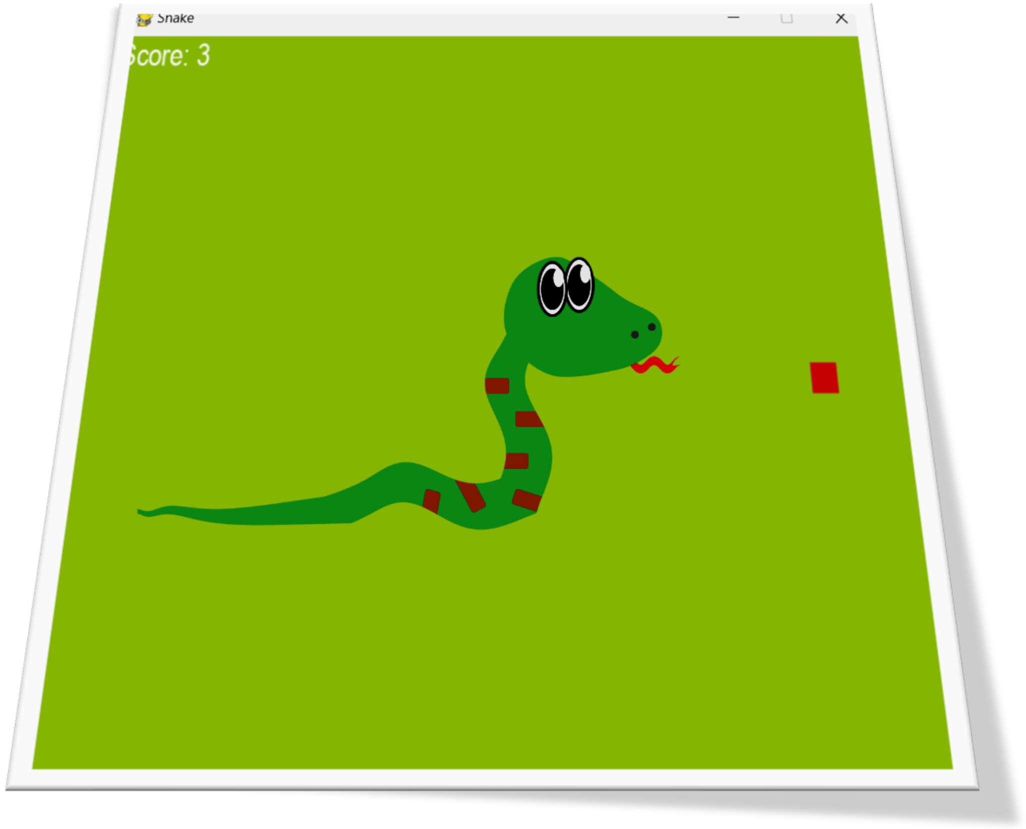
Snake Game Using Deep Reinforcement Learning



Abdullah Al Forkan
Matriculation Number: 2190314

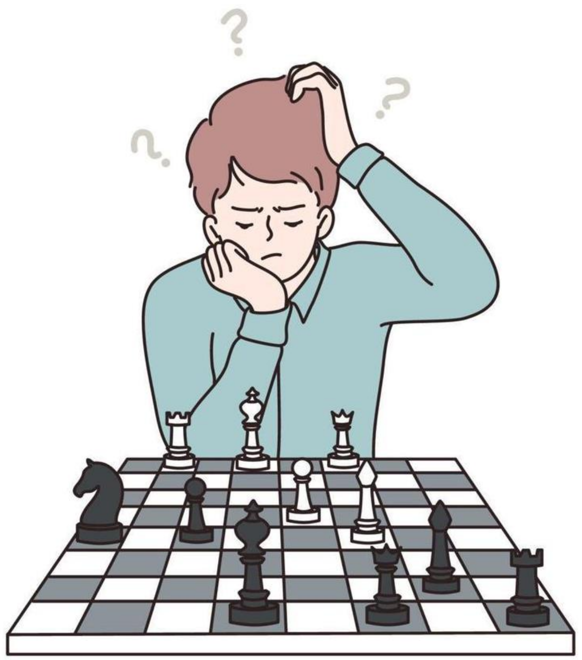
Motivation

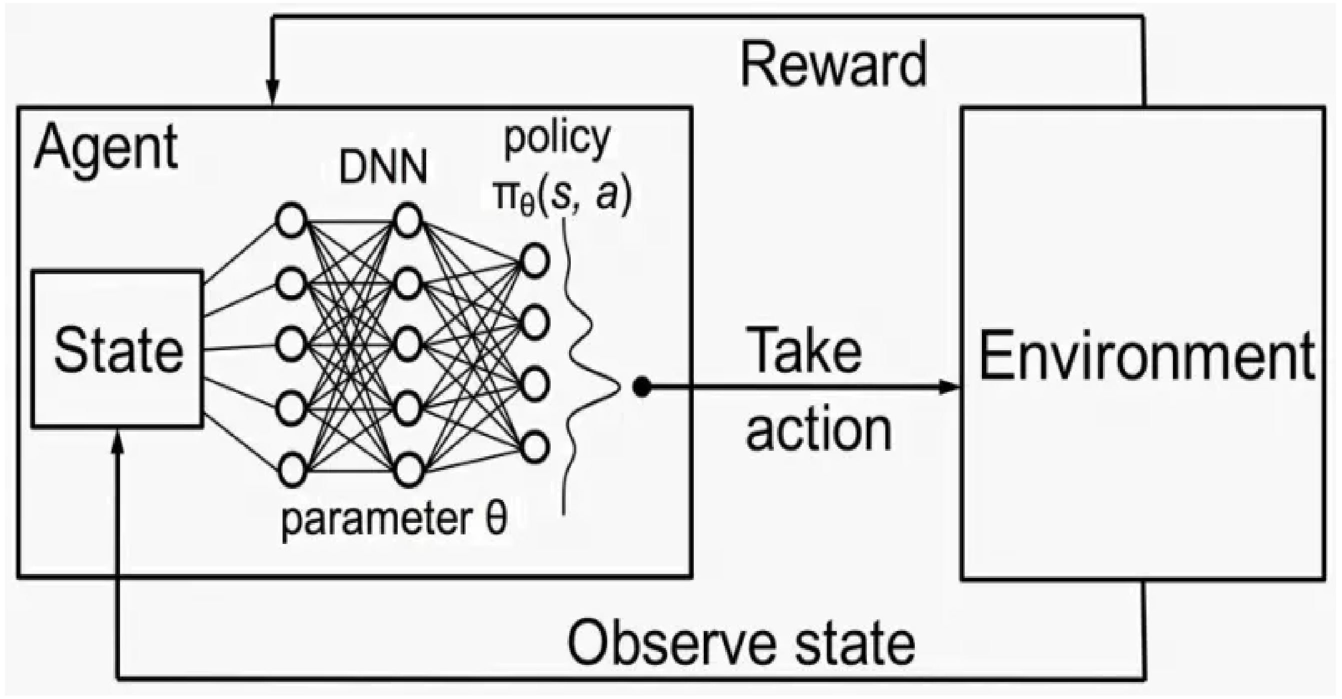
I am going to present deep learning project of automatic snake game playing based on my research paper. I will give you a comprehensive understanding of deep learning applications in game development. The main purpose is to explore and analyse the practical insights of game construction, modelling human like behaviour, results and provide recommendations. With this motivation I will discuss the following methodologies:

1. DEEP REINFORCEMENT LEARNING
2. Q-LEARNING AND DQ-LEARNING
3. DEEP Q-LEARNING PROCESS
4. IMPLEMENTING THE SNAKE GAME
5. IMPLEMENTING THE AGENT
6. IMPLIMENTING THE MODEL
7. TRAINING AND TESTING

Deep Reinforcement Learning

Combines artificial neural networks with a framework of reinforcement learning that helps software agents learn how to reach their goals. That is, it unites function approximation and target optimization, mapping states and actions to the rewards they lead to.



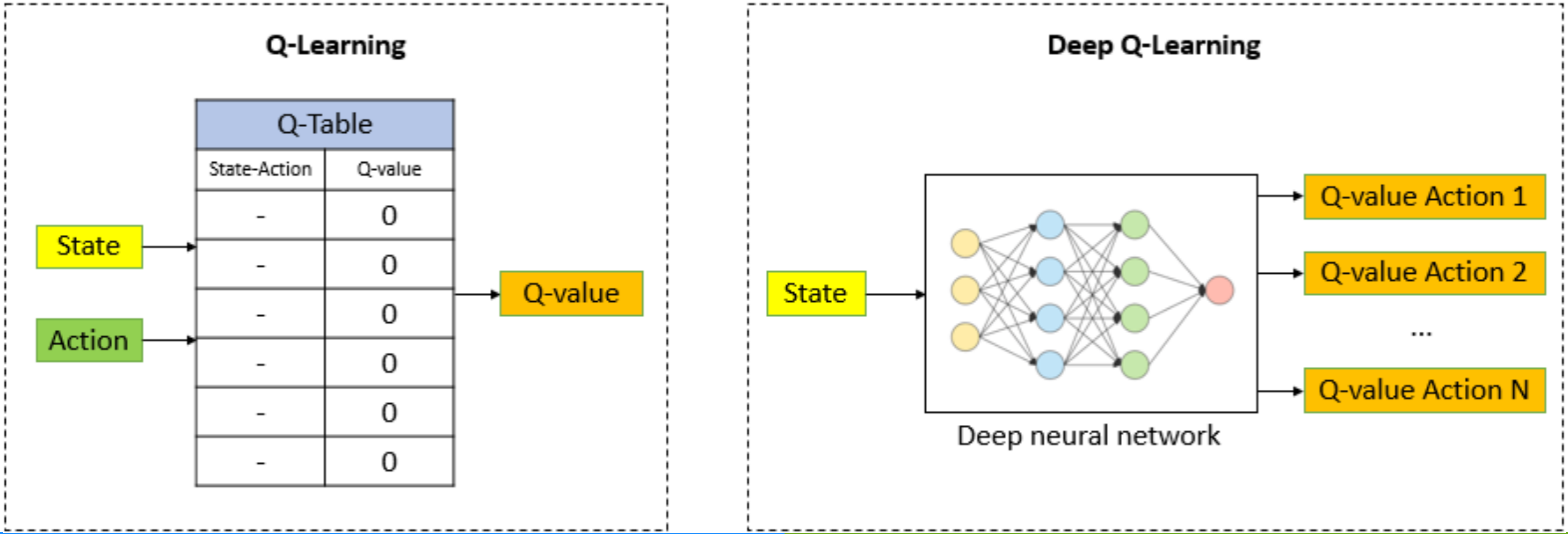


Brain Stimulation: Reward/ Penalty

DRL Process

Q-Learning and DQ-Learning

Q-Learning is a value-based algorithm that is used to find the optimal policy for an agent in a given environment. It determines the best series of actions through the Bellman Equation.
Deep Q-Learning is introduced when Q-Learning fails to calculate many states and actions.



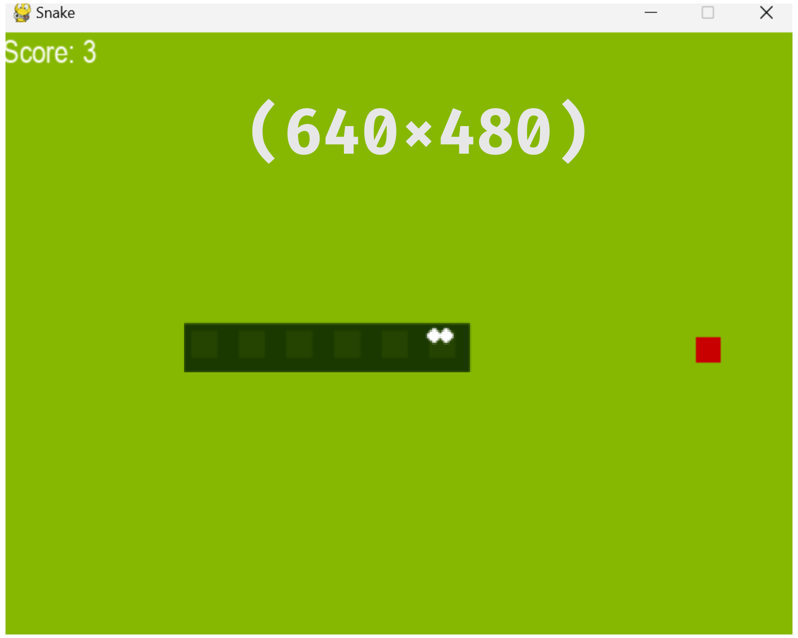
IMPLEM. THE SNAKE GAME

1. Basic Components Construction:
   - Display
   - Player/Snake

```
self.head = Point(self.w/2, self.h/2)
self.snake = [self.head,
              Point(self.head.x-BLOCK_SIZE, self.head.y),
              Point(self.head.x-(2*BLOCK_SIZE), self.head.y)]
```

   - Food

```
def _place_food(self):
    x = random.randint(0, (self.w-BLOCK_SIZE )//BLOCK_SIZE )*BLOCK_SIZE
    y = random.randint(0, (self.h-BLOCK_SIZE )//BLOCK_SIZE )*BLOCK_SIZE
    self.food = Point(x, y)
    if self.food in self.snake:
        self._place_food()
```



IMPLEM. THE SNAKE GAME

2. Play Steps

- Direction
  - Straight – [1, 0, 0]
  - Right turn – [0, 1, 0]
  - Left turn – [0, 0, 1]
- Movement
  - if direction == Direction.RIGHT: x += BLOCK SIZE
  - elif direction == Direction.DOWN: y += BLOCK SIZE
- Reward and Score: Reward = +10 and penalty = -10
- Game Over: Collisions

IMPLEM. THE AGENT

1. Initials

 - Max memory, Batch size, Learning rates, Epsilon, Discount rate

2. Direction setup

   - Point right = Point (head.x + 20, head.y)
   - Point Left = Point (head.x - 20, head.y)

   Similarly, for Up (y + 20) and Down (y – 20)

3. Get States: 11 States

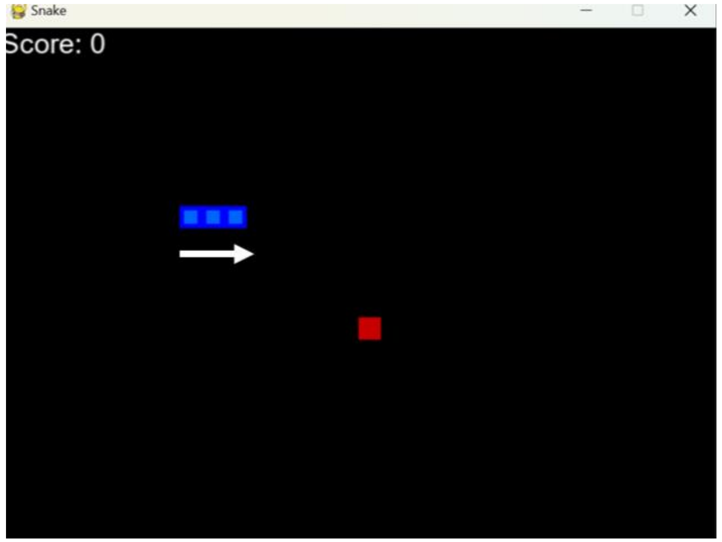
   - Move direction states = [left, right, up, down]
   - Danger zones = [danger straight, danger right, danger left]
   - Food direction = [left, right, up down]

IMPLEM. THE AGENT

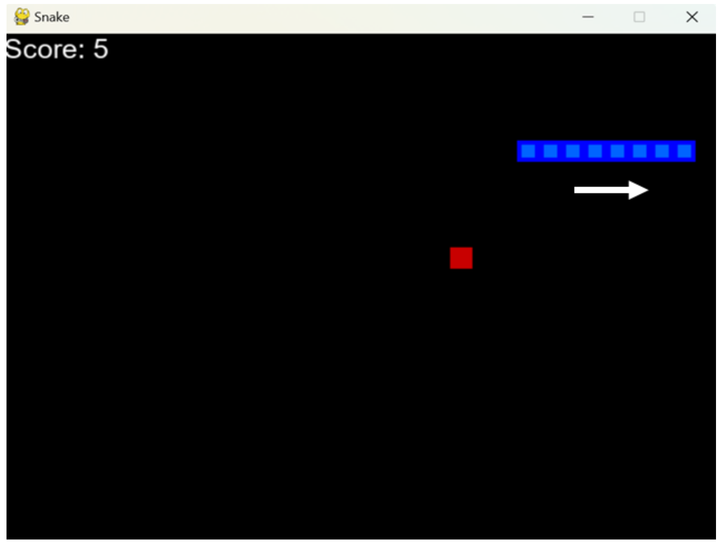
4. Scenarios

Danger zones	dan_straight, dan_right, dan_left	[0, 0, 0]
Direction states	left, right, up, down	[0, 0, 0, 0]
Food direction	left, right, up, down	[0, 0, 0, 0]

[0, 0, 0]
[0, 1, 0, 0]
[0, 1, 0, 0]



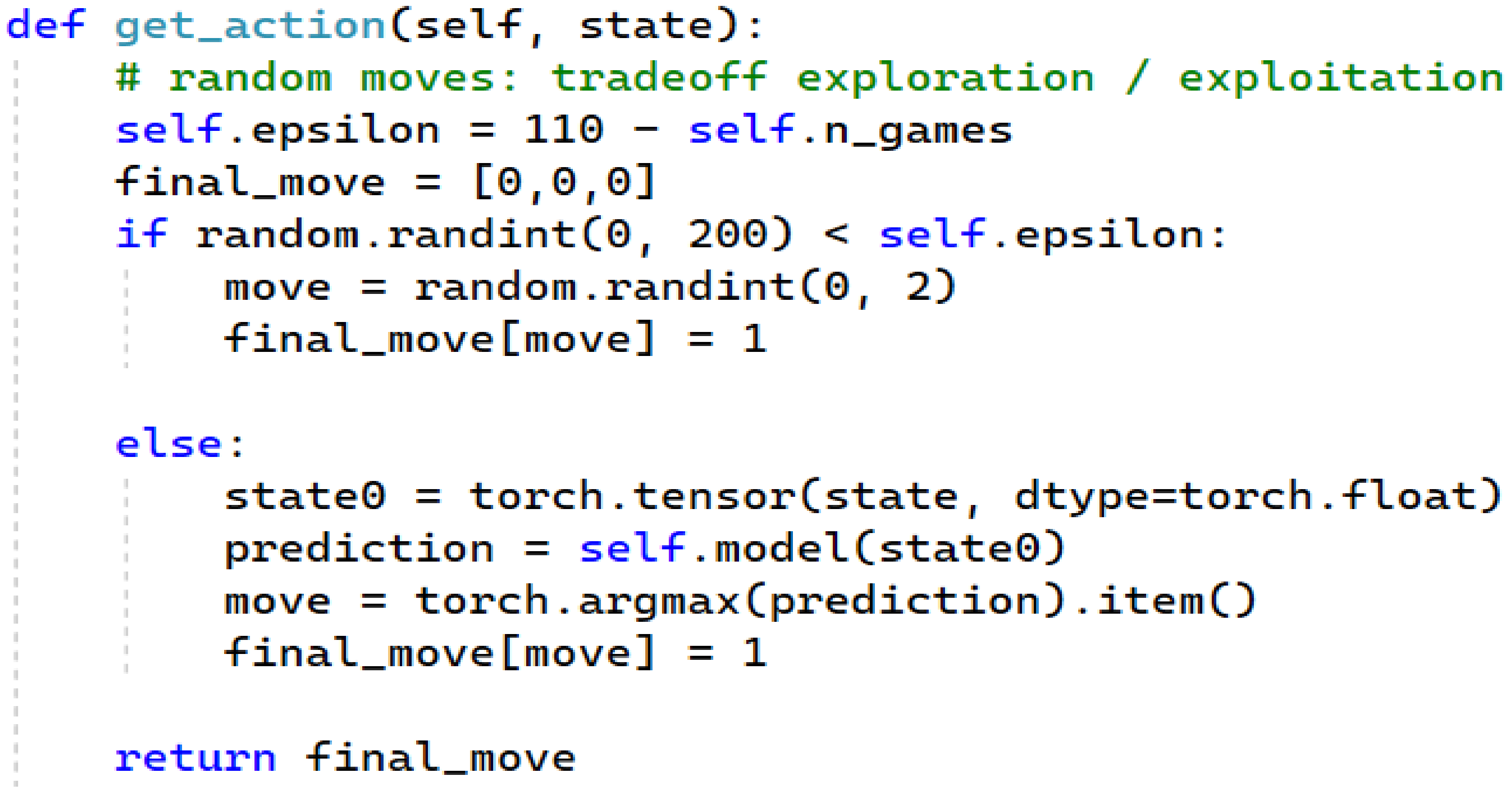
[1, 0, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]



IMPLEM. THE AGENT

Saving functions: Remember, Train short & long memory

Get action: Random exploration & Exploitation:



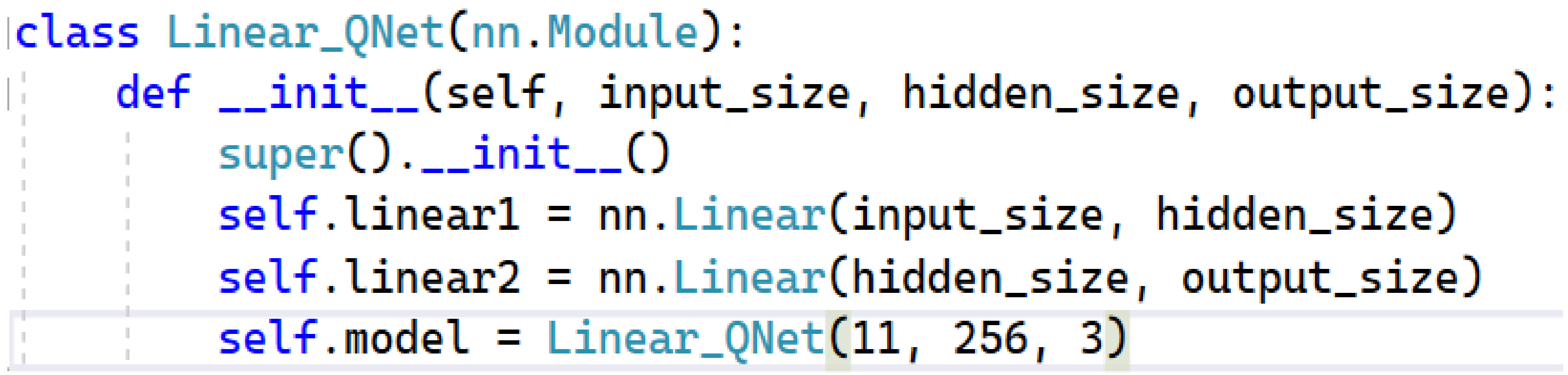
```python
def get_action(self, state):
    # random moves: tradeoff exploration / exploitation
    self.epsilon = 110 - self.n_games
    final_move = [0,0,0]
    if random.randint(0, 200) < self.epsilon:
        move = random.randint(0, 2)
        final_move[move] = 1

    else:
        state0 = torch.tensor(state, dtype=torch.float)
        prediction = self.model(state0)
        move = torch.argmax(prediction).item()
        final_move[move] = 1

    return final_move
```

IMPLIMENTING THE MODEL

1. Linear Qnet:



```python
class Linear_QNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()
        self.linear1 = nn.Linear(input_size, hidden_size)
        self.linear2 = nn.Linear(hidden_size, output_size)
        self.model = Linear_QNet(11, 256, 3)
```

2. QTrainer:

$$Loss = (Q_{new} - Q)^2$$

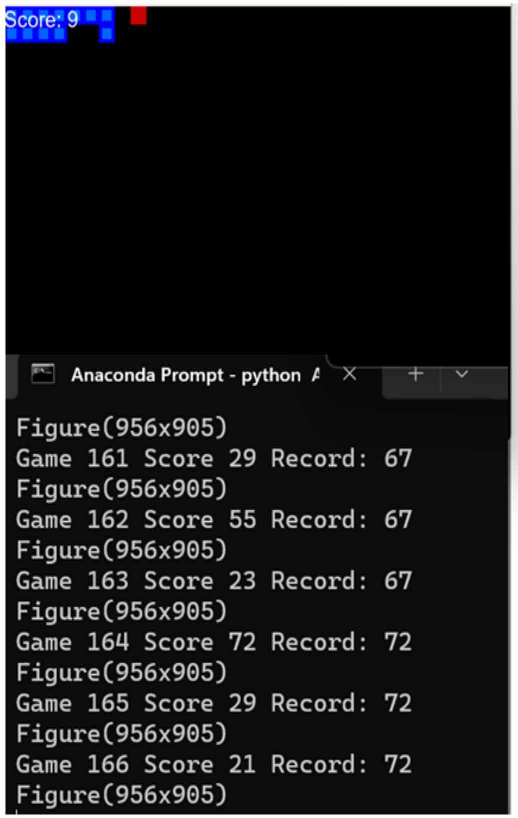
3. Q Value and Bellman Equation:

$$\underbrace{NewQ(s,a)}_{\text{New Q value for that state and that action}} = \underbrace{Q(s,a)}_{\text{Current Q value}} + \underset{\text{Learning Rate}}{\alpha}\Big[\underbrace{R(s,a)}_{\text{Reward for taking that action at that state}} + \underset{\text{Discount rate}}{\gamma}\,\underbrace{max\,Q'(s',a')}_{\text{Maximum expected future reward given the new s' and all possible actions at that new state}} - Q(s,a)\Big]$$

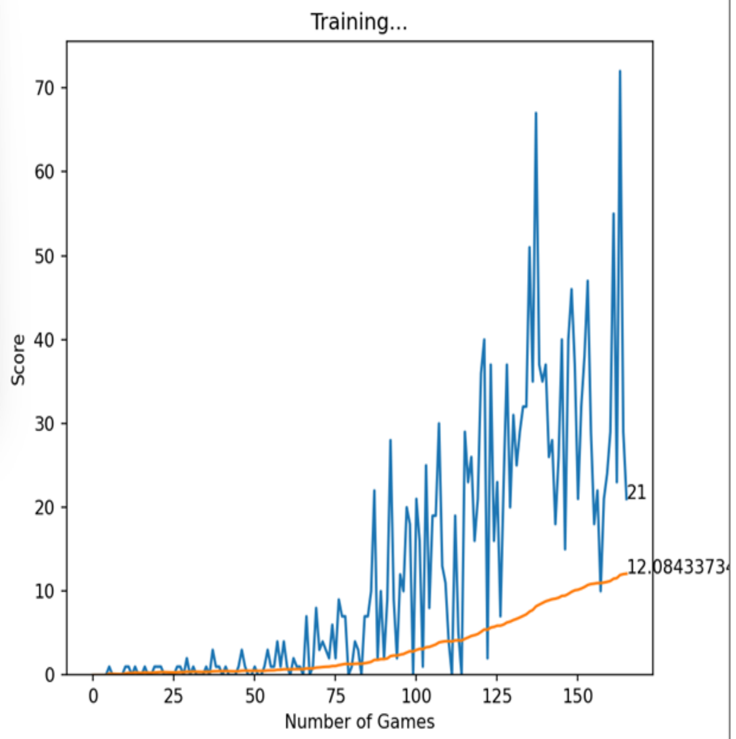
Deep Q-Learning process

1. Initialize Q-values
2. Choose action a for state s (Best Q-value)
3. Perform action a, new state s’
4. Measure reward R
5. Update Q with Bellman Equation
6. Update weights of the Neural Network

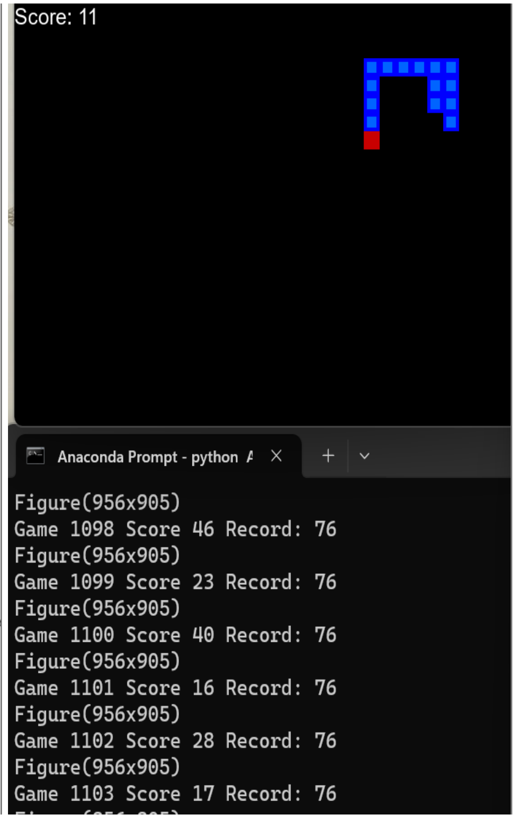
TRAINING AND TESTING



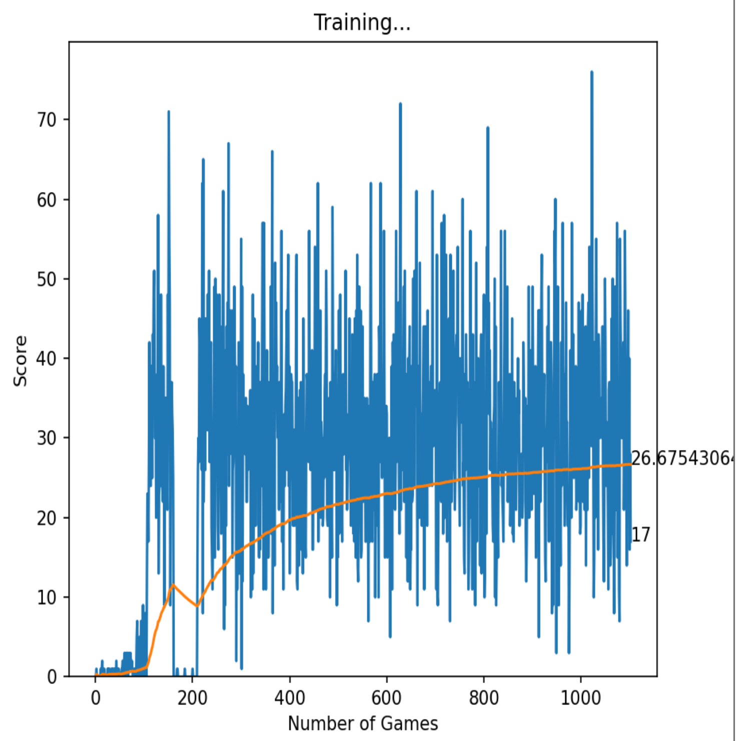
```
Figure(956x905)
Game 161 Score 29 Record: 67
Figure(956x905)
Game 162 Score 55 Record: 67
Figure(956x905)
Game 163 Score 23 Record: 67
Figure(956x905)
Game 164 Score 72 Record: 72
Figure(956x905)
Game 165 Score 29 Record: 72
Figure(956x905)
Game 166 Score 21 Record: 72
Figure(956x905)
```



Test 1



```
Figure(956x905)
Game 1098 Score 46 Record: 76
Figure(956x905)
Game 1099 Score 23 Record: 76
Figure(956x905)
Game 1100 Score 40 Record: 76
Figure(956x905)
Game 1101 Score 16 Record: 76
Figure(956x905)
Game 1102 Score 28 Record: 76
Figure(956x905)
Game 1103 Score 17 Record: 76
```



Test 2

REFERENCES

1. https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-value-functions-dqn-actor-critic-method-backpropagation-through-83a277d8c38d

2. Baeldung on Computer Science, “Q-learning vs. deep q-learning vs. deep q-network – baeldung on computer science,” 2023. [Online]. Available: https://www.baeldung.com/cs/q-learning-vs-deep-q-learningvs-deep-q-network

3. freeCodeCamp.org, “Diving deeper into reinforcement learning with q-learning,” freeCodeCamp.org, 10/04/2018. [Online]. Available: https://www.freecodecamp.org/news/diving-deeperinto-reinforcement-learning-with-q-learning-c18d0db58efe/

4. Patrick Loeber, “Python + pytorch + pygame reinforcement learning – train an ai to play snake,” 2022. [Online]. Available: https://www.youtube.com/watch?v=L8ypSXwyBds