DQN in Unity and Keras

A simple example of how to use DQN Reinforcement Learning in Unity using Keras. Included are (1) example Python scripts that illustrate single and two-agent DQN training and testing using Keras, and (2) a Unity package with two simple 2D unity games:

1. **Wall Pong:** A single agent game similar to pong. Agent moves a paddle to hit a ball against a wall.
2. **Pong:** A simple example of the classic two-agent Atari game.

The python agent connects to the unity game via a virtual (TCP) socket. To use the examples, you will need the following installed:

1. Python 2.7 - https://www.python.org/downloads/
2. Tensorflow - https://www.tensorflow.org/install/
3. Keras - https://keras.io
4. Unity - https://unity3d.com

NOTE 1: the python code has only been tested using Python 2.7, on a Mac-Book Pro. I recommend installing Keras and Tensorflow and running the python agents in a Py2.7 virtual environment.

To run the code:

1. Run a python training or testing script in terminal
2. Launch the corresponding game (either in the Unity editor or as a standalone)
3. Select AI type and click 'connect' in the game
4. Watch… and watch… and watch…and eventually a successful agent (training usually takes about 1 to 2 hours for WallPong and 2 to 4 for Multiagent Pong).

General Information:
Recently, Unity released a great toolbox for DQN using Tensorflow: https://blogs.unity3d.com/2017/09/19/introducing-unity-machine-learning-agents/?_ga=2.105619654.410151621.1506438822-872552068.1506438822).
I highly recommend you check this toolbox out.

Basically, this is much simpler version of the recently released Unity toolbox and illustrates how to do (more-or-less) the same thing using Keras.

Why Keras? Well, Keras offers a front end to Tensorflow and is much simpler to use if you are new to neural-networks and deep-learning. If you are interested in learning more about Keras and want a practical guide to how to use it for deep learning more generally, I recommend the *deep-learning in python using keras* series by Dr. Jason Brownlee at Machine Mastery: https://machinelearningmastery.com/deep-learning-with-python/

NOTE 2: Rather than using image data (as in the original DQN work), the current DQN agent(s) receive position and velocity data from the game. Which for most Unity and VR applications is preferable to image data, due to the computational cost of deep convolutional network architectures (i.e., no GPU required for these examples).

Keon Kim also has a great blog tutorial on DQN using Keras: https://keon.io/deep-q-learning/. Some of the DQN code provided here was adapted from Keon's tutorial, as well as various other educational/tutorial resources.

NOTE 3: I developed these examples in early 2017 for a number of students in my lab. I should have posted it on GitHub then…but there is never enough time in the day. Have fun!

For more reading on DQN see:

- Playing Atari with Deep Reinforcement Learning
- Human-level Control Through Deep Reinforcement Learning
- Multiagent cooperation and competition with deep reinforcement learning