Report Udacity P2 Continous Control

1 Introduction

In this report I wll describe the implemenation for the P2 Continous Control project for the Udacity Nanodegree in Deep Reinforcement Learning(RL). The Algorithm itself will be explained as well as the choice of hyperparameters. The rewards of the successful agent will be documented and possible future improvement will be discussed.

2 The DDPG-Algorithm

The algorithm is a Deep Deterministic Policy Gradient algorithm with experience replay memory(ERM). It features reinforcement learning with a 2 neural networks. The Crtic Network approximates the Q-value of a given state and action and the Actor Network approximates the best action given a state. In the training process the action chosen by the actors network is the reference for the critic network in predicting the Q-value.
Both networks feature soft updating as target networks are updated only by a linear combination with local networks, weighted with factor tau. They consist of input, output, and 2 hidden layers. On each training step a batch of experiences is sampled from ERM to update the local networks using the policy gradients, performing the soft update mentioned above. Furthermore, the networks include a dropout layer and a batch normalization layer for improved and more stable learning. The Actor network also includes a noise process which is added to the action chosen by the network.

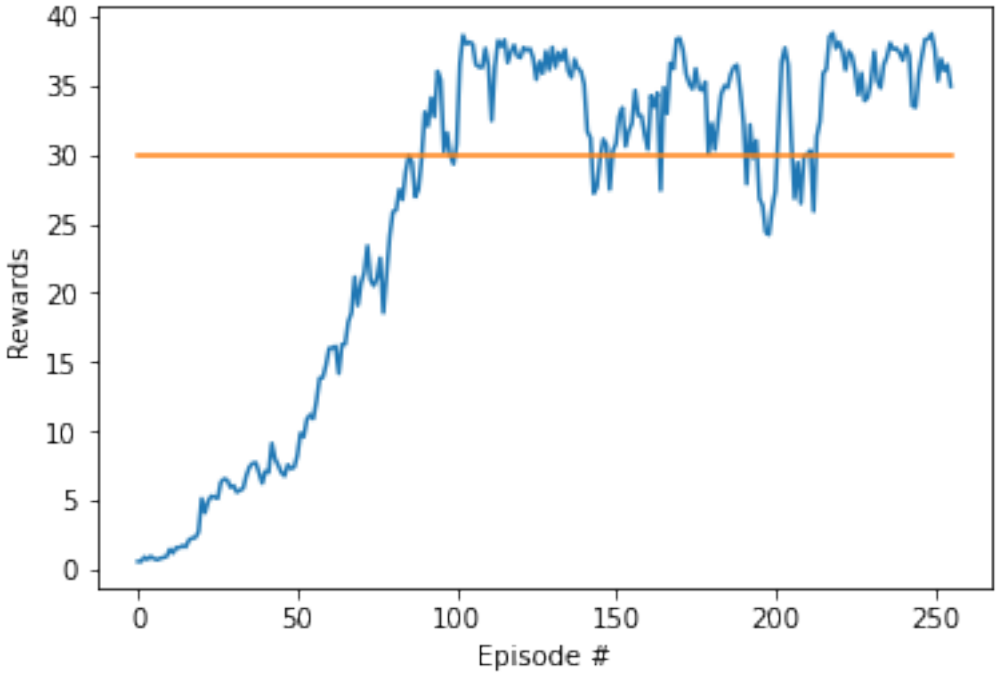
2.1 Choice Hyperparameters

I trained on various combinations of hyperparameters which are documented in *HyperParamsCheck.xlsx* and decided for hyperparameters which showed the best performance in these runs. It turned out that most parameter combinations were either learning very slow, got stuck in local optima, or degraded after more or less succesful periods. As instable rewards turned out to be the major problem, I decided for a setup that learned fast to reach episode rewards of around 30 by episode 100. After this episode I decreased the sigma parameters of the noise process and doubled the size of the ERM for the following episodes.

Name	Value	Description
gamma	0.99	Discount Factor
tau	0.001	Dampening Factor for updating Target-Networks
LR	0.0001	Learning Rate for Actor & Critic Network
batch_size	256	Batchsize of experiences drawn from replay buffer
hidden_size	256/128	Number of Neurons in first/second hidden layer
buffer_size	200k/400k	Number of elements in replay buffer
dropout	0.2	Dropout rate for Dropout Layer
Ornstein-Uhlenbeck Noise Process		
sigma	0.1/0.05	Diffusion
theta	0.15	Mean Reversion Rate
mu	0	Mean Reversion Level

3 Agents Performance

With the given hyperparameters in the main training run the agent reached the average score of 30 over 100 episodes in episode 155. The first episode reward > 30 appeared in episode 90. In the later epsiodes it stabilized on average rewards of 33-34, while successful periods where regularly interrupted by episode rewards below 30. The following plot displays the agent scores over 250 epsiodes:



4 Discussion for future Improvements

For possible improvements I would first suggest a further exploration of hyperparameters by extensive training runs over different values and prolonged periods. Learning showed to be quite vulnerable to changes in hyperparameters and most of my current exploration is based on runs of about 150 episodes or often less. This could include tests of fixed values but also different paths of hyperparameters, increasing or decreasing over time. The number of neurons in the networks showed significant impact on performance. Hence, changing the number of layers could be interesting.
Furthermore, implementation of PPO or A2C could be interesting to show how different algorithms can cope with this environment. As well Prioritized Experience Replay Memory might be very helpful. Though the main challenge would be to choose the best criterium for prioritization.