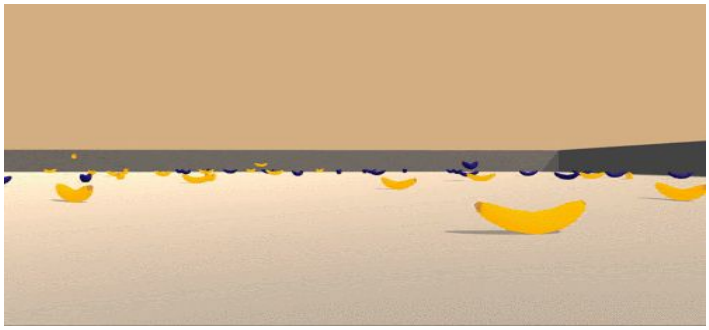


Project's goal

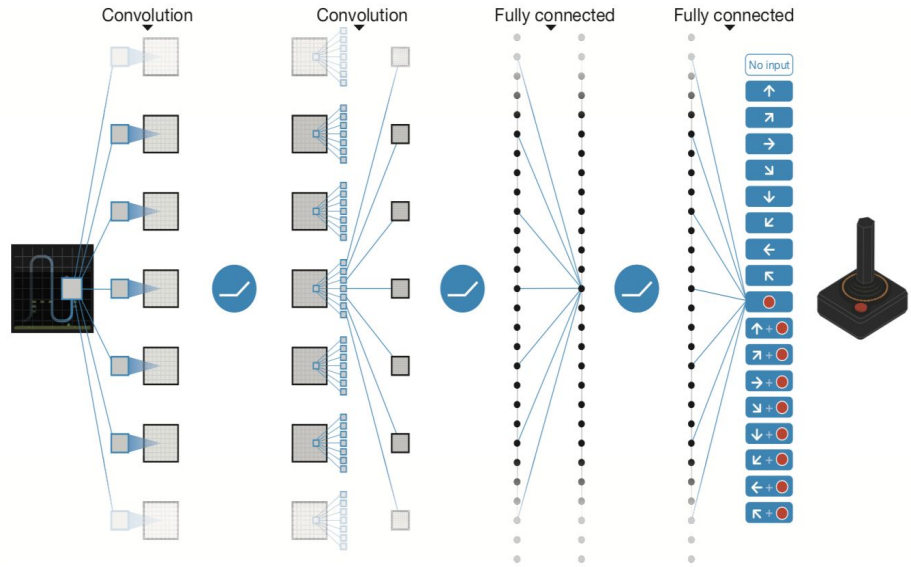
This project is for train an agent to collect yellow bananas while avoiding blue bananas in a virtual environment.

A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of your agent is to collect as many yellow bananas as possible while avoiding blue bananas.

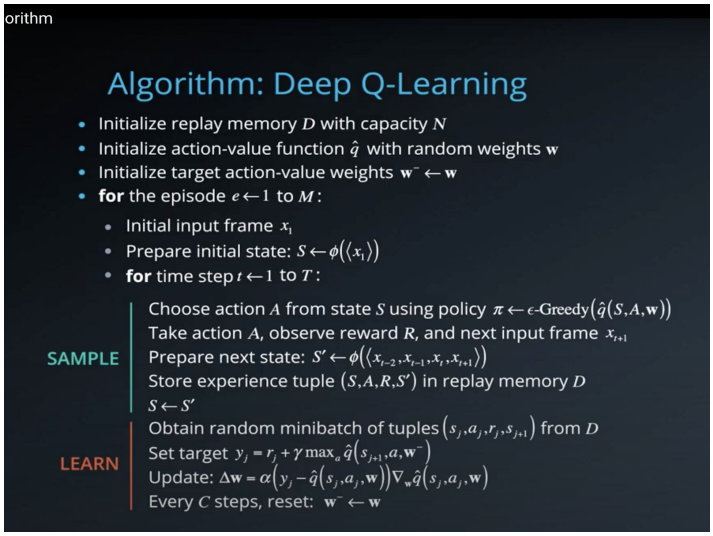
Learning Algorithm and Agent Implementation

Deep Q-Networks(DQL) is used for the train an agent.

Illustration of the DQL:



Algorithm of the Deep Q-Learning:



DQN parameters and results:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR = 5e-4               # learning rate
UPDATE_EVERY = 4        # how often to update the network
```

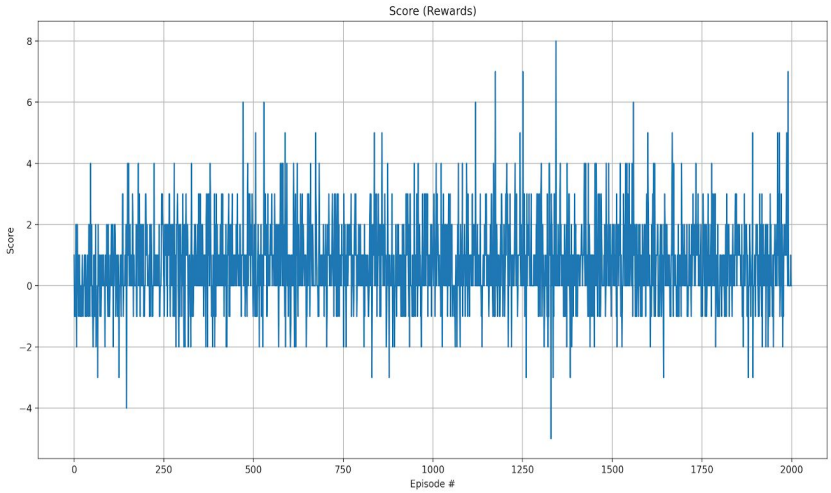
```
agent = Agent(state_size=37, action_size=4, seed=42)
```

```
# Train the agent using DQN
start_time = time.time()  # Monitor Training Time
scores = dqn(n_episodes=2000, max_t=1000, eps_start=1.0, eps_end=0.01, eps_decay=0.995)
print("\nTotal Training time = {:.1f} min".format((time.time() - start_time) / 60))
```

```
Unity brain name: BananaBrain
        Number of Visual Observations (per agent): 0
        Vector Observation space type: continuous
        Vector Observation space size (per agent): 37
        Number of stacked Vector Observation: 1
        Vector Action space type: discrete
        Vector Action space size (per agent): 4
        Vector Action descriptions: , , ,
Number of agents: 1
Number of actions: 4
States look like: [1.         0.         0.         0.         0.84408134 0.
 0.         1.         0.         0.0748472  0.         1.
 0.         0.         0.25755    1.         0.         0.
 0.         0.74177343 0.         1.         0.         0.
 0.25854847 0.         0.         1.         0.         0.09355672
 0.         1.         0.         0.         0.31969345 0.
 0.        ]
States have length: 37
Score: 0.0
Episode 100 Average Score: 0.07
Episode 200 Average Score: 0.62
Episode 300 Average Score: 0.82
Episode 400 Average Score: 0.69
Episode 500 Average Score: 0.90
Episode 600 Average Score: 0.99
Episode 700 Average Score: 0.81
Episode 800 Average Score: 0.88
Episode 900 Average Score: 0.67
Episode 1000    Average Score: 0.85
Episode 1100    Average Score: 0.75
Episode 1200    Average Score: 0.83
Episode 1300    Average Score: 0.99
Episode 1400    Average Score: 0.76
Episode 1500    Average Score: 0.82
Episode 1600    Average Score: 1.01
Episode 1700    Average Score: 0.70
Episode 1800    Average Score: 0.81
Episode 1900    Average Score: 0.59
Episode 2000    Average Score: 0.82

Total Training time = 40.6 min
```

Plot of Rewards



Conclusion and Ideas for Future Work:

Trained agent with DQN, better than agent which decide randomly

For the future work several improvements to the original Deep Q-Learning algorithm have been suggested in the classroom. These are:

1.Double DQN
Deep Q-Learning tends to overestimate action values. Double Q-Learning has been shown to work well in practice to help with this.

2.Prioritized Experience Replay
Deep Q-Learning samples experience transitions *uniformly* from a replay memory. Prioritized experienced replay is based on the idea that the agent can learn more effectively from some transitions than from others, and the more important transitions should be sampled with higher probability.

3.Dueling DQN

Currently, in order to determine which states are (or are not) valuable, we have to estimate the corresponding action values *for each action*. However, by replacing the traditional Deep Q-Network (DQN) architecture with a dueling architecture, we can assess the value of each state, without having to learn the effect of each action.