The Training Algorithm:

In order to help the agent solve the environment problem, this program implements a training algorithm that uses a Deep Q-Network that helps the agent takes the best action given a state. The learning process here is supported by an Experience Replay buffer.

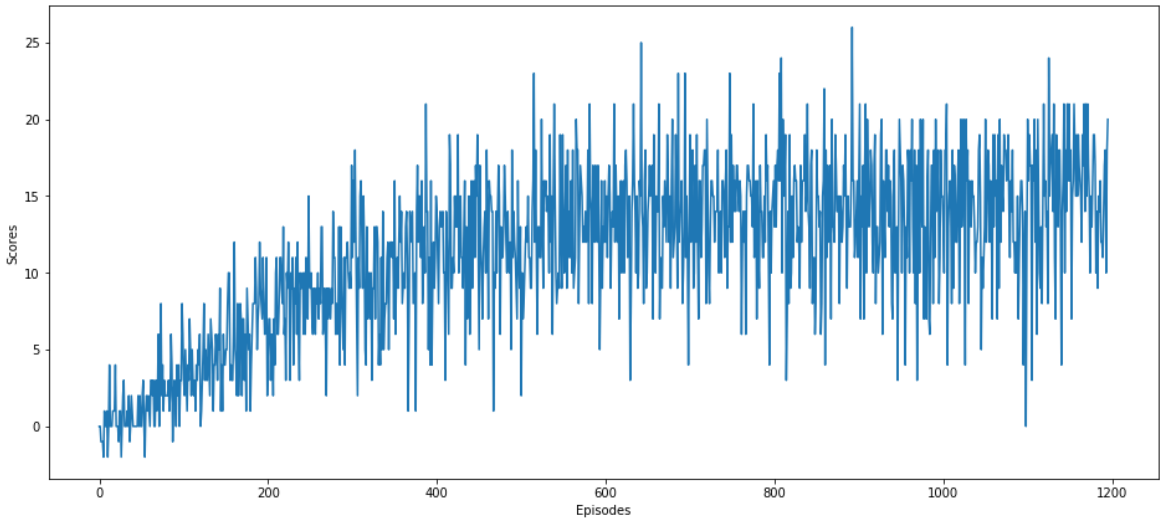
Experience Replay:
Experience replay allows us to learn more from individual tuples multiple times, recall rare occurrences, and in general make better use of our experience. This approach can help us learn a more robust policy, one that is not effected by the inherent correlation present in the sequence of observed experience tuples. The replay buffer contains a collection of experience tuples (S, A, R, S'). The tuples are gradually added to the buffer as we are interacting with the environment. The sequences take randomly break the correlation and prevents action values from oscillating or diverging catastrophically. This approach is basically building a database of samples and then learning a mapping from them. It allows Reinforcement learning apply Supervised Learning techniques in order to improve learning.

The PROGRAM HYPERPARAMS:

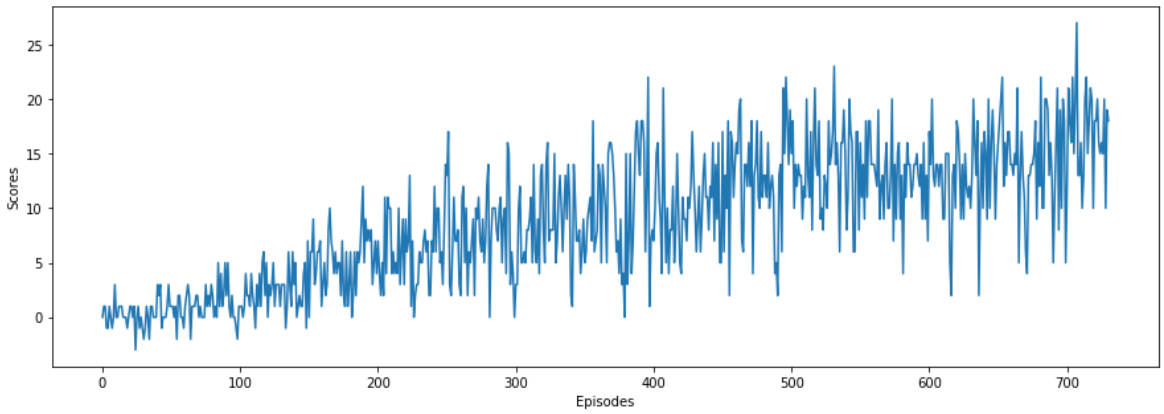
```
EPSILON
eps_start=1.0,    # start epsilon value
eps_end=0.02,     # min value for epsilon
eps_decay=0.995, # epsilon decay

BUFFER_SIZE = int(5e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size: sample 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

SOLVED = 15 # The agent target average to solve the training

Both networks used the same hyperparams
```

Training agent: Score evolution

Network model with 2 hidden layers size 64 units each:



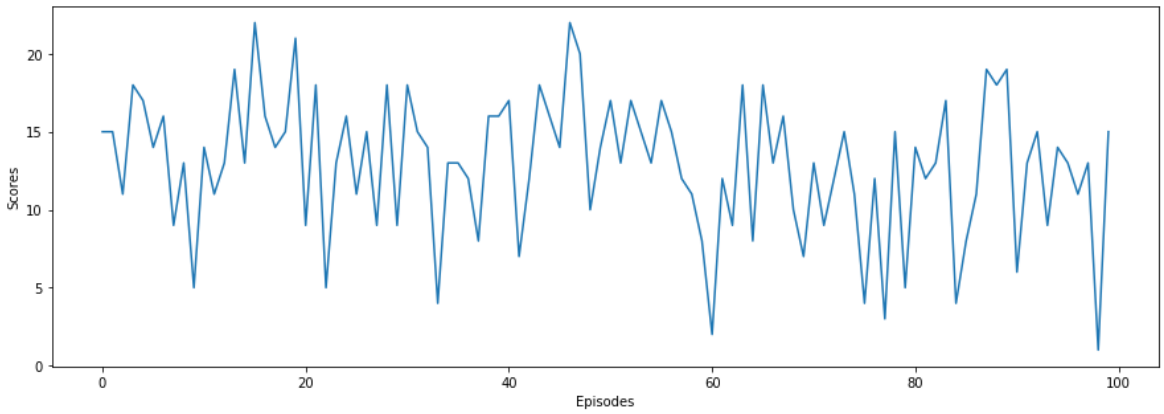
Network model with 3 hidden layers size 64 units each:



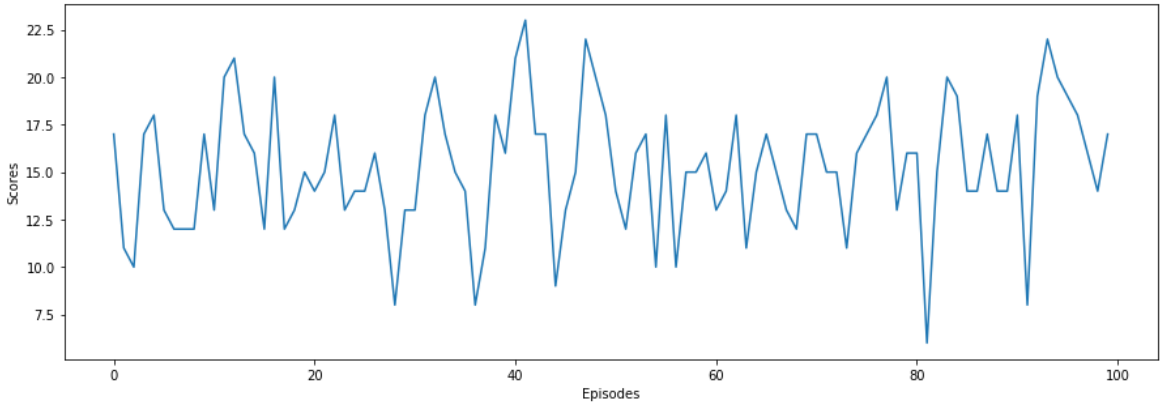
One more layer allowed the program solve the problem over less episodes.

Running trained agent: Scores evolution

Network model with 2 hidden layers size 64 units each:



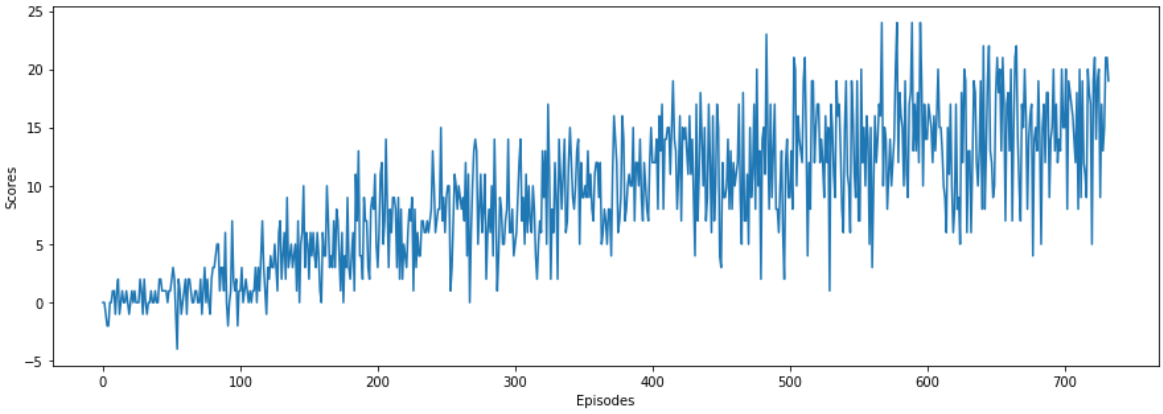
Network model with 3 hidden layers size 64 units each:



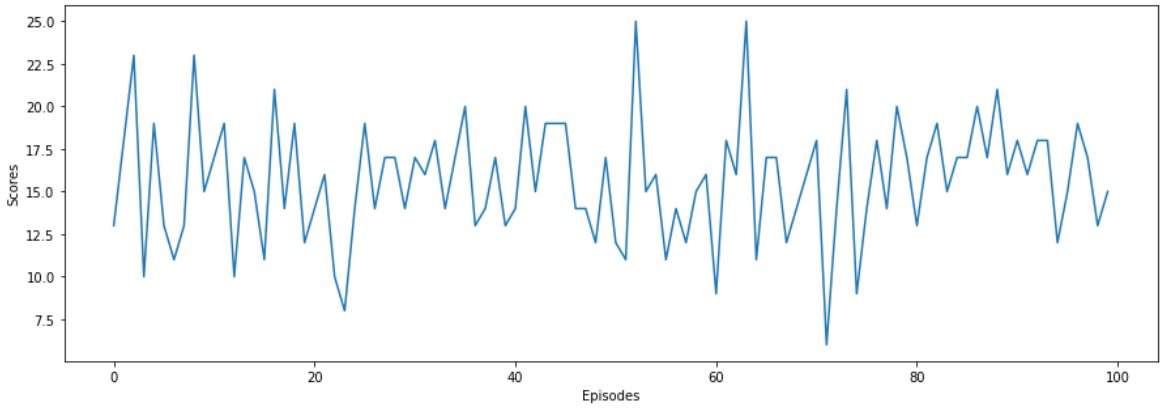
One more layer worked better during real interaction. The agent reached a higher average score and oscillated within a better range of rewards.

Testing Dueling DQN to solve the Unity Banana Collector

Training:



After training:



To the future:

The next step of this project will be the Learning from Pixels implementation. Use CNN to try solve the same banana environment problem. Whether I complete the mission, I will post the results and the source codes here.