The purpose of this homework is to learn how to play games using Deep Q-Network. In each of the games mentioned below, the objective for the agent is to win the game or maximize the total score achieved during the game:

a) **Breakout-v0.** Take a look here to learn how the game is played. The action in this game is to position the paddle right or left. Train a neural net that models the action-value function. Use discount factor = 0.95 and pre-process your images with the following code.

i) Plot (average) maximum of the Q function versus iteration to track progress of your network.

ii) After training your neural network for 24 hours, play 1000 games using your trained neural network and plot the sum of rewards in each game (no discounting here).

```
def preprocess(image):
    """ prepro 210x160x3 uint8 frame into 6400 (80x80) 2D float array """
    image = image[35:195] # crop
    image = image[::2,::2,0] # downsample by factor of 2
    image[image == 144] = 0 # erase background (background type 1)
    image[image == 109] = 0 # erase background (background type 2)
    image[image != 0] = 1 # everything else just set to 1
    return np.reshape(image.astype(np.float).ravel(), [80,80])
```

b) **MsPacman-v0.** Take a look here to learn how the game is played. Train a neural net that uses images of the game as state and models the action-value function. The discount factor is 0.99. This game has 9 actions.

You need to pre-process all the images from the game with the following function before feeding them into the neural network.

```
mspacman_color = 210 + 164 + 74
def preprocess_observation(obs):
    img = obs[1:176:2, ::2] # crop and downsize
    img = img.sum(axis=2) # to greyscale
    img[img==mspacman_color] = 0 # Improve contrast
    img = (img // 3 - 128).astype(np.int8) # normalize from -128 to 127
    return img.reshape(88, 80, 1)
```

i) Plot (average) maximum of the Q function versus iteration to track progress of your network.

ii) After training your neural network for 24 hours, play 1000 games using your trained neural network and plot the sum of rewards in each game (no discounting here).

Notes:

- Your DQN implementation must use a deep neural net, a replay buffer, and the notion of the target network.
- You are not allowed to use someones code and you are also not allowed to read or copy-and-paste someone else code.
- You can use Tensorflow and all of the functions that come with it (and all modules that are part of anaconda).
- All of you must use deepdish and your code must use a single GPU card.
- You are allowed to use any trick that can help with faster convergence.
- Your assignment score will be based on the performance of your algorithm and how much reward it gains. You will get the majority of the points as long as your code is correct and your plots show that the network is learning how to play. The following criterion will be used for to grade your assignment:
  - Top 20% performance get 100 points.
  - Those ranked from 50% to 80% get 95 points.
  - Those ranked from 20% to 50% get 92.5 points.
  - Bottmon 20% get 90 points.
  - The winner gets an extra 10 points (thus 110 points).