Deep Reinforcement Learning Project1 Report: Navigation

1 Description of implementation

The implementation of the algorithm is based on the DQN algorithm implemented in the paper 'Human-level control through deep reinforcement learning'. Two tricks were used to improve sample efficiency and stabilize training: experience replay and fixed Q-target network. The training took place in the udacity workspace environment under Unity Navigation environment. Training model is set to True to speed up training.

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

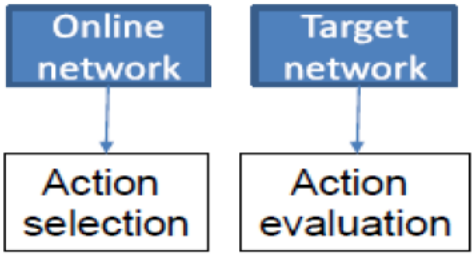
Learning Algorithm is based on the Udacity DQN algorithm implemented in Pytorch, which incorporates experience replay and fixed Q-target network to stabilize training [1]. Agents interact with environment and gets state, action, reward, next state and done information, and such experiences are stored in the experiences butter. Every few steps, agent sample a small batch from experiences and such experiences are used for training based on Q-learning algorithm. And online network will be updated. And update of the target network is based on soft update, which means a small portion of the online network is updated to the target network every few steps.

Hyperparameters

Hyperparameters	Value	Description
Number of episodes	900	Number of episodes during training
lr	0.0005	Learning rate affects how fast agents can learn
memory_size	100000	Replay memory size affects how many experiences agent can store
batch_size	64	Mini batch size affects how large sample can be pulled from memory during learning
update_frequency	4	How often to update the network
GAMMA	0.995	Discount factor affects how much future reward values with respect to now
TAU	0.001	How fast to soft update the parameters
epsilon_start	1.0	Start value of epsilon for epsilon-greedy selection

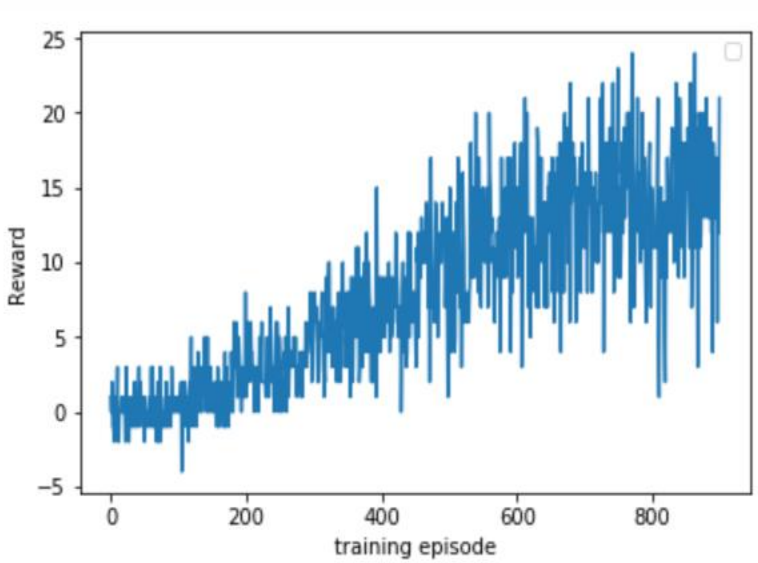
epsilon_end	0.01	End value of epsilon for epsilon greedy selection
annealing_steps	200000	Annealing step affects how fast epsilon decays

Model architecture is based on two neural networks of same size: target network and online network. They are both made of two fully connected neural networks with 64 neurons in each layer. State is input of the neural network, and after the second layer it outputs action values of each action. Target network is used for action evaluation given state. Online network and is used for training and action selection.



DQN Network Illustration

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 +13.



The number of episodes needed to solve the environment is 800 for my DQN algorithm. The average reward is 14.27 between 700-800 episodes.

4 Future Work

DQN algorithms tend to overestimate action values. future work will include implementation of double DQN which can solve overestimation problem. Also, dueling DQN have two separate estimators: state value estimation and state-dependent action advantage estimation and has been proven to lead to better learning performance and such algorithm can be implemented for future training. Moreover, prioritized experience replay can utilize more information sample information during training and is another option for future training as well.

Reference

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.