I implement Actor-Critic PPO; the implementation is based on https://github.com/rgilman33/simple-A2C-PPO. The training is organized as follows. Each episode, the agents play over 2048 frames. Then all accumulated experiences are shuffled 10 times. After each shuffling, the training stage begins. During the training, all accumulated experiences are divided into batches of size 128, and Actor and Critic losses are calculated for each batch and each agent. Further, the sum of Actor and Critic losses backpropagates for each state. To stabilize the training process, I use the following tricks:

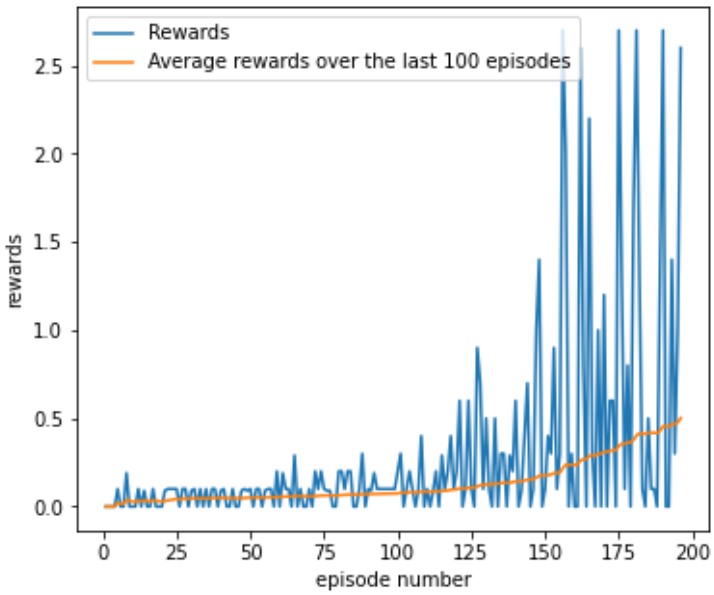
- Advantage normalization
- Gradient clipping.

After training, agents play one episode of the game, after which the reward is calculated.

Since this environment is continuous and not discrete, the classic PPO version that predicts the action probabilities is replaced with a continuous version. The continuous version is that Actor predicts the normal distribution's mean, and then the action is randomly generated. The variation vector of this normal distribution is the same for all actions and is also an optimization parameter. **It is extremely important for this environment that every action coordinate has its own variance** and not the same variance for all coordinates. Otherwise, the agents cannot achieve a score higher than 0.2.

I use the following neural network architectures. Actor and Critic neural networks have 3 layers. The first layer contains 24 neurons (the input data dimension). The second layer contains 64 neurons. The last layer contains 2 neurons (action dimension) in the case of Actor and 1 neuron for Critic. Since each action coordinate is between -1 and 1, I use the Tanh function for Actor's output.

The training process graph is shown below. The average reward over the last 100 episodes exceeds 0.5 in episode 196.



The following hyperparameter values are used:

- learning rate = 3e-4
- epsilon = 1e-5 (Adam optimizer)
- discount rate = 0.99
- tau = 0.95
- gradient clipping value = 5
- entropy coefficient = 0.

PPO is one of the most powerful algorithms to date, especially enhanced by the Actor-Critic component. However, my solution can be improved as follows:

- deeper neural network could be considered
- more advanced algorithms such as ACER could further improve performance
- frame stacking/recurrent layers could help the agents better understand motion and improve performance
- better values of the hyperparameters could be found.