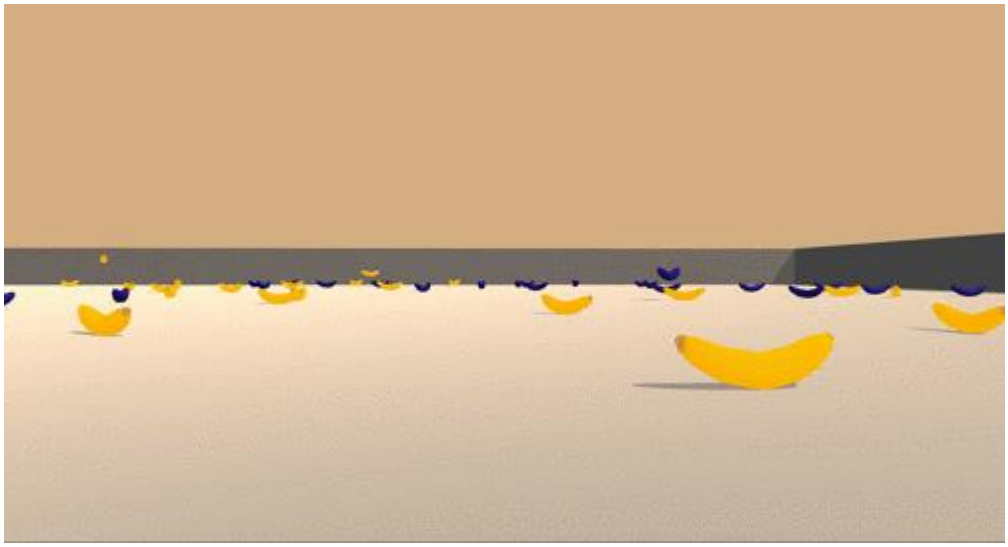
Banana Frenzy Agent

Project's goal:

To Train an agent to navigate to a virtual world and collect bananas as much as possible and ignore the blue bananas.



Environment

The environment is based on Unity ML-agents

Reward:

+1 reward = if agent collect a yellow Banana -1 reward = if agent collect blue Banana

State Space:

There are 37 dimensions in which agent can move with velocity along with ray-based perception of objects around agent's forward direction.

Actions:

Four discrete actions are available,

0 - move forward

1 - move backward

2 - turn left

3 - turn right.

Pseudo Code for Deep Q-Learning:

```
Initialize Q(s,a)for all pairs (s,a)
s = initial state
k = 0
While Loop(Still convergencing)
{
    Iterate action a and reach state s'
    if(s' is a terminal state)
    {
        target = R(s,a,s')
    }
    else
    {
        target = R(s,a,s') + Gamma MAXa1 Q(s',a')
       Loss updated
    }

    s = s'
}
```

We will be using two neural networks , Primary neural network and Target neural network(Fixed Q-targets) . Because We saw in the Deep Q Learning article that, when we want to calculate the TD error (aka the loss), we calculate the difference between the TD target (Q_target) and the current Q value (estimation of Q therefore there is a big correlation between the TD target and the parameters (w) we are changing).

$$\Delta w = \alpha[(R + \gamma \, max_a \, \hat{Q}(s', a, w)) - \hat{Q}(s, a, w)] \, \nabla_w \hat{Q}(s, a, w)$$

Change in weights

learning rate

Maximum possible Qvalue for the next_state (= Q_target)

Current predicted Q-val

TD Error

Gradient of our current predicted Q-value

Double DQNs:

To get the best action of the next state, sometimes Highest Q-value is biased. To overcome this, we compute the Q target, we use two networks to decouple the action selection from the target Q value generation. Then DQN Network to select the best action for the next state use our target to calculate the Q-value for that action at the next state.

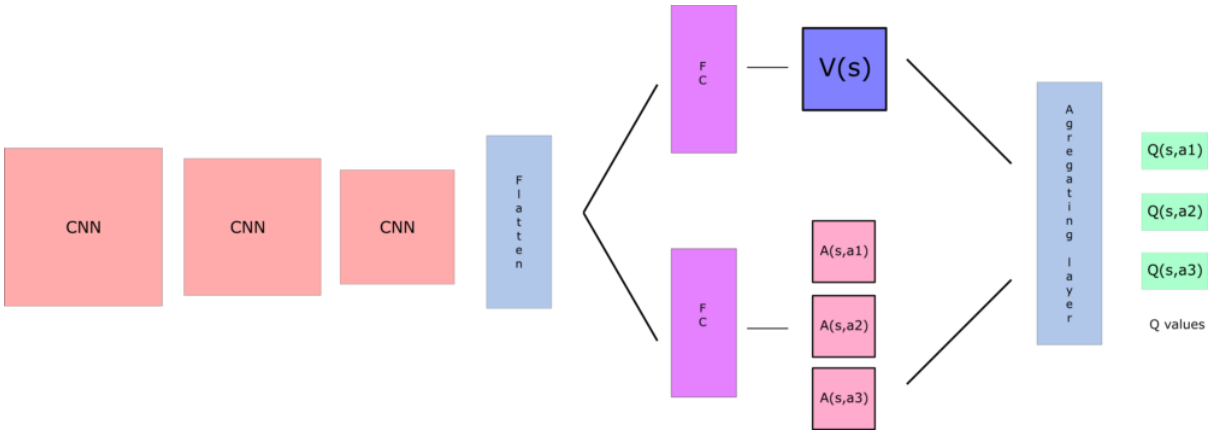
Dueling DQN (aka DDQN):

To improve the taking an action at that state Q(s,a) further.we can decompose Q(s,a) as the sum of:

V(s): the value of being at that state

A(s,a): the advantage of taking that action at that state (how much better is to take this action versus all other possible actions at that state).

i.e Q(s,a) = A(s,a) + v(s)



Further enhancements:

- Try different architectures with different weight initializations methods and different parameters
- Enhanced Prioritized Experience Replay (PER)

References:

https://cugtyt.github.io/blog/rl-notes/201807201658.html#:~:text=Fixed%20Q%2Dtargets,-We%20saw%20in&text=Using%20the%20Bellman%20equation%2C%20we,target%20and%20the%20Q%20value.