the reward never converged and achieved only a maximum if 0.1. this may be due to less no of episodes.

4.3 LunarLander

for LunarLander Env , we have ran our AC Algorithm for 1000 episodes as shown below in fig-8. but the reward converged only around 40 episodes and achieved a maximum if 300. again, this may be due to less no of episodes.

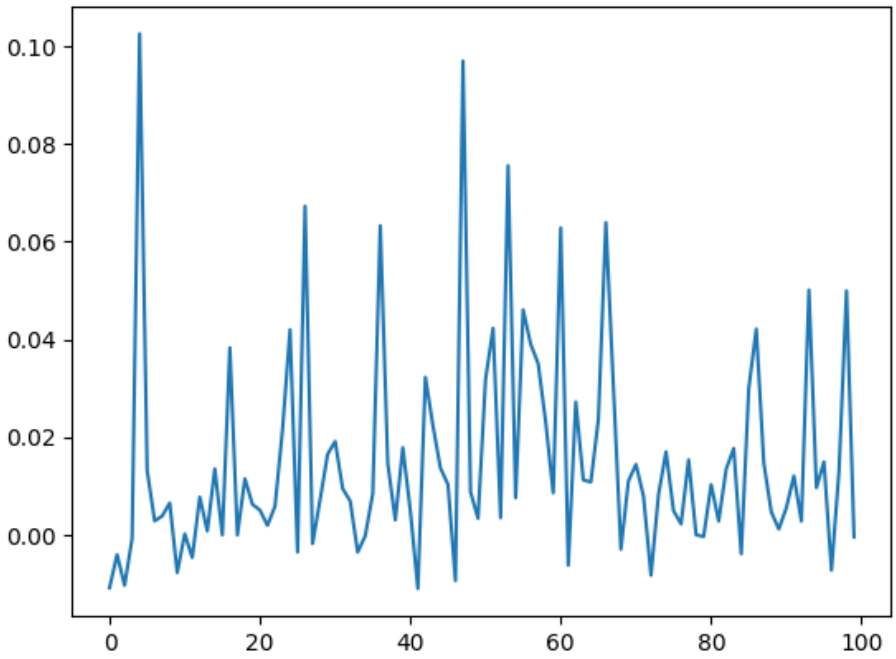

Figure 7: Mountain Car Training Reward Plot

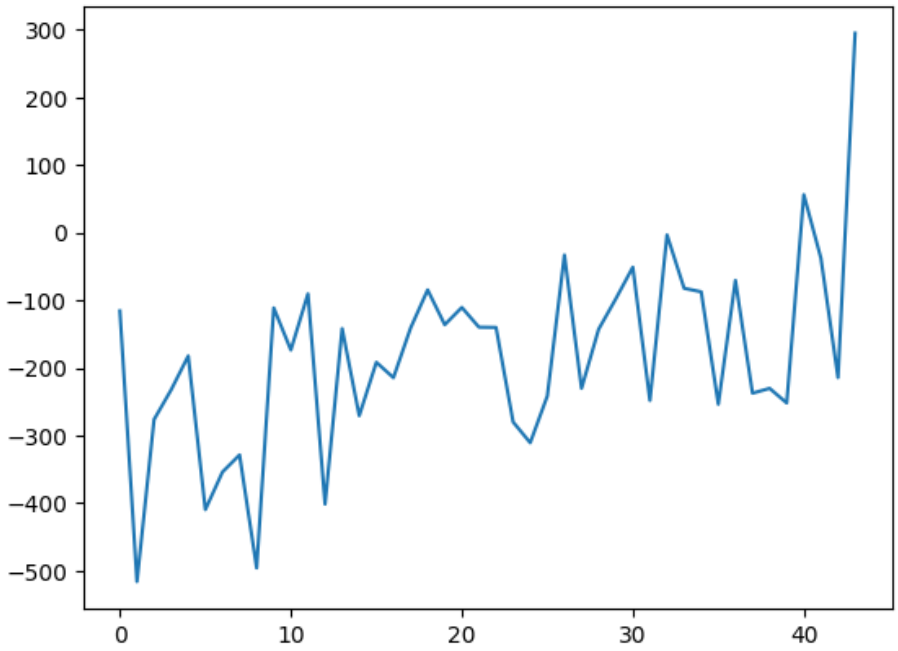

Figure 8: LunarLander Training Reward Plot

5 Evaluation Results

5.1 Cartpole

our Evaluation is not complete as the algorithm never trained fully enough for the rewards to converge.we can only achieve a maximum reward of 34 during this.

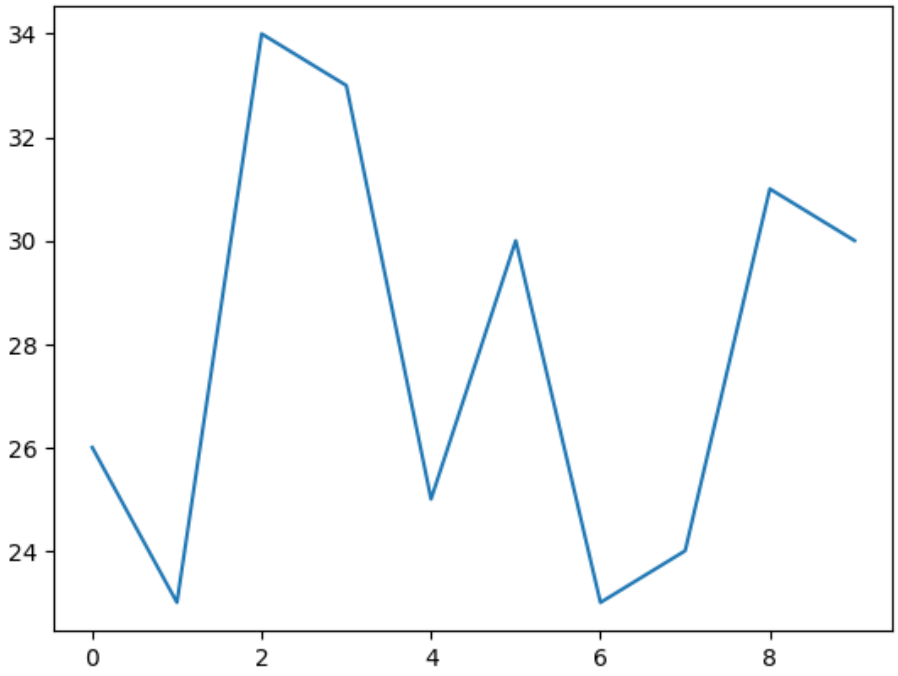

Figure 9: Cartpole Evaluation Reward Graph

5.2 MountainCar

our Evaluation is not complete as the algorithm never trained fully enough for the rewards to converge.we can only achieve a maximum reward of 0.04 during this.

5.3 LunarLander

our Evaluation is not complete as the algorithm never achieved the required avg reward across all episodes but it did achieved a reward above 200 twice and 300 once.

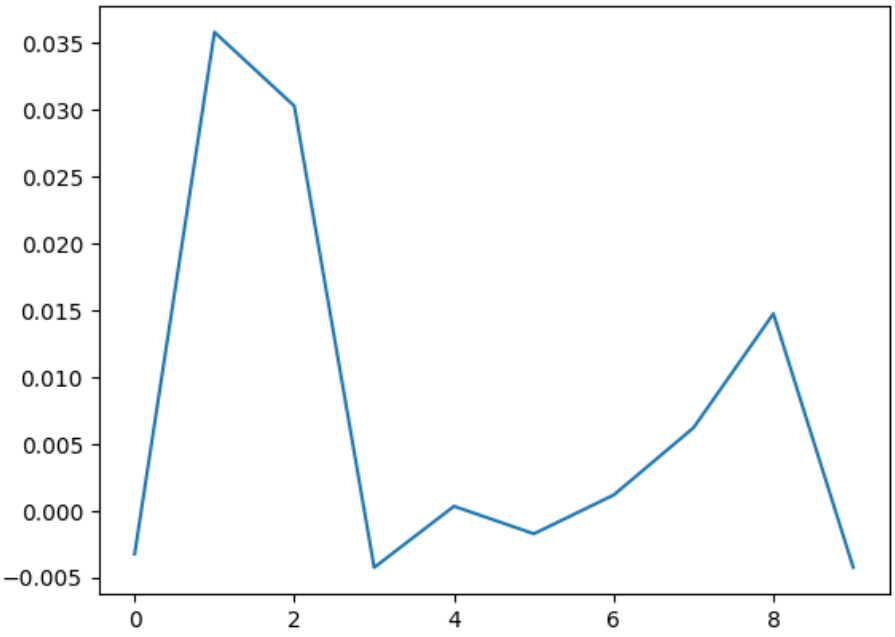

Figure 10: Mountaincar Evaluation plot

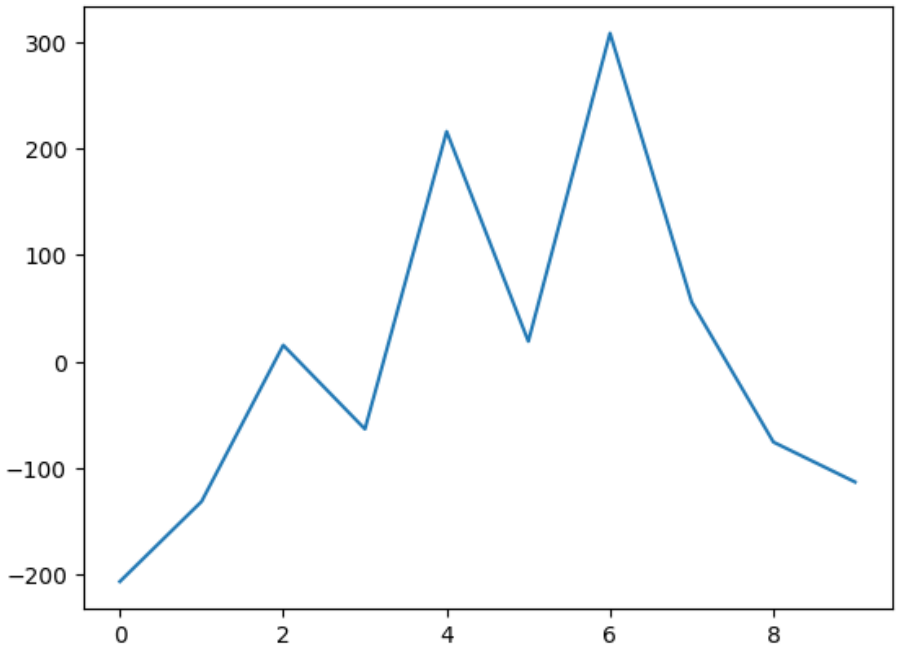

Figure 11: LunarLander Evaluation Reward Graph

6 Extra Points : Image-based

Unfortunately, for this we could only provide the trainin graph with just 10 epochs shown in fig-1, it took more than an hour to just run 10 episodes and laptop almost hung when i tried running it for 100.

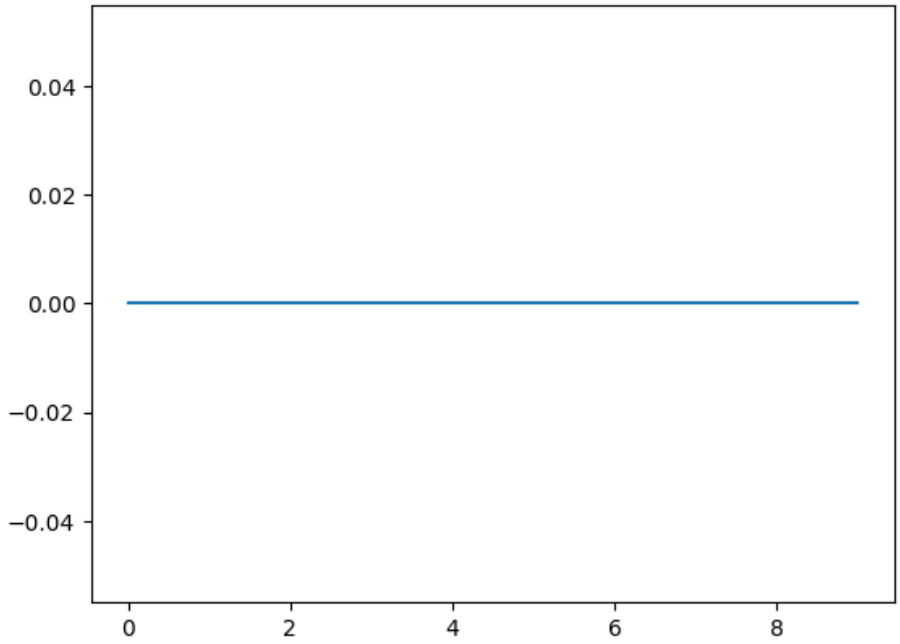

Figure 12: Atari Breakout Training Reward Graph

7 Conclusion

we have not trained our AC models for enough epochs. this is due to high training time taken for all envs(I have trained all of them on my personal laptop and it got heavy. i couldn't use CCR and colab was giving some trouble with gym versions)

we'll increase the epochs and will run the algorithm and shall use it our Final project submission.

8 References

- NIPS Styles (docx, tex)
- Overleaf (LaTex based online document generator) - a free tool for creating professional reports GYM environments

- GYM environments
- Open AI Gym (Documentation, etc)
- Lecture slides
- DQN-Paper
- Wikipedia
- Richard S. Sutton and Andrew G. Barto, "Reinforcement learning: An introduction" (pdf)
- google.com and internet in general(only for referencing not for copying)
- marvel comics