Report

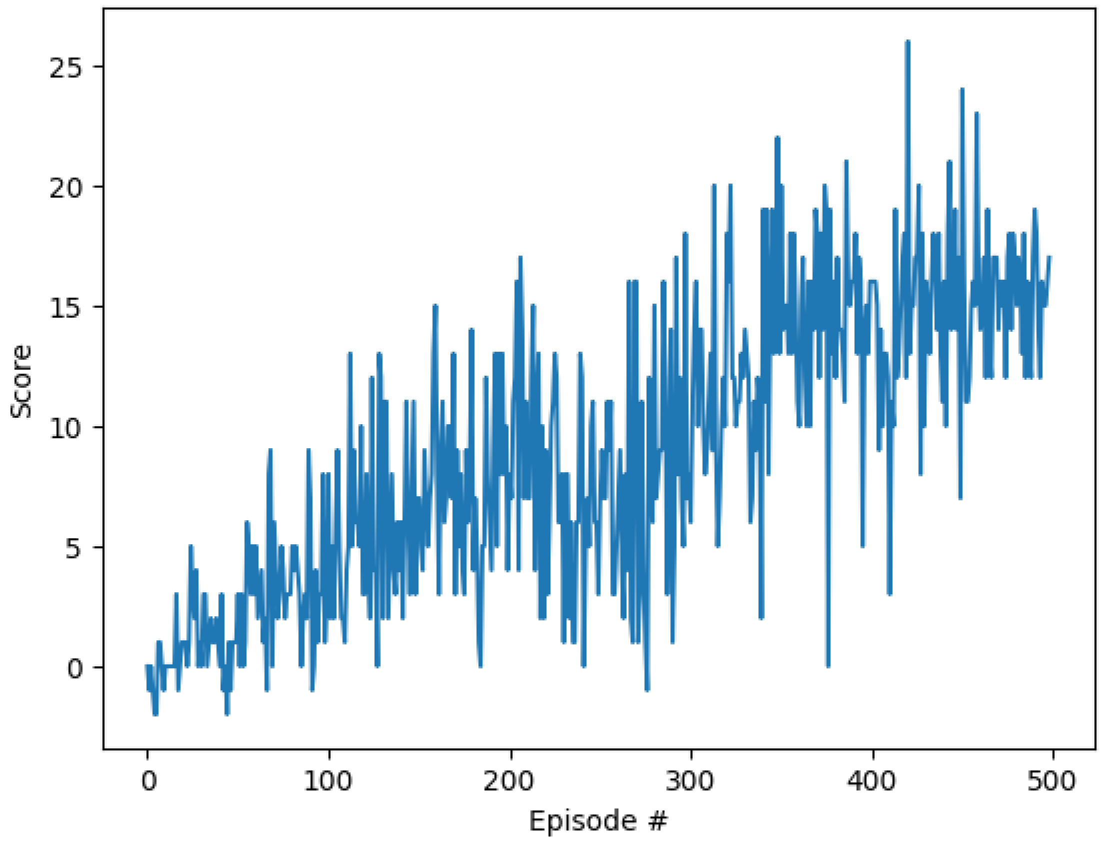
The project is about designing a Deep Reinforcement Learning (DRL) agent to navigate in an Environment to grasp yellow bananas (+1) as much as possible while avoiding blues ones (-1) in a time interval.

The state space is defined based on 37 dimensions. There are four discrete actions that set the direction of movement including moving forward, moving backward, turn left, and turn right.

I designed and evaluated several variants of neural networks models based on the number of layers and their nodes. Also, variants of DQN models, and types of experience replay buffer.
The final neural network model that is shown has **around 9 thousands parameters, 9301,** and four layers.

The DRL is based on Dueling DQN, and its experience replay buffer is prioritized sampling based sumtree. The size of the prioritized experience replay buffer is 1e6.. The epsilon has two levels of decrease rate.

The training of the model is shown in the following figure and description.



```
Episode 100	Average Score: 1.96,eps: 0.08597780270477876
Episode 200	Average Score: 6.61,eps: 0.03147065642155796
Episode 300	Average Score: 7.82,eps: 0.011519278051388249
Episode 400	Average Score: 13.35,eps: 0.01
```

Environment solved in **399 episodes! Average Score: 15.01**

The weights of the model are based on 399 episodes. In a test condition, the model leads to an average **15.8 in 20 runs.**

There are several files including model_dueiling_DQN, dqn_agent, and Navigation. Also, the weights of the model is attached as a file, “checkpoint.pth”. The checkpoint captures the weights of the model.

File:========================== **model_dueiling_DQN.py**
The model consists of 4 layers of NN with Relu activation functions. The model is based on Dueling DQN.
The number of weights is 37*(64+1)+64*(64+1)+64*(32+1)+32*(16+1)+16*(4+1)=9301 that is less than 10,000.
File:=========================== **dqn_agent_duel_pri_exp.py**
The class "Memory" includes functions to add, store, update, and sample repositories. I used a prioritized experience replay that is based on "sumtree".
I used an existing library, "starr", and one of its classes, SumTreeArray, that provides "sumtree" to update and store experiences very fast.
The class "Agent" includes making a pair of identical neural networks for the base and target.
The learning mechanism calculates errors for updating the base network and soft updating the target network.
File:============================**Navigation**
The environment, the agent, and the interaction initiated and ran through that.
File:============================**checkpoint.pth**
The weights are the neural network of the model in 399 episodes.