Report

Introduction

This document contains a brief report of Unity Navigation problem.

Problem Description

This environment lets the agent roam across a room full of yellow and blue Bananas. The goal is to collect (touch) yellow bananas and avoid the blue ones. A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana.

The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction. At each state the agent has 4 possible actions.

- move forward
- move backward
- turn left
- turn right

Our goal is to reach score of +13 over 100 consecutive episodes in order to consider the problem solved.

Network Architecture

To solve this problem a fully connected layer neural network has been used with on input layer, one output layer and two intermediate layers. The size of each layer is provided in the table below.

Layer	Size
Input	37
Fully connected 1	64
Fully connected 2	64
Output	4

The size of input layer and output put layer matches the state space size and action space size respectively.

Methods used

To solve this problem Deep Q-Learning is used. The agent utilizes the following methods.

- Experience replay
- Fixed Q-targets

Hyper-Parameters

The table below describes the hyper-parameters used.

Hyper-Parameter	Value
Batch size	64
Learning Rate	0.0005
Gamma	0.99
Tau	0.001

Advanced version

An advanced version of this solution has been also implemented. The following methods have been added.

- Double DQN
- Prioritized Experience Replay
- Adding Dropout layers to the network

The advanced solution can be found in the advanced branch of the same Github repository.

Results

The following result shows the training progress for a random run of the code.

Episode 100 Average Score: 0.74

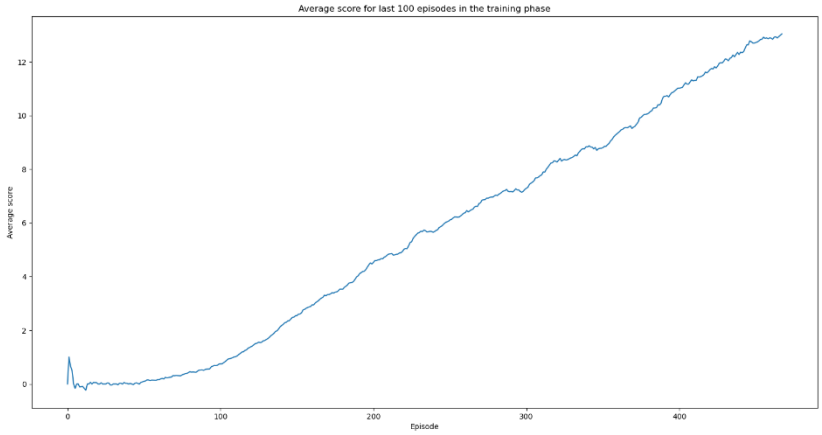
Episode 200 Average Score: 4.47

Episode 300 Average Score: 7.25

Episode 400 Average Score: 11.02

Episode 468 Average Score: 13.04

Criteria reached after 468 episodes



Ideas for future work

Although the current results show that the model is working well, there might be a few solutions to make the agent work even better or learn faster. I suggest trying the following ideas.

- Changing the network structure
    - Adding extra layers
    - Change layers size
- Try changing the hyper-parameters
- Trying Dueling DQN
- Trying Distributional DQN
- Learning from multi-step bootstrap targets
- Trying NoisyNet
- Try the Rainbow algorithm which uses a few of the mentioned ideas together