Report

The problem was solved using a policy gradient algorithm, more specifically a simple "Deep Deterministic Policy Gradient" adapted for multiple agents. This is also called "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive" as introduced in the paper [1]. The algorithm is very similar to the normal DDPG, as each agent have four neural networks, the Actor, the Critic and their respective copies called "target networks". The later ones are updated slowly and progressively at every step, by keeping 99% of the current weights and adding 1% of the weights of the respective local network. While the actor is used to predict the optimal action given the state input, the critic approximates the Q-value for a state-action pair. The architectures of both networks are very similar (256 neurons in the first hidden layer and 128 neurons in the second one, ReLU activation function after each hidden layer), with just two differences:

- The input and the output sizes:

	Actor	Critic
Input Size	State size	Number of agents * (State size + Action size)
Output Size	Action size	1

- The TANH activation function is used after the output layer of the actor (as the action space is between -1 and 1)

A similar training pipeline is used for the multi-agents case: a loss is calculated given a criterion, then this is used to calculate the gradient which is then back-propagated through the network and the weights are updated using the gradients.

The pipeline begins by using the local actor to get the action for the current state. Before the action is taken, noise is added to it. This has an important role in the exploration property, as the action space is continuous. Then, the new action is taken and the rewards and the next state are observed. This transition is stored in a shared memory buffer by all the agents, from which batches are then sampled. The role of the memory buffer is use the transitions in the learning algorithms in a way that they are uncorrelated, so the network don't become unstable. Moreover, past transitions can be reused without the need to experience the state again. The memory buffer is shared, as transition experienced by other agents can be used by the rest as well.

The loss for the critic network is computed as the mean squared error between the current Q-value (obtained from the local critic network using current state) and a predicted Q-value (obtained using the Bellman equation). The predicted Q-value is calculated adding the rewards with the discounted state value of the next state (this is obtained by getting the action for the next state using the target actor network, then the Q-value using the target critic).

The loss of the actor is computed as the negative of the mean of the Q-value resulted from the local critical network given as input the current state and the actions from the local actor. At the end of each step in the pipeline, the soft-update is done on both networks.

One of the hyperparameters chosen is the discount factor, set to 0.99. This allows for future rewards to be highly considered. Another hyperparameter is the batch size. Using 128 allows the model to better generalize. The learning rates are 0.0001 for the actor and 0.001 for the critic.

The scores obtained throughout the training algorithm is presented in figure 1. As it can be seen, the score is increasing continuously until it is stopped when the desired value was obtained. There is plenty of noise in the rewards, as it can be observed that in the last training period, the score suddenly drops and then it rises again.

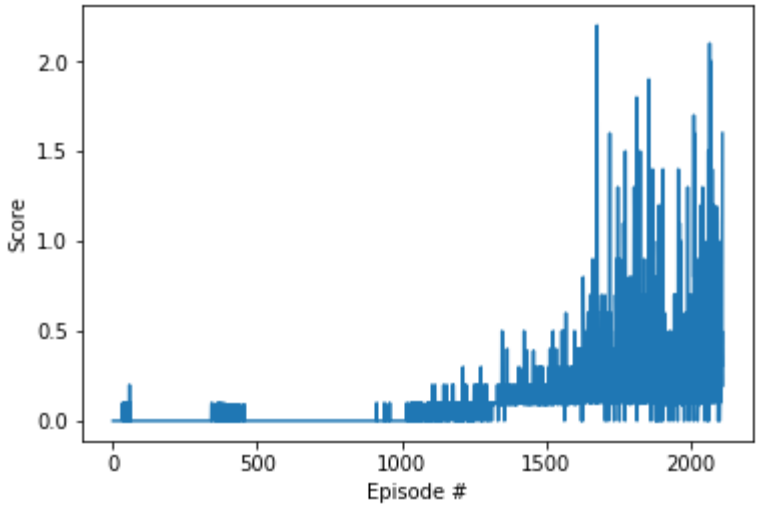

Figure 1. The scores over the episodes

Over time, the agent starts to learn the environment, thus it accumulates more positive rewards and the scores increases. It took 211 episodes in order to solve the environment with an average score of 0.511. The average score throughout the training is presented in figure 2.

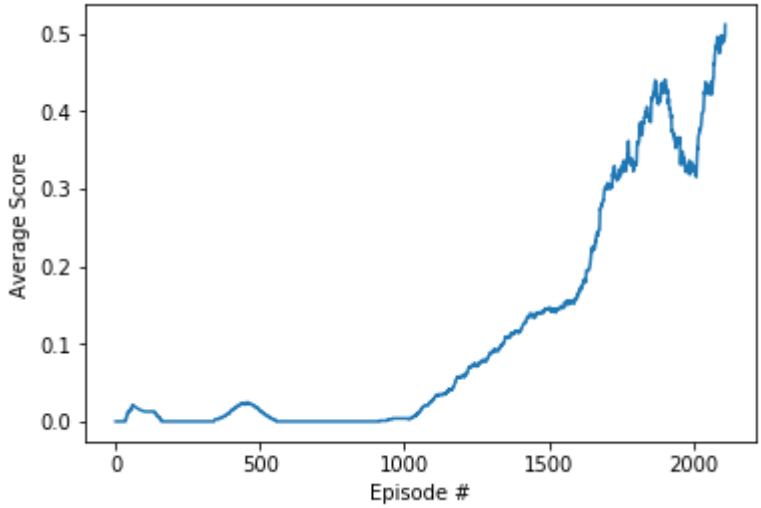

Figure 2. The average score over the episodes

One method to improve the algorithm is to implement a prioritised memory buffer, as this will speed up the training. Another improvement is to find a solution for a more stable learning curve. This will give a better estimation of the Q-value. Finally, the capacity of the network can be increased, as well as add features such as batch normalisation for an increased stability and better convergence.

REFERENCES

[1] Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Abbeel, O.P. and Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems* (pp. 6379-6390).