Project report

1 Learning algorithm

We are using an hybrid approach to solve the following environment, a combination of Q-learning – a value based temporal difference method and deep neural networks known as the Deep Q-Learning Algorithm. Neural networks, is used as a universal approximator to predict the future values of the Q-table, replacing the table involved in the Q-learning method with a deep neural network. This allows for us to find more complex relationship within the data and overcomes the limitation of the Q-table by allowing for a larger number of action-value predictions and therefore expanding the size of problems we can solve. In addition, two more features implemented to allow for better performance and stabilisation is: 1) Experience replay and 2) Fixed Q-Targets.

1 Experience replay
Experience replay allows us to make use of past experience by replaying a set of tuple (S, A, R, S') experience stored in a replay buffer. This allows us to break the temporal correlation that comes with updating values without any memory – making use of rare experience more and mixing experiences for updates.

2 Fixed Q-Targets
Fixed Q-Targets, decouples the target from the agents action allowing us to overcome the erroneous correlation that arises from the fact we constantly update the parameters at every time step. Fixed Q-Target allows us to train the agent for n time steps before updating the weights in W-, which can be represented in the equation below, labeled (1).

$$\Delta w = \alpha \cdot (R + \gamma \max\_a \, q\hat{}(S', a, w-) - q\hat{}(S, A, w)) \nabla w q\hat{}(S, A, w) \qquad (1)$$

W- is the parameter that is updated by a different target network and which is not done during the learning step.

Model - Deep Neural Network

For the neural network I used a 3 layer structure with the first layer consisting of 64 nodes and the second layer of 64 nodes.

For each layer I used a Rectified Linear Unit (RELU) to return a value between 0 and 1 (probability outcome),which is attributed towards each action in the action space. The last layer then returns probability outcomes for each action (number of nodes corresponding to the number of actions in the action space).

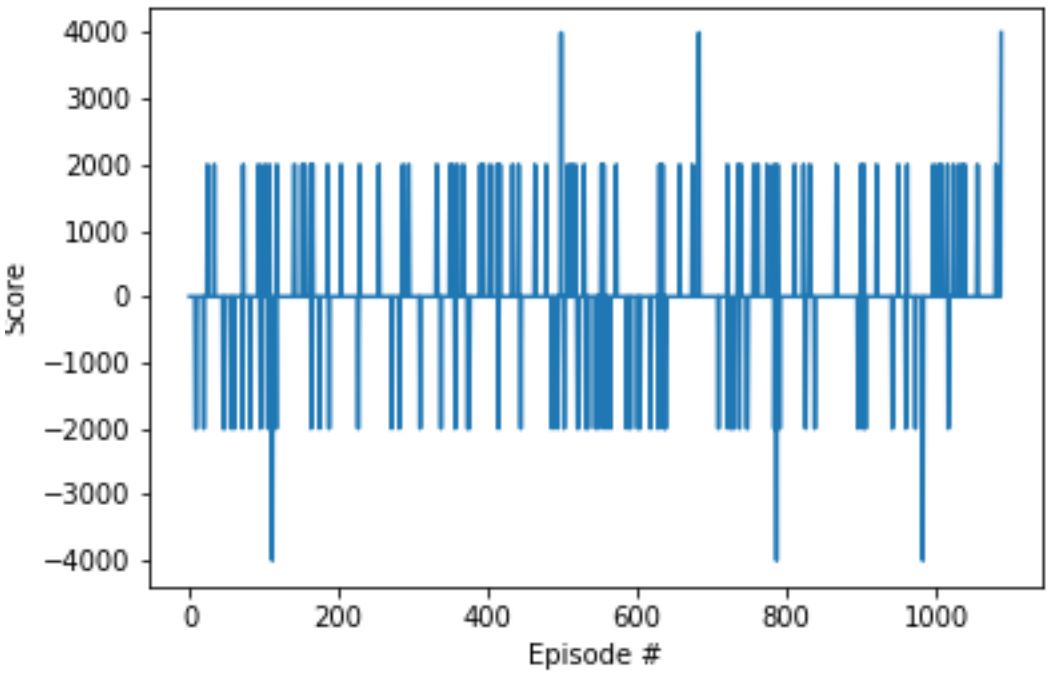
Hyperparameters

For the hyperparameters I done many trial and error, but with some changes being made with the motivation towards increasing the capabilities to replay memory or speed the learning process due to the limited number of data. Below are the list of hyperparameters which I used to solve the environment:

```
BUFFER_SIZE = int(1e6)          # replay buffer size
BATCH_SIZE = 64                 # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 1e-3                      # for soft update of target parameters
LR = 5e-1                       # learning rate
UPDATE_EVERY = 4                # how often to update the network

n_episodes = 2000
max_t = 2000
eps_start = 1.0
eps_end = 0.01
eps_decay = 0.995
```

2 Plot - Rewards



3 Future works

For future work we can try to increase the performance by using prioritised experience replay – a variation of experience replay which focuses on specific transition over other ones, since not every transition is as fruitful as one another [1].

There are two main design choices we can make when thinking of developing a replay memory capacity: which experiences to store and which experiences to replay. Prioritised experience replay tackles the latter, focusing on how to select from memory. To implement this I would change the data structure from deque to a binary heap data structure, so that when I sample the data for the memory with the highest priority transition I can retrieve it in O(1) and update priorities in O(logN) time complexity.

To do this successful I would need to use one of the many different criterion to measure what 'good' memory looks like. One such criterion is the temporal relationship towards learning and the memory retrieved, known as TD-Error. Another is stochastic prioritization which is a balance between greedy prioritization and uniform random sampling, allowing more variation and no zero probability is ever assigned as priority to a memory.

4 Sources

[1] Tom Schaul, John Quan, Ioannis Antonoglou and David Silver
Google DeepMind. Prioritized Experience Replay.
{schaul,johnquan,ioannisa,davidsilver}@google.com