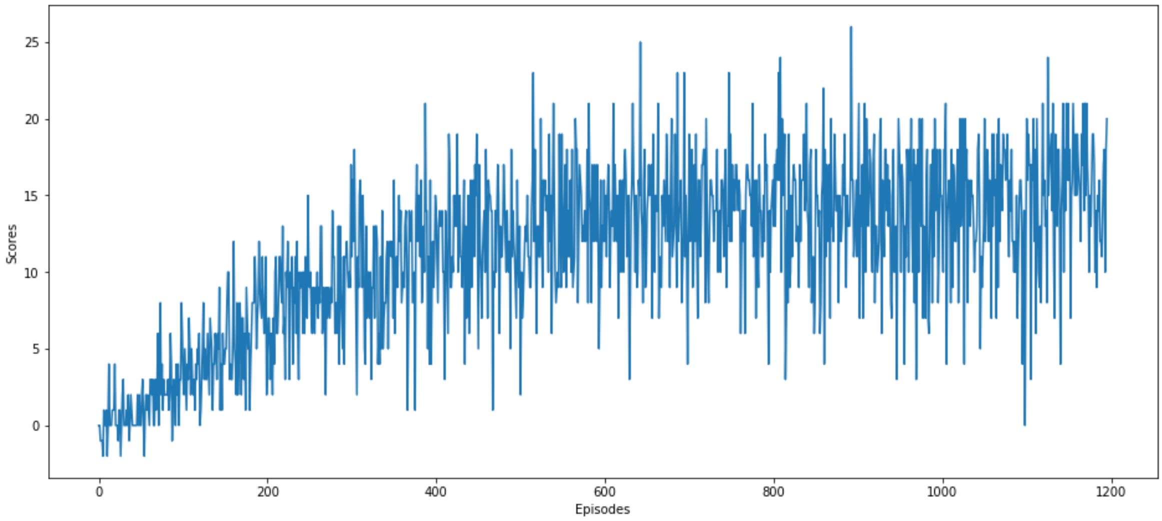
```
PROGRAM HYPERPARAMS:

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
```

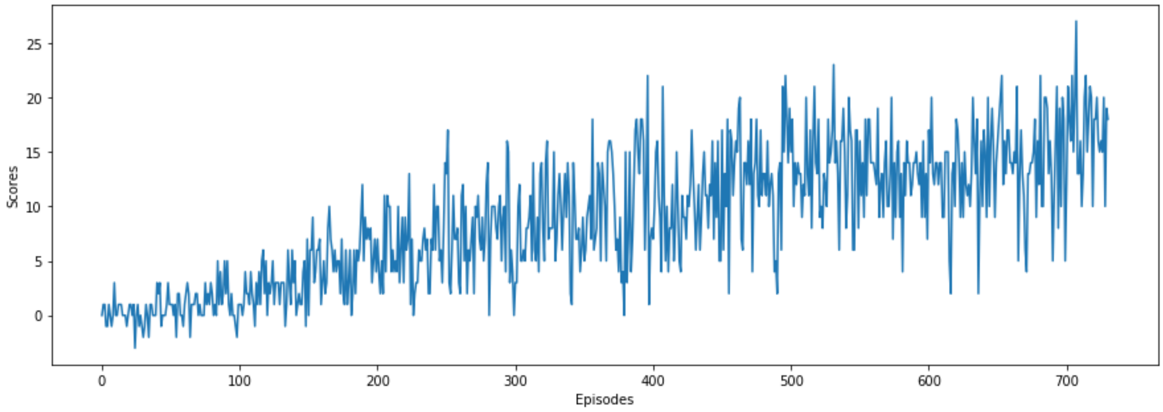
Both networks used the same hyperparams

Training agent: Score evolution

Network model with 2 hidden layers size 64 units each:



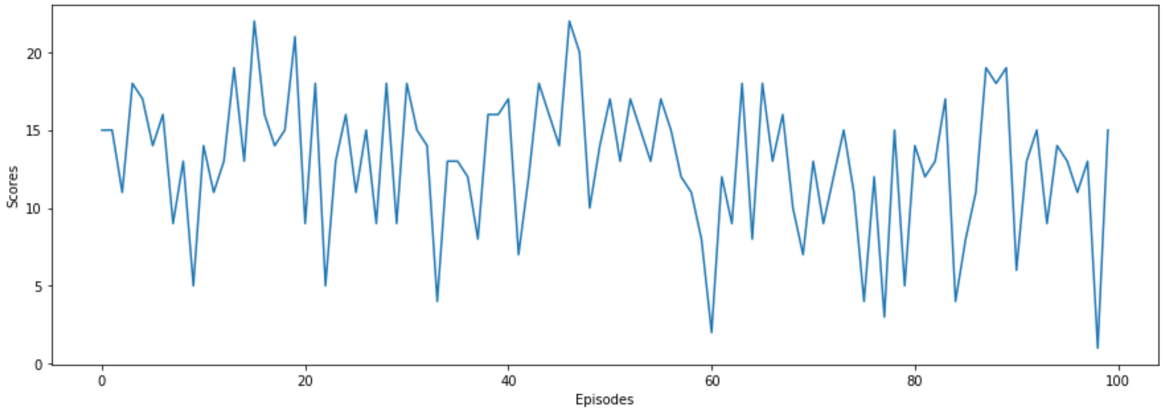
Network model with 3 hidden layers size 64 units each:



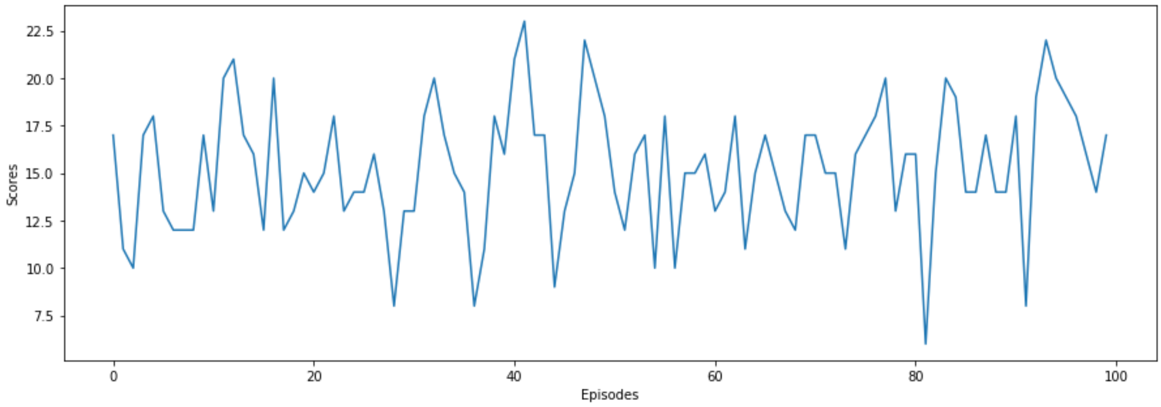
One more layer allowed the program solve the problem over less episodes.

Running trained agent: Scores evolution

Network model with 2 hidden layers size 64 units each:



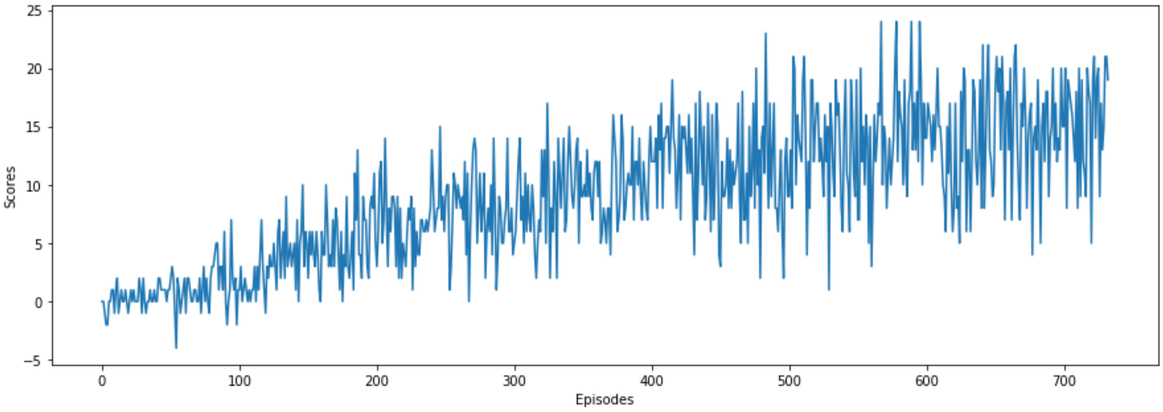
Network model with 3 hidden layers size 64 units each:



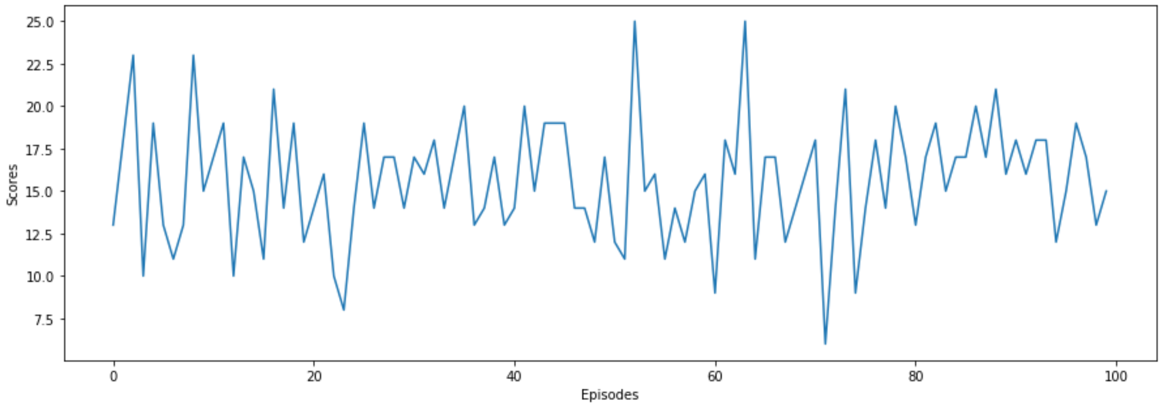
One more layer worked better during real interaction. The agent reached a higher average score and oscillated within a better range of rewards.

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.