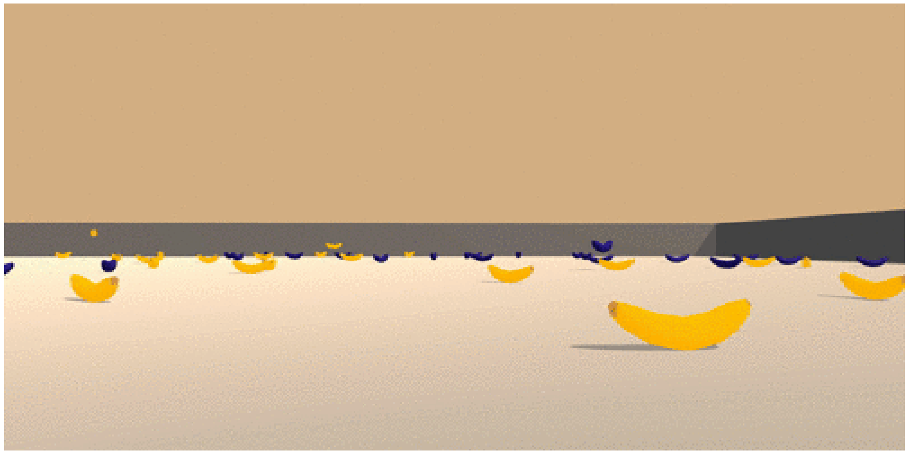
Navigation Project – Yellow Bananas

December 15, 2020 11:02 AM

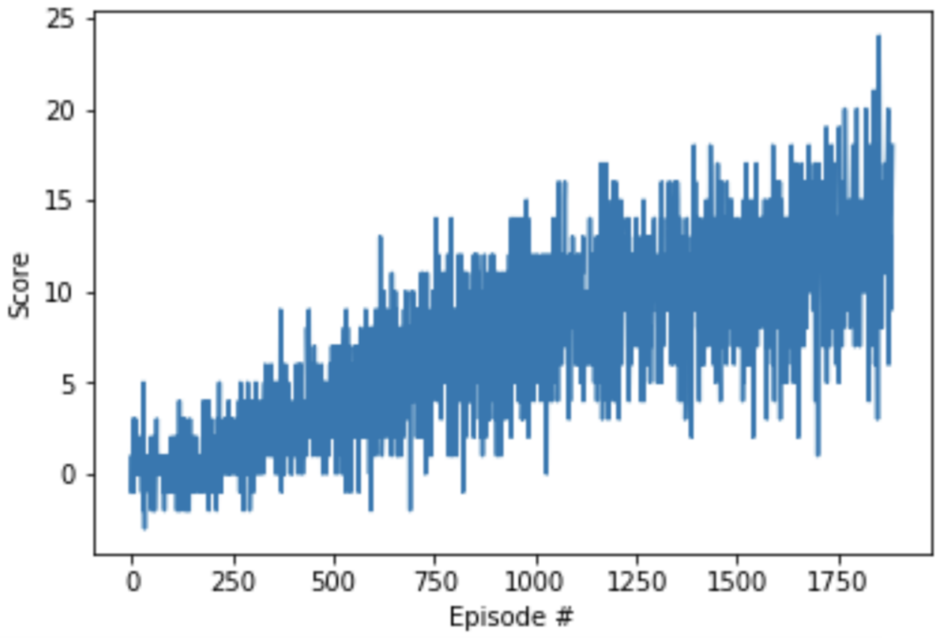
The Environment

- Project: train an agent to navigate (and collect bananas!) in a large, square world.
- Goal of Agent: Collect as many yellow bananas as possible while avoiding blue bananas.
  - A reward of +1 is provided for collecting a yellow banana
  - A reward of -1 is provided for collecting a blue banana
- State space: 37 dimensions
  - agent's velocity, ray-based perception of objects around the agent's forward direction
- Action Space (Discrete): 4 Actions
  - 0 - forward
  - 1 – backward
  - 2 – left
  - 3 – right
- The task is episodic, and in order to solve the environment, your agent must get an average score of +13 over 100 consecutive episodes.
- Note: The project environment is similar to, but **not identical to** the Food Collector environment on the Unity ML-Agents GitHub page.



Instructions

- For this project, you can use any algorithm of your choosing to solve the task.
- Strongly encouraged to do your own research, to devise your own approach towards solving this problem
- Should be able to solve the project by making only minor modifications to the DQN code provided as part of the **Deep Q-Networks** lesson.
- Please see the image below for an example of how you might expect your agent's score to evolve.



- How long it should take: we were able to solve the project in fewer than 1800 episodes.

Where to Start

1. Master the details of Deep Q-Networks (DQN)

- Read the **DQN paper** **to master all of the details. Refer to the lesson on Deep Q-Networks** to cement your understanding.

2. Study the coding exercise from the lesson.

- In the **Deep Q-Networks** lesson, you applied a DQN implementation to an OpenAI Gym task. Take the time to understand this code in great detail.
- Tweak the various hyperparameters and settings to build your intuition for what should work well (*and what doesn't!*).

3. Adapt the code from the lesson to the project.

- Adapt the code from the exercise to the project, while making as few modifications as possible. (*Remember that the code that you use to interact with the Unity environment is different from the OpenAI gym interface.*)
- Don't worry about efficiency, and just make sure the code runs. Don't worry about modifying hyperparameters, optimizers, or anything else of that nature just yet.
- You do not need to run your code on a GPU

4. Optimize the hyperparameters.

- After you have verified that your DQN code runs, try a few long training sessions while running your code on CPU.
- If your agent fails to learn, try out a few potential solutions by modifying your code.

Submission Checklist

- a **README** that describes how someone not familiar with this project should use your repository.
- the **code** that you use for training the agent, along with the trained model weights.
- a **report** describing your learning algorithm.