Project 1: Navigation

This is for the vectorized input problem.

Learning Algorithm

I have taken the previous exercise as a skeleton for this problem.
You have the same structure a dqn_agent.py file that contains the learn and acting part. And a model.py file that contains the Deep Neural Network, use to evaluate the Q function.

This implementation is allowing Deep Reinforcement learning with 3 optional features

- Double DQN
- Prioritized Experience Replay
- Dueling

You can activate and deactivate using the Boolean in the hyperparameters cell.

I have also added tools, in the ./tools folder, like the prioritized replay buffer. To manage performance of big buffer, I had to implement Binary Tree sampling like proposed in the DeepMind paper.

Hyperparameters for the DQN:

All hyperparameters are manage from this cell, and are passed a argument dictionary to the relevant function.

In [7]:
```
from dqn_agent import Agent

hyperparameters = {
        'dqn' :     {   'n_episodes':           1300,           # Number of episode of self training
                        'max_t':                1000,           # Max value of steps per episode (should be more than 200)
                        'epsilon':              { 'name': 'ε', 'profil': 'geometric', 'init': 1., 'bound': 0.01, 'steps': 300 }
                    },
        'agent' :   {   'GPU':                  True,           # Use GPU (Bool)
                        'state_size':           37,             # State Size, here 37
                        'action_size':          4,              # Action space, here 4
                        'seed':                 0,              # Random generator seed
                        'DDQN':                 True,           # Use Double DQN (Bool)
                        'Prioritised_replay':   False,          # Use Prioritised Experience Replay (Bool)
                        'Dueling':              True,           # Use Dueling model (Bool)
                        'BUFFER_SIZE':          int(1e5),       # replay buffer size
                        'BATCH_SIZE':           64,             # minibatch size
                        'GAMMA':                0.99,           # discount factor
                        'LR':                   5e-2,           # learning rate
                        'LEARN_EVERY':          4,              # how often to update the network
                        'UPDATE_EVERY':         20,             # how often to update the network
                        'alpha' :               { 'name': 'α', 'profil': 'constant', 'init': 0.7 },
                        'TD_Error_clip':        [0., 1.], #float('inf')], # Cliping of TD Error range in PER selection
                        'beta' :                { 'name': 'β', 'profil': 'linear', 'init': 0.6, 'bound': 1., 'steps': 200*75 },
                        'tau':                  { 'name': 'τ', 'profil': 'constant', 'init': 5e-2 },
                    },
}

agent = Agent(**hyperparameters['agent'])
```

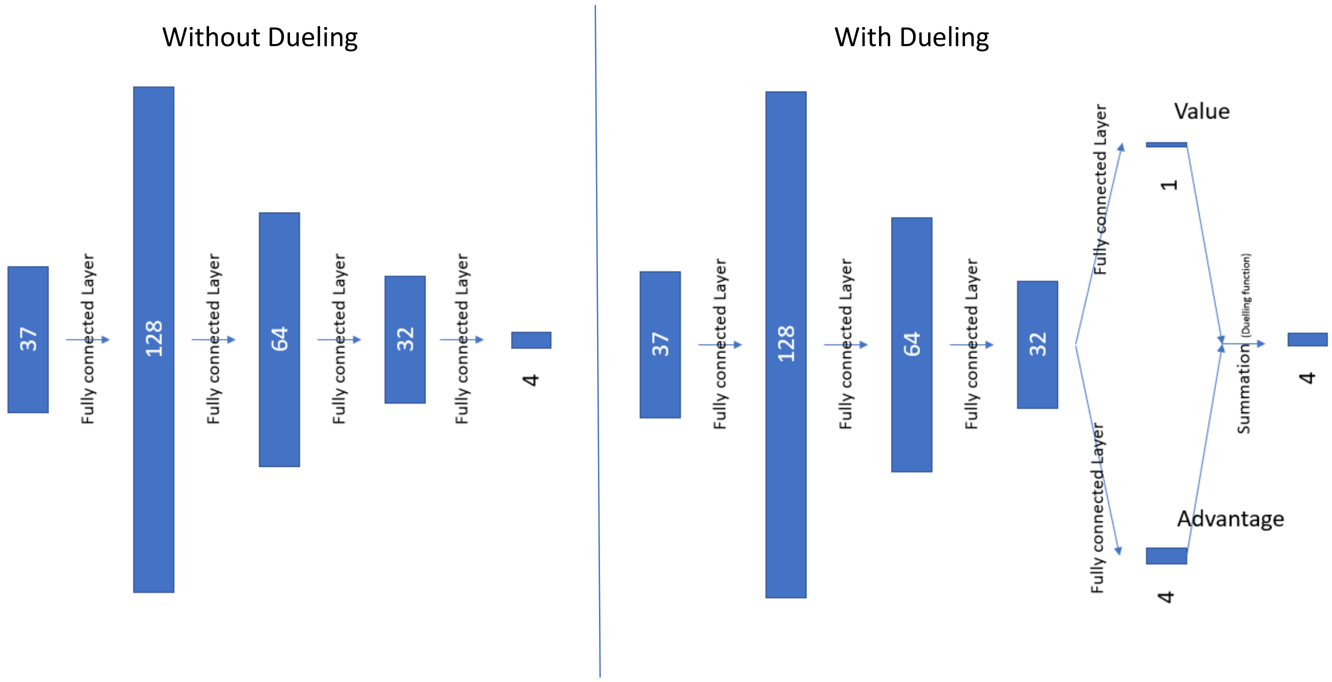
After some long trial and error, I have settled on those parameters:

- GPU active for performance.
- Double DQN active,
- Prioritized replay inactive, I have better performance without, I tried introducing clipping of TD Error, changed Alpha and Beta, etc… But still better without.
- Dueling active
- Replay buffer at 10000 like previously
- Batch size for learning at 64
- Gamma at 0.99, very standard, just not to loop.
- Learning Rate 0.05
- Trigger learning every 4 episodes
- And change the target network every 20 episodes

- Alpha, TD_Error clipping and beta not used as PER inactive
- Tau at 0.05 as the rate to move from current to target network.

For the Neural Network

I have used a simple structure for the vectorized problem,

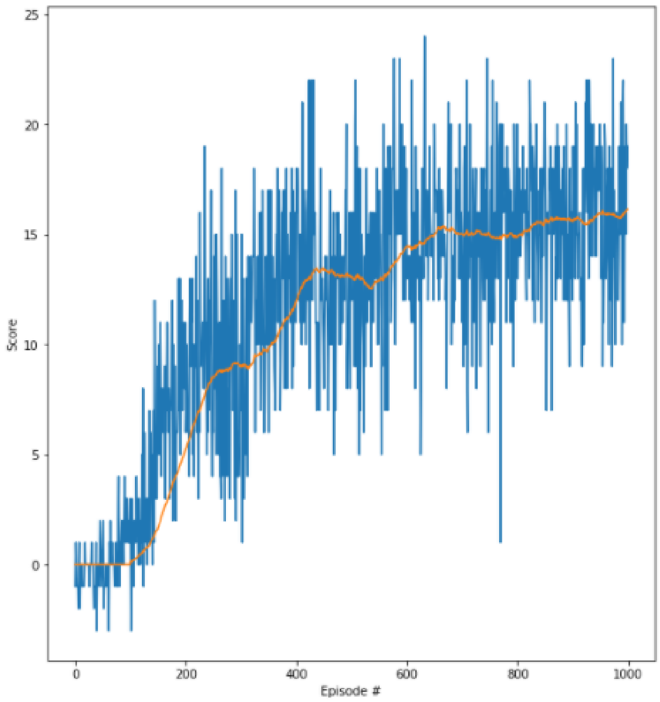


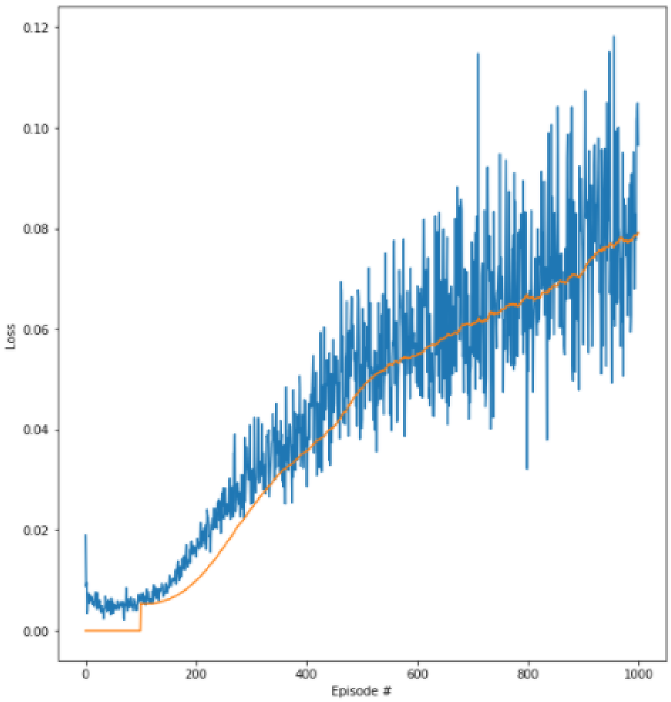
Plot of Rewards

When using the “Navigation.ipynb” you have the following result in 2 plots.

1. score and average score over the last 100 episodes per episode of training.
2. Loss function and average loss over the last 100 episodes per episode of training.

With Dueling and DDQN activate

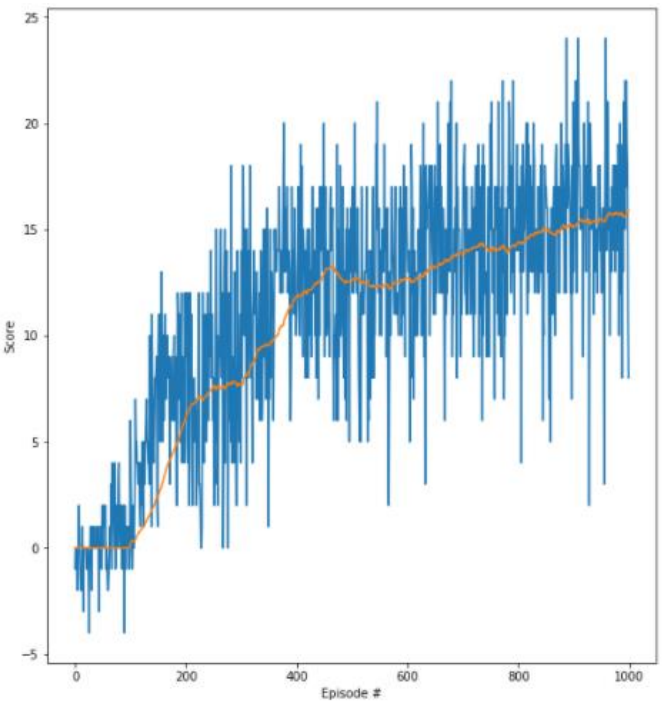


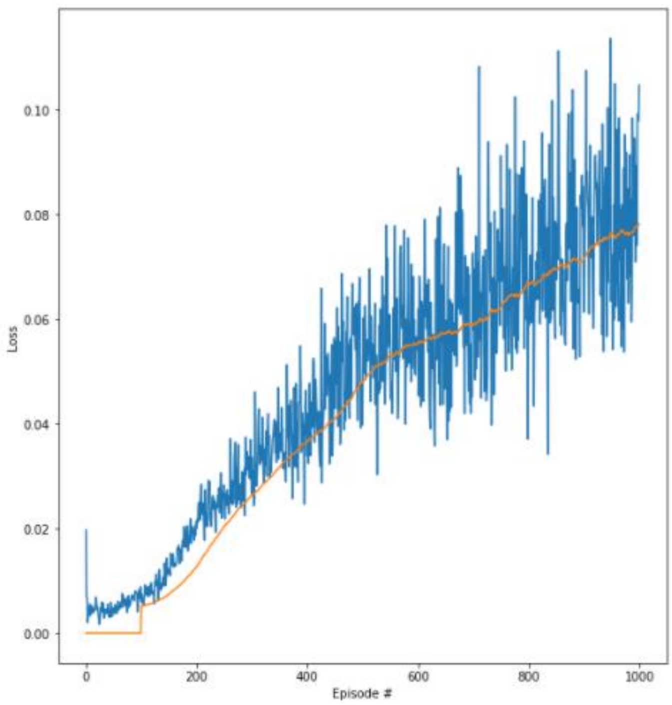


The average score over 100 episodes has reached 13 on episodes 424, as the instructor way of counting, this implies that the score of 13 was reached 100 episodes before. So 324

```
Episode  100 in 1.48s.  Score   3  Average Score:  0.14  Loss: 6.5e-03  LR: 5.0e-02 Paramaters ε=2.2e-01 γ=9.9e-01  τ=5.0e-02
Episode  200 in 1.47s.  Score  11  Average Score:  5.26  Loss: 1.5e-02  LR: 5.0e-02 Paramaters ε=4.6e-02 γ=9.9e-01  τ=5.0e-02
Episode  300 in 1.44s.  Score   7  Average Score:  8.98  Loss: 2.5e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  400 in 1.46s.  Score  11  Average Score: 11.97  Loss: 3.9e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  424 in 1.51s.  Score  22  Average Score: 13.07  Loss: 3.2e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 324 episodes! Average Score: 13.07
Episode  500 in 1.52s.  Score  13  Average Score: 13.09  Loss: 4.0e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  588 in 1.49s.  Score  23  Average Score: 14.10  Loss: 4.6e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 488 episodes! Average Score: 14.10
Episode  600 in 1.47s.  Score  18  Average Score: 14.45  Loss: 5.2e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  650 in 1.50s.  Score  20  Average Score: 15.03  Loss: 6.4e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 550 episodes! Average Score: 15.03
Episode  700 in 1.47s.  Score  15  Average Score: 15.00  Loss: 7.7e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  800 in 1.50s.  Score  20  Average Score: 14.91  Loss: 6.6e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  900 in 1.48s.  Score  12  Average Score: 15.59  Loss: 6.5e-02  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  953 in 1.51s.  Score  19  Average Score: 16.03  Loss: 6.7e-02  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 853 episodes! Average Score: 16.03
Episode 1000 in 1.49s.  Score  19  Average Score: 16.20  Loss: 9.7e-02  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
```

Without any extra feature, just DQN





The average score over 100 episodes has reached 13 on episodes 450, as the instructor way of counting, this implies that the score of 13 was reached 100 episodes before. So 350.

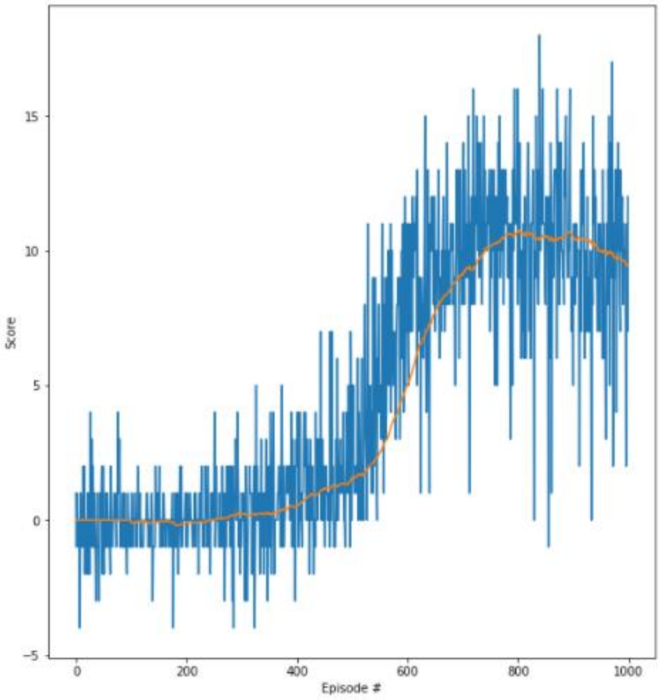
```
Episode  100 in 1.25s.  Score   0  Average Score:  0.21  Loss: 8.8e-03  LR: 5.0e-02 Paramaters ε=2.2e-01 γ=9.9e-01  τ=5.0e-02
Episode  200 in 1.29s.  Score   9  Average Score:  6.19  Loss: 2.1e-02  LR: 5.0e-02 Paramaters ε=4.6e-02 γ=9.9e-01  τ=5.0e-02
Episode  300 in 1.22s.  Score   4  Average Score:  7.66  Loss: 2.7e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  400 in 1.23s.  Score  10  Average Score: 11.86  Loss: 4.0e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  450 in 1.26s.  Score  20  Average Score: 13.02  Loss: 5.2e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 350 episodes! Average Score: 13.02
Episode  500 in 1.24s.  Score  14  Average Score: 12.60  Loss: 4.0e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  600 in 1.28s.  Score  15  Average Score: 12.72  Loss: 5.5e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  700 in 1.30s.  Score  13  Average Score: 13.77  Loss: 7.3e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  705 in 1.29s.  Score  12  Average Score: 14.01  Loss: 7.6e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 605 episodes! Average Score: 14.01
Episode  800 in 1.25s.  Score  14  Average Score: 14.40  Loss: 6.0e-02  LR: 5.0e-02 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode  844 in 1.26s.  Score  19  Average Score: 15.03  Loss: 6.2e-02  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Environment solved in 744 episodes! Average Score: 15.03
Episode  900 in 1.28s.  Score  21  Average Score: 15.15  Loss: 6.1e-02  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
Episode 1000 in 1.24s.  Score   8  Average Score: 15.75  Loss: 1.0e-01  LR: 5.0e-03 Paramaters ε=1.0e-02 γ=9.9e-01  τ=5.0e-02
```

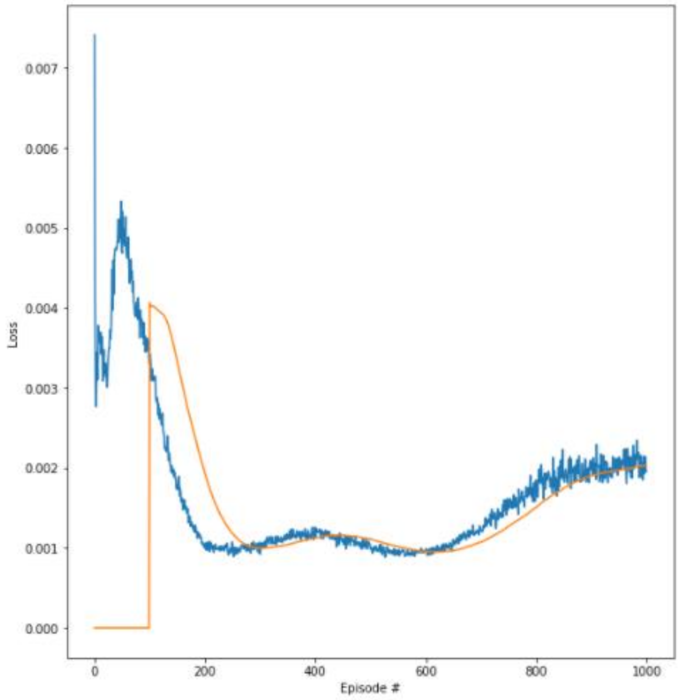
My conclusion is that without extra features it takes more time to learn and you get a lower best score. Probably due to the difficultly of randomly walking into a yellow banana, it seems to take 100 episodes to understand that grabbing yellow bananas is better than wandering around. But even more than the extra features, I fear that hyperparameters right picking is also very influential on the end result.

Ideas for Future Work

On the vector problem:

- Find a better quicker/less complex NN structure that converge and learn faster.
- Find better alpha and beta parameter that make PER more beneficial.
- Better understand the link between network structure and convergence.
- Understand why PER is bringing so much overfitting





On the vision problem:

- Change the input processing of the input image. Stop using RGB and create 1 frame for Yellow, one for Blue and one for background, on 4 frames, so 12 input channels to the convolutional network.
- Understand how long a network need to converge. (not like Deepmind brute force)
- Better understand the link between structure and convergence.
- Better understand impact/benefits of more depth or more convolutions.
- Better understand where to put residual layers.

I have spent some time on the code to make it nice, it is more now the fine tuning of parameters the next and perpetual challenge.