I implement Actor-Critic PPO for 20 agent environment. My implementation is based on https://github.com/rgilman33/simple-A2C-PPO. The agents are trained in the following way. Each episode 20 the agents play over 2000 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 2000, and Actor and Critic losses are calculated for each batch. Further, the sum of Actor and Critic losses backpropagates for each state. To stabilize the training process, I use the following tricks:

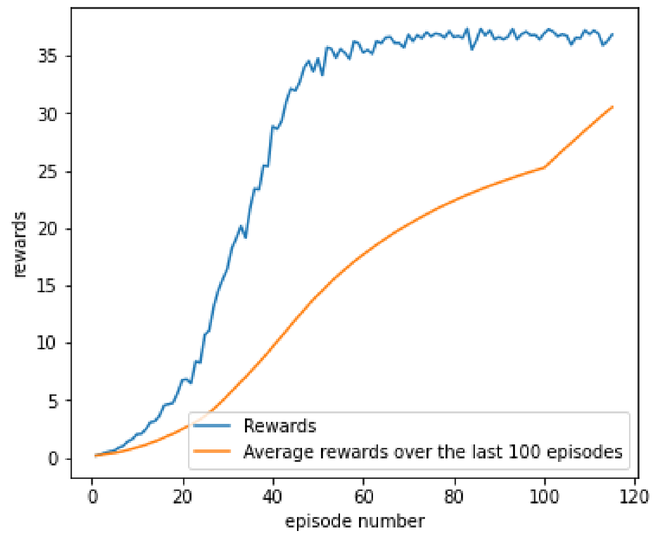
- Advantage normalization
- Gradient clipping
- Entropy loss.

After training, agents play one episode of the game, after which their average rewards are 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 of this normal distribution is the same for all actions and is also an optimization parameter.

I use the following neural network architectures. Actor and Critic neural networks have 3 layers. The first layer contains 33 neurons (the input data dimension). The second layer contains 256 neurons. The last layer contains 4 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 agent quickly learns to find the right strategy, and after 40 games starts gaining more than 30 points. The average reward over the last 100 episodes exceeds 30 in episode 114.



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 PPO modifications such as ACER could further improve performance
- better values of the hyperparameters could be found.