SOLVING THE BANANA ENVIRONMENT

PRE-REQUISITES

The reader is assumed to have a background in classical reinforcement learning.

LEARNING ALGORITHM

It is based on Deep Q-Learning with the following improvements that are combined:

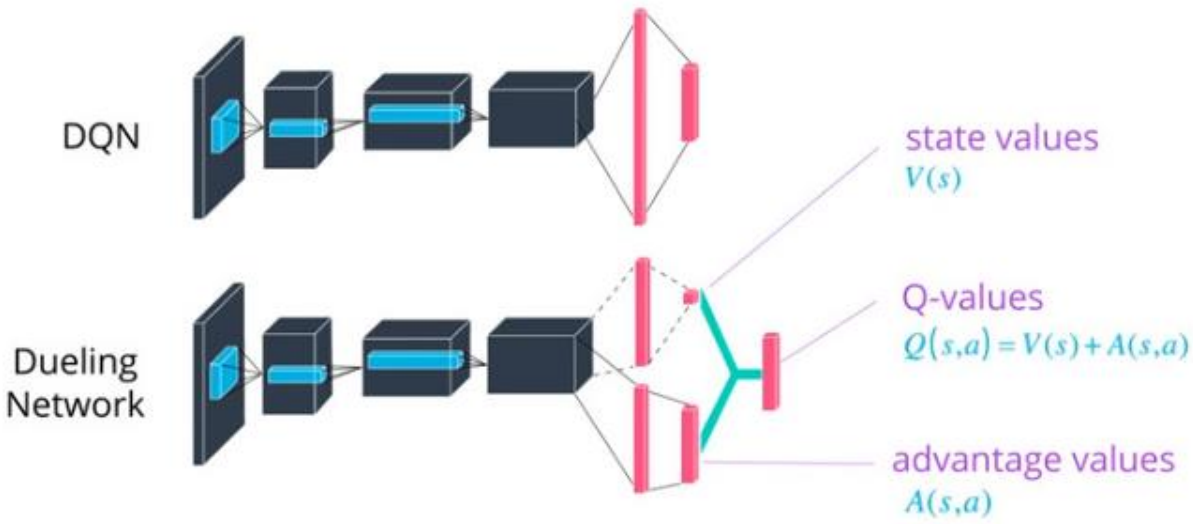
- Experience Replay
- Fixed Q-Target
- Double Deep Q-Learning
- Dueling Network
- Prioritized Experience Replay

The main idea is to implement a Sarsa Max Reinforcement Learning algorithm by using a function approximator when the environment is continuous. Instead of a linear function, a neural network is being used.

Following the research by DeepMind on the development of an agent to solve the games on Atari 2600, a Deep Q Network can converge and be prevented from oscillating when the learning is based on samples from its experiences. Indeed, consecutive experiences are known to have strong correlations that can be broken with the experience replay. In another hand, when calculating a loss, a reference is needed. When applying the Bellman equation with a neural network, the algorithm will end-up correcting the weight of that neural network based on itself. That is fixed by using as a reference, an old copy of the same network. That behavior is copied from the function of the hippocampus of human brain.

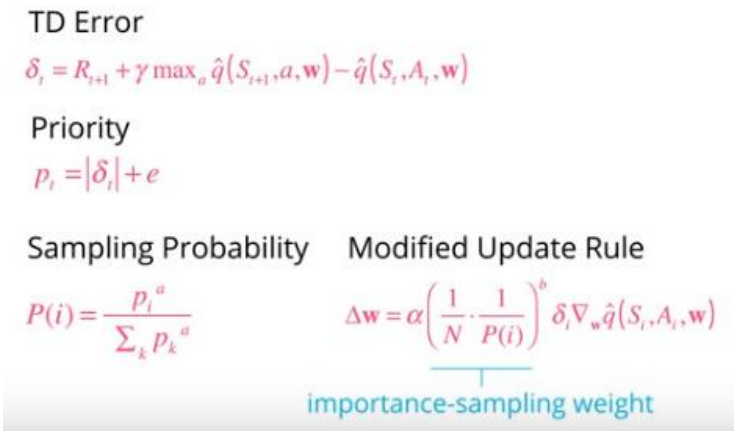
Afterwards, according to a paper by Sebastian Thrun et al., Deep Q-Learning overestimates actions values. To counter that, when estimating the action-values of the next states, the online network is being used to find out the best action whereas, the fixed network will estimate its value. That will be used as a reference during the training of the online network. In that way, if both networks do not agree, the estimated value will be less.

Then, the idea of a dueling network by Ziyu Wang et al., is to have the action-value approximator as the sum of a state-value approximator and an advantage-value approximator. To achieve this, the last stream of a DQN is replace with two parallel streams. The first stream will calculate the value state and the second, the advantage values for each action on that particular state. That is illustrated here below:



The key is that it is not necessary to estimate the value of each action for many states. However, Q=V+A is not identifiable. In fact, adding a constant to V and removing that same constant to A will still lead to the same Q function. Applying directly that formula will lead to poor performance. To fix that, the combination of V and A will be: Q = V + (A – mean(A)).

Finally, a Deep Q Learning can be improved by Prioritized Experience Replay as well according to Tom Schaul et al. When sampling from the experiences memory, the agent should retrieve those which require further exploration. That can be defined by assigning a priority to each experience. That priority is function of the error encountered by the network while learning. To reduce biases, a default priority is assigned to the experiences that are well estimated by the network. In addition to that, during backpropagation, the gradients are changed as well to be proportional to the probability of the experience.



In the previous formulas:

- e is the default minimal priority given to the experiences
- a is a fixed value
- if a is close to 1, the sampling will be based on priority only, and when close to 0, the sampling will be based on a normal distribution
- b is a value increasing from 0 to 1. Practically, b should be close to 1 as the network is converging.

IMPLEMENTATION DETAILS AND HYPERPARAMETERS

In this repository, 3 different implementations are provided:

- 1st – Double Q-Learning Algorithm with Experience Replay and fixed Target
- 2nd – Dueling Network with Double Q-Learning Algorithm with Experience Replay and fixed Target
- 3rd – Dueling Network with Double Q-Learning Algorithm with Prioritized Experience Replay and fixed Target

For performance concern, the priorities in the entire memory are not updated before every sampling but rather on a lower frequency. In that way, the algorithm spent less computation resource.

The hyperparameters and model architectures are here below:

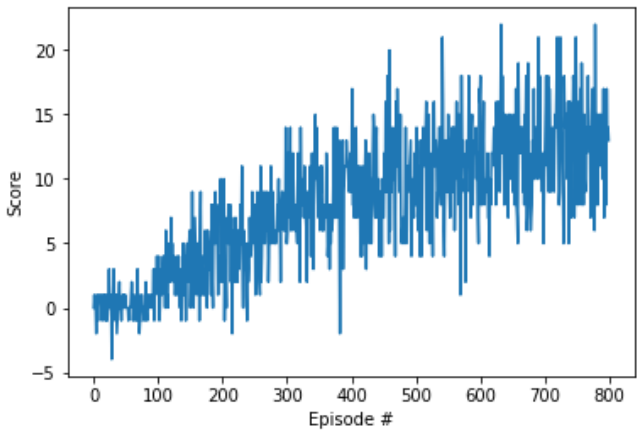
	Double Q-Learning Algorithm with Experience Replay and fixed Target	**Dueling Network with Double Q-Learning with Experience Replay and fixed Target**	**Dueling Network with Double Q-Learning with Prioritized Experience Replay and fixed Target**
Size of the Memory	10,000	10,000	10,000
Size of sampling	32	32	32
Discount Factor	0.99	0.99	0.99

	Double Q-Learning Algorithm with Experience Replay and fixed Target	Dueling Network with Double Q-Learning with Experience Replay and fixed Target		Dueling Network with Double Q-Learning with Prioritized Experience Replay and fixed Target	
Rate of transfer from local to targeted network	0.5	0.5		0.5	
Learning Rate	0.0005	0.0005		0.0005	
Frequency of update of the local network	Every step	Every step		Every step	
Frequency of update of the targeted network	Every 4 steps	Every 4 steps		Every 4 steps	
GLIE - Epsilon decay	Multiplicative factor per episode of 0.995	Multiplicative factor per episode of 0.995		Multiplicative factor per episode of 0.995	
GLIE - Minimum Epsilon	0.01	0.01		0.01	
Maximum steps in one episode	500 steps	500 steps		500 steps	
Architecture of the neural network	**Input (37)** Fully Connected (128) ReLU Fully Connected (64) ReLU Fully Connected (32) ReLU Final (4) ReLU **Output**	**Input (37)** Fully Connected (128) ReLU Fully Connected (64) ReLU Fully Connected (32) ReLU		**Input (37)** Fully Connected (128) ReLU Fully Connected (64) ReLU Fully Connected (32) ReLU	
		Fully Connected (16) ReLU Fully Connected (4) **Advantage - ReLU**	Fully Connected (16) Fully Connected (1) **State Value - ReLU**	Fully Connected (16) ReLU Fully Connected (4) **Advantage - ReLU**	Fully Connected (16) Fully Connected (1) **State Value - ReLU**
		Output = State Value + (Advantage – mean(Advantage))		**Output = State Value + (Advantage – mean(Advantage))**	
Frequency of priority update				Every 32 steps	
Default Priority				1e-10	
Priority Factor				0.7	
Importance Sampling Factor Increase				Fixed at 1	

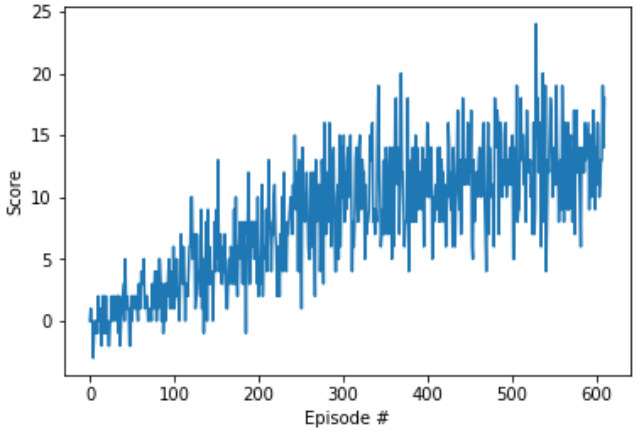
RESULTS

The table below represents the results that are achieved by each implementation:

	Double Q-Learning Algorithm with Experience Replay and fixed Target	**Dueling Network with Double Q-Learning with Experience Replay and fixed Target**	**Dueling Network with Double Q-Learning with Prioritized Experience Replay and fixed Target**
Number of episodes spent to achieve an average of more than 13 points for 100 consecutive episodes	800	611	Did not finish



Scores of Double Q-Learning with a simple network



Scores of Double Q-Learning with Dueling Network

IDEAS FOR FUTURE WORK

The implemented Prioritized Experience Replay did not converge. So, the next step is to fix that by adjusting the implementation.

Besides, that algorithm is requiring extra computing resource while calculating priorities and sampling the experiences. Moreover, regardless of having all tensors on a GPU, the memory is still sitting on the CPU. So, each calculation requires a transfer of data from the CPU to the GPU. That led to a lower speed of training. An improvement of this implementation is to have the memory sitting on the GPU.

Finally, maybe, instead of having an old copy of the online network, a better fixed target might be achieved as well by having:

- Two different networks A and B, learning on different experiences but B is fixed when A is learning and vice-versa
- Two different network A and B, learning on the same experiences but B is fixed when A is learning and vice-versa

NOTES

Some illustrations in this document are retrieved from the Udacity's Deep Reinforcement Learning Nanodegree