Navigation Project
Deep Reinforcement Learning Nanodegree

Nontawat Pattanajak

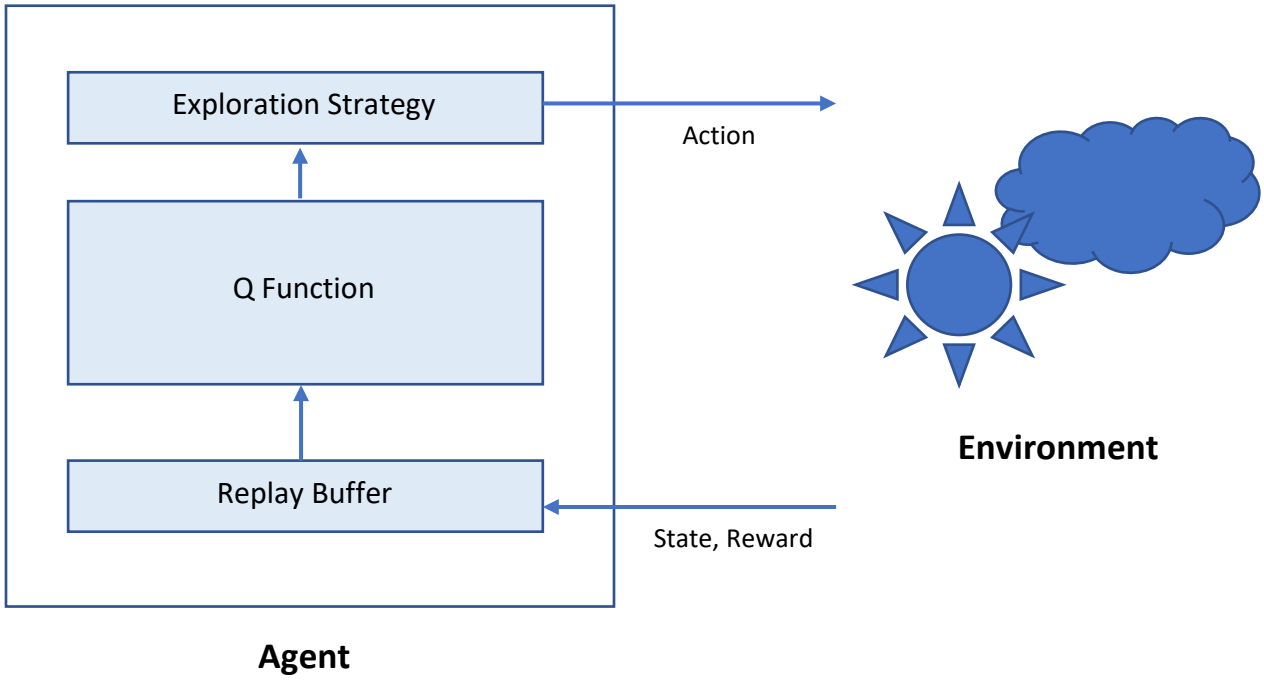
23 December 2020

1. Introduction

This project aims to develop an agent to navigate to collect yellow bananas. The environment consists of 37 states which are agent's velocity and ray-based perception of object. The agent learns environment to make an action from 4 possible actions such as move forward, move backward, turn left, and turn right, to navigate to a yellow banana. This task is episodic. In order to develop a successful agent, the agent must get an average score more than 13 for over 100 consecutive episodes.

2. Algorithm

An agent in this project is developed by one of the most successful algorithms in Deep Reinforming Learning called Deep Q-Network (DQN) [1] introduced by Google DeepMind in 2015. DQN implemented in this project consists of 3 sections such as Replay Buffer, Q-Function, and Exploration Strategy as the diagram below.



Picture 1: Process inside the agent

2.1 Replay Buffer

This section is used to collect experience of the agent when encountering the environment. Five data are stored such as `state`, `action`, `reward`, `next_state`, and `done` consecutively until they reach the limitation of buffer. Then Replay Buffer randomly selects the data to generate batch data, feeding to Q Function.

2.2 Q Function

In traditional Reinforcement Learning, Q Function is generated by a reward table calculated from action and state. However, it is unlikely to implement when the environment is a large state space. In this project, Artificial Neural Network (ANN) is chosen to replace Q Function. ANN learns data fed from Reply Buffer to create approximation function (or non-linear function) to calculate relationship (function) between state and action.

2.3 Exploration Strategy

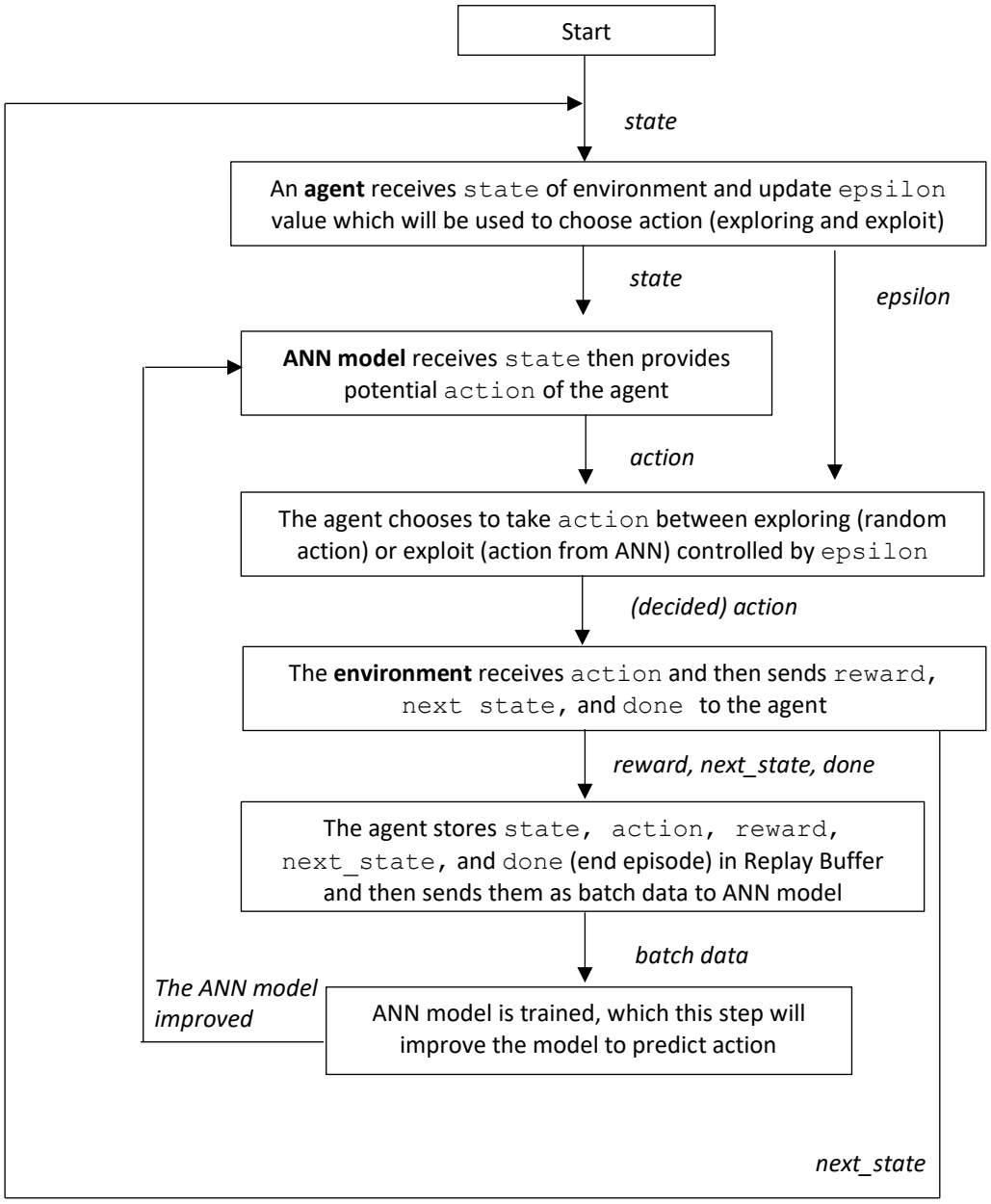
Exploration Strategy is a section to tell an agent which action should be taken between exploring and exploiting. In the early episodes, the agent should to explore rather than to exploit. After the agent learn (late episodes), the agent tends to choose to exploit. Choosing between exploring and exploiting is controlled by epsilon. At the early episodes, the epsilon is high value (such as 1). After that (in the late episodes), the epsilon is nearly 0.

3. Method and Implementation

The algorithm for building an agent is developed by PyTorch. The table below contains hyperparameters used in the development of this agent, following to the diagram showing how the algorithm works.

Table 1: Agent Hyperparameters

Hyperparameter	Detail	Value
BUFFER_SIZE	replay buffer size	100000
BATCH_SIZE	minibatch size (output from Replay Buffer)	64
GAMMA	discount factor	0.995
TAU	for soft update of target parameters	0.001
LR	learning rate	0.0005
UPDATE_EVERY	how often to update the network	4
epsilon_start	Epsilon for exploring in the early episode	1.0
epsilon_end	Epsilon for exploring in the late episode	0.01
epsilon_decay	Rate of changing Episilon	0.995



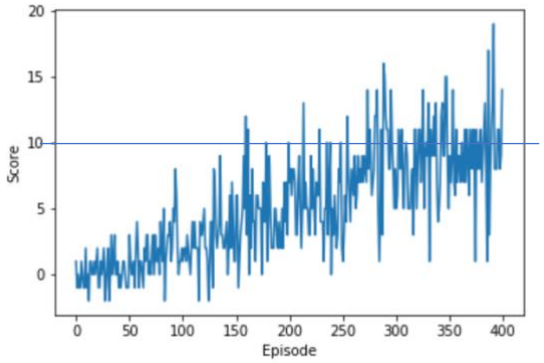
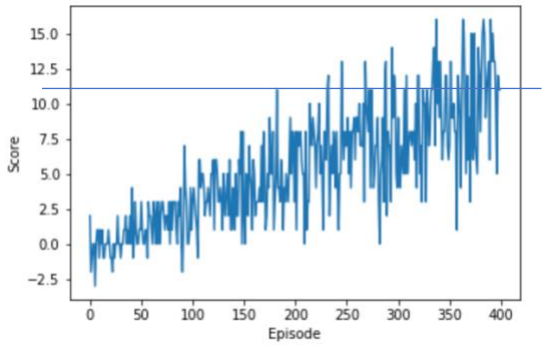
Picture 2: How the agent learns

4. Result and Discussion

4.1 Considering a model architecture

In the early state of this project, there are two architectures of an ANN model which (1) using 1024 nodes for hidden layer 1 and 2, and (2) using 1024 nodes for hidden layer 1 and 512 hidden layer 2. The result shows that the model with architecture (2) provides slightly less episodes to make a high reward (average score) than model (1). It can be also stated that model (2) provides slightly higher score at the same episode. In addition to this, smaller number of nodes could require less computational expense. Therefore, model (2) is the selected architecture.

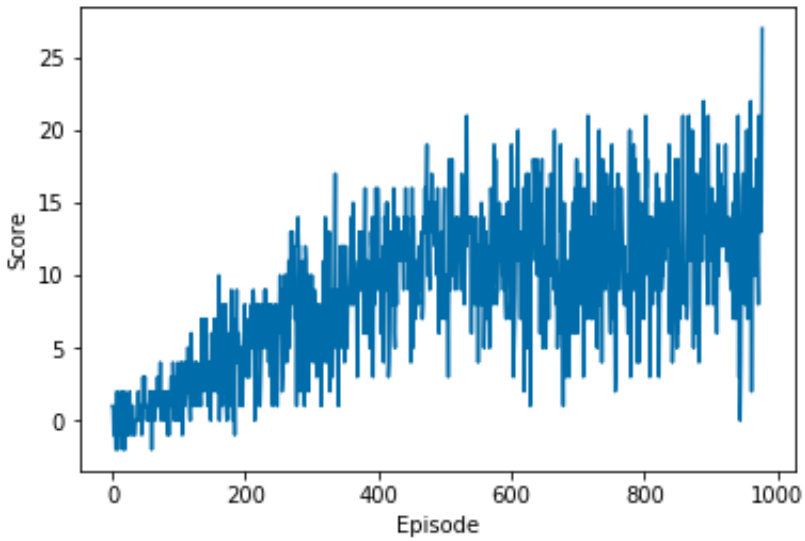
Table 2: Score comparing between model 1 and 2

Model 1 1024 nodes for hidden layer 1 and 2	Model 2 1024 and 512 nodes for hidden layer 1 and 2 respectively
	

4.2 The experimental result of training the agent

Model Architecture 2 is selected to develop the agent. The result shows that the agent requires 978 episodes to reach the average score 13.

```
Episode 100    Average Score: 0.7200
Episode 200    Average Score: 3.6200
Episode 300    Average Score: 6.8500
Episode 400    Average Score: 8.5800
Episode 500    Average Score: 11.1800
Episode 600    Average Score: 11.4100
Episode 700    Average Score: 11.4600
Episode 800    Average Score: 11.6500
Episode 900    Average Score: 12.4600
Episode 978    Average Score: 13.1200
Environment solved in 978 episode with Average Score: 13.12
```



Picture 3: Score of training the agent by episode

5. Conclusion and Future Work Suggestion

The agent in this project is developed by using Deep Q Network and having Replay Buffer for the improvement. The Q Function is provided by Artificial Neural Network with having two hidden layers (1024 and 512 nodes for hidden layer 1 and 2 respectively). The result of training an agent shows that the agent requires 978 episodes to reach the average score at 13 while benchmark requires approximately 1800 episodes to reach this level.

To sum up, this agent requires less episodes to train to reach the excepted score when comparing to the benchmark. However, there is some room for the improvement such as applying Double DQN, Priority Experience Replay, and Dueling DQN.

References

[1] Volodymyr Mnih et al, Human-level control through deep reinforcement learning, https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf