Reinforcement Learning - Project assignment 1

For this assignment a reinforcement learning agent is trained to navigate through a 2D world. The goal for the agent is to collect yellow bananas by ignoring blue bananas. The provided environment is a Unity Framework. The agent is as a double deep q-learning network (DDQN) implemented with prioritised experience replay implemented. The goal of the agent is to reach 13 points in average over 100 episodes.

1. Learning Algorithm

Reinforcement Learning is about to teach an agent to interact with an environment. The agent gets the current state of an environment and return the action with the goal to maximise a reward. With **deep Q-learning** the agent uses an artificial neural network to predict the best action for a current state. An artificial neural network is used in this case as a nonlinear function approximater to learn a state-action mapping for the environment. The agent tries to select actions, that the expected discounted reward will be maximised. Therefor the **discount-rate gamma** is used to determines the present value of future rewards. A sequences of actions are called policy. As policy-method the **epsilon greedy** method is used. This on-policy method, which choose most of the time an action with the maximal estimated action value. But with the probability of epsilon the epsilon greedy method also choose a random action. This policy method allows to explore unknown states of the environment (Sutton and Barto 2018).
A problem of DQNs is, that action values can be overestimated under certain conditions. **Double Q-learning** introduces a second neural network, which separates learning and acting. This approach reduces observed overestimations and has shown a better performance on several use cases (van Hasselt et al. 2015). Double Q-learning results in a more stable and reliable learning.
With **prioritised experience replay** the agent experience (state) at each time step is stored in a replay buffer. This replay buffer aggregates state, action, reward tuples over many episodes of the game. A batch for training the agent is uniformly sampled from this replay buffer for the agent training. So the agent learns from the action, state, reward tuple independently (Sutton and Barto 2018). This allows to learn the agent more efficient, since there is not much correlation between the experiences.

2. Model architecture and hyperparameters

The reinforcement learning agent is trained with the ?! implementation. The hyper parameters for the **discount-factor** gamma is 0.99. For tau, an interpolation parameter for updating the double den, a value of 0.001 was used. The double network architecture is **updated** with every forth run. The **replay buffer** uses a buffer size of 10000 samples, where a batch of 64 samples are randomly picked for the training of the neural network.

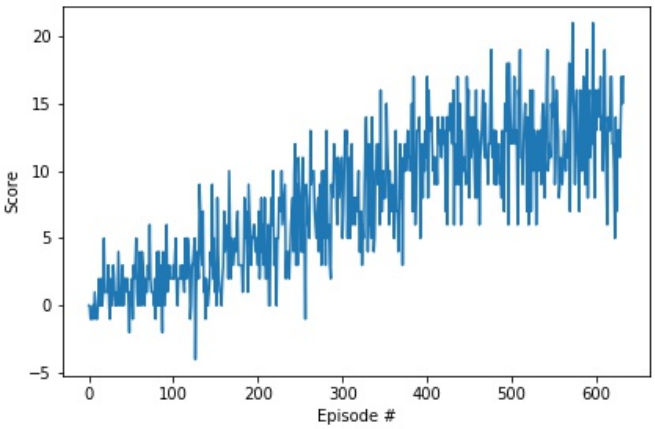
The implementation of the neural network uses a double DQN approach. For the model two identically **fully connected neural network** are used for the model. Each neural network has 2 hidden layers with 256 nodes and uses ReLus as activation functions. As an optimizer the Adam optimizer is used with a learning rate of 0.0005.

Hyperparameter	Value
Buffer size	10000
Batch size	64
Gamma	0.99
Tau	0,001
Learning rate	0.0005
Update cycle	4

3. Results

The goal of the agent is the achievement of 13 points over an average of 100 episodes. This score is reached at episode 633. So for every simulation, the agent collects in average 13 bananas. As in the table and especially in the graph shown, the agent learns linear until episode 500. After episode 500 there is a slow increase of the score until episode 500 until 13 points are reached. But still the agent has trouble to reach those 13 points with every episode. So the score can vary between 5 points and 20 points. A future improvement can be to reduce the variance of the score for the agent.

Episode	Average score
100	1.28
200	3.47
300	6.46
400	9.12
500	11.97
600	12.61
633	13.00



4. Future ideas

A future implementation idea for the agent is the implementation of the complete rainbow algorithm (Hessel et al. 2018). This means, that additional to the current implementation of a double DQN and a prioritised experience replay, four additional algorithms will be added like d**ueling** DQN, **multi-step bootstrap targets**, **distributional DQN** and **noisy DQN**.
Also another implementation will be the replacement of the neural network architecture towards a CNN architecture. So the agent can be **trained by images** as input variables.

5. Literature

Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-l earning." Thirtieth AAAI conference on artificial intelligence. 2016.