Machine Learning

Deep Reinforcement learning Project

Snake Game Using DRL

Snake Game using Reinforcement Learning

Introduction

Reinforcement learning is an area of machine learning concerned with software agents ought to take actions in an environment in order to maximise the notion of cumulative reward. **Deep reinforcement learning** (**deep RL**) is a subfield of machine learning that combines reinforcement learning (RL) and deep learning.

RL considers the problem of a computational agent learning to make decisions by trial and error. Deep RL incorporates deep learning into the solution, allowing agents to make decisions from unstructured input data without manual engineering of the state space. Deep RL algorithms are able to take in very large inputs (e.g. every pixel rendered to the screen in a video game) and decide what actions to perform to optimize an objective (e.g. maximizing the game score).

In this assignment deep reinforcement learning has been used to play snake game.

Overview:

This assignment consists of 3 parts namely game, agent and model. Snake game has been developed using pygame. Agent is the soul of the assignment as it is learning and playing the game using the Model. Model has been developed using pytorch Library.

Game (Pygame)

- Play_step(action)
- Output – reward, game_over, score

Agent

- Game
- Model

Training:

State = get_state(game)

Action = get_move(state)

➔Model.predict()

Reward, game_over,score = game.play_step(action)

Remember

Model Train

Model(Pytorch)

- Linear _QNet(DQN)
- Model.predict(state)

➔ action

STATES

- Collide Straight
- Collide Right
- Collide Left

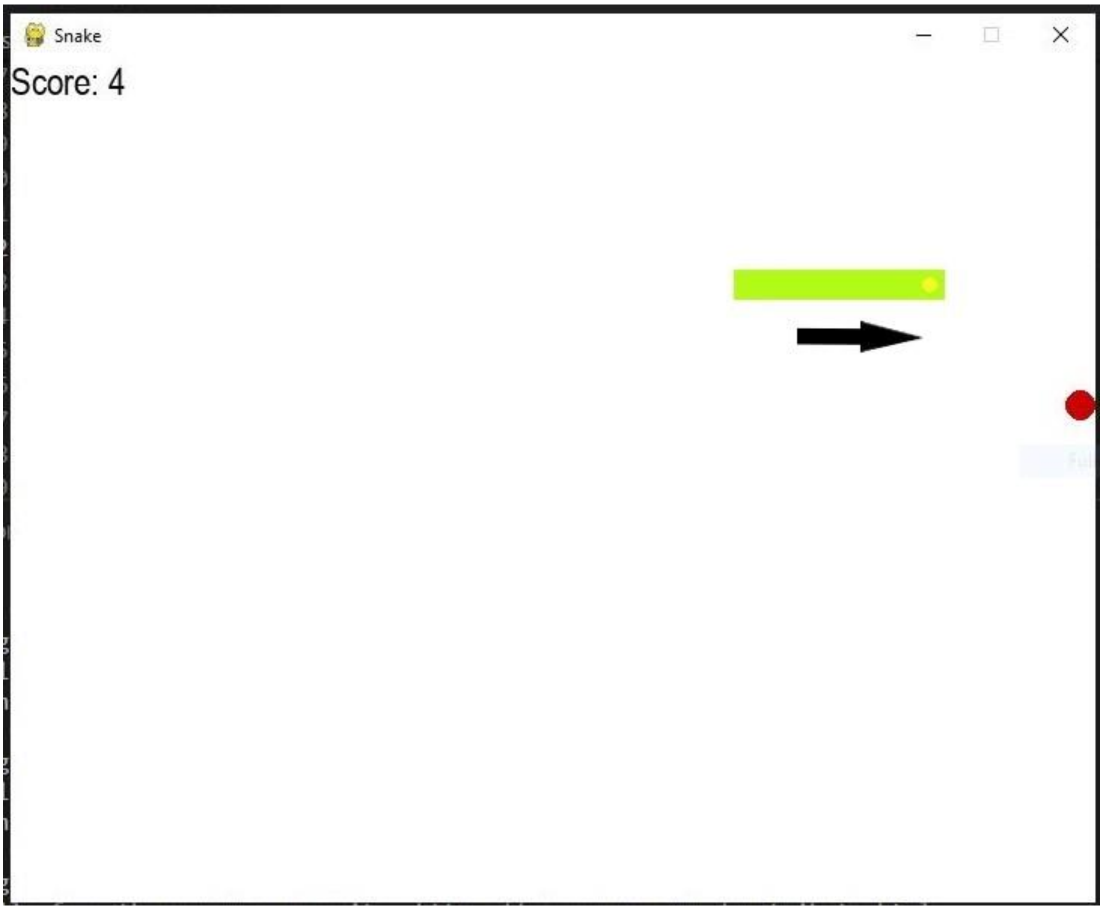
- Direction Left
- Direction Right
- Direction Up
- Direction Down

- Food Right
- Food Left
- Food up
- Food Down

Actions

- [1 , 0 , 0] - Straight
- [0 , 1 , 0] - Right Turn
- [0 , 0 , 1]   - Left Turn

Example of State:



Current State : [0 , 0 , 0

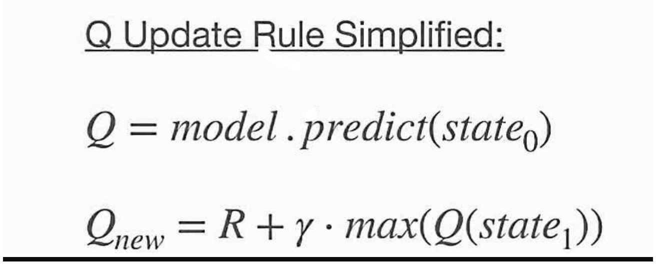
0 , 1 , 0 , 0

1 , 0 , 0 , 1]

REWARD

- Eat Food : +10
- Collide : -10
- Else : 0

Simplified Bellman Equation:

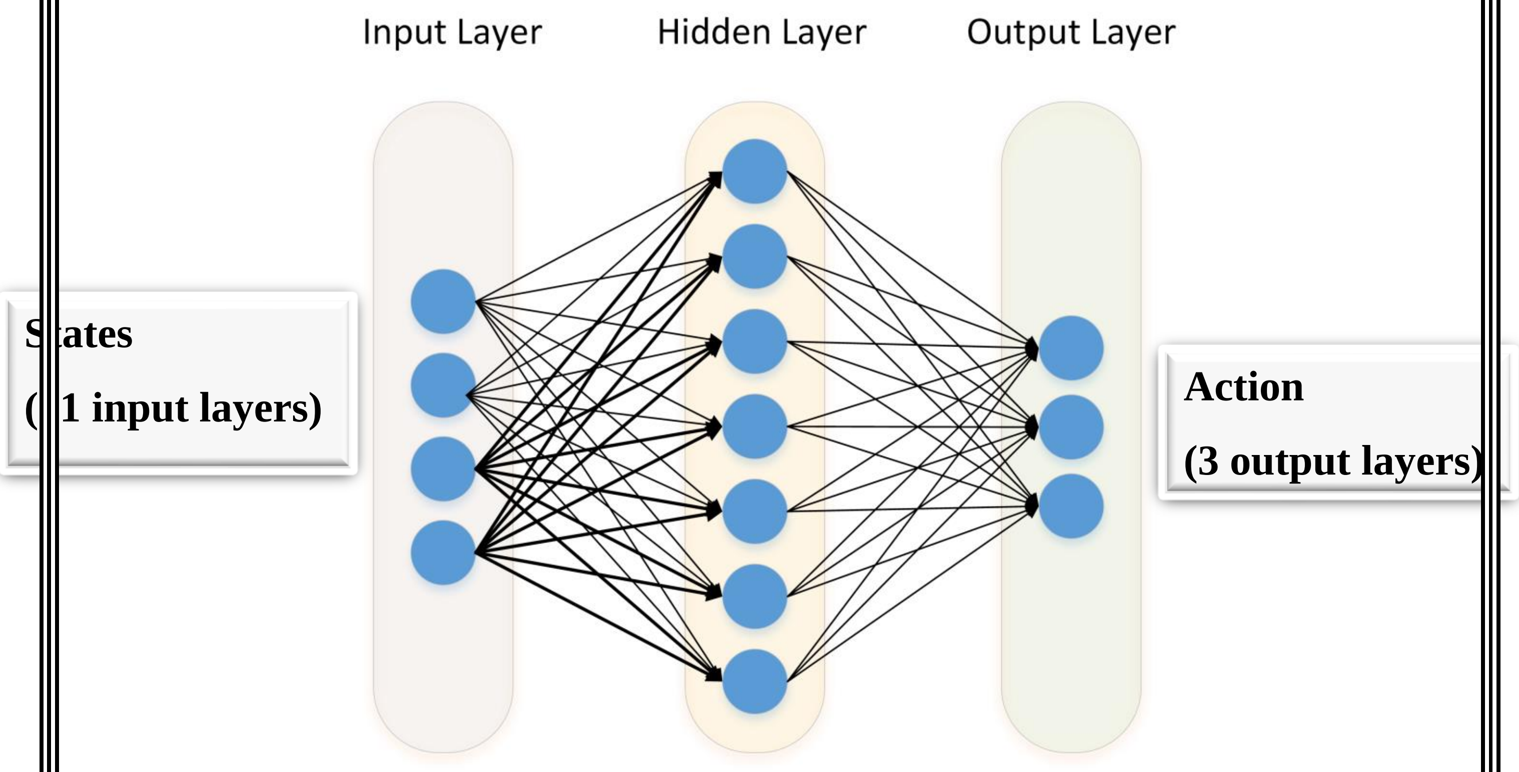


Q Update Rule Simplified:

$$Q = model\,.\,predict(state_0)$$

$$Q_{new} = R + \gamma \cdot max(Q(state_1))$$

MODEL:



States

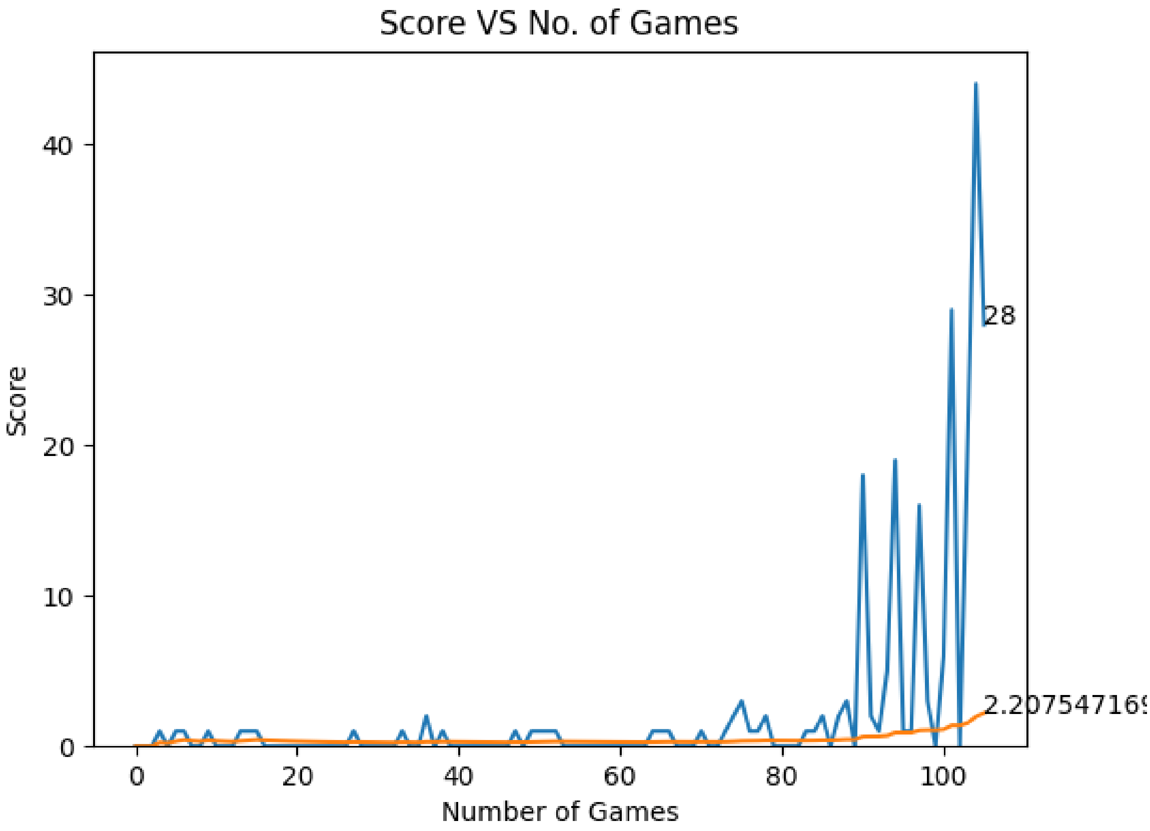
(11 input layers)

Action

(3 output layers)

Games vs Score

As the snake keeps on learning its score improve as well it is depicted in the graph below as the no. of games are increasing the mean as well the highest score is also increasing.



Conclusion

Conclusions Reinforcement learning is an effective means for adapting neural networks to the demands of many tasks. However, reinforcement-learning algorithms become much more powerful when they can take advantage of the contributions of a trainer.