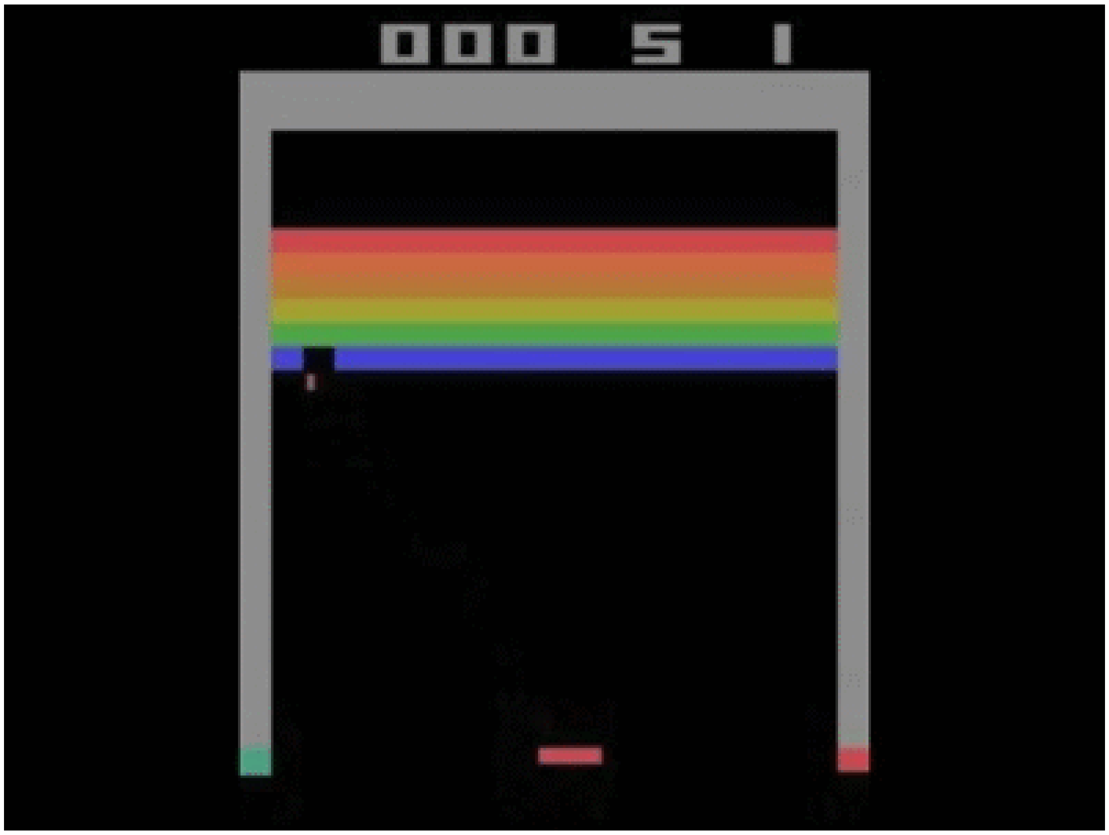
# Spring 2024 CS 444

## Assignment 5: Deep Reinforcement Learning

### Due date: Wednesday, May 1, 11:59:59PM



In this assignment, you will implement the famous Deep Q-Network (DQN) and its successor Double DQN on the game of Atari Breakout using the OpenAI Gym. The goals of this assignment are to (1) understand how deep reinforcement learning works when interacting with the pixel-level information of an environment and (2) implement a recurrent state to encode and maintain history.

### Main Assignment: DQN and Double DQN

**Download the starting code here.**

The top-level notebook (`MP5.ipynb`) will guide you through all the steps of the DQN. You will mainly implement the training of the Agent in the `agent.py` file for DQN, and `agent_double.py` for double DQN. We provide you with the neural network. For consistency of grading, **DO NOT** change the architecture of the neural network. Due to computational constraints, we only expect you to reach a mean score of 8 for the regular DQN and 10 for the Double DQN agent. Most of the hyperparameters are contained in `config.py`. You may play around with them if you want, but the provided values will be enough to reach the desired score.

**Note**: The starter code also includes `MP5_DQNLSTM_EC.ipynb` and `agent_lstm.py`. This is part of extra credit and not the main assignment. See Extra Credit section below for more details.

As you look in the iPython notebook, in our terminology, a single episode is a game played by the agent till it loses all its lives (in this case, your agent has 5 lives). In the paper, however, an episode refers to almost 30 minutes of training on the GPU and such training is not feasible for us.

As stated above, your goal is to have `agent.py` **reach a mean score of 8 and** `agent_double.py` **reach a mean score of 10**. This should take between 2000 and 2500 episodes.

Below is an example of expected rewards vs. number of episodes for the regular DQN agent to help with your debugging:

- 200 episodes: 1.5
- 400 episodes: 1.5
- 600 episodes: 1.5
- 1000 episodes: 1.75
- 1200 episodes: 2.6
- 1400 episodes: 3.8
- 1600 episodes: 4.7
- 1800 episodes: 5.95
- 2000 episodes: 8.6

And the following is an example of expected rewards vs. number of episodes for the Double DQN agent:

- 200 episodes: 1.5
- 400 episodes: 1.5
- 600 episodes: 1.5
- 1000 episodes: 1.8
- 1200 episodes: 2.9
- 1400 episodes: 4.2
- 1600 episodes: 5.0
- 1800 episodes: 6.6
- 2000 episodes: 9.1
- 2200 episodes: 10.1

**This is a computationally expensive assignment.** It is expected that your code should run for at least 4 hours to complete 2000 episodes. You can stop training each model as soon as you reach the respective target mean score.

We recommend that you look at the following links:

- Official DQN PyTorch Tutorial
- Official DQN paper
- Official Double DQN paper
- DQN Tutorial on Medium (Double DQN is the target DQN variant)

We also **highly recommend** that you understand the Official DQN PyTorch tutorial before starting this assignment. This will give you a great starting point to implement DQN and Double DQN as the tutorial implements a version of double DQN for cartpole! However, we expect you to follow our code instructions and implement code in our format. **Uploading code that does not follow our format will receive a zero.**

This assignment requires a GPU, so use your Google Cloud credits (colab could work for this assignment as well).

## Extra Credit

- Implement a DQN agent that uses an LSTM to keep track of history. This requires minimal architectural changes -- refer to `MP5_DQNLSTM_EC.ipynb` and `agent_lstm.py` for details. Unlike the agent in the main part of the assignment, this agent only sees current frames as observations and there is no explicit concatenation of successive frames. Training code should be very similar to that of DQN, but with small modifications, as mentioned throughout the code. For full credit, the LSTM agent is expected to reach a mean score of 8.

- Train a DQN agent for one or more additional Atari games from OpenAI gym and report on any implementation/hyperparameter changes you had to make, and your agent's performance.

- Implement policy gradient training or another advanced RL method for Breakout or another Atari game and compare performance (including convergence speed) to your DQN method. *You need to write your own code from scratch, not train an off-the-shelf method.*

## Environment Setup

The assignment is given to you in the `MP5.ipynb file.` If you are using a local machine, ensure that iPython is installed (https://ipython.org/install.html). You may then navigate the assignment directory in terminal and start a local iPython server using the `jupyter notebook` command. **Instructions to install dependencies are provided at the top of the notebook.** Please use environment with Python 3.7. These instructions should work for local machines, Google Cloud, and Google Colab. We have tested this assignment on PyTorch version 1.10.2, so please install this version if there are other dependency issues.

There is a separate `MP5_DQNLSTM_EC.ipynb file` for extra credit part of the assignment.

If you will be working on the assignment on a local machine then you will need a Python environment set up with the appropriate packages. We suggest that you use Conda to manage Python package dependencies (https://conda.io/docs/user-guide/getting-started.html).

Unless you have a machine with a GPU, running this assignment on your local machine will be very slow and is not recommended.

## Submission Instructions

**This is your last assignment, so feel free to use up your remaining late days if you so choose!**

1. All of your code (Python files and ipynb file) **in a single ZIP file**. The filename should be **netid1_netid2_mp5_code.zip**.
2. Upload your policy net models (in .pth format) as a separate file.
3. Your iPython notebook (`MP5.ipynb`) with output cells converted to **PDF format**. The filename should be **netid1_netid2_mp5_output.pdf**.
4. If you attempted the separate extra credit part of the assignment, the `MP5_DQN_LSTM_EC.ipynb` with output cells converted to **PDF format**. The filename should be **netid1_netid2_mp5_dqnlstm_ec_output.pdf**
5. A brief report in PDF format using **this template**. The filename you submit should be **netid1_netid2_mp5_report.pdf**.

Please refer to course policies on collaborations, late submission, etc.