1)The vanilla Deep Q-network

The network used by the Vanilla Deep Q Agent is detailed in Vanilla_DQN/pytorch_model.py and is a simple forward pass 5 layer (128, 128, 128, 64, 64).

The Hyperparameters are in the options file of the Vanilla_DQN folder Specifically, they are:

```
batch 64
memory_size 1000000
update_freq 32
lr 0.0001
discount_rate 0.9
transfer_rate 0.001
env Unity_Banana
env_seed 0
num_episodes 3000
max_iteration 1000
min_epsilon 0.1
decay 0.995
win_cond 13
```

2)The Double Q-Learning Agent

The networks used for the Double Deep Q Agent are two of the same kind (the same for the Vanilla DQN Agent): Forward pass 5 layer MLP (128,128,128, 64,64)

The Hyperparameters are in the options file of the Double_DQN folder Specifically, they are:

```
batch 64
memory_size 1000000
update_freq 32
lr 0.0001
discount_rate 0.9
transfer_rate 0.001
env Unity_Banana
env_seed 0
num_episodes 3000
max_iteration 1000
min_epsilon 0.1
decay 0.995
win_cond 13
```

2)Double Dueling DQN

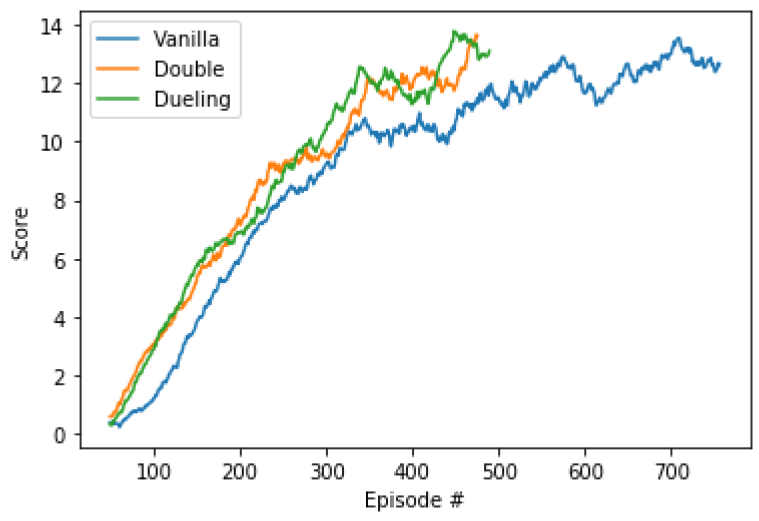
As far as the networks used: the q network and the dueling share 3 layers (128,128,64) and they each have an additional 64.

Here are the hyper-parameters for the Double-Dueling DQN Agent:

```
batch 64
memory_size 1000000
update_freq 32
```

```
lr 0.0001
discount_rate 0.9
transfer_rate 0.001
env Unity_Banana
env_seed 0
num_episodes 3000
max_iteration 1000
min_epsilon 0.1
decay 0.995
win_cond 13
```

Performance summary to win condition (50-rolling average)



Further improvements:

I have not been able to adjust the code in https://github.com/rlcode/per for prioritized replay (I'm not far, and help would be much appreciated, I'm getting a Unicode error…)

We could also implement improvements suggested in https://arxiv.org/pdf/1710.02298.pdf, like A3C or Noisy DQN, etc… all of them combined resulting in the rainbow algorithm.