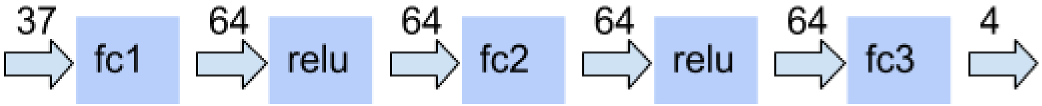
I. Model (QNetwork)

There's 37 input values and 4 output values



II. DQN (Deep Q-Network) Algorithm

Agent Structure

qnetwork_local - predict current state using the QNetwork (model shown above)
qnetwork_target - predict future state using the QNetwork (model shown above)
memory - array that holds the past state, action, reward, next state, and done values

Parameters

n_episodes	600	Number of times the environment is run
eps_start	1.0	Initial epsilon value
eps_end	0.01	Minimal epsilon value
eps_decay	0.7	Epsilon decay
seed	500	Seed of random function
BUFFER_SIZE	1e5	Replay (memory) buffer size
BATCH_SIZE	64	Minibatch size
GAMMA	0.99	Discount factor
TAU	1e-3	Soft update of target parameters
LR	5e-4	Learning rate
UPDATE_EVERY	4	How often to update the network

Overview

For each episode:

1) Reset the environment
2) Get the state of the environment

While the environment is running:

1) Use the agent to **predict action** based on the environment's state
2) Pass the predict action into the environment
3) Retrieve the next state, reward, and done variable
4) **Agent step**. Record next state, reward, and done variable to **memory** in agent
5) Set the current state variable to next state

6) Add reward to score
7) Break out of while loop if done is true

Predict action

1) Use **qnetwork_local** to predict action with state as input.
2) Re-train **qnetwork_local**
3) If a random value is greater then the epsilon value, choose the highest probably from predict action, otherwise randomly choose an action

Agent step

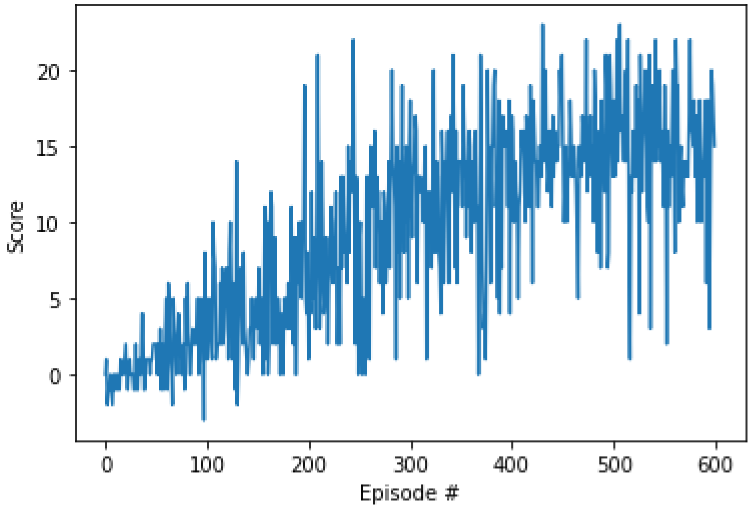
1) Add experience to memory

Do the following every 4th step (UPDATE_EVERY):

1) Randomly sample experiences from **memory**
2) Use **qnetwork_target** to Q target next states (**Q_targets_next**) with next state as input
3) Computes the Q target with the following formula
$Q\_targets = reward + (gamma * Q\_targets\_next * (1 - dones))$
4) Get the expected Q values (**Q_expected**) from local model (**qnetwork_local**) with state as input
5) Use **mse_loss** for **Q_expected** and **Q_targets_next**
6) Finally, update the weights of the target network (**qnetwork_target**) from the local network (**qnetwork_local**) using the following formula
$qnetwork\_target_{weights} = tau * qnetwork\_local_{weights} + (1 - tau) * qnetwork\_target_{weights}$

III. Results

Here's a graph of the episodes vs scores. As the episodes get closer to 600, we see the scores concentrate between 10 and 18.



IV. Ideas for Future work

There are other DQN improvements that I would like to pursue such as:

1) Double DQN
2) Prioritized Experience Replay
3) Dueling DQN