

P1 Navigation

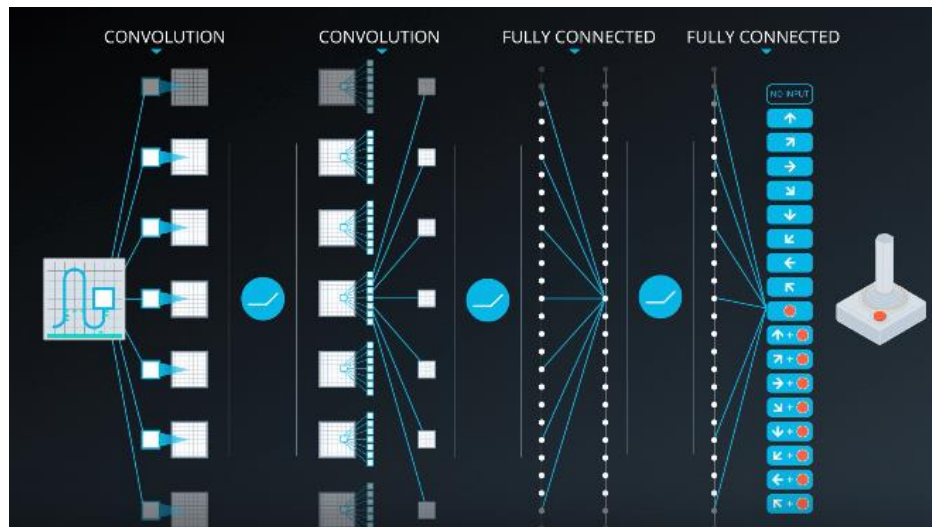
Introduction

In this project, I am going to train an agent to navigate (and collect bananas!) in a large, square world. A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of your agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available. The task is episodic, and in order to solve the environment, my agent must get an average score of +13 over 100 consecutive episodes.

DQN Network:

I am going to solve this environment using DQN network. My DQN network takes a state so no need for convolutional layers and will implement 3 linear layers where the input is the state and the final output is the action probability.



I am also deploying a replay buffer to store the tuples of $S, A, R, S+1$ to benefit from them in the network training by storing the experience.

To prevent the correlation between the target and the parameters we are changing. I am implementing local and target network with applying soft update for the parameters.

Training Procedure:

Algorithm: Deep Q-Learning

- Initialize replay memory D with capacity N
- Initialize action-value function \hat{q} with random weights \mathbf{w}
- Initialize target action-value weights $\mathbf{w}^- \leftarrow \mathbf{w}$
- **for** the episode $e \leftarrow 1$ to M :
 - Initial input frame x_1
 - Prepare initial state: $S \leftarrow \phi(\langle x_1 \rangle)$
 - **for** time step $t \leftarrow 1$ to T :

SAMPLE

Choose action A from state S using policy $\pi \leftarrow \epsilon\text{-Greedy}(\hat{q}(S, A, \mathbf{w}))$

Take action A , observe reward R , and next input frame x_{t+1}

Prepare next state: $S' \leftarrow \phi(\langle x_{t-2}, x_{t-1}, x_t, x_{t+1} \rangle)$

Store experience tuple (S, A, R, S') in replay memory D

$S \leftarrow S'$

LEARN

Obtain random minibatch of tuples (s_j, a_j, r_j, s_{j+1}) from D

Set target $y_j = r_j + \gamma \max_a \hat{q}(s_{j+1}, a, \mathbf{w}^-)$

Update: $\Delta \mathbf{w} = \alpha (y_j - \hat{q}(s_j, a_j, \mathbf{w})) \nabla_{\mathbf{w}} \hat{q}(s_j, a_j, \mathbf{w})$

Every C steps, reset: $\mathbf{w}^- \leftarrow \mathbf{w}$

Training Results:

I was able to solve the environment in 429 episodes:

