Project 1: Navigation

Description of the implementation

Starting algorithm

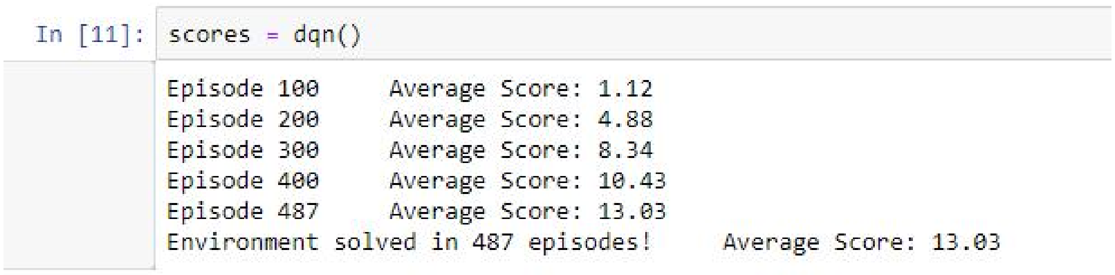
In order to solve this challenge, Deep Q-Network (DQN) was implemented by exploring different hyperparameters and neural network hidden layers & nodes.

As a start, Experience Replay and Fixed Q-Targets was used. Original hyperparameters values from Udacity Deep Q-Network solution codes (which was used to solve a different problem) was used as a starting point. This version of DQN of has 2 fully-connected (FC) layers, each containing 64 nodes (no dropouts) and the following hyperparameters.

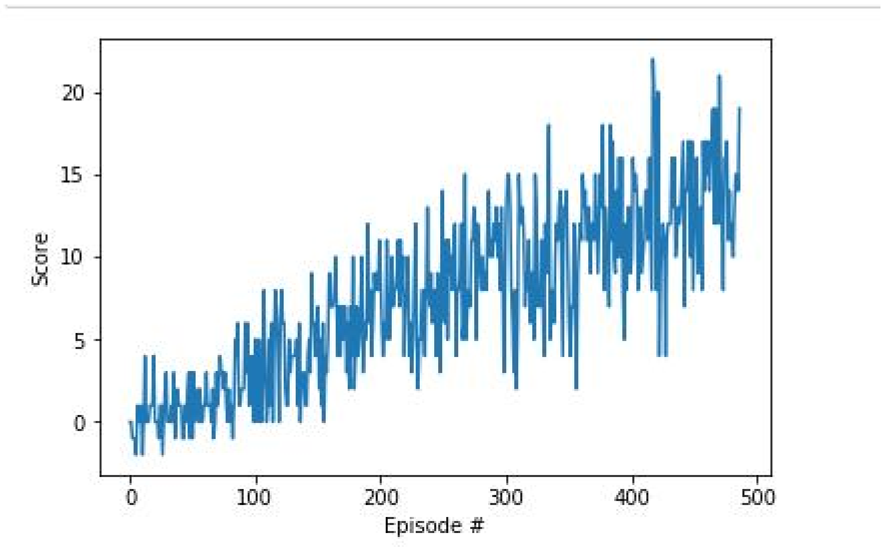
- BATCH_SIZE = 64 # minibatch size
- GAMMA = 0.99 # discount factor
- TAU = 1e-3 # for soft update of target parameters
- LR = 5e-4 # learning rate
- UPDATE_EVERY = 4 # how often to update the network
- Epsilon Decay=0.995
- Neural Network: No dropout 2 hidden layers, each with 64 nodes (64,64)
- Replay buffer size=1e5
- Gamma (discount factor)=0.99
- Number of episodes=1000
- Max number of timesteps per episode=1000
- Epsilon start=1
- Epsilon minimum=0.01

This model solved the environment in **487 episodes** and will become the baseline model for comparison with future iterations. Even though challenge was solved in 487 episodes, agent was allowed to continue learning and interacting with the environment to evaluate if "number of episodes" was the limiting factor for agent to learn.

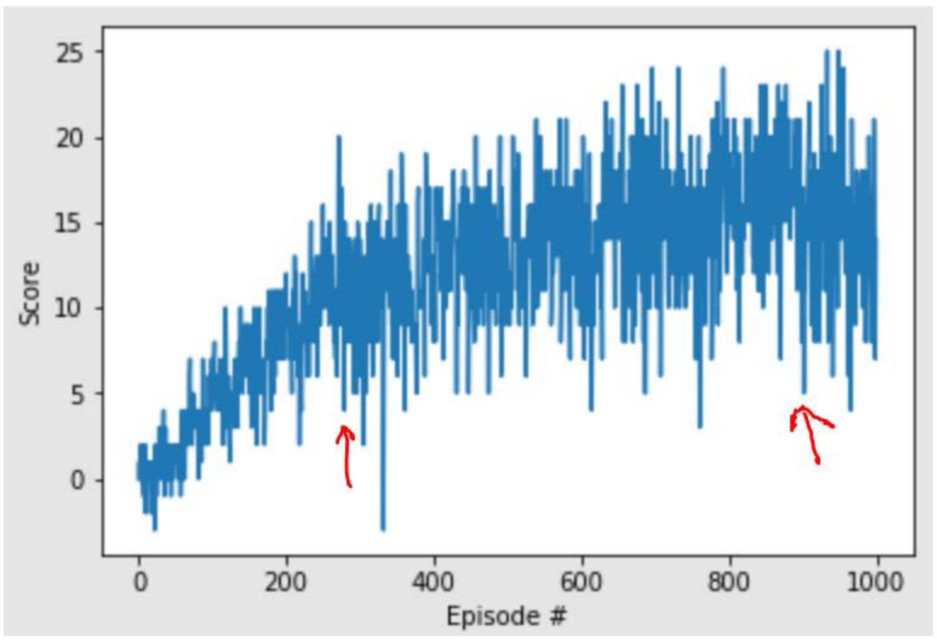
Baseline results:



```
In [11]: scores = dqn()

         Episode 100     Average Score: 1.12
         Episode 200     Average Score: 4.88
         Episode 300     Average Score: 8.34
         Episode 400     Average Score: 10.43
         Episode 487     Average Score: 13.03
         Environment solved in 487 episodes!     Average Score: 13.03
```



Baseline results (extrapolated to 1000 episodes to identify bottleneck):



The next improvements to be explored hinged upon the following 2 observations:

a) **Observation 1:** The average score started to plateau after ~ episode 300 and started dropping slightly after episode 900.

- This suggests that “increasing the number of episodes” has hit diminishing marginal return around episode 300 and we might need to provide more samples for agent to learn or help agent to learn faster

- **High level improvement suggestions:**

    - Increase batch size to provide more samples for agent to learn

    - Increase the FC layers and Nodes to help agent learn more nuanced details of the environment, and at the same time control for overfitting by introducing dropout

    - Increase learning rate to help agent learn faster

    - Increase decay rate so that agent converges faster by being more greedy earlier (at 0.995 of decay rate, epsilon is still 0.22 by episode 300 and agent might not need to explore as much anymore).

b) **Observation 2:** The fluctuation of episode scores increased significantly after ~ episode 300

- This suggests that after ~ episode 300, the agent started running into the “moving target” problem

- **High level improvement suggestions:**

    - Increase duration of soft update as number of episode increases

    - Decrease value of Tau as number of episode increases

    - Try hard update but increase duration of update more than what is used for soft update

Start of exploration to improve agent performance

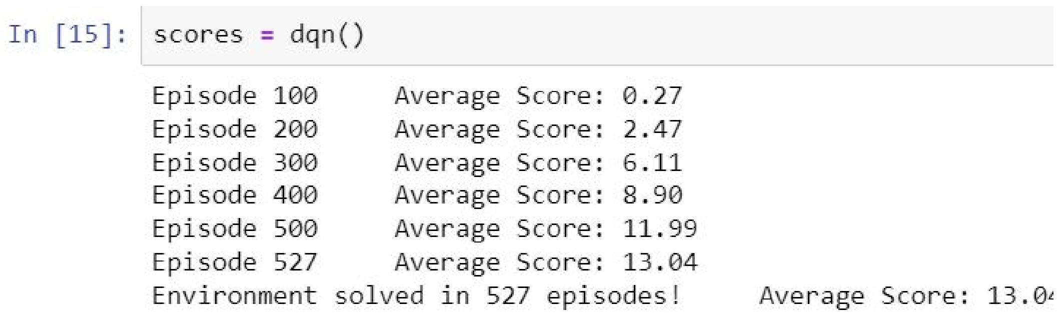
Exploration 1: Change of FC layers & nodes and introduce dropout

- Increase the FC layers and Nodes to help agent learn more nuanced details of the environment, and at the same time control for overfitting by introducing dropout

    - Increasing layer without changing number of nodes

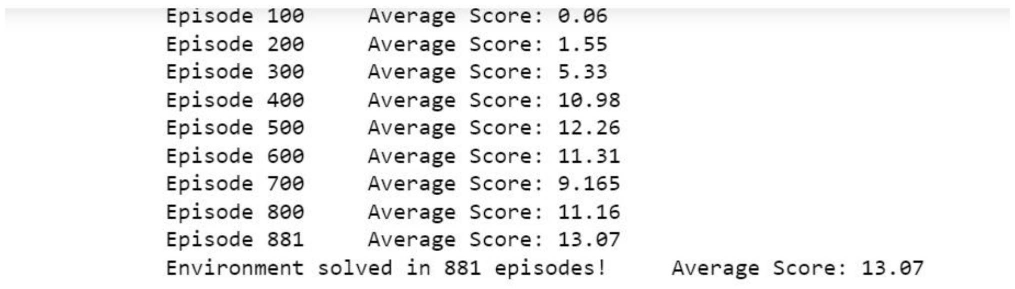
        - Tried 64,64,64,64 (4 layers):

            - With NO dropout, performance dropped slightly (might be due to stochastic nature of environment too)



```
In [15]: scores = dqn()

         Episode 100     Average Score: 0.27
         Episode 200     Average Score: 2.47
         Episode 300     Average Score: 6.11
         Episode 400     Average Score: 8.90
         Episode 500     Average Score: 11.99
         Episode 527     Average Score: 13.04
         Environment solved in 527 episodes!     Average Score: 13.0
```

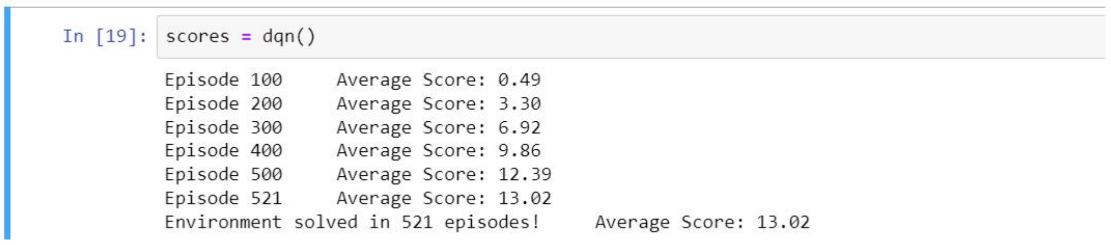
- With 0.2 drop out, performance became worse



```
Episode 100     Average Score: 0.06
Episode 200     Average Score: 1.55
Episode 300     Average Score: 5.33
Episode 400     Average Score: 10.98
Episode 500     Average Score: 12.26
Episode 600     Average Score: 11.31
Episode 700     Average Score: 9.165
Episode 800     Average Score: 11.16
Episode 881     Average Score: 13.07
Environment solved in 881 episodes!     Average Score: 13.07
```

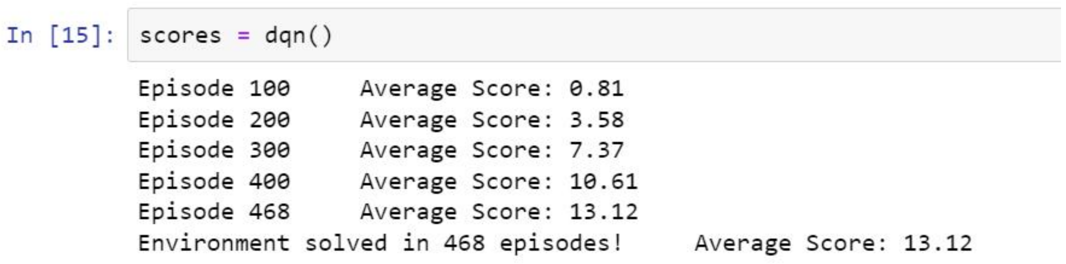
- Increasing nodes without changing number of layers

  - With NO drop out, tried 128,128 (2 layers, but double original nodes), performance dropped slightly (might be due to stochastic nature of environment too)



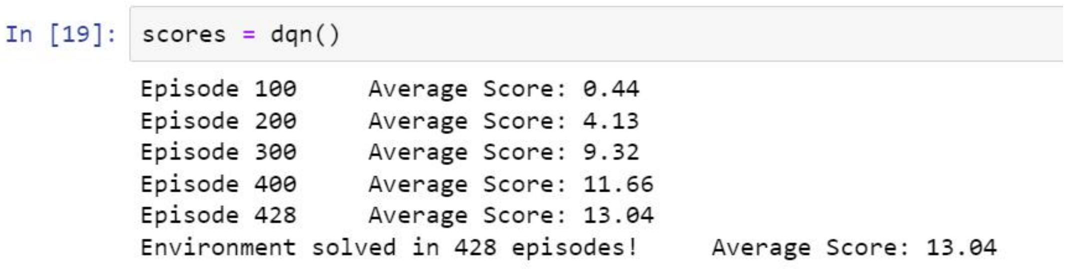
```
In [19]: scores = dqn()

         Episode 100     Average Score: 0.49
         Episode 200     Average Score: 3.30
         Episode 300     Average Score: 6.92
         Episode 400     Average Score: 9.86
         Episode 500     Average Score: 12.39
         Episode 521     Average Score: 13.02
         Environment solved in 521 episodes!     Average Score: 13.02
```

  - With dropout of 0.2, Tried 128,128 (2 layers, but double original nodes), and now the performance has **started improving!**



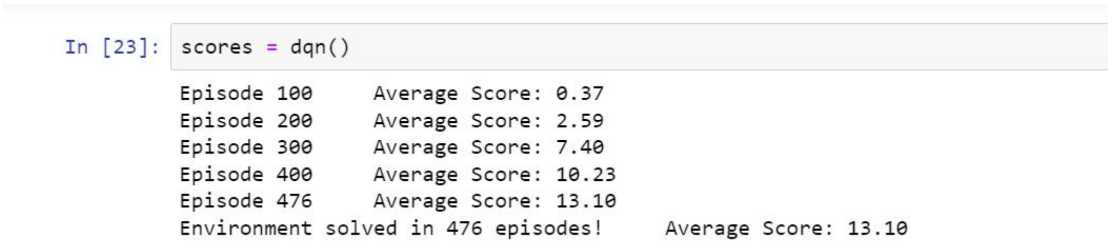
```
In [15]: scores = dqn()

         Episode 100     Average Score: 0.81
         Episode 200     Average Score: 3.58
         Episode 300     Average Score: 7.37
         Episode 400     Average Score: 10.61
         Episode 468     Average Score: 13.12
         Environment solved in 468 episodes!     Average Score: 13.12
```

- Since increasing nodes with dropout improved performance, try increasing nodes while keeping number of layers (300,300) and maintaining 0.2 dropout. This had further improved performance



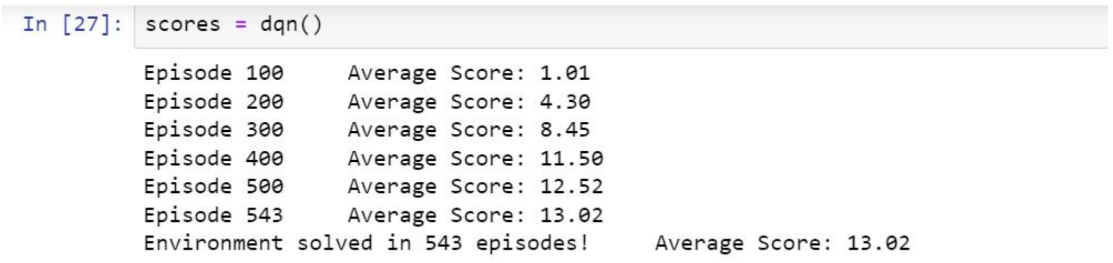
```
In [19]: scores = dqn()

         Episode 100     Average Score: 0.44
         Episode 200     Average Score: 4.13
         Episode 300     Average Score: 9.32
         Episode 400     Average Score: 11.66
         Episode 428     Average Score: 13.04
         Environment solved in 428 episodes!     Average Score: 13.04
```

- To try to see if increasing of layers with the same number of nodes helped with performance (300,300,300) while keeping 0.2 dropout. Seems like increasing layers decreased performance slightly (might be due to stochastic nature of the environment)



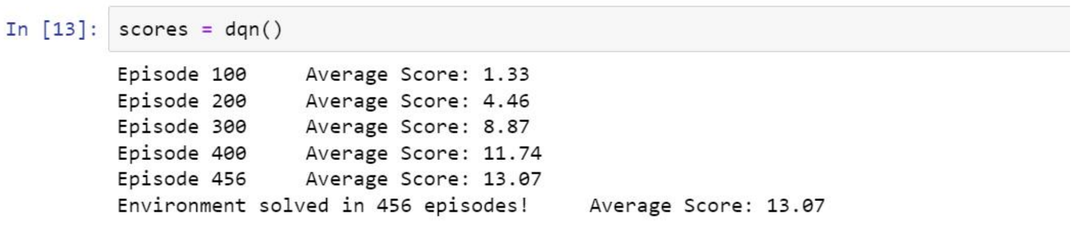
```
In [23]: scores = dqn()

         Episode 100     Average Score: 0.37
         Episode 200     Average Score: 2.59
         Episode 300     Average Score: 7.40
         Episode 400     Average Score: 10.23
         Episode 476     Average Score: 13.10
         Environment solved in 476 episodes!     Average Score: 13.10
```

- Keeping layers back to 2 but double the nodes again (600,600) with dropout of 0.2. This caused performance to drop



```
In [27]: scores = dqn()

         Episode 100     Average Score: 1.01
         Episode 200     Average Score: 4.30
         Episode 300     Average Score: 8.45
         Episode 400     Average Score: 11.50
         Episode 500     Average Score: 12.52
         Episode 543     Average Score: 13.02
         Environment solved in 543 episodes!     Average Score: 13.02
```

- Finally, changing layer to (300,150) with dropout 0.2. This did gave a worse performance compared to layer (300,300) with dropout 0.2.



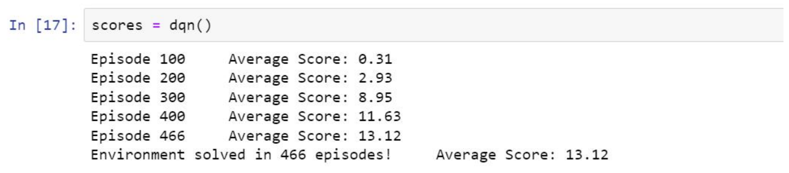
```
In [13]: scores = dqn()

         Episode 100     Average Score: 1.33
         Episode 200     Average Score: 4.46
         Episode 300     Average Score: 8.87
         Episode 400     Average Score: 11.74
         Episode 456     Average Score: 13.07
         Environment solved in 456 episodes!     Average Score: 13.07
```

Mini conclusion from Exploration 1: Without drop out, Increasing nodes or layers of neural network did not help improve performance. With dropout of 0.2, increasing the number of nodes (keeping layers=2) to (300,300) helped improve performance so (300,300) will be used as an optimal version (with **428 episodes** as the new target upon which to improve) before moving on to Exploration 2 in changing hyperparameters.

Exploration 2: Change of Hyperparameters

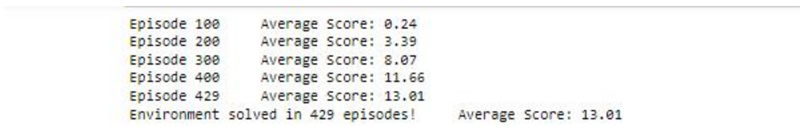
- Increase batch size to provide more samples for agent to learn

  - Increase batch size from 64 to 128 but this did not improve performance.



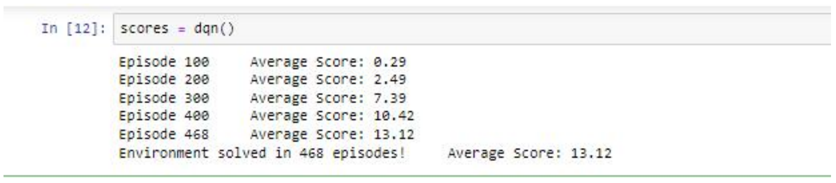
```
In [17]: scores = dqn()

Episode 100	Average Score: 0.31
Episode 200	Average Score: 2.93
Episode 300	Average Score: 8.95
Episode 400	Average Score: 11.63
Episode 466	Average Score: 13.12
Environment solved in 466 episodes!	Average Score: 13.12
```

  - batch size was increased from 64 to 300 but again performance did not improve.



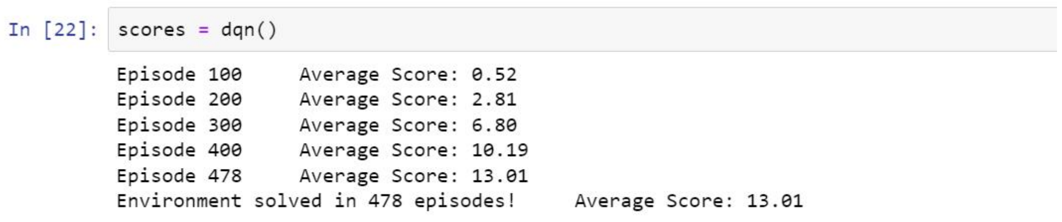
```
Episode 100	Average Score: 0.24
Episode 200	Average Score: 3.39
Episode 300	Average Score: 8.07
Episode 400	Average Score: 11.66
Episode 429	Average Score: 13.01
Environment solved in 429 episodes!	Average Score: 13.01
```

  - Batch size increased to 2000. Took a long time to run but no improvement in performance. Suggest to stick to original 64



```
In [12]: scores = dqn()

Episode 100	Average Score: 0.29
Episode 200	Average Score: 2.49
Episode 300	Average Score: 7.39
Episode 400	Average Score: 10.42
Episode 468	Average Score: 13.12
Environment solved in 468 episodes!	Average Score: 13.12
```

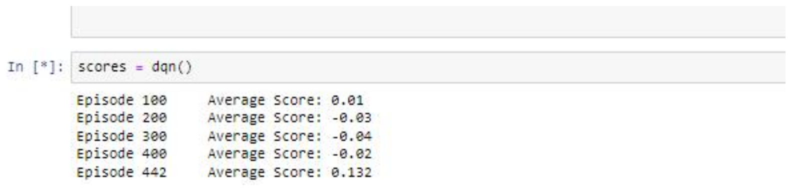
  - Just for experimentation, batch size was decreased from 64 to 32. As expected, performance worsened so for this problem, suggest to not decrease batch size at all



```
In [22]: scores = dqn()

Episode 100	Average Score: 0.52
Episode 200	Average Score: 2.81
Episode 300	Average Score: 6.80
Episode 400	Average Score: 10.19
Episode 478	Average Score: 13.01
Environment solved in 478 episodes!	Average Score: 13.01
```

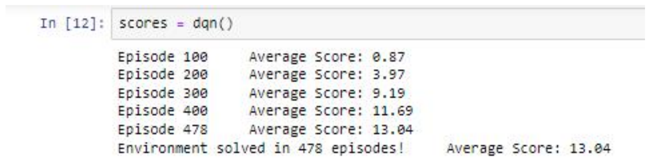
- Increase learning rate to help agent learn faster

  - Increase learning rate from 5e-4 to 5e-3. However, this increase was too high as agent performance could not converge so learning was ended abruptly to avoid wasting time waiting for a non-converging agent trying to converge.



```
In [*]: scores = dqn()

        Episode 100     Average Score: 0.01
        Episode 200     Average Score: -0.03
        Episode 300     Average Score: -0.04
        Episode 400     Average Score: -0.02
        Episode 442     Average Score: 0.132
```

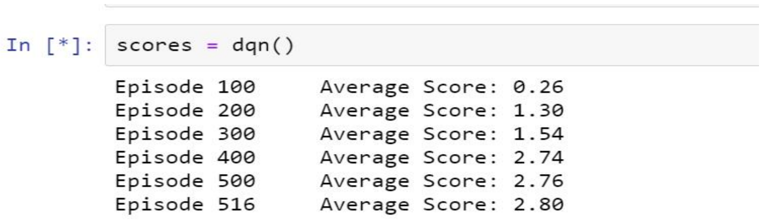
- Increase learning rate but not too much, from 5e-4 to 7e-4. This too did not improve performance hence the original 5e-4 will be used



```
In [12]: scores = dqn()

         Episode 100     Average Score: 0.87
         Episode 200     Average Score: 3.97
         Episode 300     Average Score: 9.19
         Episode 400     Average Score: 11.69
         Episode 478     Average Score: 13.04
         Environment solved in 478 episodes!     Average Score: 13.04
```

➢ Decrease value of Tau

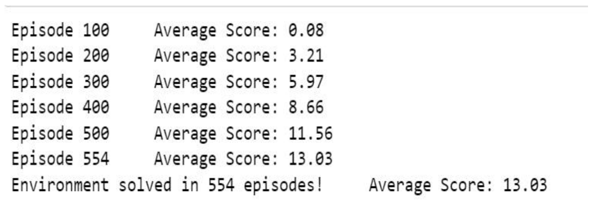
- Decrease value of tau from 1e-3 to 1e-4. This decreased performance drastically as the score was still hovering at 2.8 after 500 episodes so learning was ended abruptly to avoid wasting time waiting for a non-converging agent trying to converge.



```
In [*]: scores = dqn()

        Episode 100     Average Score: 0.26
        Episode 200     Average Score: 1.30
        Episode 300     Average Score: 1.54
        Episode 400     Average Score: 2.74
        Episode 500     Average Score: 2.76
        Episode 516     Average Score: 2.80
```

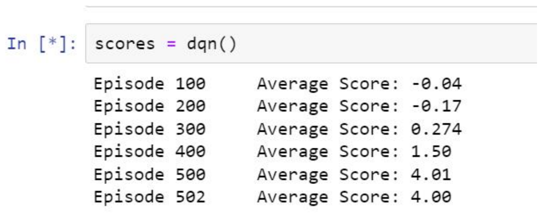
➢ Increase duration of soft update .

- Double duration of soft update from 4 to 8. This decreased performance too



```
Episode 100     Average Score: 0.08
Episode 200     Average Score: 3.21
Episode 300     Average Score: 5.97
Episode 400     Average Score: 8.66
Episode 500     Average Score: 11.56
Episode 554     Average Score: 13.03
Environment solved in 554 episodes!     Average Score: 13.03
```

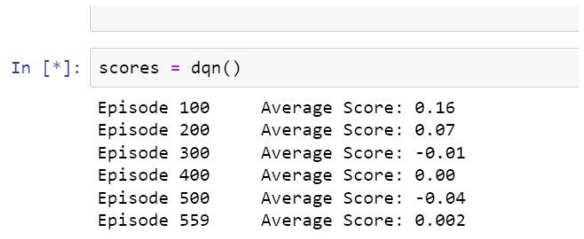
- Just to introduce some variation in experimentation, double duration of soft update from 4 to 20 and increase Tau to 0.2. However, this also worsened performance



```
In [*]: scores = dqn()

        Episode 100     Average Score: -0.04
        Episode 200     Average Score: -0.17
        Episode 300     Average Score: 0.274
        Episode 400     Average Score: 1.50
        Episode 500     Average Score: 4.01
        Episode 502     Average Score: 4.00
```

➢ Try hard update (tau=1) but increase duration of update to more than what is used for soft update

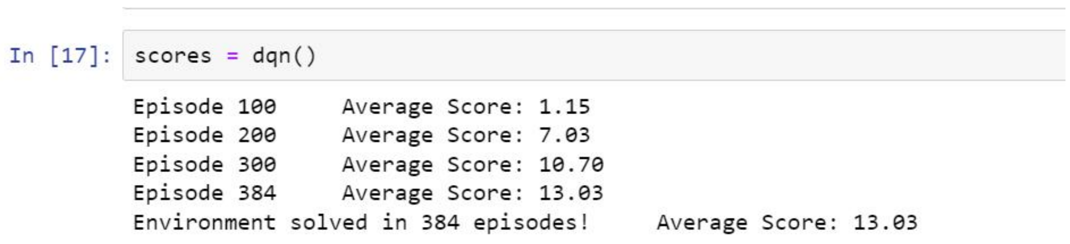
- Increase value of tau to 1 but increase duration of update to 10 but agent performance could not converge even after 500 episodes so learning was stopped abruptly.



```
In [*]: scores = dqn()

Episode 100     Average Score: 0.16
Episode 200     Average Score: 0.07
Episode 300     Average Score: -0.01
Episode 400     Average Score: 0.00
Episode 500     Average Score: -0.04
Episode 559     Average Score: 0.002
```

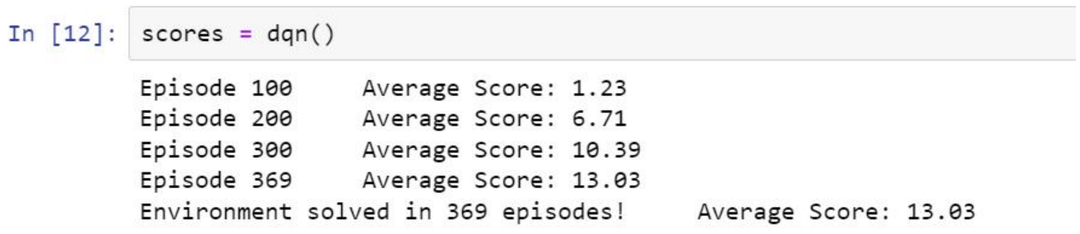
➢ Increase decay rate of epsilon so that agent will converge faster

- Speeding up decay from 0.995 to 0.990 so that by episode 300, epsilon can be reduced to ~0.05 to avoid wasting time exploring environment as the agent's knowledge becomes more reliable. This caused the agent to learn much faster and solving the environment in 384 episodes!



```
In [17]: scores = dqn()

Episode 100     Average Score: 1.15
Episode 200     Average Score: 7.03
Episode 300     Average Score: 10.70
Episode 384     Average Score: 13.03
Environment solved in 384 episodes!     Average Score: 13.03
```

- Just for experimentation's sake, speed up decay from 0.995 to 0.980 and this improved performance slightly to 369 episodes (this might be due to stochastic nature of the environment though but we will still use 0.980 this as the final hyperparameter)



```
In [12]: scores = dqn()

Episode 100     Average Score: 1.23
Episode 200     Average Score: 6.71
Episode 300     Average Score: 10.39
Episode 369     Average Score: 13.03
Environment solved in 369 episodes!     Average Score: 13.03
```

Final list of hyperparameters and neural network architecture used (changes made to the original hyperparameters are highlighted in red below):

✓ Hyperparameters explored:

- BATCH_SIZE = 64 # minibatch size
- GAMMA = 0.99 # discount factor
- TAU = 1e-3 # for soft update of target parameters
- LR = 5e-4 # learning rate

- UPDATE_EVERY = 4   # how often to update the network
- Epsilon Decay=0.98 (changed)
- Neural Network: dropout of 0.2, 2 hidden layers, each with 300 nodes (300,300) (changed)

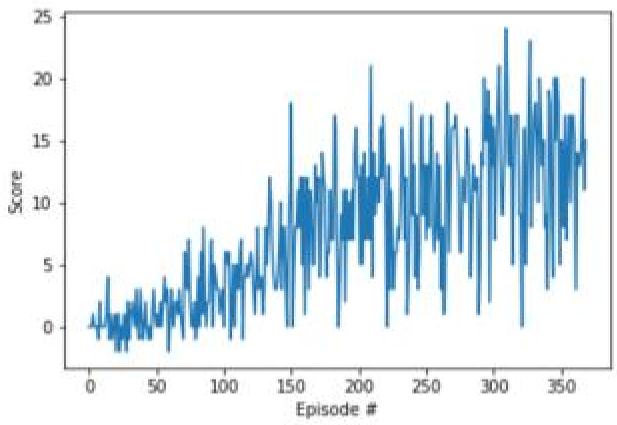
✓ Hyperparameters NOT explored:

- Replay buffer size=1e5
- Gamma (discount factor)=0.99
- Number of episodes=1000
- Max number of timesteps per episode=1000
- Epsilon start=1
- Epsilon minimum=0.01

Plot of Rewards

This graph below shows that the Agent is able to receive an average reward of at least +13 over 100 episodes by **Episode 369**.

```
Episode 100	Average Score: 1.23
Episode 200	Average Score: 6.71
Episode 300	Average Score: 10.39
Episode 369	Average Score: 13.03
Environment solved in 369 episodes!	Average Score: 13.03
```



Conclusion

Adding more nodes to the fully connected layers of neural network (without increasing number of layers) with 0.2 dropout improved original agent performance from 498 to 428 episodes. After tuning multiple hyperparameters, the only one which generated noticeable improvements to agent performance was the speeding up of epsilon decay rate from 0.995 to 0.980 as the agent did not have to spend too much effort exploring environment as the number of episodes increased. Changing of this decay rate improved agent performance further from 428 episodes to **369 episodes**.

Ideas for Future Work

1. To improve agent's performance:
   a) A combination of some or all of the following can be explored:
      i. double DQN
      ii. dueling DQN
      iii. prioritized experience replay
   b) Further experimentation with hyperparameters using scheduler (as number of episodes changes, hyperparameters changes accordingly). Some examples:
      i. Since the target Q becomes more reliable as an agent gains more experience with more episodes, we can:
         - Decrease Tau as number of episodes increases
         - Increase the UPDATE_EVERY variable as number of episodes increases
         - Decrease epsilon (by increasing decay rate) as number of episodes increases as an agent can rely lesser on exploration and exploit on the knowledge gained after many episodes to maximize future reward
2. To increase confidence level of the outcome of hyperparameter or neural network architecture change, GridSearch with Cross-Validation could be explored to ensure each iteration of improvements was not due to luck but rather the actual effect of model change.
3. To remove the need of hand-crafting of states by humans , which sometimes might not be the best representation of the environment (especially when it comes to solving new environments), we could learn directly from pixels with the addition of CNN to process game frames.
   a) While this might make it harder for agent to learn, it helps with generalization in the sense that we can use the same agent (same hyperparameters and neural network architecture) across different environments and still might get decent performance.