The report clearly describes the learning algorithm, along with the chosen hyperparameters. It also describes the model architectures for any neural networks.

A plot of rewards per episode is included to illustrate that the agent is able to receive an average reward (over 100 episodes) of at least +13. The submission reports the number of episodes needed to solve the environment.

**Project report**

**Implementation of learning algorithm**

The first learning algorithm I implanted is vanilla Deep Q Learning with fixed Q-target and random experience replay by making only minor modifications to the DQN code provided as part of the Deep Q-Networks lesson. This allows me to familiar with the environment and get my code working with the Unity Machine Learning Agents.

But the input is not an image and so fully connected layers are used instead of convolutional layers. The details of the network is provided below:

* Fully connected layer - input: 37 (state size) output: 64 activation : Relu
* Fully connected layer - input: 64 output 64 activation : Relu
* Fully connected layer - input: 64 output: 4 (action size)

Hyperparameters:

* Replay buffer size = 1e5
* Batch size = 128
* Discount factor (GAMMA) = 0.99
* TAU (for soft update of target parameters) = 1e-3
* Learning rate = 5e-4
* Update network frequency: every 4 steps
* Maximum steps per episode: 2000
* Maximum episodes = 2000
* Starting epsilon = 1.0
* Ending epsilon = 0.01
* Epsilon decay rate = 0.995

Episode 100 Average Score: 0.95

Episode 200 Average Score: 4.46

Episode 300 Average Score: 7.67

Episode 400 Average Score: 10.23

Episode 500 Average Score: 12.33

Episode 515 Average Score: 13.04

Environment solved in 415 episodes! Average Score: 13.04

Dueling DQN

Episode 100 Average Score: 0.23

Episode 200 Average Score: 2.67

Episode 300 Average Score: 6.31

Episode 400 Average Score: 8.15

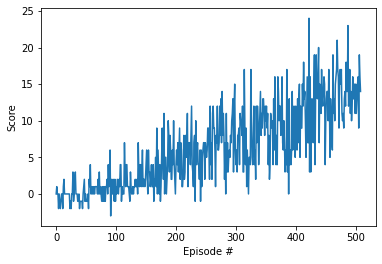
Episode 500 Average Score: 12.67

Episode 509 Average Score: 13.01

Environment solved in 409 episodes! Average Score: 13.01

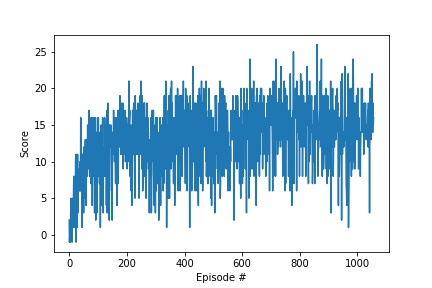
2021-04-16 18:32:40.095683

2021-04-16 18:45:23.795010



**Results**

**Training Plot**

Plot showing the RL training with respect to average rewards over last 100 episodes. [](https://github.com/karthi0804/DQN_banana/blob/master/images/avg_score_plot.jpg)

**Training Output**

Episode 100 Average Score: 7.89

Episode 200 Average Score: 9.83

Episode 300 Average Score: 10.65

Episode 400 Average Score: 11.18

Episode 500 Average Score: 11.60

Episode 600 Average Score: 11.84

Episode 700 Average Score: 12.20

Episode 800 Average Score: 12.51

Episode 900 Average Score: 12.75

Episode 1000 Average Score: 12.88

Episode 1057 Average Score: 13.00

Environment solved in 957 episodes! Average Score: 13.00

**Ideas for future work**

1. Extensive hyperparameter optimization
2. Double Deep Q Networks
3. Prioritized Experience Replay
4. Dueling Deep Q Networks
5. Rainbow: Combining Improvements in Deep Reinforcement Learning

There are many aspects in which I can improve this project. Implementing dueling DQN or prioritized experience replay would likely improve the number of episodes until solved.

Now that this environment is solved, I should try out solving the same game, but with a different state representation. Representing the state as raw pixels would be more realistic since raw pixels are what we see, and we should allow the agent to see the same.