# Reinforcement Learning

COMS4047A/COMS7053A

Lab 3 - Deep Q-Network

Tamlin Love (tamlin.love1@students.wits.ac.za)
Shahil Mawjee (shahil.mawjee@students.wits.ac.za)

## 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 algorithm presented in research paper.
- Understand hyper-parameter optimisation for Deep RL.

## Compute

Training DQN agents can be computationally intensive, so we suggest using Google Colab if you do not have access to an Nvidia GPU. Ensure you attach the notebook to a kernel with GPU access, this can by the following:

- Select *Runtime* in toolbar
- then select *Change runtime type*
- and then select **GPU** as *Hardware accelerator*

Refer to https://colab.research.google.com/notebooks/intro.ipynb for more information about Google Colab.

## Submission

Work in groups of 3 to 4 people. List group members in the *group_members.txt* file.

For this lab, we have provided starter-code which makes use of Pytorch, so you will need to edit it if you plan to use TensorFlow.

**You do not have to use the starter code** and can implement your DQN from scratch. If you are using Google Colab, save your Colab notebook as a notebook (.ipynb) file and include in your submission.

Include the following in your submission (as zip file):

1. A summary of the DQN algorithm. Make sure to include the following points:
    - Explain the training process outlined in figure 3.
    - Q-Network architecture.
    - List of hyper-parameters, provide the following for each parameter:
        - Name of parameter
        - Value used
        - Brief description of the role of the parameter
2. Your code files and saved model.
3. A plot of training reward curves, as shown in the paper.
4. A plot the loss curve during training.
5. A GIF of your trained DQN agent playing Pong.

Submit your zip file to Moodle.

# 1 Deep Q-Learning (DQN)

Read to the following papers in order to understand and implement the DQN:

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

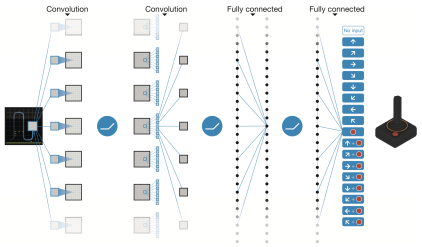


Figure 1: Deep Q Network (Source: Nature paper)

**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 $\epsilon$ 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).

```
pip install gym[atari]
```

In this environment, the observation is an RGB image of the screen, which is an array of shape (210, 160, 3)

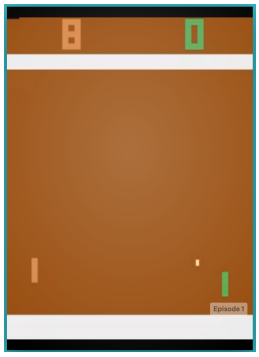

Figure 2: Pong Game (Source - OpenAI Gym website)

## 1.2 Training Loop

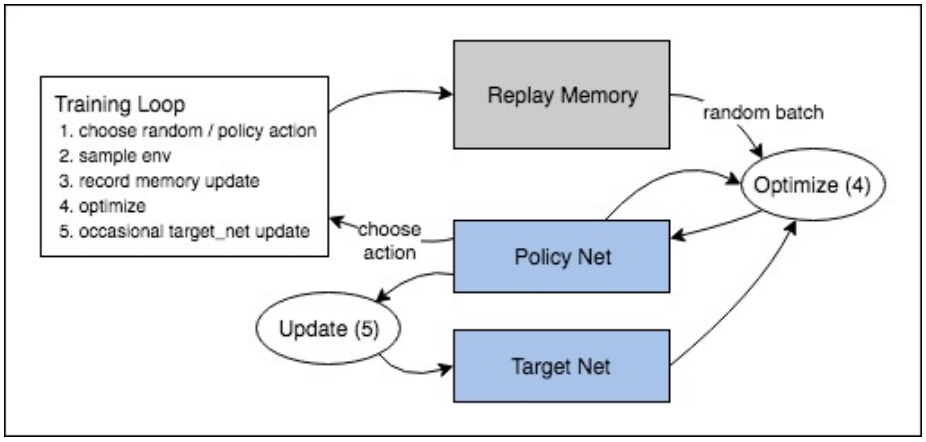


Figure 3: Training loop overview

# 2 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]

### Gym Monitor

The gym monitor wrapper will assist you in capture a recording of your agent playing the game.

```
env = gym.wrappers.Monitor(env, "path/to/store/recordings")
```