COMS4047A - Reinforcement Learning
Lab 3 - Deep Q-Network

Shahil Mawjee

Overview

The lab will focus on the implementation of the Deep Q-Learning (DQN) agent to play Atari games. For this lab we will use the Pong (link) environment which are contained within the OpenAI Gym **Atari** package.

Goals:

- Implement model from research paper.
- Understand hyperparameter optimisation.

OpenAI Gym Wrapper

Wrappers are used to transform an environment in a modular way. Link to gym wrappers.

```
env = gym.make('PongNoFrameskip-v4')
env = MyWrapper(env)
```

WarpFrame Wrapper

Warp frames to 84x84 as done in the Nature paper and later work. Expects inputs to be of shape height x width x number_of_channels

PyTorchFrame Wrapper

Pytorch expects images in the from [number_of_channels, height, width] hence this wrapper transforms image from [height, width, number_of_channels] to [number_of_channels, height, width]

1 Deep Q-Learning (DQN)

Refer to the following papers in order to implement the DQN:

- Human-level control through deep reinforcement learning (link)
- Playing Atari with Deep Reinforcement Learning (link)

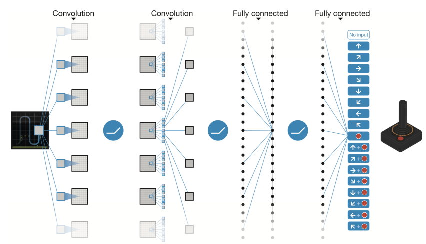

Figure 1: Deep Q Network (Source: Nature paper)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
    end for
end for

1.1 Environment: Atari Pong

For this lab we will use Atari's Pong game (link). In this environment, the observation is an RGB image of the screen, which is an array of shape (210, 160, 3)

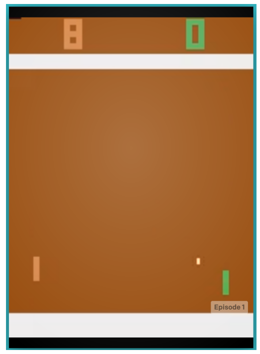

Figure 2: Pong Game (Source - OpenAI Gym website)

1.2 Training Loop

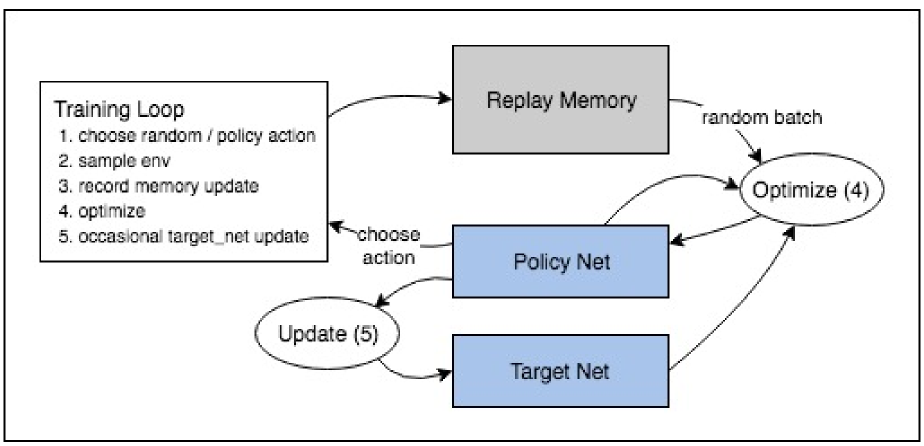

Figure 3: Training loop overview

Submission

Include the following in your zip file:

- Summary of process write up (half a page)
- Your code
- Plot of reward per episode (as done in the paper)
- GIF of your trained agent playing Pong

Submit your zip file to Moodle.