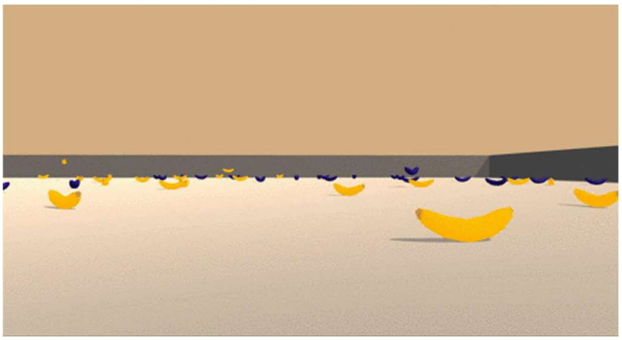
Project 1 – Navigation

Introduction:

The purpose of this project is to train an agent to play in a plain squared Unity 3D environment to collect yellow bananas and avoid blue bananas. A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana.

The state space has 37 dimensions given by Unity, which contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select between the available action:

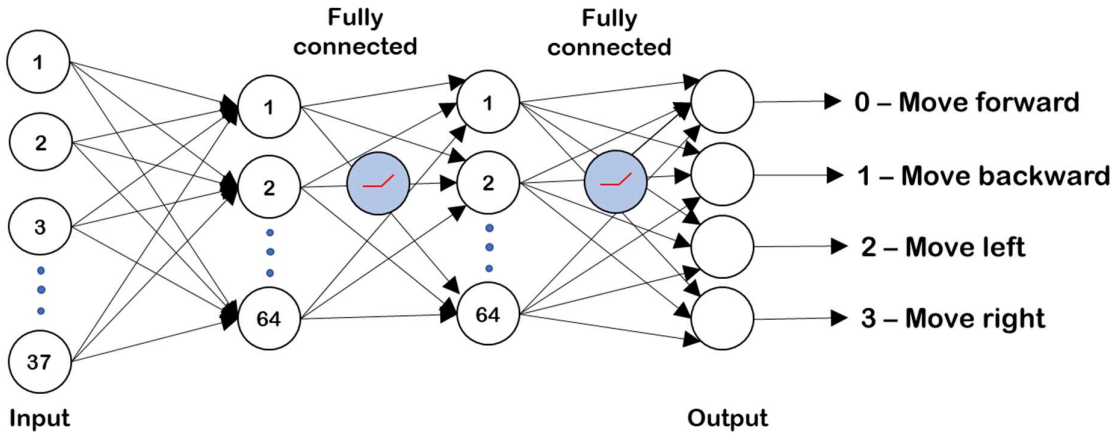


0 - move forward.
1 - move backward.
2 - turn left.
3 - turn right.

The task is episodic, and in order to solve the environment, the agent must get an average score of +13 over 100 consecutive episodes.

Learning Algorithm:

The algorithm used in my implementation is a Deep Q-Learning with experience replay, the architecture of the neural network used is:



Thanks to the implementation of **experience replay**, the algorithm stores states which are rare and actions that are costly to be able to recall them, each experience (Which is formed by State, Action, Reward and Next State) is stored in a buffer as the agent is interacting with the environment, then sample a small batch of these experiences in order to learn from them. Thanks to this, it learns from individual state-actions multiple times, recall rare occurrences and make a better use of the experience obtained.

The buffer of experiences is sample randomly, to this helps breaking the correlation and prevent actions from oscillating or diverging in wrong ways.

The algorithm includes the implementation of **Fixed Q-targets,** The idea behind fixed q-targets is to fix the parameter w used in the calculation of the target. This is achieved by having two separate networks, one

is the "local" network being learned and the other being the target network. The weights of the target network are taken from the "local" network itself by freezing the model parameters for a few iterations and updating it periodically after a few steps. Therefore, it ensures that the target network parameters are significantly different from the "local" network parameters.

Another improvement implemented in the learning algorithm is **Double DQN**, in double DQN we have two copies of the Neural network model, one learns during the experience, and the other one is a copy of the last episode, and is used to compute the Q-value. This prevents the algorithm from propagating high rewards that may be obtained by chance.

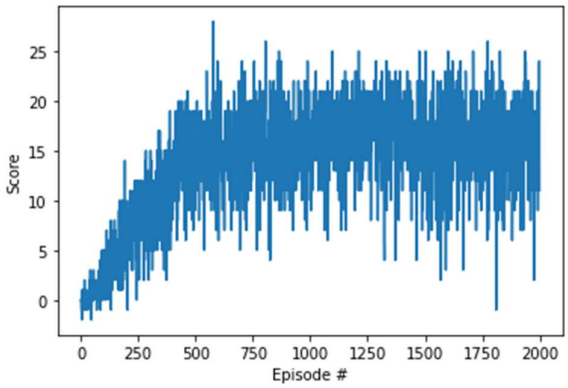
Algorithm performance:

The algorithm have been trained for 2000 episodes with different hyperparameters setups, the first setup used have been the following:

Replay buffer size (BUFFER_SIZE)	$1*e^{5}$
Minibatch size (BATCH_SIZE)	64
Discount Factor (GAMMA)	0.99
Soft update of target parameters (TAU)	$1*e^{-3}$
Learning Rate (LR)	$5*e^{-4}$
Network update episodes (UPDATE_EVERY)	4

The algorithm have been trained for 2000 episodes, getting the following performances

```
Episode 100     Average Score: 0.75
Episode 200     Average Score: 4.06
Episode 300     Average Score: 7.40
Episode 400     Average Score: 10.17
Episode 500     Average Score: 14.02
Episode 600     Average Score: 14.71
Episode 700     Average Score: 14.42
Episode 800     Average Score: 15.50
Episode 900     Average Score: 15.48
Episode 1000    Average Score: 15.77
Episode 1100    Average Score: 16.31
Episode 1200    Average Score: 16.77
Episode 1300    Average Score: 17.19
Episode 1400    Average Score: 16.66
Episode 1500    Average Score: 15.87
Episode 1600    Average Score: 15.31
Episode 1700    Average Score: 16.15
Episode 1800    Average Score: 15.97
Episode 1900    Average Score: 15.43
Episode 2000    Average Score: 15.80
```

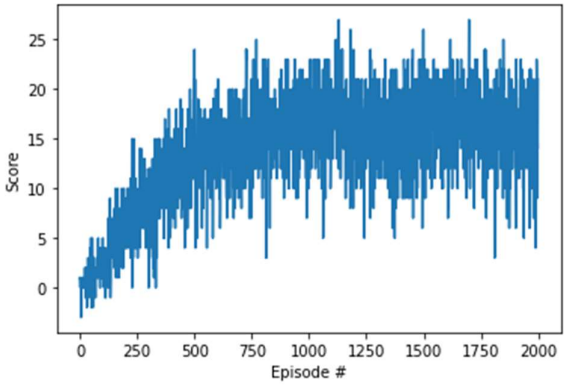


A change in the parameters have been done to test if the performance might be improved to the following ones, reducing the batch size and changing the LR to mach the hyperparamters used in "Human-level control through DRL" paper:

Replay buffer size (BUFFER_SIZE)	$1*e^{5}$
Minibatch size (BATCH_SIZE)	32
Discount Factor (GAMMA)	0.99
Soft update of target parameters (TAU)	$1*e^{-3}$
Learning Rate (LR)	0.00025
Network update episodes (UPDATE_EVERY)	4

No significant improvement have been done to the performance, therefore the isolation of both hyperparamter chances was not done to check what change created the improvement.

```
Episode 100     Average Score: 1.12
Episode 200     Average Score: 4.06
Episode 300     Average Score: 7.69
Episode 400     Average Score: 9.48
Episode 500     Average Score: 12.14
Episode 600     Average Score: 13.80
Episode 700     Average Score: 14.32
Episode 800     Average Score: 15.64
Episode 900     Average Score: 16.14
Episode 1000    Average Score: 15.79
Episode 1100    Average Score: 16.28
Episode 1200    Average Score: 16.91
Episode 1300    Average Score: 15.74
Episode 1400    Average Score: 16.16
Episode 1500    Average Score: 16.30
Episode 1600    Average Score: 17.17
Episode 1700    Average Score: 16.93
Episode 1800    Average Score: 16.45
Episode 1900    Average Score: 15.91
Episode 2000    Average Score: 15.79
```

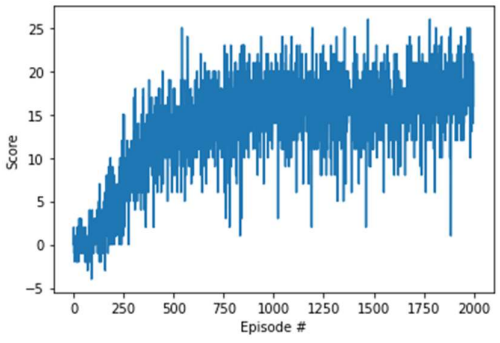


Because this change in the hyperparameters didn't show any performance imrpovement, we get back to the "original" hyperparamters.

In all these previous performance figures we can see that an **overfitting** happens around episode 1.600, and the performance goes slightly down.

The following change done to try to improve the performance was to do a Double DQN network, the main improvement shown is that the **overfitting** that happened with the simple DQN did not happen with the Double DQN for those 2000 episodes, the algorithm performance kept improving

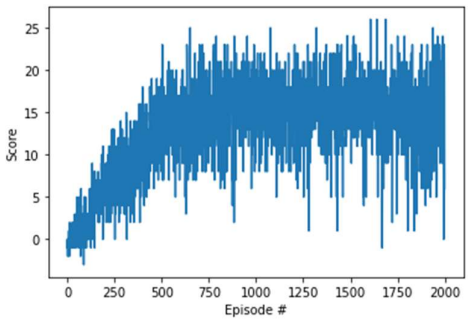
```
Episode 100     Average Score: 0.11     MinScore: -4.00         MaxScore: 4.00
Episode 200     Average Score: 2.38     MinScore: -3.00         MaxScore: 10.00
Episode 300     Average Score: 6.34     MinScore: 0.00  MaxScore: 15.00
Episode 400     Average Score: 10.63    MinScore: 0.00  MaxScore: 19.00
Episode 500     Average Score: 12.63    MinScore: 0.00  MaxScore: 20.00
Episode 600     Average Score: 13.48    MinScore: 0.00  MaxScore: 25.00
Episode 700     Average Score: 14.80    MinScore: 0.00  MaxScore: 22.00
Episode 800     Average Score: 14.67    MinScore: 0.00  MaxScore: 23.00
Episode 900     Average Score: 15.97    MinScore: 0.00  MaxScore: 23.00
Episode 1000    Average Score: 16.48    MinScore: 0.00  MaxScore: 24.00
Episode 1100    Average Score: 16.77    MinScore: 0.00  MaxScore: 24.00
Episode 1200    Average Score: 16.43    MinScore: 0.00  MaxScore: 25.00
Episode 1300    Average Score: 16.31    MinScore: 0.00  MaxScore: 23.00
Episode 1400    Average Score: 16.18    MinScore: 0.00  MaxScore: 25.00
Episode 1500    Average Score: 15.93    MinScore: 0.00  MaxScore: 26.00
Episode 1600    Average Score: 16.35    MinScore: 0.00  MaxScore: 23.00
Episode 1700    Average Score: 16.69    MinScore: 0.00  MaxScore: 24.00
Episode 1800    Average Score: 17.18    MinScore: 0.00  MaxScore: 26.00
Episode 1900    Average Score: 17.64    MinScore: 0.00  MaxScore: 25.00
Episode 2000    Average Score: 17.65    MinScore: 0.00  MaxScore: 25.00
```



It's been tested with a smaller Neural Network (32 instead of 64 nodes in both hidden layers) in the model to check we saw a performance improvement, but it didn't show any additional improvement in the performance in the simple DQN nor in the Double DQN. Same test with 32 nodes and "changed" parameters was done without significant improvement in the performance.

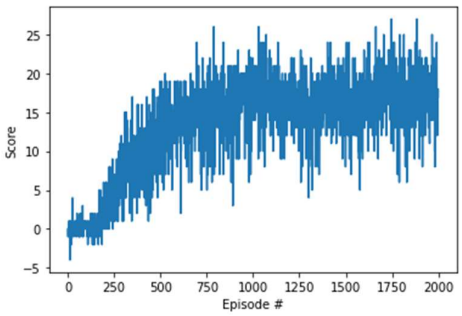
The next improvement search is through the optimization of the learning rate while the training. This have been done through the pytorch LR optimizer (*torch.optim.lr_scheduler.StepLR*), previously in the Simple DQN:

```
Episode 100     Average Score: 0.76     MinScore: -3.00     MaxScore: 6.00
Episode 200     Average Score: 4.03     MinScore: -1.00     MaxScore: 11.00
Episode 300     Average Score: 7.40     MinScore: 0.00      MaxScore: 14.00
Episode 400     Average Score: 8.89     MinScore: 0.00      MaxScore: 18.00
Episode 500     Average Score: 12.27    MinScore: 0.00      MaxScore: 19.00
Episode 600     Average Score: 13.86    MinScore: 0.00      MaxScore: 23.00
Episode 700     Average Score: 14.10    MinScore: 0.00      MaxScore: 25.00
Episode 800     Average Score: 15.16    MinScore: 0.00      MaxScore: 24.00
Episode 900     Average Score: 14.89    MinScore: 0.00      MaxScore: 24.00
Episode 1000    Average Score: 16.15    MinScore: 0.00      MaxScore: 24.00
Episode 1100    Average Score: 15.84    MinScore: 0.00      MaxScore: 25.00
Episode 1200    Average Score: 15.53    MinScore: 0.00      MaxScore: 22.00
Episode 1300    Average Score: 15.46    MinScore: 0.00      MaxScore: 23.00
Episode 1400    Average Score: 16.33    MinScore: 0.00      MaxScore: 25.00
Episode 1500    Average Score: 16.07    MinScore: 0.00      MaxScore: 24.00
Episode 1600    Average Score: 16.18    MinScore: 0.00      MaxScore: 24.00
Episode 1700    Average Score: 15.74    MinScore: -1.00     MaxScore: 26.00
Episode 1800    Average Score: 15.09    MinScore: 0.00      MaxScore: 24.00
Episode 1900    Average Score: 14.94    MinScore: 0.00      MaxScore: 23.00
Episode 2000    Average Score: 15.38    MinScore: 0.00      MaxScore: 25.00
```



And later in a Double DQN, but none shown a significant increase in the performance.

```
Episode 100     Average Score: 0.16     MinScore: -4.00 MaxScore: 4.00
Episode 200     Average Score: 0.71     MinScore: -2.00 MaxScore: 6.00
Episode 300     Average Score: 4.46     MinScore: 0.00  MaxScore: 12.00
Episode 400     Average Score: 8.96     MinScore: 0.00  MaxScore: 17.00
Episode 500     Average Score: 10.02    MinScore: 0.00  MaxScore: 19.00
Episode 600     Average Score: 13.03    MinScore: 0.00  MaxScore: 20.00
Episode 700     Average Score: 14.39    MinScore: 0.00  MaxScore: 24.00
Episode 800     Average Score: 14.92    MinScore: 0.00  MaxScore: 26.00
Episode 900     Average Score: 15.06    MinScore: 0.00  MaxScore: 24.00
Episode 1000    Average Score: 16.33    MinScore: 0.00  MaxScore: 23.00
Episode 1100    Average Score: 17.19    MinScore: 0.00  MaxScore: 26.00
Episode 1200    Average Score: 16.62    MinScore: 0.00  MaxScore: 24.00
Episode 1300    Average Score: 16.45    MinScore: 0.00  MaxScore: 23.00
Episode 1400    Average Score: 15.37    MinScore: 0.00  MaxScore: 22.00
Episode 1500    Average Score: 16.63    MinScore: 0.00  MaxScore: 24.00
Episode 1600    Average Score: 16.23    MinScore: 0.00  MaxScore: 24.00
Episode 1700    Average Score: 16.99    MinScore: 0.00  MaxScore: 26.00
Episode 1800    Average Score: 17.59    MinScore: 0.00  MaxScore: 27.00
Episode 1900    Average Score: 17.03    MinScore: 0.00  MaxScore: 27.00
Episode 2000    Average Score: 17.12    MinScore: 0.00  MaxScore: 25.00
```



Future improvements:

Future improvements for this process may be:

- Use prioritized experience replay to break the correlation between consecutive experiences and stabilize our learning algorithm.
- Implementation of Dueling DQN
- Continue testing the tunning of the hyperparameters to search for a better performance.