Report on DQN

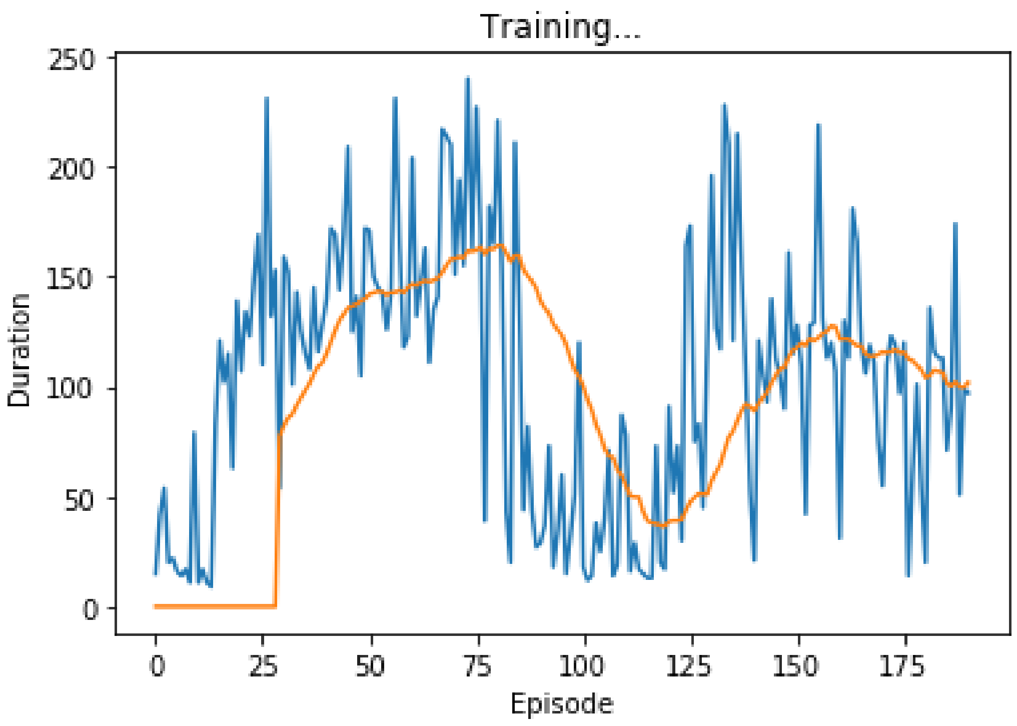
February 22, 2018

```
In [6]: # Set up
        %matplotlib inline
        from IPython.display import clear_output
        import dqn
```

0.1 Training with image pixel input

The network is designed to have 3 convolutional layers (with batch normalization and relu), and 1 fully connected layer.

```
In [19]: # image pixel input
         dqn.args.image = True
         dqn.args.batch_size = 128
         dqn.args.gamma = 0.999
         dqn.main(dqn.args)
```

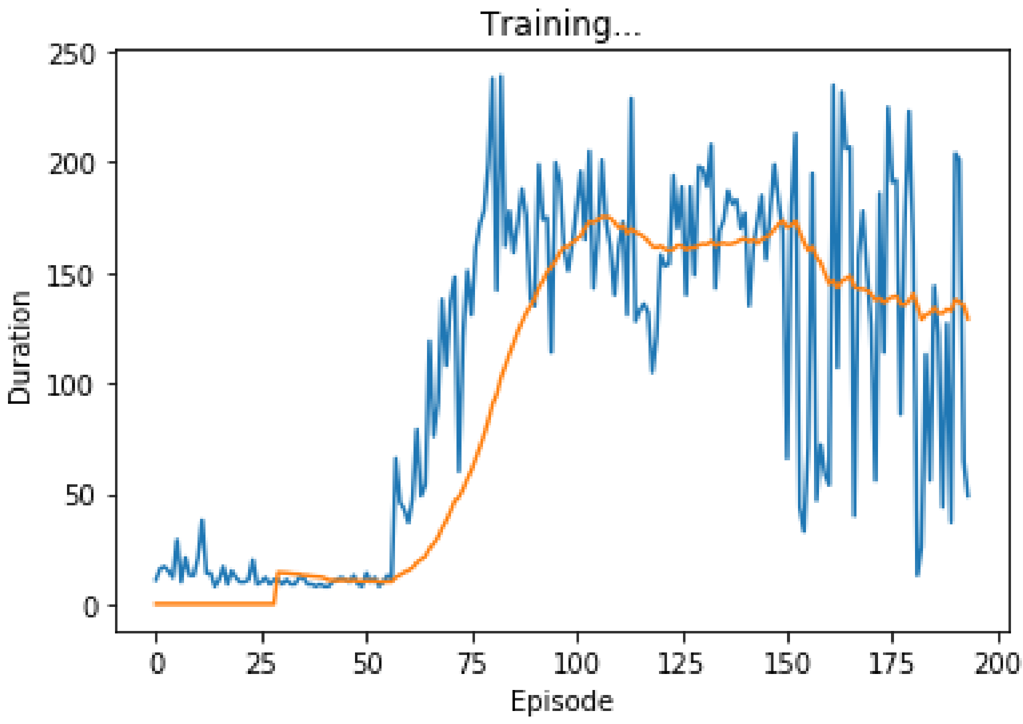


The training yields rewards over 150 in less than 200 epochs. The training is rather time-consuming, and it takes quite long time to recover from a descending direction in exploration.

0.2 Training with state vector input

The network is designed to have 3 fully connected layers (with batch normalization and relu).

```
In [17]: # state vetor input
         dqn.args.image = False
         dqn.args.batch_size = 128
         dqn.args.gamma = 0.85
         dqn.main(dqn.args)
```



The training yields better result while significantly more efficient than that with image pixel input, as the input data capture mian infomation with only 4 variables, so the parameters to be trained is far less and the training is easier. This comparison demonstrates the importance of data reduction.

In addition, it is notable that with less input variables, the discount γ should be smaller than that with image pixel input (less than 0.9). Otherwise, the training gets stuck on existing experience and fails to update the policy due to lack of flexibility.