Navigation Report

Andy Gooden

This report summarizes my solution for the Navigation Project for Udacity's Deep Reinforcement Learning course. The goal of the project was to learn a policy for an agent that is trying to collect bananas. We use a Deep Q-Network to approximate the optimal Action-Value function, which, given the agent's current state, will return the value for all possible actions. The policy can then choose either the action with the highest value, or a random action if an epsilon-greedy policy is used.

Model Architecture:

The model, a Deep Q-Network (DQN), is defined in the `model.py` file. I used a three layer feed forward neural network. The first two layers have 256 nodes with the ReLU activation function. The final (output) layer has four nodes, one for each possible action. The output value for each of the actions represents the Q-Networks value for that action given a state. I experimented with various numbers of nodes for the hidden layers, ranging from 32 hidden nodes up to 512 hidden nodes. I was able to achieve the best performance (highest average score over 100 episodes) with 256 hidden nodes. I did not see any appreciable performance gain with 512 hidden nodes. The experimental output from my best configuration is listed in Appendix A, with the other permutations that I tried listed in Appendix B.

Learning Algorithm

Training the DQN is handled in the `dqn_agent.py` file. The goal of the learning algorithm is to get our DQN as close as possible to the optimal Action-Value function (the target-function). This is done by adjusting the weights of our DQN to minimize the square error between the target-function and our current function. However, we do not know the target-function, so we approximate it by using a second DQN (target_dqn). At each learning step, we take the current reward plus the value of our target_dqn at the next state (using a greedy policy) and compare that to the value of our current DQN. The difference between our current_dqn and target_dqn is the error signal that is fed into backpropagation. I used Torch's `mse_loss` function coupled with the `adam` optimizer and I fed in batches of size 128 for each training loop. I found that larger batches gave slightly better performance and slightly lower variance of scores within the 100 episode window. Additionally, I did not have the agent learn every time step in an episode, but rather after N time steps, specified by the LEARN_EVERY parameter. This was set to 16 in my final implementation.

Fixed Q-Targets:

If the weights of our target_dqn and current_dqn were the same, then these functions would be correlated, inhibiting proper learning. To solve this, we fix the weights of the target_dqn for several learning cycles while the weights of the current_dqn are updated. After some fixed number of learning cycles, the weights from the current_dqn are copied over to the target_dqn. This technique decouples the two DQNs and is called "Fixed Q-Targets". I experimented with different intervals for updating the target_dqn weights, as well as using a "soft-update" where the weights of the target_dqn were updated every learning step as a linear combination (via parameter TAU) of the target_dqn and current_dqn weights. Via experimentation, my best setup used a TAU of 1e-3 (a soft update) every learning cycle.

Experience Replay

At each step, we create an experience tuple (ET), which contains the current state, current action, reward, next state and a boolean denoting if we hit the goal state. The ET's in a sequence are highly correlated with each other and can negatively impact the training of the DQN. To combat this, we create an ET buffer, implemented as a deque, and push in ET's as they arrive. When it's time to train the DQN, we randomly sample ET's from the buffer to form our batch of data. This random sampling breaks up correlated patterns in the ET's. My implementation stores 10,000 ET's in the buffer.

Policy

My agent operates under an epsilon greedy policy. During training, epsilon is initially set to 1.0 such that all actions are random. We decay epsilon by .995 at the end of each episode until we hit a minimum epsilon of 0.01

Best Experimental Results

The different experiments that I ran are shown in Appendix B. I varied many of the hyperparameters such as learning rate, fixed Q-target update interval, learning interval, learning batch size, and the model's hidden layer sizes.

My best run solved the environment in 740 episodes with an average score over 100 runs of 13.08. I continued to allow the agent to learn for the full 2000 episodes and reached a maximum average score of 15.10. Figure 1 shows a plot of the rewards per episode. While the plot is quite noisy, it shows the average reaching 13 around episode 800 and climbing slightly after that. Tabular output of average score, score variance vs episode number is shown below as well as the chosen hyperparameters. The model that first solved the environment at episode 740 is saved in the Github repo as `checkpoint.pth`. The model that reached the maximum average score at episode 2000 is saved in the Github repo as `final.pth`.

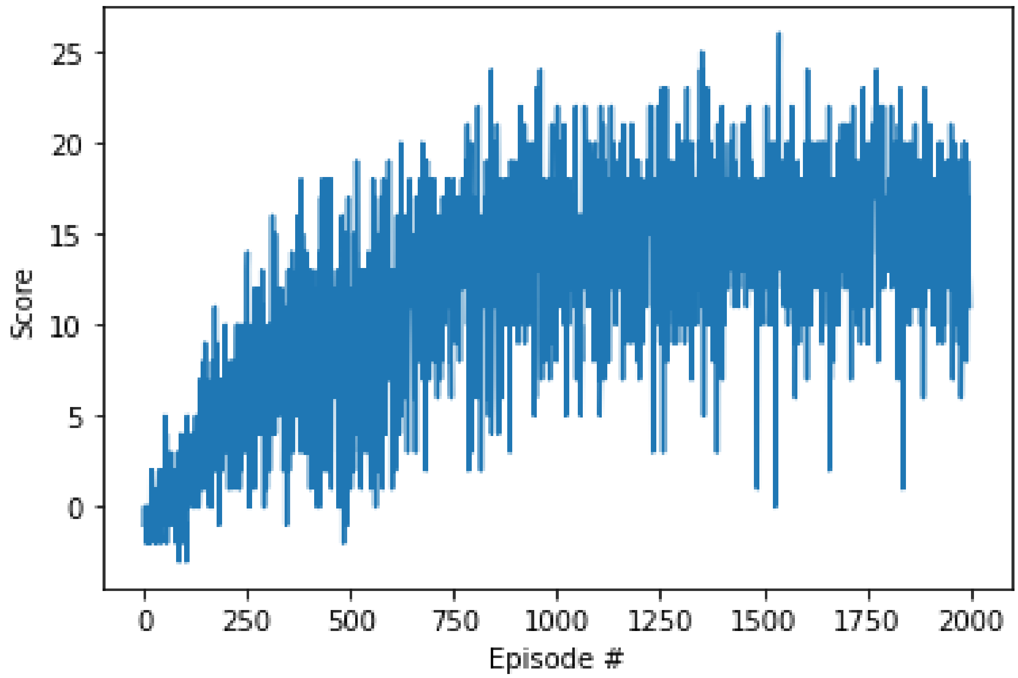


Figure 1: Score for Each Episode During Training

Chosen Hyperparameters

BUFFER_SIZE = int(1e5)	# replay buffer size
BATCH_SIZE = 128	# minibatch size
GAMMA = 0.99	# discount factor
TAU = 1e-3	# for soft update of target parameters
LR = 5e-5	# learning rate
LEARN_EVERY = 16	# how often to learn and update weights
UPDATE_TARGET_EVERY = 1	# how often to update target network w soft update
EPS_START = 1.0	
EPS_END = 0.01	
EPS_DECAY = 0.995	
MODEL_LAYER1_SIZE = 256	
MODEL_LAYER2_SIZE = 256	

Future Improvements

One of the first things I would like to try is Prioritized Experience Replay, which weights samples according to their effect on the update. In a similar vein, I would like to try weighting experience tuples in the sampling process based on their reward. This would mean states that directly led to capturing a banana would be sampled more often, helping to propagate that value to other states potentially more quickly.

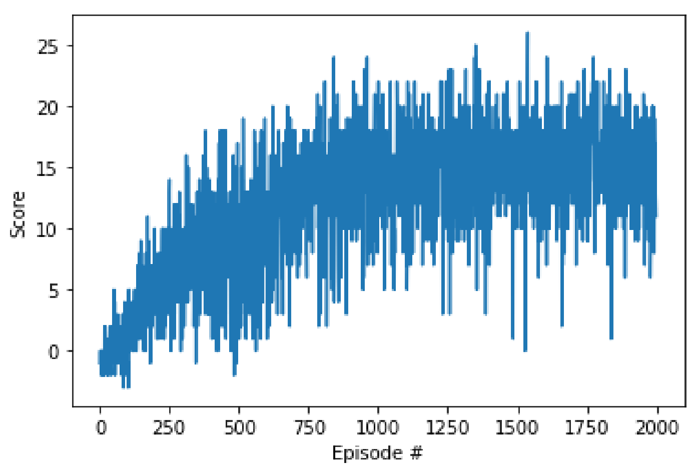
The second improvement would be to implement a Double DQN. Here, we would use the weights from our current_dqn to choose the next action which is used to evaluate the target_dqn. This helps to compensate for the over-estimation of the DQN's value.

Finally, I would like to experiment with the value of the discount factor, gamma. I would like to gather some statistics on how many decisions are made to capture a banana, and see if reducing the discount factor has any impact on longer or shorter decision sequences.

Appendix A - Best Model

```
BUFFER_SIZE = int(1e5)          # replay buffer size
BATCH_SIZE = 128                # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 1e-3                      # for soft update of target parameters
LR = 5e-5                       # learning rate
LEARN_EVERY = 16                # how often to learn and update weights
UPDATE_TARGET_EVERY = 1         # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 256
MODEL_LAYER2_SIZE = 256
```

```
Episode 100     Average Score: 0.33     Score Variance: 2.28
Episode 200     Average Score: 3.51     Score Variance: 7.39
Episode 300     Average Score: 5.91     Score Variance: 9.16
Episode 400     Average Score: 8.13     Score Variance: 14.09
Episode 500     Average Score: 8.07     Score Variance: 22.97
Episode 600     Average Score: 8.53     Score Variance: 21.43
Episode 700     Average Score: 11.73    Score Variance: 17.26
Episode 740     Average Score: 13.08
Environment solved in 740 episodes!     Average Score: 13.08
Episode 800     Average Score: 13.42    Score Variance: 12.78
Episode 900     Average Score: 13.45    Score Variance: 17.23
Episode 1000    Average Score: 14.57    Score Variance: 14.81
Episode 1100    Average Score: 14.88    Score Variance: 12.85
Episode 1200    Average Score: 14.92    Score Variance: 14.61
Episode 1300    Average Score: 14.85    Score Variance: 13.87
Episode 1400    Average Score: 15.37    Score Variance: 17.11
Episode 1500    Average Score: 14.91    Score Variance: 11.20
Episode 1600    Average Score: 15.07    Score Variance: 14.53
Episode 1700    Average Score: 15.39    Score Variance: 13.26
Episode 1800    Average Score: 15.75    Score Variance: 11.85
Episode 1900    Average Score: 15.61    Score Variance: 12.24
Episode 2000    Average Score: 15.10    Score Variance: 11.91
Final Model had Average Score: 15.10.  Saving model to: final.pth
total train time:  1669.865140914917
```



Appendix B - Other Experimental Runs

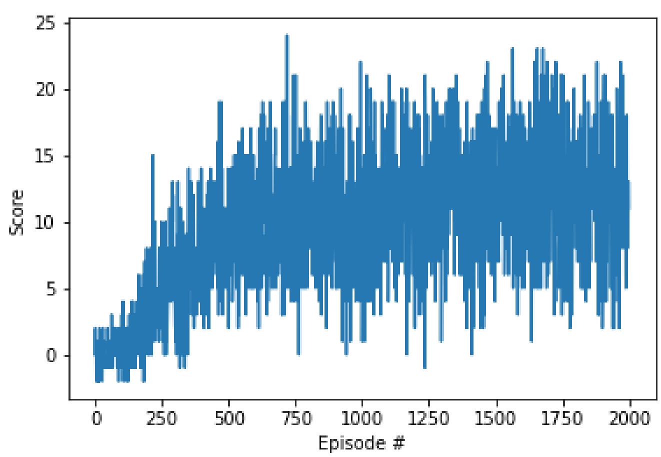
Experiment 1

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 1e-3 # for soft update of target parameters
LR = 5e-3 # learning rate
UPDATE_EVERY = 4 # how often to update the network
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995

MODEL: fc1_units=256, fc2_units=256

```
Episode 100	Average Score: 0.09
Episode 200	Average Score: 1.48
Episode 300	Average Score: 4.93
Episode 400	Average Score: 6.71
Episode 500	Average Score: 8.76
Episode 600	Average Score: 10.10
Episode 700	Average Score: 10.26
Episode 800	Average Score: 10.71
Episode 900	Average Score: 10.87
Episode 1000	Average Score: 9.41
Episode 1100	Average Score: 10.89
Episode 1200	Average Score: 12.44
Episode 1300	Average Score: 11.48
Episode 1400	Average Score: 12.76
Episode 1500	Average Score: 11.61
Episode 1600	Average Score: 12.48
Episode 1700	Average Score: 12.56
Episode 1800	Average Score: 12.36
Episode 1900	Average Score: 11.65
Episode 2000	Average Score: 12.15
```

```
total train time:  2173.3915758132935
```



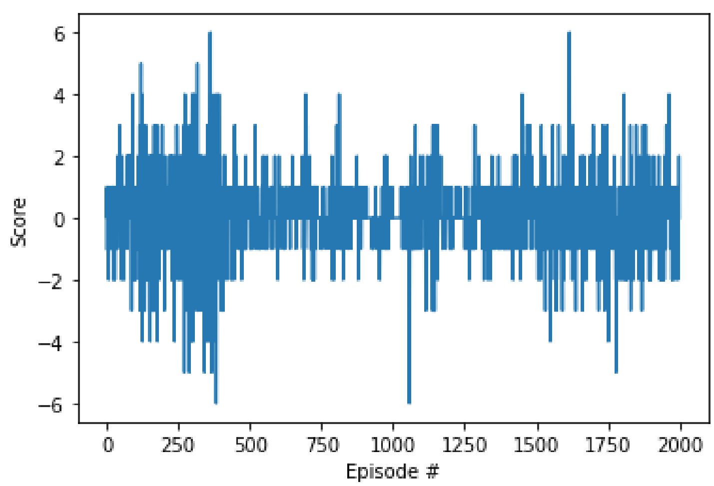
Experiment 2

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 1.0 # for soft update of target parameters
LR = 5e-3 # learning rate
UPDATE_EVERY = 16 # how often to update the network
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995

MODEL: fc1_units=256, fc2_units=256

```
Episode 100     Average Score: 0.15
Episode 200     Average Score: 0.001
Episode 300     Average Score: 0.013
Episode 400     Average Score: -0.07
Episode 500     Average Score: 0.032
Episode 600     Average Score: 0.09
Episode 700     Average Score: 0.26
Episode 800     Average Score: 0.00
Episode 900     Average Score: 0.071
Episode 1000    Average Score: -0.03
Episode 1100    Average Score: -0.09
Episode 1200    Average Score: 0.082
Episode 1300    Average Score: 0.042
Episode 1400    Average Score: 0.001
Episode 1500    Average Score: 0.341
Episode 1600    Average Score: 0.001
Episode 1700    Average Score: 0.16
Episode 1800    Average Score: -0.18
Episode 1900    Average Score: 0.032
Episode 2000    Average Score: -0.15
```

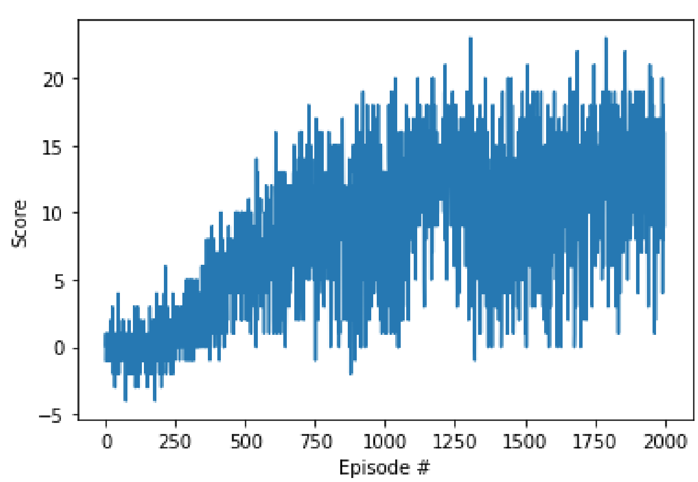
```
total train time:  1524.0444948673248
```



Experiment 3

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR = 5e-5               # learning rate
UPDATE_EVERY = 16       # how often to update the network
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 32
MODEL_LAYER2_SIZE = 32
```

```
Episode 100     Average Score: -0.08
Episode 200     Average Score: 0.031
Episode 300     Average Score: 0.922
Episode 400     Average Score: 2.77
Episode 500     Average Score: 5.33
Episode 600     Average Score: 6.85
Episode 700     Average Score: 7.74
Episode 800     Average Score: 9.87
Episode 900     Average Score: 8.92
Episode 1000    Average Score: 10.06
Episode 1100    Average Score: 9.374
Episode 1200    Average Score: 12.67
Episode 1300    Average Score: 12.85
Episode 1400    Average Score: 10.18
Episode 1500    Average Score: 10.15
Episode 1600    Average Score: 10.44
Episode 1700    Average Score: 11.54
Episode 1800    Average Score: 11.85
Episode 1900    Average Score: 12.76
Episode 2000    Average Score: 12.49

total train time:  1491.1912860870361
```

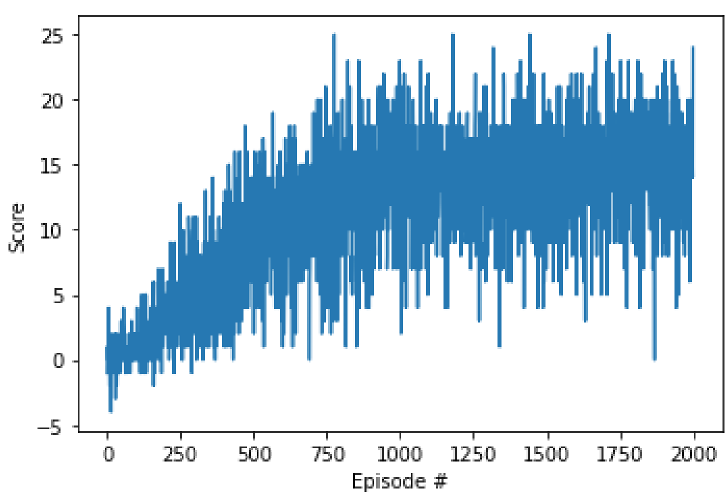


Experiment 4

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR = 5e-5               # learning rate
UPDATE_EVERY = 16       # how often to update the network
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 512
MODEL_LAYER2_SIZE = 512
```

```
Episode 100     Average Score: 0.44
Episode 200     Average Score: 2.06
Episode 300     Average Score: 4.28
Episode 400     Average Score: 5.58
Episode 500     Average Score: 8.40
Episode 600     Average Score: 9.996
Episode 700     Average Score: 10.56
Episode 800     Average Score: 11.40
Episode 900     Average Score: 12.35
Episode 1000    Average Score: 14.31
Episode 1100    Average Score: 13.46
Episode 1200    Average Score: 13.83
Episode 1300    Average Score: 13.77
Episode 1400    Average Score: 14.17
Episode 1500    Average Score: 14.12
Episode 1600    Average Score: 14.28
Episode 1700    Average Score: 14.64
Episode 1800    Average Score: 15.20
Episode 1900    Average Score: 14.37
Episode 2000    Average Score: 14.81
```

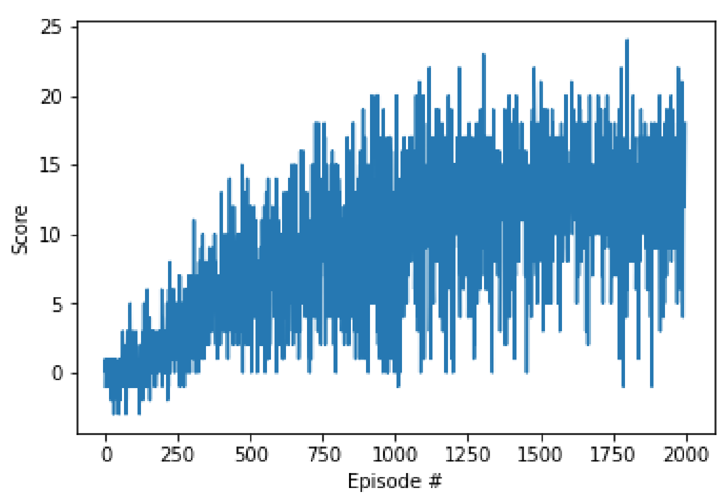
```
total train time:  1739.7480118274689
```



Experiment 5

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 1e-3 # for soft update of target parameters
LR = 5e-6 # learning rate
UPDATE_EVERY = 16 # how often to update the network
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 512
MODEL_LAYER2_SIZE = 512

```
Episode 100     Average Score: 0.07
Episode 200     Average Score: 0.89
Episode 300     Average Score: 2.76
Episode 400     Average Score: 4.75
Episode 500     Average Score: 6.86
Episode 600     Average Score: 7.10
Episode 700     Average Score: 7.86
Episode 800     Average Score: 9.18
Episode 900     Average Score: 9.17
Episode 1000    Average Score: 9.58
Episode 1100    Average Score: 11.00
Episode 1200    Average Score: 11.50
Episode 1300    Average Score: 11.76
Episode 1400    Average Score: 12.00
Episode 1500    Average Score: 13.07
Episode 1600    Average Score: 12.36
Episode 1700    Average Score: 13.31
Episode 1800    Average Score: 12.52
Episode 1900    Average Score: 12.99
Episode 2000    Average Score: 13.29


total train time:  1710.029676914215
```

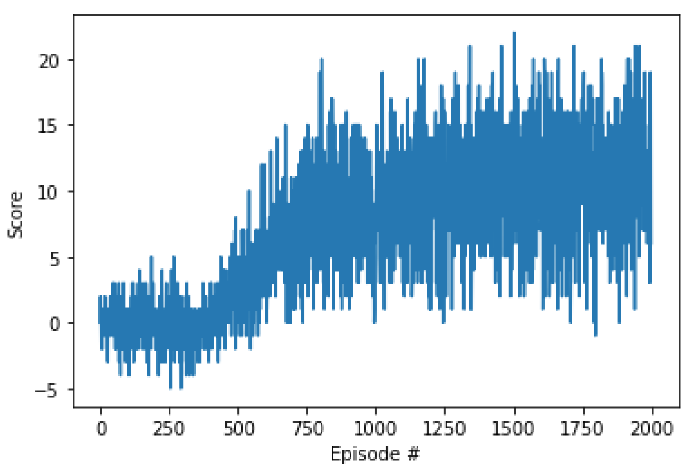


Experiment 6

```
BUFFER_SIZE = int(1e5)          # replay buffer size
BATCH_SIZE = 64                 # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 1.0                       # for soft update of target parameters
LR = 5e-5                       # learning rate
LEARN_EVERY = 4                 # how often to learn and update weights
UPDATE_TARGET_EVERY = 10        # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 256
MODEL_LAYER2_SIZE = 256
```

```
Episode 100     Average Score: 0.06
Episode 200     Average Score: 0.141
Episode 300     Average Score: -0.06
Episode 400     Average Score: -0.35
Episode 500     Average Score: 1.641
Episode 600     Average Score: 3.72
Episode 700     Average Score: 5.62
Episode 800     Average Score: 7.80
Episode 900     Average Score: 8.55
Episode 1000    Average Score: 8.27
Episode 1100    Average Score: 9.09
Episode 1200    Average Score: 9.40
Episode 1300    Average Score: 9.62
Episode 1400    Average Score: 10.15
Episode 1500    Average Score: 10.92
Episode 1600    Average Score: 11.07
Episode 1700    Average Score: 10.24
Episode 1800    Average Score: 10.92
Episode 1900    Average Score: 11.30
Episode 2000    Average Score: 12.33
```

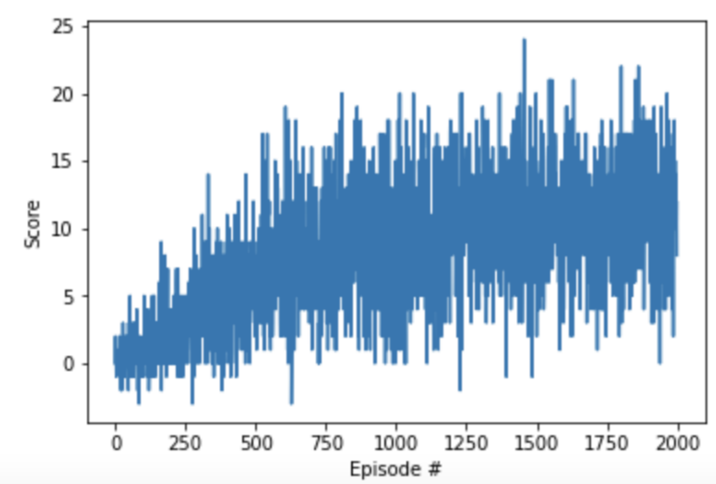
```
total train time:  2057.204227924347
```



Experiment 7

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 1.0 # for soft update of target parameters
LR = 5e-4 # learning rate
LEARN_EVERY = 4 # how often to learn and update weights
UPDATE_TARGET_EVERY = 40 # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 256
MODEL_LAYER2_SIZE = 256

```
Episode 100     Average Score: 0.51
Episode 200     Average Score: 1.68
Episode 300     Average Score: 2.48
Episode 400     Average Score: 4.18
Episode 500     Average Score: 5.14
Episode 600     Average Score: 7.09
Episode 700     Average Score: 8.07
Episode 800     Average Score: 7.75
Episode 900     Average Score: 9.42
Episode 1000    Average Score: 8.61
Episode 1100    Average Score: 9.42
Episode 1200    Average Score: 9.53
Episode 1300    Average Score: 10.80
Episode 1400    Average Score: 10.38
Episode 1500    Average Score: 11.07
Episode 1600    Average Score: 10.89
Episode 1700    Average Score: 11.90
Episode 1800    Average Score: 10.33
Episode 1900    Average Score: 12.11
Episode 2000    Average Score: 10.48


total train time:  2036.573081254959
```



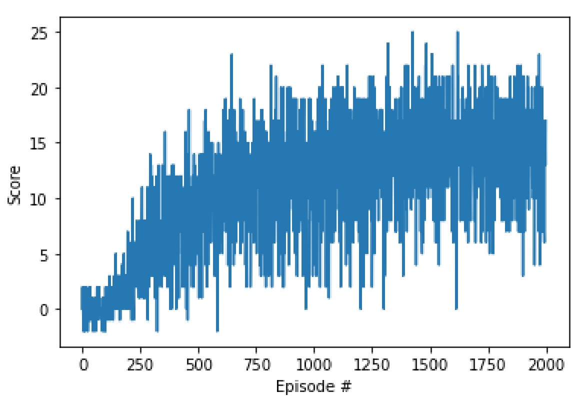
Experiment 8

BUFFER_SIZE = int(1e5)	# replay buffer size
BATCH_SIZE = 64	# minibatch size
GAMMA = 0.99	# discount factor
TAU = 1.0	# for soft update of target parameters
LR = 5e-5	# learning rate
LEARN_EVERY = 4	# how often to learn and update weights
UPDATE_TARGET_EVERY = 40	# how often to update target network w soft update
EPS_START = 1.0	
EPS_END = 0.01	
EPS_DECAY = 0.995	
MODEL_LAYER1_SIZE = 256	
MODEL_LAYER2_SIZE = 256	

Episode 100	Average Score: 0.11	Score Variance: 1.16
Episode 200	Average Score: 1.35	Score Variance: 2.43
Episode 300	Average Score: 5.00	Score Variance: 10.18
Episode 400	Average Score: 6.42	Score Variance: 11.96
Episode 500	Average Score: 7.91	Score Variance: 12.28
Episode 600	Average Score: 9.92	Score Variance: 12.45
Episode 700	Average Score: 11.59	Score Variance: 16.44
Episode 800	Average Score: 11.02	Score Variance: 16.78
Episode 900	Average Score: 11.99	Score Variance: 18.83
Episode 1000	Average Score: 11.88	Score Variance: 18.87
Episode 1100	Average Score: 12.88	Score Variance: 21.87
Episode 1103	Average Score: 13.03	
Environment solved in 1103 episodes!		Average Score: 13.03
Episode 1200	Average Score: 13.61	Score Variance: 20.56
Episode 1300	Average Score: 13.13	Score Variance: 20.17
Episode 1400	Average Score: 13.48	Score Variance: 17.35
Episode 1500	Average Score: 14.32	Score Variance: 17.44
Episode 1600	Average Score: 14.50	Score Variance: 15.01
Episode 1700	Average Score: 13.93	Score Variance: 17.51
Episode 1800	Average Score: 14.39	Score Variance: 15.22
Episode 1900	Average Score: 14.62	Score Variance: 14.52
Episode 2000	Average Score: 14.07	Score Variance: 17.55

Final Model had Average Score: 14.07. Saving model to: final.pth

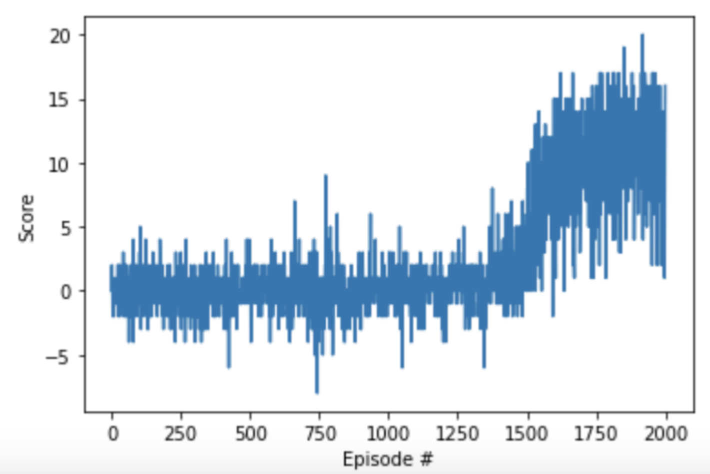
total train time: 2011.3368470668793



Experiment 9

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 1.0 # for soft update of target parameters
LR = 5e-6 # learning rate
LEARN_EVERY = 4 # how often to learn and update weights
UPDATE_TARGET_EVERY = 40 # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 256
MODEL_LAYER2_SIZE = 256

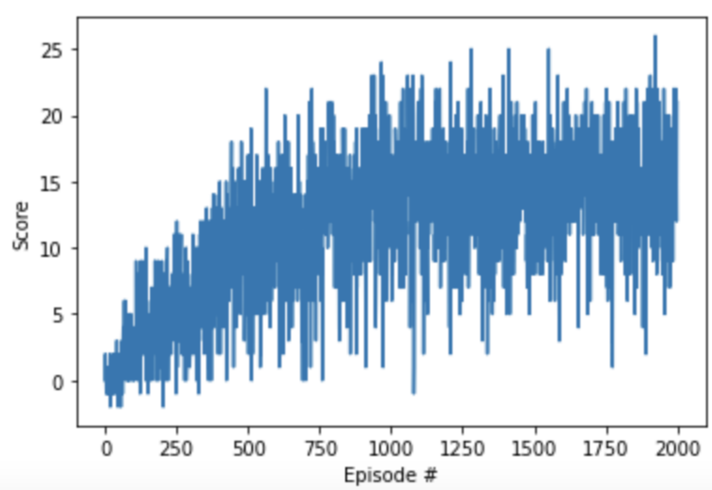
```
Episode 100     Average Score: -0.11
Episode 200     Average Score: 0.081
Episode 300     Average Score: 0.051
Episode 400     Average Score: -0.25
Episode 500     Average Score: -0.03
Episode 600     Average Score: 0.121
Episode 700     Average Score: -0.09
Episode 800     Average Score: 0.032
Episode 900     Average Score: -0.18
Episode 1000    Average Score: 0.07
Episode 1100    Average Score: 0.21
Episode 1200    Average Score: 0.10
Episode 1300    Average Score: 0.38
Episode 1400    Average Score: 0.451
Episode 1500    Average Score: 1.69
Episode 1600    Average Score: 6.62
Episode 1700    Average Score: 9.51
Episode 1800    Average Score: 11.00
Episode 1900    Average Score: 11.54
Episode 2000    Average Score: 11.29


total train time:  1963.8735659122467
```



Experiment 10

```
BUFFER_SIZE = int(1e5)              # replay buffer size
BATCH_SIZE = 64                     # minibatch size
GAMMA = 0.99                        # discount factor
TAU = 1.0                           # for soft update of target parameters
LR = 5e-5                           # learning rate
LEARN_EVERY = 4                     # how often to learn and update weights
UPDATE_TARGET_EVERY = 100           # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 256
MODEL_LAYER2_SIZE = 256
```

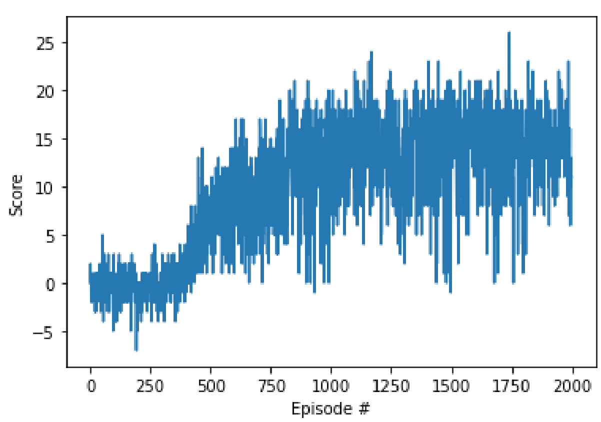
```
Episode 100     Average Score: 1.03
Episode 200     Average Score: 3.84
Episode 300     Average Score: 5.18
Episode 400     Average Score: 6.90
Episode 500     Average Score: 9.76
Episode 600     Average Score: 10.40
Episode 700     Average Score: 10.58
Episode 800     Average Score: 13.04
Episode 900     Average Score: 12.22
Episode 1000    Average Score: 13.88
Episode 1100    Average Score: 13.98
Episode 1200    Average Score: 13.88
Episode 1300    Average Score: 13.60
Episode 1400    Average Score: 13.60
Episode 1500    Average Score: 13.47
Episode 1600    Average Score: 13.62
Episode 1700    Average Score: 15.38
Episode 1800    Average Score: 14.77
Episode 1900    Average Score: 14.40
Episode 2000    Average Score: 15.09


total train time:  1898.848296880722
```



Experiment 11

BUFFER_SIZE = int(1e5)   # replay buffer size
BATCH_SIZE = 64   # minibatch size
GAMMA = 0.99   # discount factor
TAU = 1.0   # for soft update of target parameters
LR = 5e-5   # learning rate
LEARN_EVERY = 4   # how often to learn and update weights
UPDATE_TARGET_EVERY = 40   # how often to update target network w soft update
EPS_START = 1.0
EPS_END = 0.01
EPS_DECAY = 0.995
MODEL_LAYER1_SIZE = 32
MODEL_LAYER2_SIZE = 32

Episode 100   Average Score: -0.11   Score Variance: 2.34
Episode 200   Average Score: -0.43   Score Variance: 3.05
Episode 300   Average Score: -0.32   Score Variance: 2.02
Episode 400   Average Score: 0.40   Score Variance: 2.56
Episode 500   Average Score: 4.69   Score Variance: 10.31
Episode 600   Average Score: 7.20   Score Variance: 10.64
Episode 700   Average Score: 8.97   Score Variance: 12.71
Episode 800   Average Score: 9.54   Score Variance: 15.87
Episode 900   Average Score: 12.19   Score Variance: 17.33
Episode 1000   Average Score: 11.91   Score Variance: 21.20
Episode 1073   Average Score: 13.03
Environment solved in 1073 episodes!   Average Score: 13.03
Episode 1100   Average Score: 13.50   Score Variance: 13.01
Episode 1200   Average Score: 14.32   Score Variance: 12.52
Episode 1300   Average Score: 12.85   Score Variance: 16.07
Episode 1400   Average Score: 13.17   Score Variance: 18.52
Episode 1500   Average Score: 13.27   Score Variance: 25.28
Episode 1600   Average Score: 14.17   Score Variance: 15.36
Episode 1700   Average Score: 14.23   Score Variance: 17.52
Episode 1800   Average Score: 13.87   Score Variance: 16.93
Episode 1900   Average Score: 14.02   Score Variance: 12.26
Episode 2000   Average Score: 13.85   Score Variance: 12.77
Final Model had Average Score: 13.85. Saving model to: final.pth
total train time: 1812.9740772247314

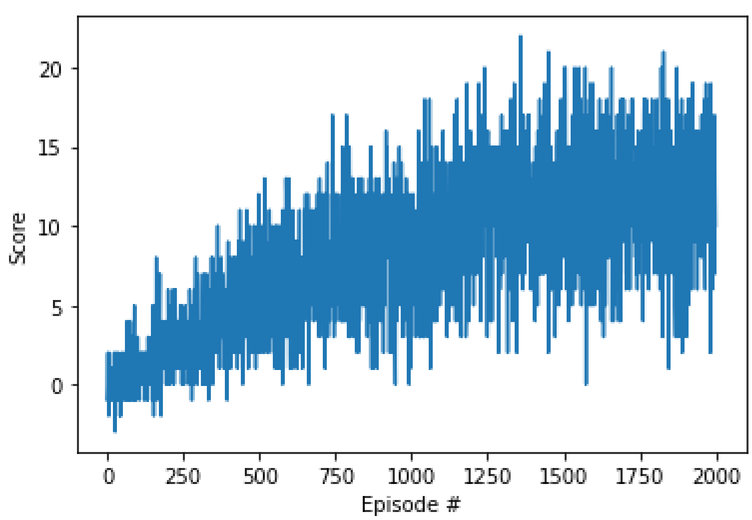


Experiment 12

BUFFER_SIZE = int(1e5)	# replay buffer size
BATCH_SIZE = 64	# minibatch size
GAMMA = 0.99	# discount factor
TAU = 1e-3	# for soft update of target parameters
LR = 5e-5	# learning rate
LEARN_EVERY = 1	# how often to learn and update weights
UPDATE_TARGET_EVERY = 1	# how often to update target network w soft update
EPS_START = 1.0	
EPS_END = 0.01	
EPS_DECAY = 0.998	
MODEL_LAYER1_SIZE = 256	
MODEL_LAYER2_SIZE = 256	

Episode 100	Average Score: 0.44	Score Variance: 1.81
Episode 200	Average Score: 1.34	Score Variance: 2.46
Episode 300	Average Score: 2.87	Score Variance: 3.85
Episode 400	Average Score: 3.73	Score Variance: 5.84
Episode 500	Average Score: 4.98	Score Variance: 6.94
Episode 600	Average Score: 6.01	Score Variance: 9.97
Episode 700	Average Score: 6.76	Score Variance: 8.42
Episode 800	Average Score: 7.88	Score Variance: 10.09
Episode 900	Average Score: 7.94	Score Variance: 10.30
Episode 1000	Average Score: 8.63	Score Variance: 11.39
Episode 1100	Average Score: 8.82	Score Variance: 13.63
Episode 1200	Average Score: 10.09	Score Variance: 14.14
Episode 1300	Average Score: 11.23	Score Variance: 11.56
Episode 1400	Average Score: 11.53	Score Variance: 13.17
Episode 1500	Average Score: 11.08	Score Variance: 12.65
Episode 1600	Average Score: 12.27	Score Variance: 17.86
Episode 1700	Average Score: 11.38	Score Variance: 15.72
Episode 1800	Average Score: 12.16	Score Variance: 10.27
Episode 1900	Average Score: 11.43	Score Variance: 18.93
Episode 2000	Average Score: 12.39	Score Variance: 14.90

Final Model had Average Score: 12.39. Saving model to: final.pth
total train time: 3890.78329205513



Experiment 13

BUFFER_SIZE = int(1e5)	# replay buffer size
BATCH_SIZE = 64	# minibatch size
GAMMA = 0.99	# discount factor
TAU = 1e-3	# for soft update of target parameters
LR = 5e-5	# learning rate
LEARN_EVERY = 16	# how often to learn and update weights
UPDATE_TARGET_EVERY = 1	# how often to update target network w soft update
EPS_START = 1.0	
EPS_END = 0.01	
EPS_DECAY = 0.995	
MODEL_LAYER1_SIZE = 256	
MODEL_LAYER2_SIZE = 256	

Episode 100	Average Score: 0.34	Score Variance: 1.76
Episode 200	Average Score: 2.65	Score Variance: 4.19
Episode 300	Average Score: 5.09	Score Variance: 8.72
Episode 400	Average Score: 6.29	Score Variance: 12.77
Episode 500	Average Score: 8.52	Score Variance: 14.51
Episode 600	Average Score: 10.11	Score Variance: 17.88
Episode 700	Average Score: 9.84	Score Variance: 21.31
Episode 800	Average Score: 10.89	Score Variance: 18.62
Episode 888	Average Score: 13.01	
Environment solved in 888 episodes!		Average Score: 13.01
Episode 900	Average Score: 12.98	Score Variance: 13.52
Episode 1000	Average Score: 13.84	Score Variance: 15.75
Episode 1100	Average Score: 14.76	Score Variance: 9.12
Episode 1200	Average Score: 14.62	Score Variance: 12.48
Episode 1300	Average Score: 14.34	Score Variance: 16.44
Episode 1400	Average Score: 14.84	Score Variance: 11.21
Episode 1500	Average Score: 14.68	Score Variance: 13.64
Episode 1600	Average Score: 14.72	Score Variance: 16.78
Episode 1700	Average Score: 14.58	Score Variance: 15.20
Episode 1800	Average Score: 13.86	Score Variance: 13.18
Episode 1900	Average Score: 14.83	Score Variance: 16.10
Episode 2000	Average Score: 14.88	Score Variance: 13.13

Final Model had Average Score: 14.88. Saving model to: final.pth