# Individual Assignment
# AI in Games

Amir Sepehr Aminian
210453289

December 17, 2021

**Abstract**

In this assignment, we tried establishing a DQN to play Atari Breakout. We used openAI baseline gym wrappers and to play the game we used ROMs available online.

## Question1

It is necessary to act according to $\epsilon$-greedy policy because, by selecting a greedy policy, the agent only selects the Q value that yields the highest expected reward. In other words, it exploits the Q value it already knows. Such a policy will not allow the agent to explore other options. So to make the agent explore other solutions, an $\epsilon$-greedy policy is often used instead. Before an action is chosen, a random number is generated. If that number exceeds the $\epsilon$ the greedy action is being selected, otherwise, a random action will be selected.

## Question2

The authors of [MKS+15] argue that the use of target Q-network in addition to an online Q-network is due to the improvements it causes in stability of the network. This is achieved by cloning the network every C updates to obtain a target network $Q_C$. Then use $Q_C$ to generate the Q-learning targets for the next C updates. A delay is added between the time an update to Q is made and the time the update affects the targets. This will result in lower likelihood of divergence or oscillations.

## Question3

The former implementation computes the Q value by first adding the reward to the expected reward multiplied by the discount factor and then checks if the game is in its final frame(This is achieved via the done variable supplied by OpenAI gym 'step' function). If the game is not in the terminal frame, nothing happens during this check. But, if it is in the final frame, the updated Q value is set to -1. This implies that in the final frame, even if there is some reward gained, it is not taken into account in the Q value. However, this is not a problem in Atari Breakout as the agent cannot gain a reward and die in the same frame due to the nature of the game. Though in different games, where getting reward and dying is possible simultaneously, this may prove a problem.
In the correct implementation, the reward is always added when the Q value is being updated and whether the game is in the final frame or not affects only the future reward.

# Question4

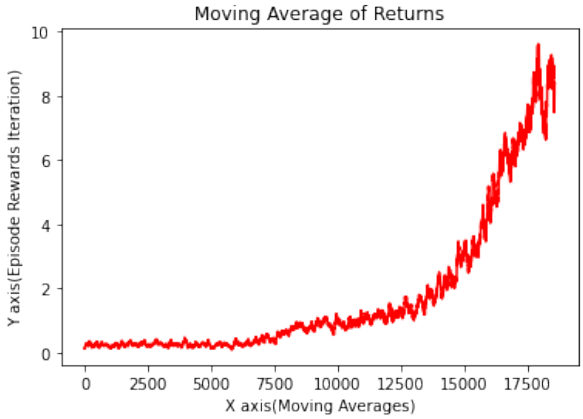


Figure 1: Moving average of the returns from episode_reward_history with 100 as window length

Due to the limitations of Google Colab, I was able to run my model 1540000 times instead of 2 million. As we can see in figure 1, the reward gradually increases. The final values should converge to 10 if we could run the model for 2 million times.

# Question5

Wrappers will allow us to add functionality to environments, such as modifying observations and rewards to be fed to our agent. It is common in reinforcement learning to preprocess observations in order to make them more easy to learn from Here are some wrappers that were deployed for Atari breakout:

- MaxAndSkipEnv: This wrapper activates every "skip'th" frame. returns the sum of rewards and max_frame over the 'skip' observations(default is 4).
- EpisodicLifeEnv : This Wrapper makes end-of-life == end-of-episode, but only reset on true game over. It makes loss of life terminal or when all of lives are exhausted but sometimes stays for a few frames after the death(eg. Qbert).
- WrapFrame : This wrapper modifies the observations returned by the environment.What it does is basically re-frames what it gets to an 84 by 84 pixels image.
- ScaledFloatFrame: This wrapper is an observation type. the ScaledFloatFrame only converts the input information from the 0-255 range 8-bit integers to a 0-1 32-bit float.
- ClipRewardEnv: This wrapper modifies the rewards by returning the -1,0,1 based on the sign of the values of the reward.
- FrameStack: This wrapper only stacks the k last frames meaning that only events in the last k frames are visible in the video.

# References

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.