Deep Q-Learning for Pusher-v2 Environment in OpenAI Gym

Abstract

This report presents an implementation of Deep Q-Network (DQN) for the Pusher-v2 environment in OpenAI Gym. The DQN agent is trained to manipulate the environment to achieve a specific goal. The training process consists of 100 episodes, each consisting of several time steps. The agent's memory is used to store experience tuples, which are used to update the Q-network during training. The performance of the agent is analysed by plotting the total rewards obtained per episode.

Introduction

The goal of this project is to train a DQN agent to solve the Pusher-v2 environment provided by OpenAI Gym. In this environment, the agent is required to control a robotic arm to push objects towards a target location. The agent receives a continuous state space describing the environment and must output continuous action values.

The implementation is based on the deep Q-learning algorithm, which combines Q-learning with deep neural networks to approximate the optimal action-value function. The agent's policy is derived from the action-value function and is used to select actions that maximize the expected cumulative rewards.

Methodology

Environment

The environment used in this project is Pusher-v2 provided by OpenAI Gym. The state_size and action_size are extracted directly from the environment's observation and action space, respectively.

DQN Agent

The DQN agent is initialized with the state and action size. The agent's settings include:

- A replay memory with a capacity to store experience tuples.
- An exploration rate (epsilon), which controls the balance between exploration and exploitation.
- A neural network architecture for approximating the action-value function.

Training

The agent is trained using the following steps:

- Initialize the environment and reset it to get the initial state.
- Loop through the episodes.
- For each episode, loop until the episode is done (terminal state is reached).
- Select an action using the agent's policy based on the current state.
- Execute the action in the environment and observe the next state, reward, and done flag.
- Calculate the total reward using the individual reward components provided by the environment.
- Store the experience (state, action, reward, next_state, done) in the agent's memory.
- Update the current state to the next state.

- If the agent's memory has enough samples, perform a batch update of the Q-network using the stored experiences.
- Print the episode's results, including the total reward and average loss.

Performance Evaluation

The performance of the DQN agent is evaluated by plotting the total rewards obtained per episode. The plot illustrates the agent's learning progress and provides insights into the effectiveness of the chosen parameters and architecture.

Results

The results of the training process are presented as follows:

episode: 0/100, e: 0.99, total_reward: 139.44

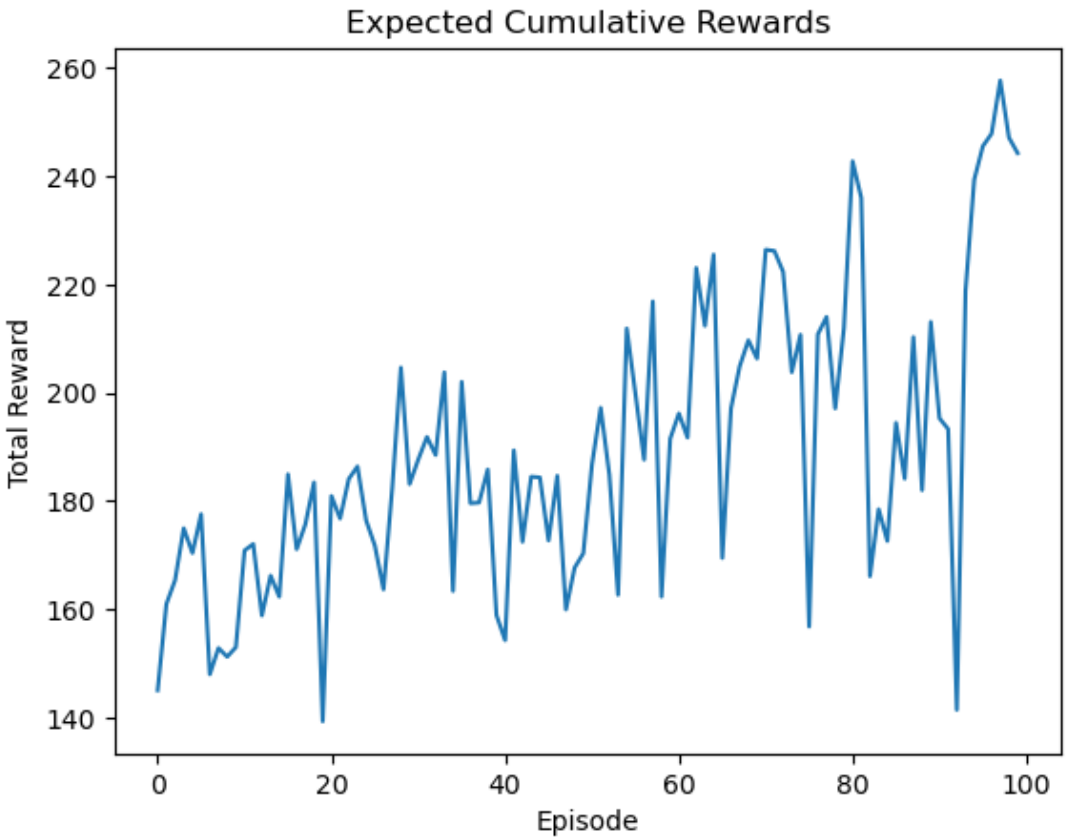
episode: 1/100, e: 0.99, total_reward: 150.93

...

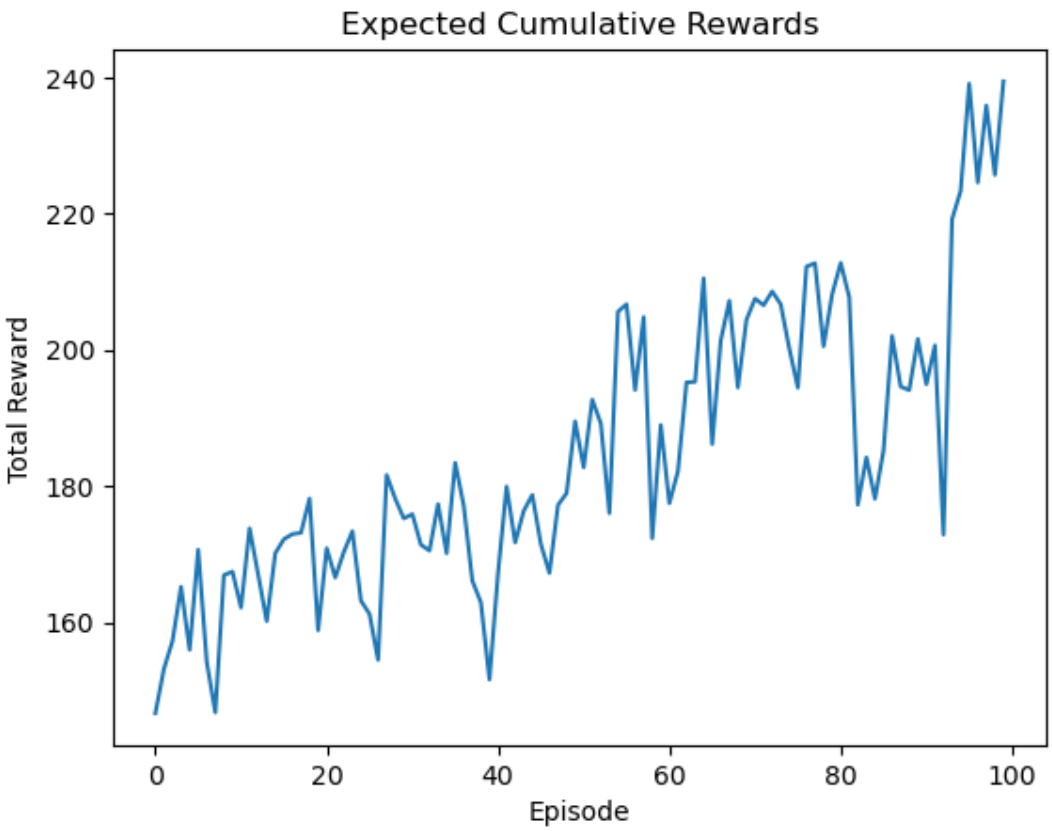
episode: 98/100, e: 0.01, total_reward: 243.15

episode: 99/100, e: 0.01, total_reward: 205.55

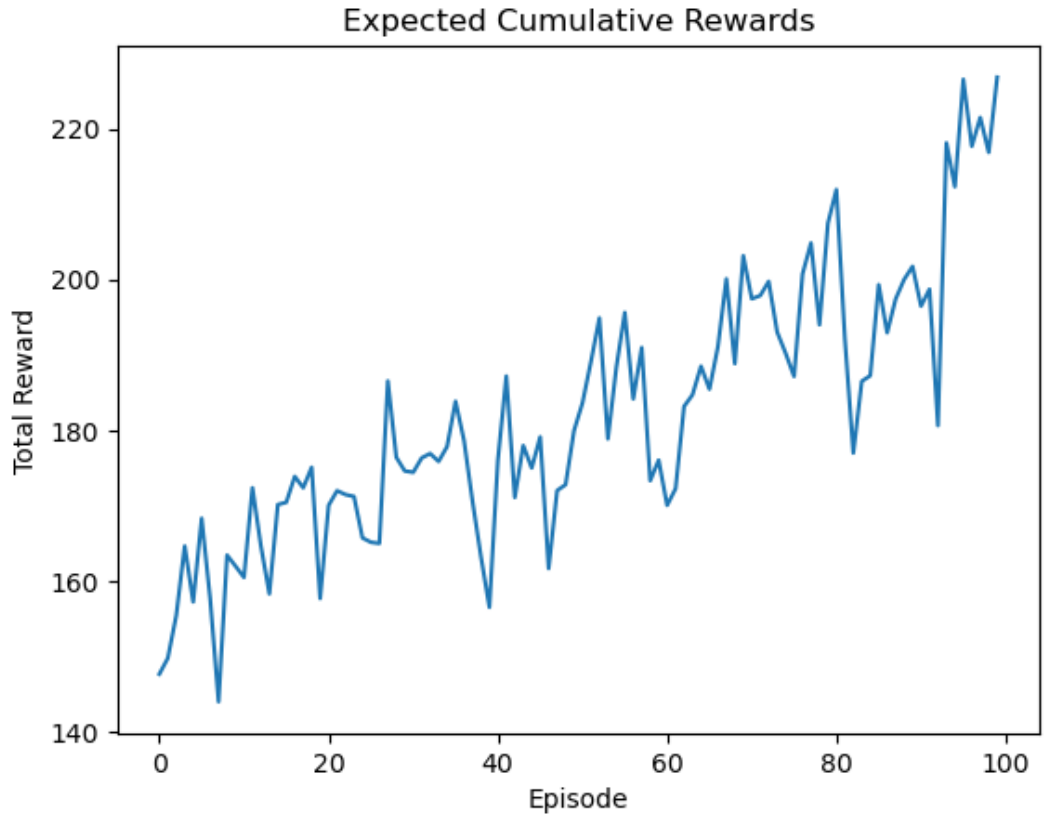
Plots



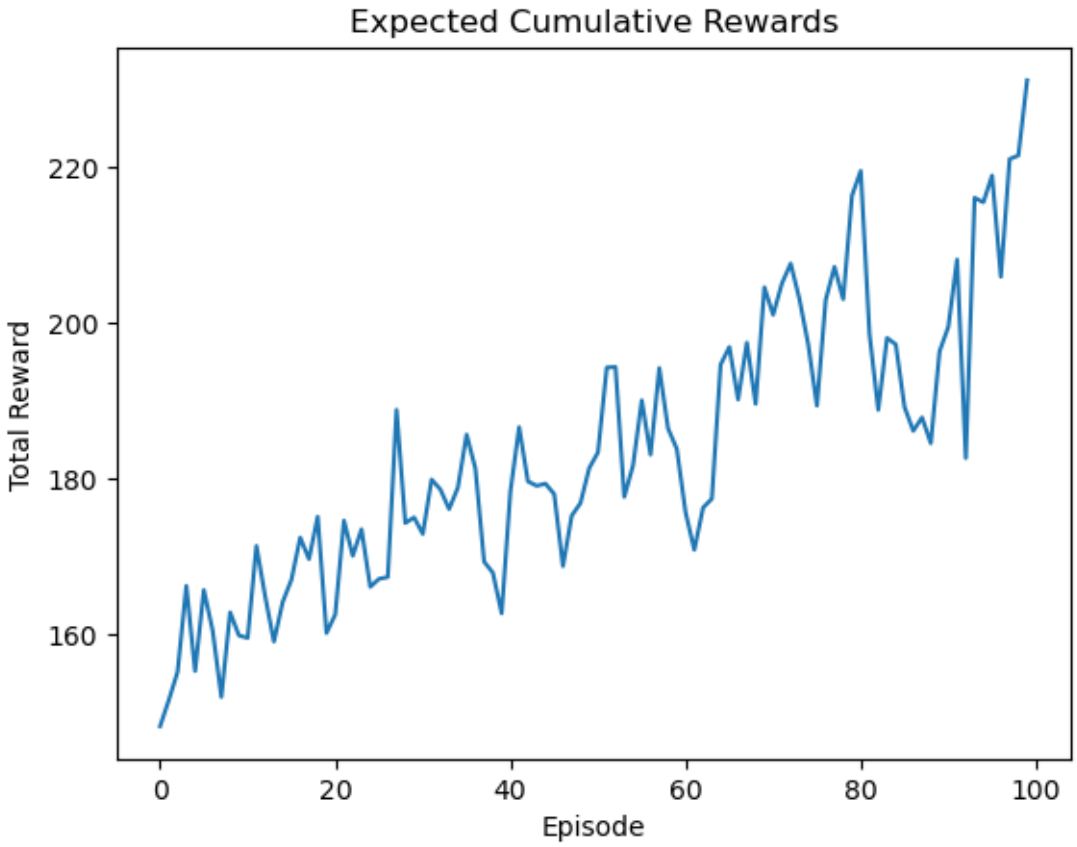
Plot: 2nd Iteration



Plot: 5th Iteration



Plot: 7th Iteration



Plot: 9th Iteration

The plots demonstrate the learning progress of the DQN agent. The agent starts with a positive total reward and gradually improves its performance over time. The fluctuations in the reward curves initially indicate that the agent is still exploring and updating its policy.

As we increase the number of iterations and average out the results, the graphs become fine-grained.

Conclusion

This report presents the implementation and results of training a DQN agent for the Pusher-v2 environment in OpenAI Gym. The agent demonstrates improvement in performance over the course of 100 episodes, indicating that the deep Q-learning algorithm is effective for this problem. Further improvements could potentially be achieved by tuning the hyperparameters, updating the neural network architecture, or employing other advanced techniques such as Double DQN or Dueling DQN.