

Report

Learning Algorithm:

Our agent uses dqn learning algorithm to train itself. In dqn we train a neural network to estimate the q value for every action.

We use experience replay to avoid the harmful correlation obtained by taking samples from sequential order. We also use fixed Q targets to avoid carrot stick dangling and avoid noise.

Fixed Target

The diagram shows the DQN weight update equation:
$$\Delta \mathbf{w} = \alpha \left(R + \gamma \max_a \hat{q}(S', a, \mathbf{w}^-) - \hat{q}(S, A, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w})$$
 Annotations include: a purple curved arrow labeled 'decoupled' pointing from the target term $\hat{q}(S', a, \mathbf{w}^-)$ to the current term $\hat{q}(S, A, \mathbf{w})$; and a green arrow labeled 'fixed' pointing down to the target term $\hat{q}(S', a, \mathbf{w}^-)$.

Hyperparameters:

Epsilon is used in epsilon greedy policy the value of epsilon determines the randomness

eps_start=1.0, eps_end=0.01, eps_decay=0.995

replay buffer size = 100000

minibatch size for training = 64

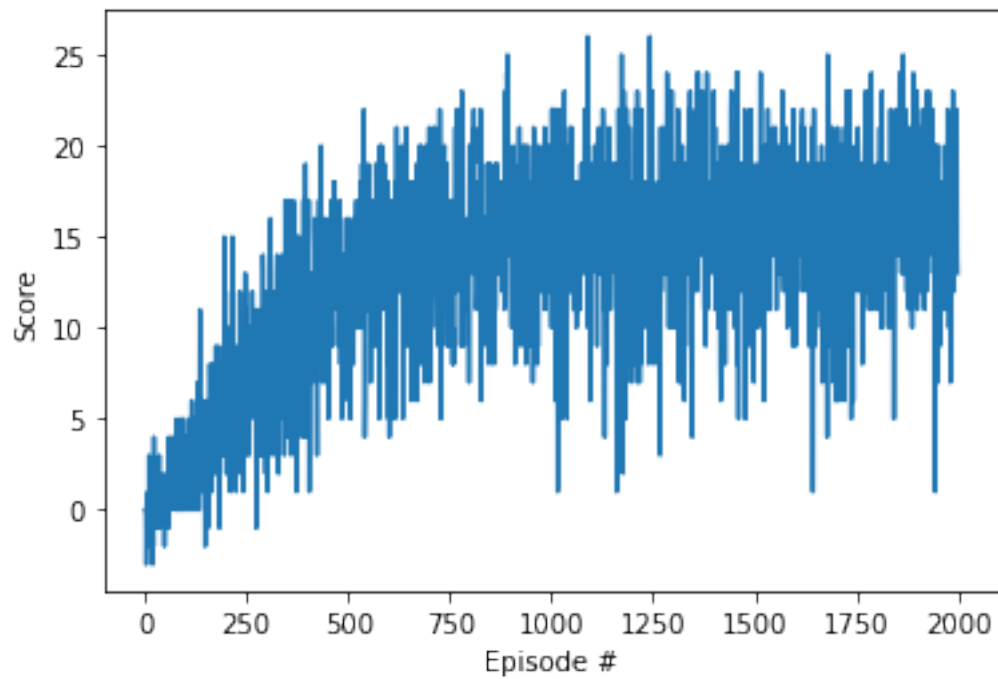
discount factor GAMMA = 0.99

for soft update of target parameters TAU = 1e-3

LR for the optimizer = 5e-4

UPDATE_EVERY = 4 steps for every learning step

plot of average rewards:



QNetwork deep learning Model:

First layer: 37 X 64

Second layer: 64 X 64

Final layer: 64 X 4