Deep Reinforcement Learning Project3 Report:

Collaboration and Competition

1 Description of implementation

The implementation of the algorithm is based on the DDPG algorithm implemented in the Udacity Github repo and based on DDPG paper: 'Continuous control with deep reinforcement learning' [1]. Two agents share the same DDPG networks. The algorithm is used for tennis environment in Unity environment. Instead of using policy gradient where a probability of taking each action given state is generated and action are chosen based on probability. DDPG algorithms actor network choose actions based on maximum probability, which is why it is called deterministic policy gradient.

Experience replay and fixed target network are used similar to DQN algorithm. Both actor and critic have two networks, one is fixed and the other one is used for training. Actor choose actions and critics evaluate such actions and such evaluation is used for actor to update its network. The training took place in the udacity's tennis environment.

2 Description of the learning algorithm, along with the chosen hyperparameters, model architectures for any neural networks

Learning Algorithm is based on the Udacity DDPG algorithm implemented in Pytorch, which incorporates experience replay and two networks of actor-critic network to stabilize training. Two Agents use the same network, which uses actor local network to choose action and critic network is used to evaluate state-action pairs. Agents interact with environment and gets local state, action, reward, next state and done information, and such experiences are stored in the experiences buffer. Every few steps, agent sample a small batch from experiences and such experiences are used for training based on DDPG algorithm. And actor and critic network networks will be updated. And update of the target network is based on soft update, which means a small portion of the local network is updated to the target network every few steps.

Hyperparameters

Hyperparameters	Value	Description
Number of episodes	3000	Number of episodes during for maximum training
LR_ACTOR	1e-4	Learning rate of actor network
LR_CRITIC	1e-4	Learning rate of critic network
BUFFER_SIZE	Int(1e6)	Replay memory size affects how many experiences agent can store
BATCH_SIZE	256	Mini batch size affects how large sample can be pulled from memory during learning

GAMMA	0.99	Discount factor affects how much future reward values with respect to now
TAU	0.001	How fast to soft update the parameters

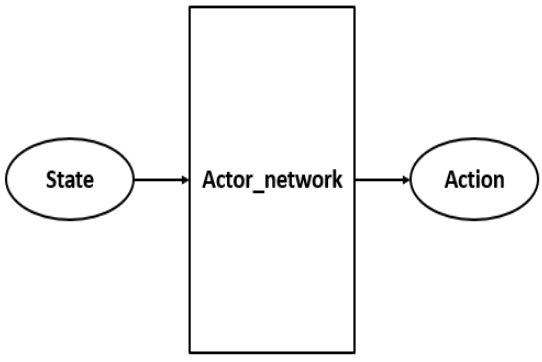
Model architecture: DDPG has actor network and critic network. And each network has a local network and target network of the same size. Actor network has state as input and has two fully connected layers of 128 size each with batch normalization and output action values with action size. Critic network has state and action as input and has two fully connected layers of 128 size each with batch normalization and outputs state-action value of the given input, which is size 1.

1 Actor Local network: State is input of the neural network, and after the second layer, it output each action values. And actor local network is used for choosing actions and updating.

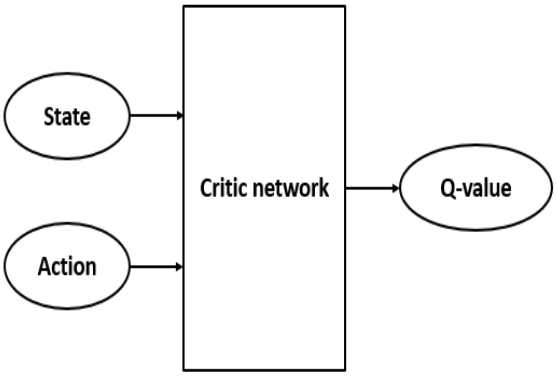
2 Actor target network: actor target has same size as actor local. And is updated after a certain period from actor local network. Its chosen action is used for evaluation of the critic network update. For a given next state, it outputs actions for next state and such value will be used for critic network TD error calculation.

3 Critic local network: is used for critic network update based on TD error.

4 Critic target network: it gathers reward plus Q value of next state, action(from actor target network) and outputs R + Q(s_, a_) and such value minus Q(s, a) from the critic local network is used for updating the critic local network.

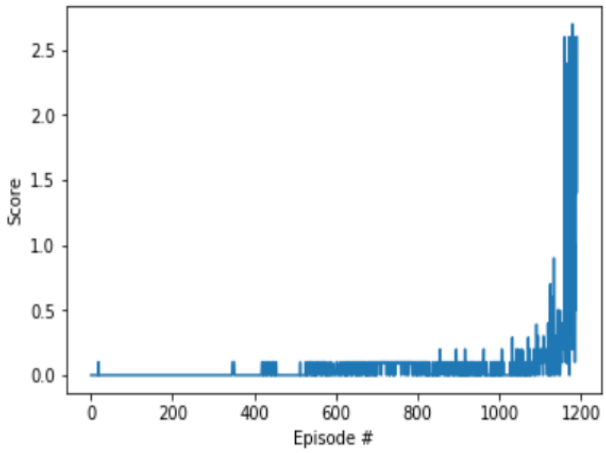


DDPG Actor Network



DDPG Critic Network

3 plot of rewards per episode is included to illustrate that the agent is able to receive an average reward (over 100 episodes) of at least +0.5.



The number of episodes needed to solve the environment is 1190 for my DDPG algorithm with the given parameters. The average reward is 0.5061 after 1190 episodes.

4 Future Work

Multi-agent DDPG algorithms could be applied to this training environment as it is special for multi-agent training. Noise factor can include something similar to epsilon-greedy, noise is initially large and gradually decrease to a small number. This could potentially lead to better training performance. Experience replay could be also another consideration for future work to speed up training.

Reference

[1] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.