Project Report P1-navigation

Aim was to solve the Banana collection environment to collect an average reward of 13 in 100 subsequent runs

Student:

Rahul Kaplesh

Course:

Deep Reinforcement learning Nanodegree by Udacity

Challenge 1

In this challenge an agent must be developed to collect the bananas on an environment which returns the state as a vector space.

Three methods were used to solve this challenge. Their results are as follows:

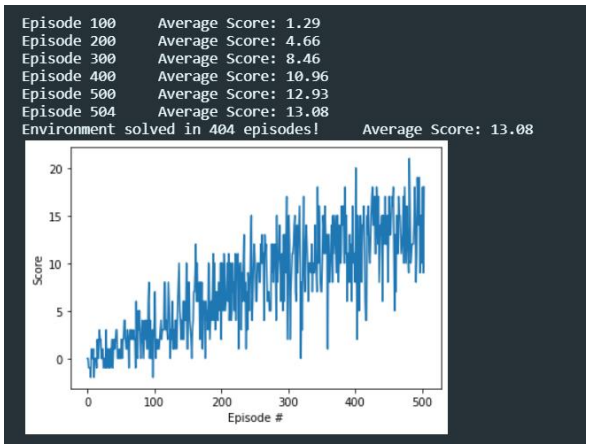
Double DQN

In this method a double layered deep Q Network was used to train the agent. Agent maintained a target Q Network and a local Q Network. Local Q Network was used to make predictions, but the target Q Network was used for training the weights of the model. The parameters for the local Q Network was translated to the Target Q Network using a factor tau. The training notebook can be found in the “DoubleDQN/ DoubleDQNFromEx.ipynb”. The agent definition is in the file “DoubleDQN/agent.py”. The model definition is found in “DoubleDQN/model.py”.

Model summary:

```
    summary(agent.qnetwork_local,(state_size,))
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                   [-1, 64]           2,432
            Linear-2                   [-1, 64]           4,160
            Linear-3                    [-1, 4]             260
================================================================
Total params: 6,852
Trainable params: 6,852
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.03
Estimated Total Size (MB): 0.03
----------------------------------------------------------------
```

Results:



Experiments gets trained in 504 episodes.

Double DQN

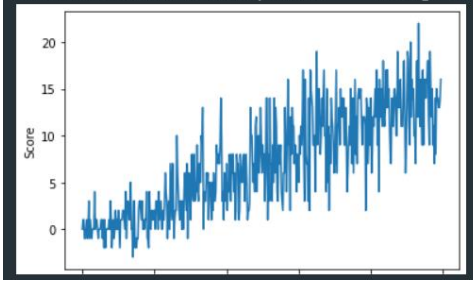
This is like the double DQN except the model is a fully connected and deeper layered network. The training notebook can be found in the “DoubleDQN_moreFCLayers/ DQN3Layers.ipynb”. The agent definition is in the file “DoubleDQN_moreFCLayers/agent.py”. The model definition is found in “DoubleDQN_moreFCLayers/model.py”.

Model summary:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                 [-1, 128]           4,864
            Linear-2                  [-1, 64]           8,256
            Linear-3                  [-1, 32]           2,080
            Linear-4                   [-1, 4]             132
================================================================
Total params: 15,332
Trainable params: 15,332
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.06
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

Results:

```
Episode 100     Average Score: 0.61
Episode 200     Average Score: 4.03
Episode 300     Average Score: 7.00
Episode 400     Average Score: 10.61
Episode 498     Average Score: 13.05
Environment solved in 398 episodes!     Average Score: 13.05
```



Agent gets trained in 498 episodes.

Double DQN With Exp Replay

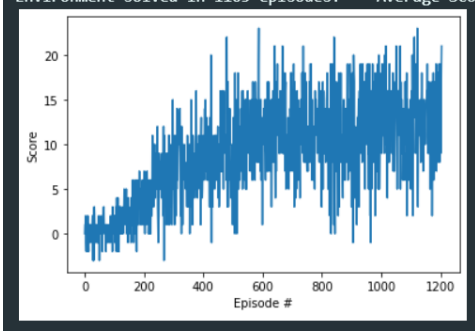
This is like the double DQN, but the experiences used for training the model are selected based on temporal difference error and biases. The training notebook can be found in the "DoubleDQN4LayersExpReplay/ ExperienceReplay.ipynb". The agent definition is in the file "DoubleDQN4LayersExpReplay/agent.py". The model definition is found in "DoubleDQN4LayersExpReplay/model.py".

Model summary:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                 [-1, 128]           4,864
            Linear-2                  [-1, 64]           8,256
            Linear-3                  [-1, 32]           2,080
            Linear-4                   [-1, 4]             132
================================================================
Total params: 15,332
Trainable params: 15,332
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.06
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

Results:

```
Episode 700     Average Score: 11.71
Episode 800     Average Score: 11.57
Episode 900     Average Score: 10.98
Episode 1000    Average Score: 11.95
Episode 1100    Average Score: 12.58
Episode 1200    Average Score: 12.79
Episode 1205    Average Score: 13.04
Environment solved in 1105 episodes!    Average Score: 13.04
```



Agent gets trained in 1205 episodes. This training is very slow maybe a faster implementation of this method can be explored.

Double DQN With Dueling Network

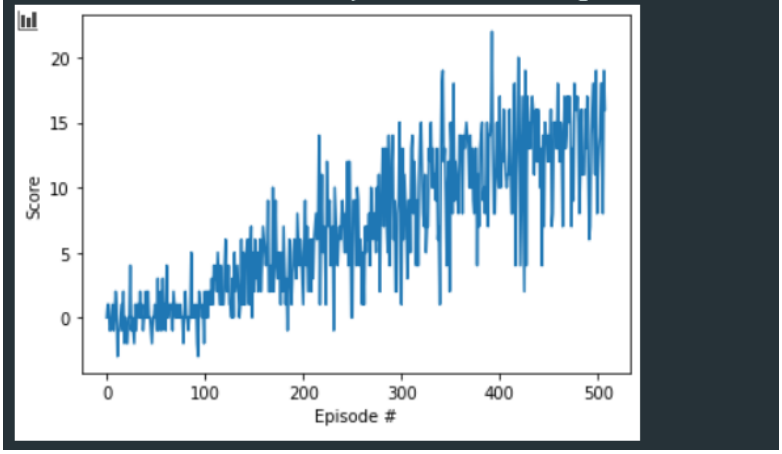
This is like the double DQN, but the state value is also computed using another similar neural network. The training notebook can be found in the “Dueling_NN_DQN/ Duelling_NN_DQN.ipynb”. The agent definition is in the file “Dueling_NN_DQN/agent.py”. The model definition is found in “Dueling_NN_DQN/model.py”.

Model Summary:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 128]           4,864
            Linear-2                 [-1, 64]           8,256
            Linear-3                 [-1, 32]           2,080
            Linear-4                  [-1, 4]             132
            Linear-5                  [-1, 1]              33
================================================================
Total params: 15,365
Trainable params: 15,365
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.06
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

Results:

```
Episode 100     Average Score: 0.22
Episode 200     Average Score: 3.40
Episode 300     Average Score: 6.85
Episode 400     Average Score: 10.64
Episode 500     Average Score: 12.87
Episode 508     Average Score: 13.03
Environment solved in 408 episodes!     Average Score: 13.03
```



Agent gets trained in 508 episodes.

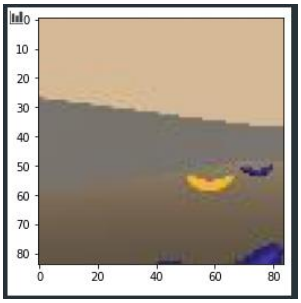
Final observations:

- A normal DQN with more layers is a better model for this environment

Challenge 2

In this challenge an agent must be developed to collect the bananas on an environment which returns the state as a RGB image of what the agent is looking at. I reduced the reward which the agent needs to attain for this challenge to 12.

State was built as frame comprising of 4 such images and was passed to a neural network implementing a 3d convolution to extract features info from a stack of frames. These features were used for then estimating the agent actions. State space as captured from environment:

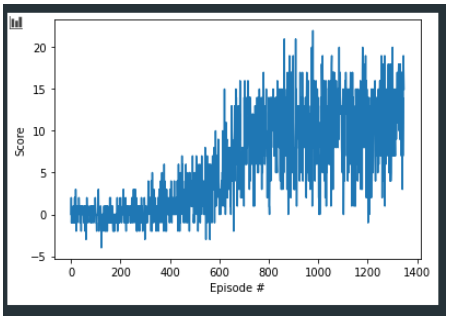


Model summary:

```
QNetwork(
  (conv_layer1): Conv3d(3, 128, kernel_size=(1, 3, 3), stride=(1, 3, 3))
  (btm_layer1): BatchNorm3d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv_layer2): Conv3d(128, 256, kernel_size=(1, 3, 3), stride=(1, 3, 3))
  (btm_layer2): BatchNorm3d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv_layer3): Conv3d(256, 512, kernel_size=(4, 3, 3), stride=(1, 3, 3))
  (btm_layer3): BatchNorm3d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=4608, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=32, bias=True)
)
```

A deep Q Network with the above-mentioned architecture was chosen for the above problem.

Results:




```
INFO: 2020-09-23 18:23:24,223 Episode 1341 Average Score: 11.74
INFO: 2020-09-23 18:23:48,307 Episode 1342 Average Score: 11.72
INFO: 2020-09-23 18:24:13,514 Episode 1343 Average Score: 11.81
INFO: 2020-09-23 18:24:38,976 Episode 1344 Average Score: 11.80
INFO: 2020-09-23 18:25:05,187 Episode 1345 Average Score: 11.76
INFO: 2020-09-23 18:25:27,728 Episode 1346 Average Score: 11.89
INFO: 2020-09-23 18:25:50,351 Episode 1347 Average Score: 11.95
INFO: 2020-09-23 18:26:13,220 Episode 1348 Average Score: 12.00
INFO: 2020-09-23 18:26:13,220 Environment solved in 1348 episodes! Average Score: 12.00
```

Agent gets trained in 1348 episodes. Average score attained by the agent over a period of 100 episodes is 12.

Further Work

Further work can be done at improving the neural network to extract the features out of the video. The implementation of experience replay is very slow in executing, this can be improved. Other techniques for implementing deep Q Networks can also be explored. Even an implementation to achieve an average score of 13 can be worked on.