Reinforcement Learning-Based Pong Game from Atari with Deep Q-Network Agent : A Detailed Report

1. Introduction

This RL project is based on the interative arcade game called Pong created by Allan Alcorn, an engineer at Atari,Inc, an American Video game developer company in 1972. Here, the game is played and learned by an AI agent through Reinforcement Learning techniques. Using the Deep Q-Network(DQN) algorithm, the agent gets trained to control one of the paddles and compete against a human opponent. The agent learns over time progressively, interacting with the environment and updates its policy using the experience it gains from training/playing.

Here, a feature is also added for the agent to learn from an human opponent while playing against them in real time. Thus, enhancing the ability of the agent to play through dynamic and continuous learning.

2. What is Pong game?

Pong is a classic 2D arcade game in which two paddles move vertically on either side of the screen to hit a ball back back and forth continuously. The goal is to score points by sending the ball past the other paddle that is controlled by another opponent. The game mechanics include:

- **Ball Movement**: The ball moves with a constant velocity and bounces off the paddles and the top and bottom walls.
- **Paddle Movement**: Paddle moves with a constant velocity and vertically to intercept the ball
- **Scoring Points**: A point is scored if the any of the players fail to hit the ball and let it pass through the paddle boundaries

3. Reinforcement Learning (RL) in Pong

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by performing actions in an environment and receiving rewards or penalties based on the outcomes of these actions. In our Pong game:

- the agent's goal is to maximise the cumulative reward by learning how to move the paddle to successfully hit the ball and prevent the opponent from scoring the point.
- The only actions here to be performed is the paddle movements i.e up and down
- Rewards are given bases Rewards are given based on whether the agent hits the ball (+1), loses a point (-10), or scores (+10).

Reinforcement Learning General Framework

- **Agent**: An entity that learns and makes decisions based on the observed environment and the rewards it receives.
- **State (S)**: The current state of the environment, which is represented by the ball's position,its velocity, and the paddle position.
- **Action (A)**: The set of actions (here it is only two actions) available to the agent.
- **Reward (R)**: The feedback recieved from the environment after taking an action from the action pool based on the circumstance/state of the environment. Positive rewards are for success(ex, hitting the ball), negative rewards are for losing a point or letting the ball go.
- **Policy (π)**: The policy is a strategy that will be used by the agent to follow a set of actions based on the current state to maximise rewards.
- **Value Function(V)**: The expected cumulative reward that the agent will receive, starting from a given state and following the policy.

The Pong game environment for this project is modelled as a Markov Decision Process(MDP) where:

- The next state depends on the current state and action taken which is the Markov Property
- The agent learns to approximate the **Q-value function** using *Neural Networks*.

4. Deep Q-Network (DQN) for Pong Game

This project uses Deep Q-Network(DQN) algorithm to train the AI Agent.

4.1. Q-Learning

Q-learning is a form of model-free reinforcement learning where the agent learns a Q-function $Q(s, a)$, which estimates the value (expected future rewards) of taking action a in state s. The update rule for Q-learning is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

Where:

- α is the learning rate.
- r is the reward obtained from the environment.
- γ is the discount factor, which determines the importance of future rewards.
- s' is the new state after taking action a.

4.2. Deep Q-Network (DQN)

DQN extends Q-learning by using a neural network to approximate the Q-function $Q(s, a)$. The agent interacts with the environment, collects experiences, and uses these experiences to train a neural network to predict the Q-values.

Network Architecture for DQN

The neural network in DQN has the given architecture:

- **Input Layer**: The state of the Pong environment (ball position, velocity, paddle positions).
- **Hidden Layers**: Several fully connected layers that process the input to extract useful features.
- **Output Layer**: Two output values, each representing the Q-value of one possible action (move up or move down).

4.3. Experience Replay

One crucial feature of DQN is **experience replay** where instead of updating the Q-values after each action, the agent stores the experiences it gains (s, a, r, s') in a memory buffer which it periodically samples as a batch of experiences to update the neural network. This helps break the correlation between consecutive experiences and stabilizes training.

4.4. Target Network

DQN also uses a **target network** to stabilize the Q-value updates. This target network is a copy of the Q-network, but its weights are updated less frequently which reduces oscillations and divergence during training to maintain a consistent agent and its plays.

4.5. Epsilon-Greedy Policy (Exploration vs Explotation condition balance)

To balance exploration and exploitation, the DQN agent follows an **epsilon-greedy policy** where it explores by taking random actions with probability ϵ, and exploits its current knowledge (choosing the action with the highest Q-value) with probability $1 - \epsilon$. Over time, ϵ decays to reduce exploration as the agent becomes more confident in its policy.

5. Tech Stack

- **Turtle Graphics**: Used for rendering the Pong game environment, including paddles, ball movement, and the game screen.
- **PyTorch**: To implement the DQN model and for building and training the neural network that approximates the Q-function.
- **NumPy**: For numerical operations and handling state representations.
- **Collections (Deque)**: For managing experience replay by storing past experiences.

6. Implementation Procedure

6.1. Game Setup

The game environment is created and setup using Python's Turtle module. The screen is initialized, and the paddles and ball are rendered. The game mechanics are coded, including ball bouncing, scoring, and paddle control.

6.2. Agent Training

- The DQN agent interacts with the Pong environment, where the state is represented by a vector of six values (ball position, ball velocity, left paddle position, right paddle position).
- The agent receives rewards based on performance of its actions and the impacts it has on the environment (e.g., hitting the ball or scoring points).
- Over the iteration of episodes, the agent's Q-network is updated using the Q-learning update rule, where the expected future reward is approximated through a neural network.

6.3. Playing Against the Agent

After training the agent, the model is saved and can be loaded for gameplay. The human controls the right paddle using keyboard inputs, while the trained agent controls the left paddle. The agent continues to learn during the game using experience replay, improving its performance as it plays against the human.

Primary Equations Used

Q-Learning Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

Where:

- $Q(s,a)$: Current Q-value for state s and action a.
- r: Immediate reward from the environment.
- γ: Discount factor, controlling the weight of future rewards.
- α: Learning rate.

DQN Loss Function:

The goal is to minimize the difference between the predicted Q-values (from the neural network) and the target Q-values (which represent the "true" value the agent should have learned for a particular state-action pair). This difference is captured using the Mean Squared Error (MSE). The loss function used to update the neural network is the mean squared error (MSE) between the predicted Q-values and the target Q-values:

$$L = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)^2 \right]$$

- L: This is the loss, specifically the mean squared error between the predicted Q-values and the target Q-values.
- $\mathbb{E}$: Represents the expected value, or the average over a batch of experiences sampled from the replay memory.
- r: The **immediate reward** received after taking action a in state s. This is the short-term feedback the agent gets from the environment after its action.

- γ: The **discount factor**, which determines the importance of future rewards. A value between 0 and 1, where:
  - $\gamma = 0$ means the agent only cares about immediate rewards.
  - γ close to 1 means the agent values future rewards more, encouraging it to plan ahead.
- $\max_{a'} Q(s', a')$: The maximum predicted Q-value for the **next state** s' across all possible next actions a'. This represents the agent's estimate of the best future reward it can obtain from the next state.
- $Q(s, a)$: The **predicted Q-value** for taking action a in the current state s.
- $r + \gamma \max_{a'} Q(s', a')$: This term represents the **target Q-value**. It's the immediate reward r plus the discounted maximum future reward $\max_{a'} Q(s', a')$. This is the "true" Q-value that the neural network is trying to approximate.

Challenges Encountered and Steps taken

1. **Exploration-Exploitation Tradeoff**: A balance between exploring new actions and exploiting known strategies had to be achieved by using an epsilon-greedy policy.
2. **Training Stability**: Using experience replay and a target network was necessary to stabilize the learning process.
3. **Human-AI Interaction**: Had to ensure a smooth gameplay between the human and AI agent required balancing the game's speed, AI responsiveness, and the agent's continuous learning during gameplay.

In [1]:
```
import random
import numpy as np
import turtle
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from collections import deque
```

In [2]:
```
# --- Pong Environment ---

class PongEnv:
    def __init__(self):
        # Initialize the screen
        self.screen = turtle.Screen()
        self.screen.title("Pong RL")
        self.screen.bgcolor("white")
        self.screen.setup(width=1000, height=600)

        # Left paddle (controlled by the DQN agent)
        self.left_paddle = turtle.Turtle()
        self.left_paddle.speed(0)
        self.left_paddle.shape("square")
        self.left_paddle.color("black")
        self.left_paddle.shapesize(stretch_wid=6, stretch_len=2)
        self.left_paddle.penup()
        self.left_paddle.goto(-400, 0)
```

```
        # Right paddle (controlled by simple AI)
        self.right_paddle = turtle.Turtle()
        self.right_paddle.speed(0)
        self.right_paddle.shape("square")
        self.right_paddle.color("black")
        self.right_paddle.shapesize(stretch_wid=6, stretch_len=2)
        self.right_paddle.penup()
        self.right_paddle.goto(400, 0)

        # Ball
        self.ball = turtle.Turtle()
        self.ball.speed(40)
        self.ball.shape("circle")
        self.ball.color("blue")
        self.ball.penup()
        self.ball.goto(0, 0)
        self.ball.dx = 5  # Horizontal velocity
        self.ball.dy = -5  # Vertical velocity

        # Score
        self.left_score = 0
        self.right_score = 0

    def reset(self):
        """ Reset the ball position and randomly assign a new direction """
        self.ball.goto(0, 0)
        self.ball.dx *= random.choice([-1, 1])
        self.ball.dy *= random.choice([-1, 1])
        return self.get_state()

    def get_state(self):
        """ Get the current state: [ball_x, ball_y, ball_dx, ball_dy, left_paddl
        return np.array([self.ball.xcor(), self.ball.ycor(),
                         self.ball.dx, self.ball.dy,
                         self.left_paddle.ycor(), self.right_paddle.ycor()])

    def step(self, action):
        """
        Perform the selected action and move the paddles and ball accordingly.
        Return the new state, reward, and whether the episode has ended (done).
        """
        # Move left paddle based on the agent's action (0 = move up, 1 = move do
        if action == 0 and self.left_paddle.ycor() < 250:
            self.left_paddle.sety(self.left_paddle.ycor() + 20)
        elif action == 1 and self.left_paddle.ycor() > -240:
            self.left_paddle.sety(self.left_paddle.ycor() - 20)

        # Simple AI for the right paddle (follows the ball)
        if self.right_paddle.ycor() < self.ball.ycor() and self.right_paddle.ycc
            self.right_paddle.sety(self.right_paddle.ycor() + 20)
        elif self.right_paddle.ycor() > self.ball.ycor() and self.right_paddle.y
            self.right_paddle.sety(self.right_paddle.ycor() - 20)

        # Move the ball
        self.ball.setx(self.ball.xcor() + self.ball.dx)
        self.ball.sety(self.ball.ycor() + self.ball.dy)

        # Ball collision with top and bottom walls
        if self.ball.ycor() > 290:
```

```
            self.ball.sety(290)
            self.ball.dy *= -1

        if self.ball.ycor() < -290:
            self.ball.sety(-290)
            self.ball.dy *= -1

        # Ball collision with paddles
        if (self.ball.xcor() > 360 and self.ball.xcor() < 370) and \
        (self.ball.ycor() < self.right_paddle.ycor() + 50 and self.ball.ycor() >
            self.ball.setx(360)
            self.ball.dx *= -1  # Reverse direction upon hitting the right paddl

        if (self.ball.xcor() < -360 and self.ball.xcor() > -370) and \
        (self.ball.ycor() < self.left_paddle.ycor() + 50 and self.ball.ycor() >
            self.ball.setx(-360)
            self.ball.dx *= -1  # Reverse direction upon hitting the left paddle

        # Reward for paddle positioning (intermediate reward)
        paddle_position_reward = 1 - abs(self.left_paddle.ycor() - self.ball.ycc

        # Check for scoring
        reward = 0
        done = False

        # Left paddle scores
        if self.ball.xcor() > 500:
            self.left_score += 1
            reward = 10  # Positive reward for scoring
            done = True  # End of episode (for training)

        # Right paddle scores
        if self.ball.xcor() < -500:
            self.right_score += 1
            reward = -10  # Penalty for opponent scoring
            done = True  # End of episode (for training)

        # Combine rewards (shaping reward + intermediate reward)
        reward += paddle_position_reward  # Add positioning reward to the total

        # Get the new state after the step
        state = self.get_state()
        return state, reward, done
```

In [3]:

```
# --- DQN Agent ---

class DQN(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(DQN, self).__init__()
        # Define the neural network layers
        self.fc1 = nn.Linear(state_dim, 128)  # Input layer (state_dim -> 128 un
        self.fc2 = nn.Linear(128, 128)        # Hidden layer (128 -> 128 units)
        self.fc3 = nn.Linear(128, action_dim) # Output layer (128 -> action_dim)

    def forward(self, x):
        # Pass input through the network
        x = F.relu(self.fc1(x))  # ReLU activation
        x = F.relu(self.fc2(x))  # ReLU activation
        return self.fc3(x)       # Output action values
```

Concept of Prioritized Experience Replay (PER)

Prioritized Experience Replay (PER) is an improvement to the standard experience replay mechanism used in DQN which aims to improve learning rate of the agent efficiently. In the standard experience replay system, the agent stores the experiences (state, action, reward, next state, done) into a buffer and samples random mini-batches from this buffer to train the neural network. But, this random sampling treats all experiences equally and its not optimal for our case. Priority-based sampling mechanism increases the learning efficiency by considering only the most "important" experiences, which is where the agent has learnt the most. This is improved using the **Temporal Difference (TD) error**, which measures the difference between the predicted Q-value and the target Q-value. Experiences with a higher TD-error indicates that the agent's prediction is high and it has to learn from experiences more to reduce this error.

Key Concepts:

1. **Temporal Difference (TD) Error**:
    - TD error is the difference between the Q-value predicted by the Q-network and the target Q-value. It reflects how much the agent's prediction deviates from the actual outcome.
    - TD error = |Q(s, a) - (r + γ * max Q(s', a'))|
    - Larger TD errors imply that the agent made a larger prediction error, signaling that these experiences are more valuable for learning.
2. **Prioritized Sampling**:
    - Instead of sampling experiences randomly, PER assigns a priority to each experience based on its TD error. Experiences with larger TD errors are more likely to be sampled for training because they offer a higher potential for reducing the agent's prediction error.
    - The probability of sampling an experience i is proportional to its priority p_i:

      $$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

      where α controls the degree of prioritization (when α = 0, prioritization follows uniform random sampling).

In [4]:
```
class DQNAgent:
    def __init__(self, state_dim, action_dim):
        self.q_network = DQN(state_dim, action_dim)  # Primary Q-network
        self.target_network = DQN(state_dim, action_dim)  # Target Q-network
        self.optimizer = optim.Adam(self.q_network.parameters(), lr=0.001)  # Op

        self.replay_buffer = deque(maxlen=10000)  # Experience replay buffer
        self.priorities = deque(maxlen=10000)     # Store TD-errors as prioritie
        self.batch_size = 64
        self.gamma = 0.99  # Discount factor for future rewards
        self.epsilon = 1.0  # Exploration rate (initially explore)
```

```
        self.epsilon_decay = 0.995  # Decay rate for epsilon
        self.epsilon_min = 0.01  # Minimum value for epsilon
        self.update_target_steps = 100  # How often to update the target network
        self.steps_done = 0

    def select_action(self, state):
        """ Select an action using epsilon-greedy exploration """
        if random.random() < self.epsilon:
            return random.randint(0, 1)  # Explore (random action)
        else:
            state = torch.FloatTensor(state).unsqueeze(0)  # Convert state to te
            with torch.no_grad():
                q_values = self.q_network(state)  # Get action values from the Q
            return q_values.argmax().item()  # Exploit (select best action)

    def update(self):
        """ Sample a batch from replay buffer based on priorities and update the
        if len(self.replay_buffer) < self.batch_size:
            return  # Not enough samples to update

        # Normalize priorities to create a probability distribution
        priorities = np.array(self.priorities)
        probabilities = priorities / priorities.sum()

        # Sample experiences based on the calculated probabilities
        indices = np.random.choice(len(self.replay_buffer), self.batch_size, p=p
        batch = [self.replay_buffer[i] for i in indices]

        # Convert batch into tensors
        states, actions, rewards, next_states, dones = zip(*batch)
        states = torch.FloatTensor(states)
        actions = torch.LongTensor(actions)
        rewards = torch.FloatTensor(rewards)
        next_states = torch.FloatTensor(next_states)
        dones = torch.FloatTensor(dones)

        # Calculate Q-values and targets
        q_values = self.q_network(states).gather(1, actions.unsqueeze(1)).squeez
        next_q_values = self.target_network(next_states).max(1)[0]
        targets = rewards + self.gamma * next_q_values * (1 - dones)

        # Calculate loss and update the Q-network
        loss = F.mse_loss(q_values, targets.detach())
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

        # Update the target network periodically
        self.steps_done += 1
        if self.steps_done % self.update_target_steps == 0:
            self.target_network.load_state_dict(self.q_network.state_dict())

    def add_experience(self, state, action, reward, next_state, done):
        """ Add experience to the replay buffer with its TD-error (priority) """
        # Calculate the TD-error (temporal difference)
        state_tensor = torch.FloatTensor(state).unsqueeze(0)
        next_state_tensor = torch.FloatTensor(next_state).unsqueeze(0)

        with torch.no_grad():
            q_value = self.q_network(state_tensor)[0, action].item()
```

```
            next_q_value = self.target_network(next_state_tensor).max(1)[0].item

        target = reward + (1 - done) * self.gamma * next_q_value
        td_error = abs(target - q_value)

        # Append experience and its TD-error to the buffers
        self.replay_buffer.append((state, action, reward, next_state, done))
        self.priorities.append(td_error + 0.01)  # Small value added to avoid ze
```

In [5]:
```
# --- Training ---

def train_dqn_agent():
    # initiating environment and agent
    env = PongEnv()
    state_dim = 6  # State: [ball_x, ball_y, ball_dx, ball_dy, left_paddle_y, ri
    action_dim = 2  # Actions: 0 = move up, 1 = move down
    agent = DQNAgent(state_dim, action_dim)

    num_episodes = 1000
    for episode in range(num_episodes):
        state = env.reset()
        total_reward = 0
        done = False

        while not done:
            # agent selects an action
            action = agent.select_action(state)

            # environment responds to the action (state is changed)
            next_state, reward, done = env.step(action)

            # gained experience added to replay buffer and agent is updated
            agent.add_experience(state, action, reward, next_state, done)
            agent.update()

            state = next_state
            total_reward += reward

        # Decay epsilon (exploration rate) after each episode to decrease the ex
        agent.epsilon = max(agent.epsilon_min, agent.epsilon * agent.epsilon_dec
        print(f"Episode {episode + 1}: Total Reward = {total_reward}")
```

In [ ]:
```
if __name__ == "__main__":
    train_dqn_agent()
```

```
C:\Users\suhas\AppData\Local\Temp\ipykernel_10652\428573313.py:42: UserWarning: C
reating a tensor from a list of numpy.ndarrays is extremely slow. Please consider
converting the list to a single numpy.ndarray with numpy.array() before convertin
g to a tensor. (Triggered internally at C:\cb\pytorch_1000000000000\work\torch\cs
rc\utils\tensor_new.cpp:281.)
  states = torch.FloatTensor(states)
```

```
Episode 1: Total Reward = 34.76666666666666
Episode 2: Total Reward = -1.0333333333333332
Episode 3: Total Reward = 90.53333333333333
Episode 4: Total Reward = 103.59999999999997
Episode 5: Total Reward = 154.93333333333325
Episode 6: Total Reward = 100.03333333333335
Episode 7: Total Reward = 78.73333333333335
Episode 8: Total Reward = -5.233333333333333
Episode 9: Total Reward = 41.16666666666666
Episode 10: Total Reward = 75.66666666666667
Episode 11: Total Reward = 71.50000000000004
Episode 12: Total Reward = 63.09999999999999
Episode 13: Total Reward = -40.233333333333334
Episode 14: Total Reward = 119.56666666666672
Episode 15: Total Reward = 159.9999999999999
Episode 16: Total Reward = 41.1
Episode 17: Total Reward = -22.63333333333334
Episode 18: Total Reward = -8.03333333333333
Episode 19: Total Reward = 14.1
Episode 20: Total Reward = 80.86666666666662
Episode 21: Total Reward = 60.66666666666673
Episode 22: Total Reward = 158.46666666666658
Episode 23: Total Reward = 117.63333333333338
Episode 24: Total Reward = -35.16666666666666
Episode 25: Total Reward = 41.66666666666662
Episode 26: Total Reward = -9.033333333333339
Episode 27: Total Reward = 54.33333333333334
Episode 28: Total Reward = 137.2
Episode 29: Total Reward = 131.9000000000001
Episode 30: Total Reward = -27.23333333333333
Episode 31: Total Reward = -21.96666666666667
Episode 32: Total Reward = 123.50000000000013
Episode 33: Total Reward = 74.53333333333336
Episode 34: Total Reward = 111.96666666666664
Episode 35: Total Reward = 110.93333333333344
Episode 36: Total Reward = 127.30000000000001
Episode 37: Total Reward = -33.833333333333336
Episode 38: Total Reward = 77.36666666666666
Episode 39: Total Reward = -25.833333333333336
Episode 40: Total Reward = 137.23333333333335
Episode 41: Total Reward = 61.433333333333316
Episode 42: Total Reward = 64.7333333333333
Episode 43: Total Reward = -28.033333333333335
Episode 44: Total Reward = 56.43333333333335
Episode 45: Total Reward = -29.03333333333333
Episode 46: Total Reward = 86.16666666666674
Episode 47: Total Reward = 47.09999999999995
Episode 48: Total Reward = 125.0
Episode 49: Total Reward = 66.69999999999997
Episode 50: Total Reward = 22.366666666666678
Episode 51: Total Reward = -11.766666666666667
Episode 52: Total Reward = 173.7333333333333
Episode 53: Total Reward = 119.53333333333333
Episode 54: Total Reward = 6.1
Episode 55: Total Reward = 163.36666666666665
Episode 56: Total Reward = 68.90000000000006
Episode 57: Total Reward = 56.83333333333333
Episode 58: Total Reward = 59.16666666666667
Episode 59: Total Reward = 47.86666666666666
Episode 60: Total Reward = 266.7999999999997
```

```
Episode 61: Total Reward = -8.366666666666667
Episode 62: Total Reward = 49.099999999999994
Episode 63: Total Reward = -37.76666666666666
Episode 64: Total Reward = 51.53333333333332
Episode 65: Total Reward = 131.60000000000002
Episode 66: Total Reward = 132.20000000000007
Episode 67: Total Reward = 56.0333333333333
Episode 68: Total Reward = -34.833333333333336
Episode 69: Total Reward = 160.4999999999999
Episode 70: Total Reward = 59.53333333333336
Episode 71: Total Reward = -28.433333333333334
Episode 72: Total Reward = 72.40000000000002
Episode 73: Total Reward = 49.19999999999995
Episode 74: Total Reward = 49.43333333333334
Episode 75: Total Reward = 61.96666666666667
Episode 76: Total Reward = -32.03333333333333
Episode 77: Total Reward = 78.23333333333332
Episode 78: Total Reward = -38.96666666666667
Episode 79: Total Reward = -40.7
Episode 80: Total Reward = 57.00000000000003
Episode 81: Total Reward = 139.0333333333333
Episode 82: Total Reward = 132.73333333333343
Episode 83: Total Reward = 66.06666666666676
Episode 84: Total Reward = 44.06666666666666
Episode 85: Total Reward = 47.3
Episode 86: Total Reward = 61.76666666666665
Episode 87: Total Reward = -41.96666666666667
Episode 88: Total Reward = 44.66666666666667
Episode 89: Total Reward = 44.400000000000006
Episode 90: Total Reward = 70.43333333333335
Episode 91: Total Reward = 1.8333333333333321
Episode 92: Total Reward = 44.36666666666667
Episode 93: Total Reward = 4.966666666666667
Episode 94: Total Reward = 130.70000000000002
Episode 95: Total Reward = 21.799999999999986
Episode 96: Total Reward = 112.5
Episode 97: Total Reward = -42.233333333333334
Episode 98: Total Reward = 37.03333333333324
Episode 99: Total Reward = 150.00000000000009
Episode 100: Total Reward = 129.63333333333333
Episode 101: Total Reward = -43.10000000000001
Episode 102: Total Reward = 66.8
Episode 103: Total Reward = 182.1666666666667
Episode 104: Total Reward = 159.59999999999988
Episode 105: Total Reward = 188.5666666666667
Episode 106: Total Reward = -23.96666666666667
Episode 107: Total Reward = 47.10000000000001
Episode 108: Total Reward = 77.46666666666665
Episode 109: Total Reward = 170.99999999999983
Episode 110: Total Reward = 168.53333333333336
Episode 111: Total Reward = 17.63333333333334
Episode 112: Total Reward = 166.00000000000006
Episode 113: Total Reward = 45.266666666666644
Episode 114: Total Reward = 155.63333333333347
Episode 115: Total Reward = -44.43333333333333
Episode 116: Total Reward = 148.8333333333334
Episode 117: Total Reward = 116.46666666666663
Episode 118: Total Reward = 42.566666666666656
Episode 119: Total Reward = -41.766666666666666
Episode 120: Total Reward = -40.96666666666667
```

```
Episode 121: Total Reward = 42.533333333333346
Episode 122: Total Reward = -41.166666666666664
Episode 123: Total Reward = 40.93333333333328
Episode 124: Total Reward = -40.166666666666664
Episode 125: Total Reward = 55.36666666666668
Episode 126: Total Reward = 111.66666666666666
Episode 127: Total Reward = 59.900000000000006
Episode 128: Total Reward = 72.20000000000002
Episode 129: Total Reward = -39.3
Episode 130: Total Reward = 116.60000000000002
Episode 131: Total Reward = 118.66666666666671
Episode 132: Total Reward = 111.63333333333328
Episode 133: Total Reward = 38.63333333333334
Episode 134: Total Reward = 41.73333333333334
Episode 135: Total Reward = 43.7333333333334
Episode 136: Total Reward = 40.099999999999994
Episode 137: Total Reward = 40.96666666666667
Episode 138: Total Reward = 40.399999999999984
Episode 139: Total Reward = 42.69999999999998
Episode 140: Total Reward = 41.900000000000006
Episode 141: Total Reward = 116.79999999999991
Episode 142: Total Reward = -41.03333333333333
Episode 143: Total Reward = 39.03333333333333
Episode 144: Total Reward = 117.80000000000001
Episode 145: Total Reward = 31.299999999999983
Episode 146: Total Reward = 35.66666666666666
Episode 147: Total Reward = 28.06666666666665
Episode 148: Total Reward = 32.3
Episode 149: Total Reward = 42.43333333333335
Episode 150: Total Reward = 23.26666666666666
Episode 151: Total Reward = 121.66666666666676
Episode 152: Total Reward = 45.699999999999946
Episode 153: Total Reward = 44.63333333333342
Episode 154: Total Reward = -42.83333333333334
Episode 155: Total Reward = -40.16666666666667
Episode 156: Total Reward = 26.866666666666667
Episode 157: Total Reward = -41.89999999999999
Episode 158: Total Reward = -41.099999999999994
Episode 159: Total Reward = 118.83333333333329
Episode 160: Total Reward = 48.96666666666662
Episode 161: Total Reward = 49.66666666666668
Episode 162: Total Reward = 105.86666666666682
Episode 163: Total Reward = 33.3333333333333
Episode 164: Total Reward = 76.09999999999995
Episode 165: Total Reward = 86.00000000000003
Episode 166: Total Reward = 34.46666666666667
Episode 167: Total Reward = 56.89999999999997
Episode 168: Total Reward = 86.69999999999996
Episode 169: Total Reward = 50.99999999999997
Episode 170: Total Reward = -14.433333333333337
Episode 171: Total Reward = 13.533333333333337
Episode 172: Total Reward = 1.0
Episode 173: Total Reward = 8.099999999999994
Episode 174: Total Reward = 11.966666666666665
Episode 175: Total Reward = 15.099999999999998
Episode 176: Total Reward = 1.266666666666664
Episode 177: Total Reward = 48.4
Episode 178: Total Reward = 103.26666666666671
Episode 179: Total Reward = 117.69999999999992
Episode 180: Total Reward = 114.23333333333333
```

```
Episode 181: Total Reward = -2.666666666666665
Episode 182: Total Reward = 91.93333333333337
Episode 183: Total Reward = 105.86666666666666
Episode 184: Total Reward = -5.366666666666667
Episode 185: Total Reward = 114.3666666666667
Episode 186: Total Reward = 200.30000000000004
Episode 187: Total Reward = 187.46666666666658
Episode 188: Total Reward = 59.566666666666656
Episode 189: Total Reward = 294.9999999999998
Episode 190: Total Reward = 304.8333333333331
Episode 191: Total Reward = 308.00000000000006
Episode 192: Total Reward = 40.233333333333334
Episode 193: Total Reward = 127.36666666666672
Episode 194: Total Reward = 197.33333333333326
Episode 195: Total Reward = 39.099999999999994
Episode 196: Total Reward = 31.699999999999996
Episode 197: Total Reward = 192.73333333333338
Episode 198: Total Reward = 160.30000000000007
Episode 199: Total Reward = 97.70000000000003
Episode 200: Total Reward = 100.93333333333337
Episode 201: Total Reward = 25.59999999999998
Episode 202: Total Reward = 43.5
Episode 203: Total Reward = 772.8499999999988
Episode 204: Total Reward = 140.13333333333335
Episode 205: Total Reward = 208.96666666666664
Episode 206: Total Reward = 197.13333333333327
Episode 207: Total Reward = 171.46666666666664
Episode 208: Total Reward = 216.2333333333334
Episode 209: Total Reward = 560.7833333333335
Episode 210: Total Reward = 232.39999999999992
Episode 211: Total Reward = -36.56666666666667
Episode 212: Total Reward = 172.83333333333337
Episode 213: Total Reward = 108.50000000000006
Episode 214: Total Reward = 14.8
Episode 215: Total Reward = 148.06666666666666
Episode 216: Total Reward = 182.3999999999999
Episode 217: Total Reward = 130.06666666666683
Episode 218: Total Reward = 205.09999999999994
Episode 219: Total Reward = 165.00000000000009
Episode 220: Total Reward = 113.6333333333334
Episode 221: Total Reward = 202.03333333333336
Episode 222: Total Reward = 232.99999999999986
Episode 223: Total Reward = 154.0333333333333
Episode 224: Total Reward = 18.53333333333334
Episode 225: Total Reward = 213.13333333333327
Episode 226: Total Reward = 135.86666666666662
Episode 227: Total Reward = 268.33333333333314
Episode 228: Total Reward = 221.10000000000002
Episode 229: Total Reward = 183.73333333333323
Episode 230: Total Reward = 120.90000000000008
Episode 231: Total Reward = -18.833333333333332
Episode 232: Total Reward = 204.89999999999984
Episode 233: Total Reward = 301.5999999999998
Episode 234: Total Reward = 253.3999999999998
Episode 235: Total Reward = 257.2666666666666
Episode 236: Total Reward = 141.3999999999999
Episode 237: Total Reward = 204.89999999999986
Episode 238: Total Reward = 259.2
Episode 239: Total Reward = 166.59999999999997
Episode 240: Total Reward = 229.06666666666666
```

```
Episode 241: Total Reward = 170.86666666666636
Episode 242: Total Reward = 288.1666666666663
Episode 243: Total Reward = 174.13333333333327
Episode 244: Total Reward = 343.39999999999986
Episode 245: Total Reward = 414.5666666666663
Episode 246: Total Reward = 132.09999999999994
Episode 247: Total Reward = 671.8499999999999
Episode 248: Total Reward = 335.09999999999974
Episode 249: Total Reward = 300.29999999999967
Episode 250: Total Reward = 282.2333333333331
Episode 251: Total Reward = 276.0999999999998
Episode 252: Total Reward = 243.49999999999991
Episode 253: Total Reward = 260.4666666666666
Episode 254: Total Reward = 252.8
Episode 255: Total Reward = 272.13333333333304
Episode 256: Total Reward = 239.2666666666666
Episode 257: Total Reward = 211.83333333333323
Episode 258: Total Reward = 201.99999999999991
Episode 259: Total Reward = 488.16666666666634
Episode 260: Total Reward = 198.36666666666653
Episode 261: Total Reward = 13.266666666666675
Episode 262: Total Reward = 79.7
Episode 263: Total Reward = 577.6499999999994
Episode 264: Total Reward = 134.00000000000009
Episode 265: Total Reward = 242.03333333333327
Episode 266: Total Reward = 301.3999999999999
Episode 267: Total Reward = 184.1333333333333
Episode 268: Total Reward = -13.399999999999999
Episode 269: Total Reward = 209.03333333333327
Episode 270: Total Reward = 143.16666666666666
Episode 271: Total Reward = 47.73333333333333
Episode 272: Total Reward = 608.4833333333331
Episode 273: Total Reward = 713.2166666666662
Episode 274: Total Reward = 262.39999999999986
Episode 275: Total Reward = 35.66666666666666
Episode 276: Total Reward = 319.06666666666644
Episode 277: Total Reward = 580.1833333333335
Episode 278: Total Reward = 635.8166666666665
Episode 279: Total Reward = 617.5500000000003
Episode 280: Total Reward = 49.73333333333335
Episode 281: Total Reward = 1017.7166666666665
Episode 282: Total Reward = 1282.1166666666695
Episode 283: Total Reward = 915.8833333333329
Episode 284: Total Reward = 218.8333333333332
Episode 285: Total Reward = 9.233333333333313
Episode 286: Total Reward = 459.2000000000001
Episode 287: Total Reward = 73.29999999999995
Episode 288: Total Reward = 150.8
Episode 289: Total Reward = 694.2166666666672
Episode 290: Total Reward = 225.73333333333323
Episode 291: Total Reward = 416.9666666666666
Episode 292: Total Reward = 755.2833333333332
Episode 293: Total Reward = 119.36666666666666
Episode 294: Total Reward = 892.3500000000006
Episode 295: Total Reward = 339.93333333333317
Episode 296: Total Reward = 48.56666666666666
Episode 297: Total Reward = 390.8
Episode 298: Total Reward = 52.06666666666666
Episode 299: Total Reward = 37.63333333333332
Episode 300: Total Reward = 73.76666666666662
```

```
Episode 301: Total Reward = 62.800000000000004
Episode 302: Total Reward = 86.93333333333332
Episode 303: Total Reward = 124.23333333333325
Episode 304: Total Reward = 254.26666666666668
Episode 305: Total Reward = 788.8166666666666
Episode 306: Total Reward = 515.333333333333
Episode 307: Total Reward = 609.2833333333331
Episode 308: Total Reward = 383.133333333333
Episode 309: Total Reward = 59.93333333333332
Episode 310: Total Reward = 16.000000000000004
Episode 311: Total Reward = 273.1833333333332
Episode 312: Total Reward = 125.33333333333327
Episode 313: Total Reward = 565.6166666666661
Episode 314: Total Reward = -3.1666666666666723
Episode 315: Total Reward = 189.73333333333323
Episode 316: Total Reward = 42.63333333333334
Episode 317: Total Reward = 543.9499999999998
Episode 318: Total Reward = 31.000000000000007
Episode 319: Total Reward = 392.59999999999985
Episode 320: Total Reward = 174.2666666666668
Episode 321: Total Reward = 16.666666666666664
Episode 322: Total Reward = 157.56666666666672
Episode 323: Total Reward = 58.16666666666668
Episode 324: Total Reward = 87.76666666666668
Episode 325: Total Reward = 204.50000000000003
Episode 326: Total Reward = 528.4833333333332
Episode 327: Total Reward = 117.26666666666674
Episode 328: Total Reward = 50.3
Episode 329: Total Reward = 26.299999999999994
Episode 330: Total Reward = 56.70000000000001
Episode 331: Total Reward = 61.93333333333332
Episode 332: Total Reward = 390.65000000000003
Episode 333: Total Reward = 250.2166666666663
Episode 334: Total Reward = 47.96666666666666
Episode 335: Total Reward = 57.76666666666667
Episode 336: Total Reward = 13.599999999999994
Episode 337: Total Reward = 107.53333333333333
Episode 338: Total Reward = 115.6333333333333
Episode 339: Total Reward = 106.10000000000002
Episode 340: Total Reward = 49.33333333333331
Episode 341: Total Reward = 12.133333333333336
Episode 342: Total Reward = -20.9
Episode 343: Total Reward = 65.13333333333334
Episode 344: Total Reward = 139.53333333333333
Episode 345: Total Reward = 45.233333333333306
Episode 346: Total Reward = 64.93333333333334
Episode 347: Total Reward = 64.33333333333334
Episode 348: Total Reward = 214.43333333333334
Episode 349: Total Reward = 108.36666666666686
Episode 350: Total Reward = 297.09999999999974
Episode 351: Total Reward = 18.499999999999996
Episode 352: Total Reward = 133.80000000000007
Episode 353: Total Reward = 4.76666666666666
Episode 354: Total Reward = 451.0499999999999
Episode 355: Total Reward = 579.4166666666665
Episode 356: Total Reward = 478.81666666666644
Episode 357: Total Reward = 114.63333333333328
Episode 358: Total Reward = 468.0833333333331
Episode 359: Total Reward = 315.8833333333331
Episode 360: Total Reward = 529.9499999999994
```

```
Episode 361: Total Reward = 494.78333333333285
Episode 362: Total Reward = 310.74999999999966
Episode 363: Total Reward = 665.5833333333333
Episode 364: Total Reward = 236.16666666666643
Episode 365: Total Reward = 475.0166666666663
Episode 366: Total Reward = 446.04999999999967
Episode 367: Total Reward = 143.9666666666667
Episode 368: Total Reward = 78.6666666666667
Episode 369: Total Reward = 79.13333333333333
Episode 370: Total Reward = 58.566666666666684
Episode 371: Total Reward = 171.3000000000001
Episode 372: Total Reward = 60.800000000000026
Episode 373: Total Reward = 172.2666666666666
Episode 374: Total Reward = 238.59999999999997
Episode 375: Total Reward = 243.8
Episode 376: Total Reward = 209.23333333333326
Episode 377: Total Reward = 57.03333333333333
Episode 378: Total Reward = 57.56666666666667
Episode 379: Total Reward = 336.8666666666664
Episode 380: Total Reward = 41.99999999999998
Episode 381: Total Reward = 48.00000000000001
Episode 382: Total Reward = 39.16666666666665
Episode 383: Total Reward = 46.366666666666674
Episode 384: Total Reward = 39.63333333333333
Episode 385: Total Reward = 118.23333333333338
Episode 386: Total Reward = 186.7333333333334
Episode 387: Total Reward = 185.53333333333333
Episode 388: Total Reward = 303.1999999999998
Episode 389: Total Reward = 202.6333333333332
Episode 390: Total Reward = 40.46666666666667
Episode 391: Total Reward = 218.53333333333336
Episode 392: Total Reward = 40.0
Episode 393: Total Reward = 31.26666666666667
Episode 394: Total Reward = 220.06666666666652
Episode 395: Total Reward = 319.9999999999999
Episode 396: Total Reward = 144.3
Episode 397: Total Reward = 121.26666666666672
Episode 398: Total Reward = 273.3999999999998
Episode 399: Total Reward = 189.1
Episode 400: Total Reward = 523.0499999999995
Episode 401: Total Reward = 222.56666666666678
Episode 402: Total Reward = 239.03333333333327
Episode 403: Total Reward = 190.3333333333333
Episode 404: Total Reward = 230.53333333333313
Episode 405: Total Reward = 690.5166666666668
Episode 406: Total Reward = 617.7499999999997
Episode 407: Total Reward = 63.76666666666668
Episode 408: Total Reward = 61.933333333333344
Episode 409: Total Reward = 343.7000000000001
Episode 410: Total Reward = 400.1499999999999
Episode 411: Total Reward = 66.2
Episode 412: Total Reward = 145.4333333333334
Episode 413: Total Reward = 8.066666666666652
Episode 414: Total Reward = 156.83333333333331
Episode 415: Total Reward = 102.83333333333337
Episode 416: Total Reward = 78.96666666666671
Episode 417: Total Reward = 179.1
Episode 418: Total Reward = 31.06666666666667
Episode 419: Total Reward = 86.03333333333336
Episode 420: Total Reward = 8.599999999999998
```

```
Episode 421: Total Reward = 22.200000000000003
Episode 422: Total Reward = 140.29999999999998
Episode 423: Total Reward = 222.2666666666667
Episode 424: Total Reward = 238.79999999999984
Episode 425: Total Reward = 168.93333333333325
Episode 426: Total Reward = -49.699999999999996
Episode 427: Total Reward = 155.46666666666658
Episode 428: Total Reward = 203.20000000000002
Episode 429: Total Reward = -19.83333333333333
Episode 430: Total Reward = 154.50000000000003
Episode 431: Total Reward = -51.03333333333332
Episode 432: Total Reward = -51.366666666666674
Episode 433: Total Reward = 415.88333333333316
Episode 434: Total Reward = 47.399999999999984
Episode 435: Total Reward = 153.79999999999995
Episode 436: Total Reward = 235.59999999999974
Episode 437: Total Reward = 69.53333333333332
Episode 438: Total Reward = 65.70000000000005
Episode 439: Total Reward = 369.6999999999997
Episode 440: Total Reward = 49.300000000000004
Episode 441: Total Reward = 150.16666666666646
Episode 442: Total Reward = 195.69999999999993
Episode 443: Total Reward = 91.53333333333332
Episode 444: Total Reward = 62.033333333333324
Episode 445: Total Reward = 108.20000000000003
```