Individual Assignment – Abdulnour Dualeh

This report aims to answer questions posed in the individual assignment.

Q1)

During the training phase the Q-network has 2 simultaneous goals:

i) Explore to improve its understanding of the environment

ii) Exploit by taking a greedy action depending on the Q values

So, the epsilon greedy takes a random action with probability epsilon & takes a greedy action with respect to the Q values otherwise. This guarantees that the Q-values converge to the Q values corresponding to the optimal policy.
On the other hand, only greedy updates will converge quickly but to a suboptimal policy which is undesirable.

This is contrary to the testing phase where we just need to take an action with the maximum Q(s,a) (Action value) i.e. only be greedy & ignore exploration stage.

Q2)
the paper states that the target Q network is used to address instabilities.

The online Q-network is updated every 4 frames so the Q values fluctuate quite a bit. The target Q-network was designed to avoid these fluctuations in the Q-values.

In the paper

The idea proposed in paper was to keep updating the Q-network frequently but using a stable target Q-network for get the Q-values. This target Q-network is then synchronised with the online Q-network regularly.

In our case, the online Q-network updates every 4 frames & the target Q-network updates for every 10K frames thus, is significantly more stable.

Q3)

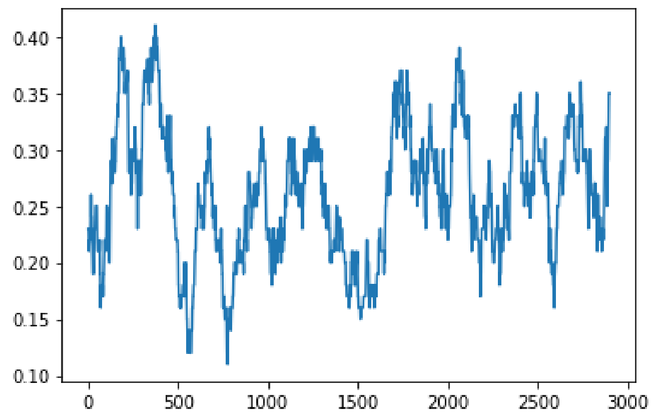
<u>Baseline implementation:</u>
It first updates the q value according to bellman's optimality equation & then considers the probability of the episode ending in the current step & deduct that payoff.
This is incorrect primarily because one step reward would be received even if the episode ends after the current action which is contrary to what the update step implements

<u>Our implementation:</u>
The episode will end with probability done_sample in which case we will get just the current reward & otherwise we will get the long term cumulative reward according to bellman's optimality equations

i.e with p = done_sample :   reward_sample
With p = 1-done_sample :   reward_sample + gamma*tf.max(future_rewards)

Which gives updated_q = reward_sample + (1-done_sample)*gamma*tf.max(future_rewards)
Exactly what our update rule does. (It is clean & makes perfect sense)

Q4)
This is my plot for moving average of the returns. The code for this can be found in the notebook and the dqn_test.py file.



Q5)

The various OpenAI Gym wrappers differ primarily in how an episode is generated & how the reward function is defined in the environment. This is particularly important because different rewards can reinforce different behaviour of the agent.

Role of a few OpenAI Gym wrappers for Atari :

i) <u>MaxAndSkipEnv</u>
It only returns a maxed state over every k observations & the remaining are skipped to avoid redundancy & quick training, testing even if the forward pass latency is high.

ii) <u>EpisodicLifeEnv</u>
The episode ends when life ends but the environment resets only after all lives are exhausted. There is a reward for breaking the pieces & a negative reward for losing life.

iii) <u>WarpFrame</u>
Converts the frame to grayscale & warp it into a 84*84 grid & that is the state.
Rewards are similar to episodic life env with an additional reward for making the ball fall on the deflector.

iv) <u>ScaledFloatFrame</u>
The ScaledFloatFrame normalizes the observations to 0~1

v) <u>ClipRewardEnv</u>

Very similar to episodic env but all rewards are in {-1,0,+1} depending on the sign of the reward

vi) FrameStack
The state is defined as the previous k frame stacked together.
This is very memory efficient as it takes cumulative decisions & the forward is speed up due to matrix operations

References

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540):529.