Pong Game: Q-Learning vs. Deep Q-Learning

Baoying Feng
fengb1@arizona.edu

1 Introduction

This project examines the performance of Q-learning and Deep Q-learning agents in the classic game of Pong. The exploration aims to gain deeper insights into the distinct ways reinforcement learning algorithms function in a gaming environment. By comparing the interactions between Q-learning and Deep Q-learning agents in Pong, this study seeks to understand their strengths and limitations. Rigorous experimentation across varying levels of complexity offers a nuanced perspective on how these algorithms adapt to dynamic gaming environments.

2 Methodology

3.1 Experiment Setup

The experiment is implemented within a Flask web application, enabling seamless communication between the Pong game interface and reinforcement learning agents. The HTML interface presents the game environment while JavaScript functions handle paddle movement and ball collision detection. The Flask application computes rewards and adjusts paddle positions for both agents, ensuring consistent initial conditions by resetting all data, including Q-tables and neural network weights, at each test iteration.

3.2 Algorithms

The Q-learning algorithm employed in this experiment is a foundational reinforcement learning technique known for its simplicity and effectiveness in discrete action spaces. It maintains a Q-table, where each entry represents the expected cumulative reward for taking a specific action in a given state. As the agent interacts with the environment, the Bellman equation is used to update the Q-values iteratively, based on immediate and future rewards. While Q-learning can handle small, discrete state spaces directly, discretization is often applied to handle larger or continuous environments. This approach facilitates structured exploration and exploitation.

The learning rate (alpha) for Q-learning is set to 0.1, balancing stability and adaptability by controlling how much new information overrides existing knowledge. The discount factor (gamma), set at 0.99, ensures long-term rewards are prioritized. The exploration rate (epsilon), set to 0.1, strikes a balance between exploration and exploitation by occasionally selecting random actions to avoid getting stuck in local optima. The reward system is designed to encourage the agent to learn successful gameplay behaviors, providing +1 for hitting the ball and +0.5 for approaching it. This iterative learning process helps the agent gradually identify optimal strategies to maximize cumulative rewards.

Deep Q-learning builds on Q-learning principles by using deep neural networks to approximate the Q-function directly from raw state observations. The neural network allows the agent to learn intricate state-action mappings, particularly in high-dimensional and continuous state spaces, without requiring explicit discretization. This network architecture can offer more sophisticated decision-making abilities, especially in challenging environments. The primary network estimates Q-values for each action given the current state, while the target network provides a stable reference for training.

The learning rate (alpha) is set to 0.01, as higher rates cause the paddle to get stuck in specific positions like the bottom right corner. This learning rate helps control weight adjustments during backpropagation, preventing over-adjustment that could destabilize learning. The discount factor (gamma) is 0.99 to prioritize future rewards, and the exploration rate (epsilon) starts at 0.1 to encourage action exploration, gradually reducing over time as the agent gains confidence and begins exploiting optimal strategies. The reward system remains consistent with Q-learning, providing incentives for approaching (+0.5) and hitting (+1) the ball.

Deep Q-learning incorporates two additional mechanisms to stabilize learning. Experience replay stores past transitions in a buffer, which are randomly sampled to break temporal correlations and provide a broader range of training examples. The target network updates less frequently than the primary network to offer stable Q-value estimates, minimizing learning divergence and ensuring training consistency. By gradually minimizing the difference between predicted and target Q-values, the agent refines its policies over time.

Although both algorithms share similar principles, they differ significantly in implementation. Q-learning relies on discretization and direct updates to Q-tables, while Deep Q-learning uses neural networks to approximate complex policies, enabling adaptation to more challenging environments with high-dimensional inputs. Both algorithms iteratively refine their strategies to maximize cumulative rewards through exploration and exploitation.

3.3 Challenges

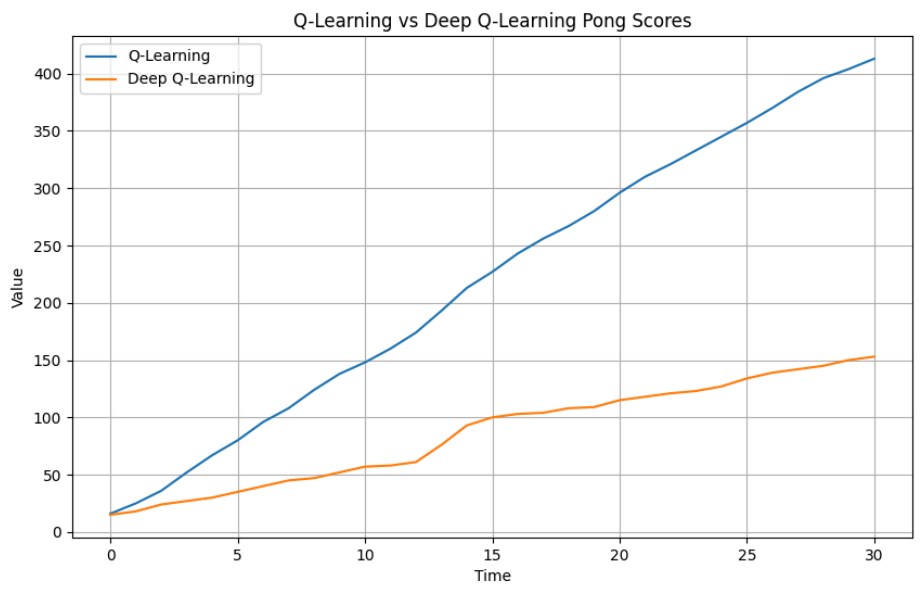
Implementing and fine-tuning these reinforcement learning algorithms presented several challenges. Difficulties included integrating them within the Pong environment, managing the exploration-exploitation balance, and optimizing hyperparameters. However, with iterative testing and guidance from external resources, ChatGPT, the setup was successfully established.

4 Result

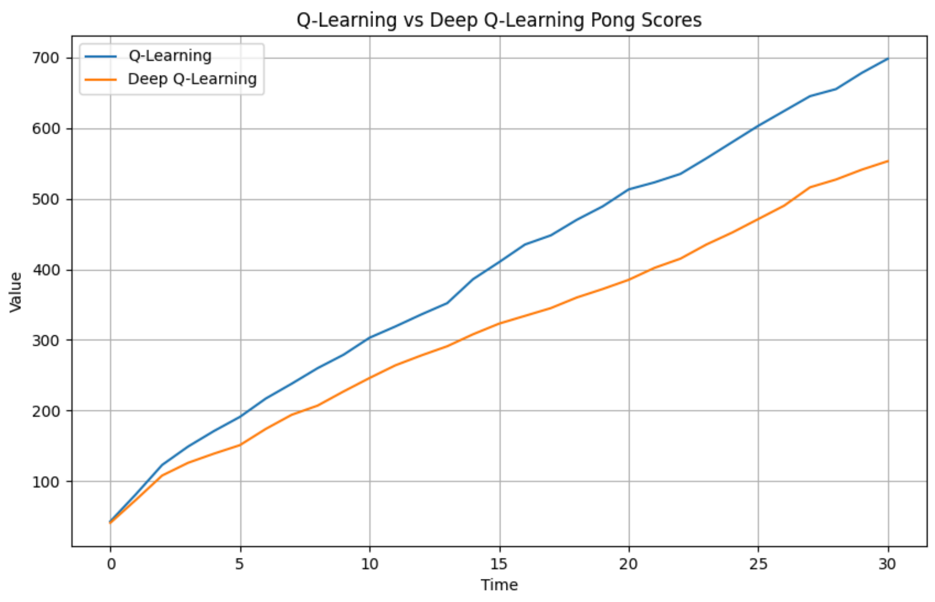
4.1 Result Graphs

The results are displayed through four graphs, each comparing Q-learning and Deep Q-learning agents in different Pong game scenarios:

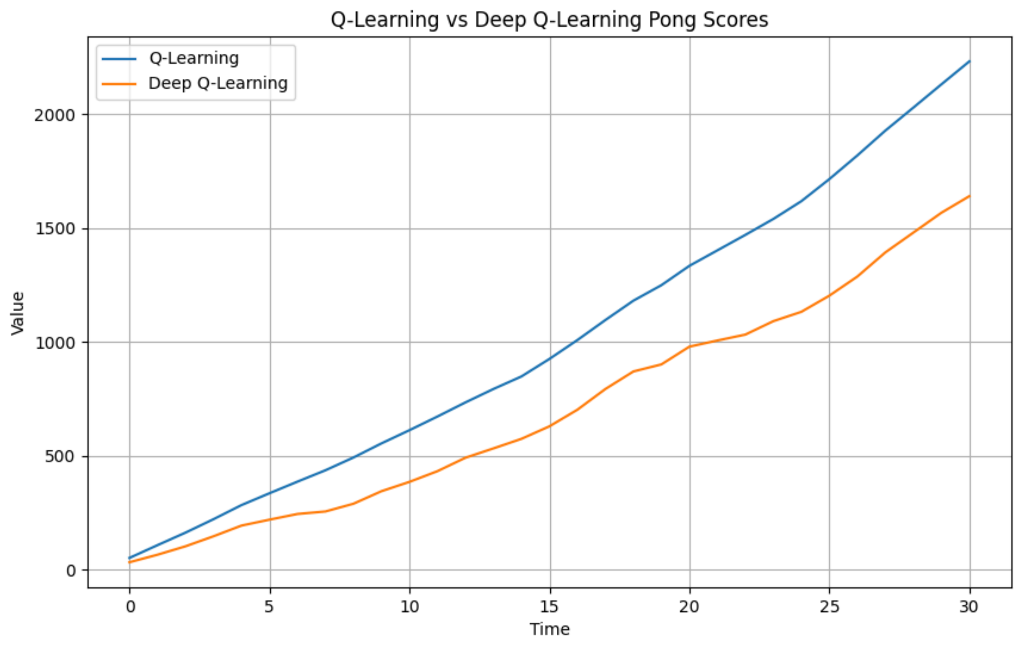
1. Single Ball, Large Paddle



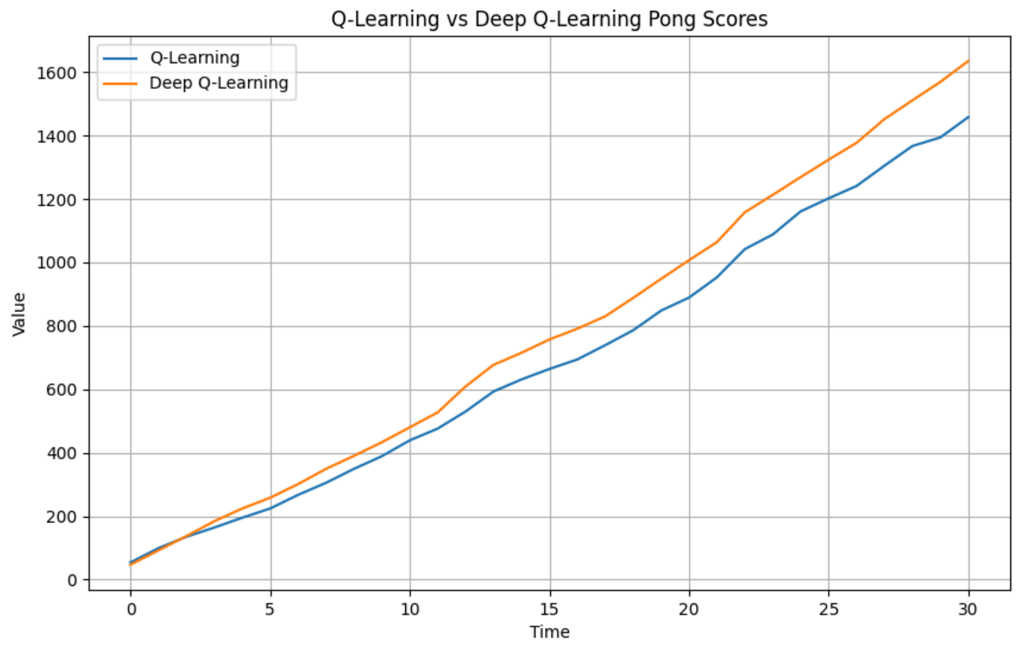
2. Three Balls, Large Paddle



3. Three Balls, Reduced Paddle Size



4. Five Balls, Small Paddle



4.2 Analysis

The result graphs offer valuable insights into the comparative performance of Q-learning and Deep Q-learning algorithms across various Pong scenarios. In the initial graphs, Q-learning consistently demonstrates superior adaptability and performance. In the scenario involving a single ball and a large paddle, Q-learning steadily improves its scores over time, showcasing rapid learning and reliable progress. Deep Q-learning also improves but at a slower and less consistent pace.

As complexity increases with three balls and a large paddle, Q-learning maintains steady improvement, while Deep Q-learning progresses similarly but lags behind. The performance gap widens when the paddle size is reduced, emphasizing Q-learning's adaptability and decision-making under tighter constraints. Deep Q-learning struggles in this scenario, revealing challenges in managing high-dimensional action spaces and requiring additional fine-tuning.

However, in the final scenario with five balls and a small paddle, Deep Q-learning overtakes Q-learning and maintains a consistent lead. Despite earlier results showing Q-learning outperforming Deep Q-learning, this result indicates that Deep Q-learning can adapt and thrive in complex environments with sufficient training data. The neural network model excels at identifying intricate patterns and multitasking strategies. This final graph illustrates that, while Q-learning may be advantageous in simpler environments, Deep Q-learning's ability to handle high-dimensional spaces becomes more apparent as complexity increases.

Overall, these results highlight the strengths and limitations of both approaches. Q-learning is more efficient and stable in simpler settings, while Deep Q-learning, with its robust neural network architecture, excels in complex environments where deeper exploration and learning are necessary.