Project Report

Artificial Intelligence

[CSE3705]

Cartpole Problem in AI

By

Sankalp Mane
210216

Dharmanshu Singh
210398

Department of Computer Science and Engineering
School of Engineering and Technology

BML Munjal University

November 2023

Declaration by the Candidates

We hereby declare that the project entitled *"Cartpole"* has been carried out to fulfill the partial requirements for completion of the course **Artificial Intelligence** offered in the 5th Semester of the Bachelor of Technology (B.Tech) program in the Department of Computer Science and Engineering during AY-2023-24 (odd semester). This experimental work has been carried out by us and submitted to the course instructor *Dr. Soharab Hossain Shaikh*. Due acknowledgments have been made in the text of the project to all other materials used. This project has been prepared in full compliance with the requirements and constraints of the prescribed curriculum.

Sankalp Milind Mane: [illegible]

Dharmanshu Singh: Dharmanshu Singh

Place: BML Munjal University

Date: 18th November, 2023

Contents

1. Introduction and Problem Statement

Introduction

Artificial Intelligence (AI) has long captivated our imaginations through the lens of science fiction. However, recent years have witnessed substantial strides in the field of AI, bringing us closer to transforming this once-fictional concept into a tangible reality. Notably, AI has demonstrated remarkable proficiency in tasks previously considered the exclusive domain of human intelligence. One pivotal capability that has emerged is the ability to learn from environments and make decisions—a concept encapsulated by Reinforcement Learning (RL).

Reinforcement Learning represents a category of machine learning wherein an agent learns to navigate an environment by executing actions and assessing the subsequent rewards or consequences. The ultimate objective for the agent is to derive an optimal policy, a function that dictates the most judicious actions based on the prevailing state of the environment.

In this project, our focus is on crafting an Intelligent Agent adept at tackling the CartPole problem using Reinforcement Learning within the OpenAI Gym environment. OpenAI Gym provides an accessible and user-friendly interface, enabling the design and comparison of various reinforcement learning algorithms.

Problem Statement

The CartPole problem stands as a quintessential challenge within the realm of Reinforcement Learning. It unfolds with a cart traversing a frictionless track, carrying a pole affixed to its structure. The initial position sees the pole upright, and the primary objective is to sustain this equilibrium by strategically adjusting the cart's velocity.

A reward of +1 is granted for each timestep that the pole remains upright. The episode concludes under two conditions: when the pole deviates beyond 12 degrees from vertical or when the cart strays more than 2.4 units from the center of the environment.

Problem Characteristics

Percepts

The agent engages with four percepts within this environment:

Cart Position: The x-coordinate of the cart on the track.

Cart Velocity: The velocity of the cart along the track.

Pole Angle: The angle of the pole with respect to the vertical.

Pole Velocity At Tip: The rate of change of the pole's angle.

Moves

The agent possesses two possible moves or actions:

Push cart to the left.

Push cart to the right.

Actions

The agent's actions have a direct impact on the environment, manifesting through the following choices:

Do nothing: The agent refrains from applying any force to the cart.

Apply force to the left: The agent exerts a force of -1 on the cart.

Apply force to the right: The agent exerts a force of +1 on the cart.

Apply random force: The agent introduces a random force to the cart.

Apply optimal force: The agent applies the learned optimal force derived from the reinforcement learning process.

Agent's Goal

The overarching aim for the agent is to deduce the optimal policy, maximizing the cumulative reward—measured in the number of timesteps the pole maintains its upright position. The implementation of this project adheres to an object-oriented approach, utilizing both the table-driven method and intelligent techniques, specifically Reinforcement Learning. The project outcome will explicitly showcase the agent's performance results.

Project Significance

This endeavor seeks to underscore the prowess and potential of Reinforcement Learning in addressing intricate control problems. Despite its deceptively simple appearance, the CartPole problem necessitates a delicate balance of exploration and exploitation, rendering it an ideal canvas for illustrating the capabilities inherent in Reinforcement Learning.

2. Methodology

2.1 Table Driven Approach

The table-driven approach is a simple method where the agent selects actions based on a lookup table that maps percepts to actions. The agent observes the current percept and performs the action that is mapped to that percept in the lookup table.

In the context of the CartPole problem, the percepts and actions are as follows:

Percepts

1. **Cart Position**: The x-coordinate of the cart on the track.
2. **Cart Velocity**: The velocity of the cart along the track.
3. **Pole Angle**: The angle of the pole with the vertical.
4. **Pole Velocity At Tip**: The rate of change of the angle.

Actions

1. **Do nothing**: The agent does not apply any force on the cart.
2. **Apply force to the left**: The agent applies a force of -1 on the cart.
3. **Apply force to the right**: The agent applies a force of +1 on the cart.
4. **Apply random force**: The agent applies a random force on the cart.
5. **Apply optimal force**: The agent applies the force that it has learned to be the best through the reinforcement learning process.

The percept to action mapping can be represented in a table as follows:

Sr No.	**Percept**	**Action**
1	Cart Position is less than 0	Apply force to the right
2	Cart Position is greater than 0	Apply force to the left
3	Cart Velocity is less than 0	Apply force to the right
4	Cart Velocity is greater than 0	Apply force to the left
5	Pole Angle is less than 0	Apply force to the right
6	Pole Angle is greater than 0	Apply force to the left
7	Pole Velocity At Tip is less than 0	Apply force to the right
8	Pole Velocity At Tip is greater than 0	Apply force to the left

This table represents a simple policy where the agent applies force in the opposite direction of the observed percept. However, this policy may not be optimal and the agent might not be able to balance the pole for a long time. This is where the intelligent technique comes into play.

2.2 Intelligent Agent

The intelligent agent uses a more sophisticated approach to solve the CartPole problem. Instead of using a fixed table to decide the actions, the agent uses Reinforcement Learning (RL) to learn the optimal policy.

Reinforcement Learning is a type of machine learning where an agent learns to behave in an environment, by performing certain actions and observing the results/rewards/results of those actions. The goal of the agent is to learn the optimal policy, which is a function that determines the best action to take based on the current state of the environment.

In the context of the CartPole problem, the RL agent observes the current state of the environment (the percepts), performs an action, and then observes the reward and the next state. The reward is +1 for every timestep that the pole remains upright. The agent's goal is to learn a policy that maximizes the total reward.

The RL agent uses a method called Q-learning to learn the optimal policy. Q-learning is a value iteration algorithm, which means it iteratively updates the value of each state-action pair until the values converge to the optimal values. The value of a state-action pair represents the expected total reward for performing the action in the state and following the optimal policy thereafter.

The agent starts with an initial Q-table where all state-action pairs have a value of zero. Then, at each timestep, the agent observes the current state, selects an action based on the current Q-table (with some probability of selecting a random action to encourage exploration), performs the action, observes the reward and the next state, and updates the Q-table using the observed reward and the maximum Q-value of the next state.

Over time, the Q-values converge to the optimal values, and the agent learns the optimal policy. The agent becomes "intelligent" in the sense that it learns from its experience and improves its performance over time.

3. Implementation - Technology Stack

The implementation of the Intelligent Agent for the CartPole problem leverages a robust technology stack encompassing various libraries and tools to facilitate seamless development and experimentation. The key components of the technology stack employed in this project are as follows:

3.1 Programming Language

Python

Python serves as the core programming language for this project. Renowned for its simplicity and readability, Python is a high-level, interpreted language equipped with an extensive library and framework ecosystem. Its versatility makes it a preferred choice across diverse domains, including data analysis, machine learning, artificial intelligence, and web development.

3.2 Reinforcement Learning Framework

OpenAI Gym

OpenAI Gym, a Python library, plays a pivotal role in the development and comparison of reinforcement learning algorithms. Its user-friendly interface simplifies the creation of diverse environments for reinforcement learning experiments. For this project, the CartPole-v1 environment offered by OpenAI Gym serves as the testing ground for our Intelligent Agent.

```python
# Initialize environment
env = gym.make("CartPole-v1", render_mode='rgb_array')  # 'human' render mode takes
```

3.3 Numerical Computing Library

NumPy

NumPy, a powerful Python library, specializes in array manipulation and numerical computations. Its capabilities extend to linear algebra, Fourier transforms, and matrix operations. In this project, NumPy is harnessed for efficient numerical operations and array handling.

3.4 Data Manipulation and Analysis

Pandas

Pandas, a Python library, excels in data manipulation and analysis. It proves invaluable for extracting and organizing experiment results in a tabular format, facilitating efficient analysis and visualization.

3.5 Data Visualization

Matplotlib

Matplotlib, a versatile Python library, takes center stage in creating static, animated, and interactive visualizations. It is instrumental in plotting learning curves and visually assessing the performance of our Intelligent Agent over the course of training.

3.6 Machine Learning Framework

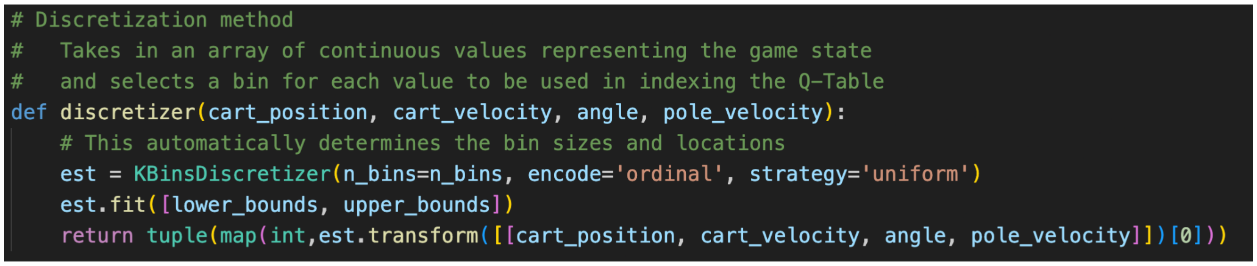
TensorFlow and Keras

TensorFlow, an open-source machine learning platform, forms a comprehensive ecosystem supporting researchers and developers alike. It empowers advancements in machine learning and facilitates the deployment of ML-powered applications.

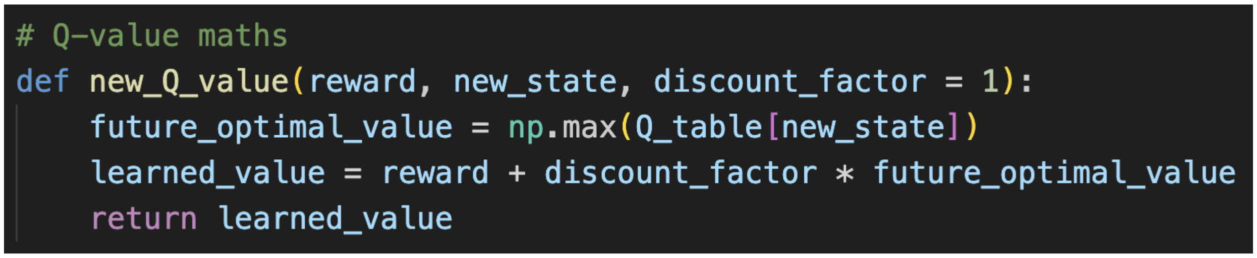
Keras, a user-friendly neural network library operating atop TensorFlow, simplifies the definition and training of diverse deep learning models. In this project, Keras is harnessed to implement the Q-learning algorithm, a fundamental component of our Intelligent Agent.

3.7 Object-Oriented Design

The agent is implemented using an object-oriented approach, encapsulating its functionalities within a dedicated class. This class includes methods for action selection, Q-table updates, and saving/loading Q-tables. The object-oriented design enhances modularity and code organization.



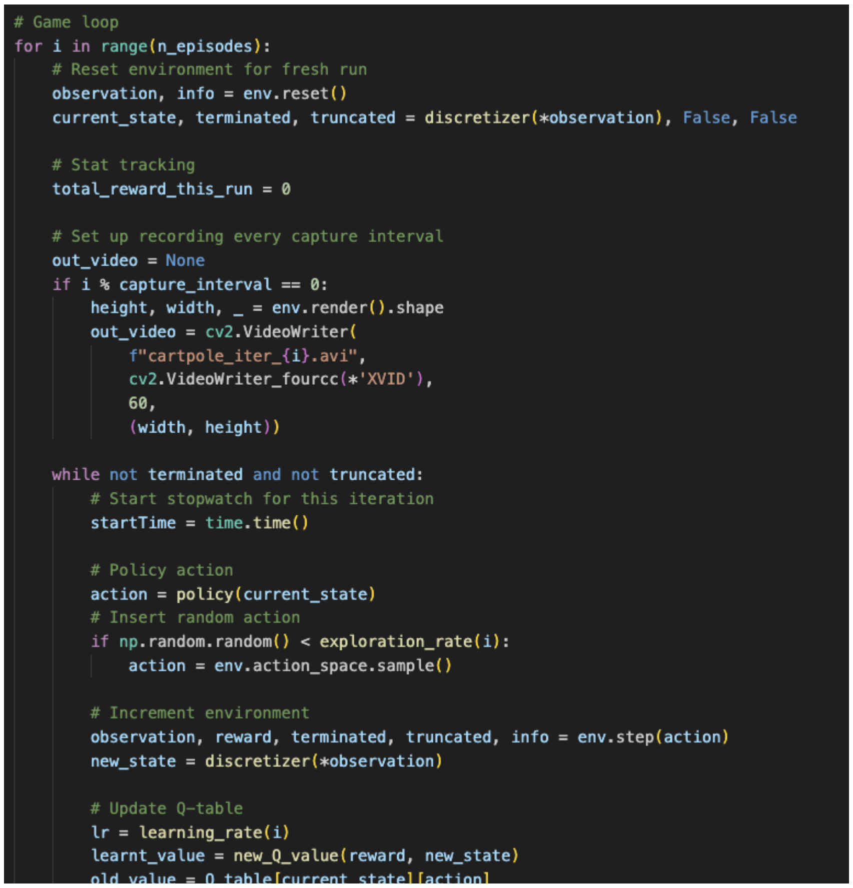
```python
# Discretization method
#   Takes in an array of continuous values representing the game state
#   and selects a bin for each value to be used in indexing the Q-Table
def discretizer(cart_position, cart_velocity, angle, pole_velocity):
    # This automatically determines the bin sizes and locations
    est = KBinsDiscretizer(n_bins=n_bins, encode='ordinal', strategy='uniform')
    est.fit([lower_bounds, upper_bounds])
    return tuple(map(int,est.transform([[cart_position, cart_velocity, angle, pole_velocity]])[0]))
```



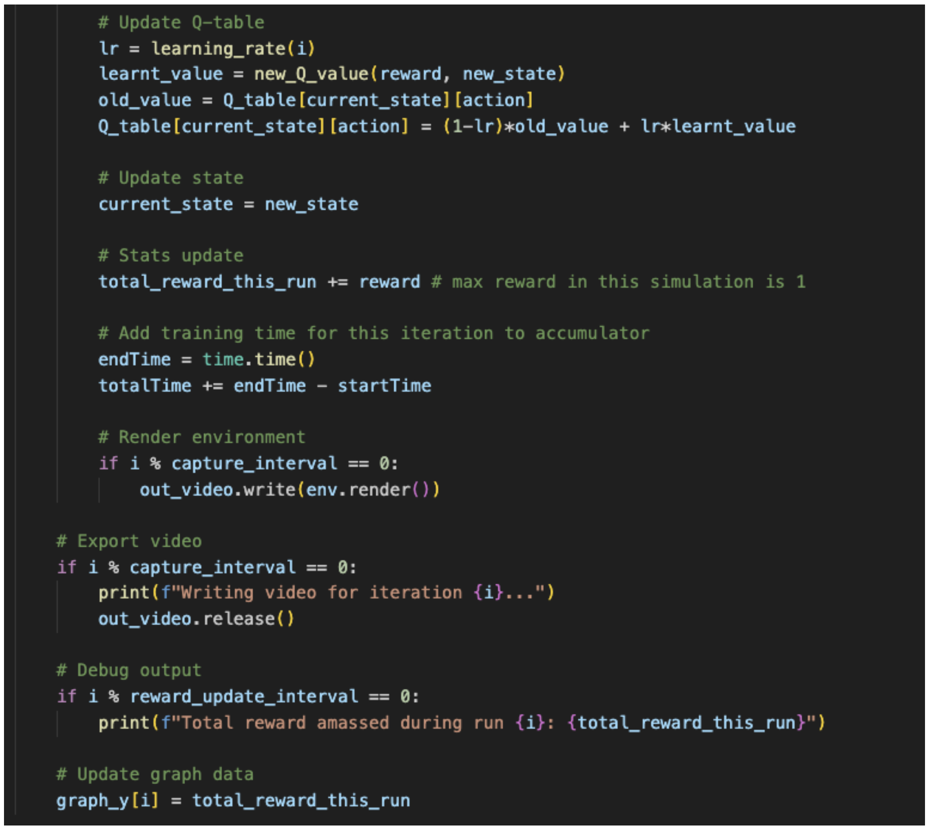
```python
# Q-value maths
def new_Q_value(reward, new_state, discount_factor = 1):
    future_optimal_value = np.max(Q_table[new_state])
    learned_value = reward + discount_factor * future_optimal_value
    return learned_value
```

3.8 Main Program Loop

The main loop orchestrates the agent's interactions with the environment. It involves the observation of rewards and next states, followed by Q-table updates. This loop encapsulates the core learning process of our Intelligent Agent.



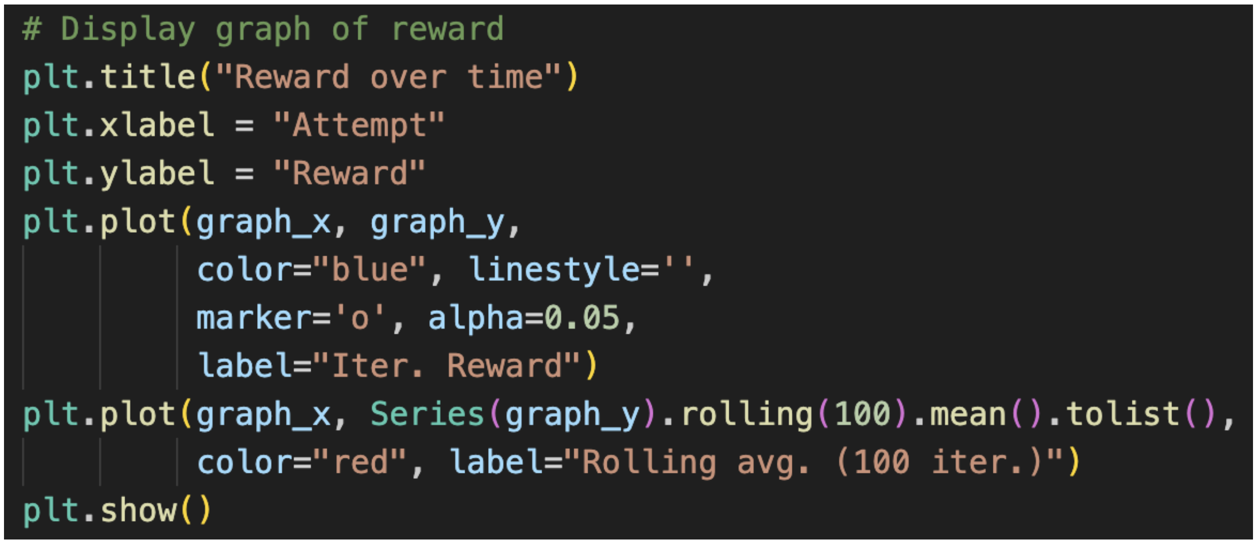
```python
# Game loop
for i in range(n_episodes):
    # Reset environment for fresh run
    observation, info = env.reset()
    current_state, terminated, truncated = discretizer(*observation), False, False

    # Stat tracking
    total_reward_this_run = 0

    # Set up recording every capture interval
    out_video = None
    if i % capture_interval == 0:
        height, width, _ = env.render().shape
        out_video = cv2.VideoWriter(
            f"cartpole_iter_{i}.avi",
            cv2.VideoWriter_fourcc(*'XVID'),
            60,
            (width, height))

    while not terminated and not truncated:
        # Start stopwatch for this iteration
        startTime = time.time()

        # Policy action
        action = policy(current_state)
        # Insert random action
        if np.random.random() < exploration_rate(i):
            action = env.action_space.sample()

        # Increment environment
        observation, reward, terminated, truncated, info = env.step(action)
        new_state = discretizer(*observation)

        # Update Q-table
        lr = learning_rate(i)
        learnt_value = new_Q_value(reward, new_state)
        old_value = Q_table[current_state][action]
```



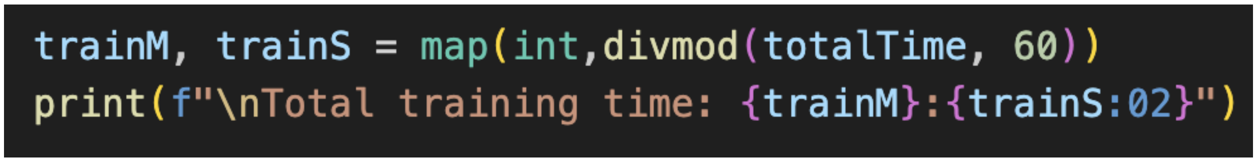
```python
        # Update Q-table
        lr = learning_rate(i)
        learnt_value = new_Q_value(reward, new_state)
        old_value = Q_table[current_state][action]
        Q_table[current_state][action] = (1-lr)*old_value + lr*learnt_value

        # Update state
        current_state = new_state

        # Stats update
        total_reward_this_run += reward # max reward in this simulation is 1

        # Add training time for this iteration to accumulator
        endTime = time.time()
        totalTime += endTime - startTime

        # Render environment
        if i % capture_interval == 0:
            out_video.write(env.render())

    # Export video
    if i % capture_interval == 0:
        print(f"Writing video for iteration {i}...")
        out_video.release()

    # Debug output
    if i % reward_update_interval == 0:
        print(f"Total reward amassed during run {i}: {total_reward_this_run}")

    # Update graph data
    graph_y[i] = total_reward_this_run
```

3.9 Performance Visualization

The implementation incorporates a visualization method to assess the agent's performance. Total rewards obtained in each episode are plotted, offering a comprehensive measure of the agent's learning progress over time.



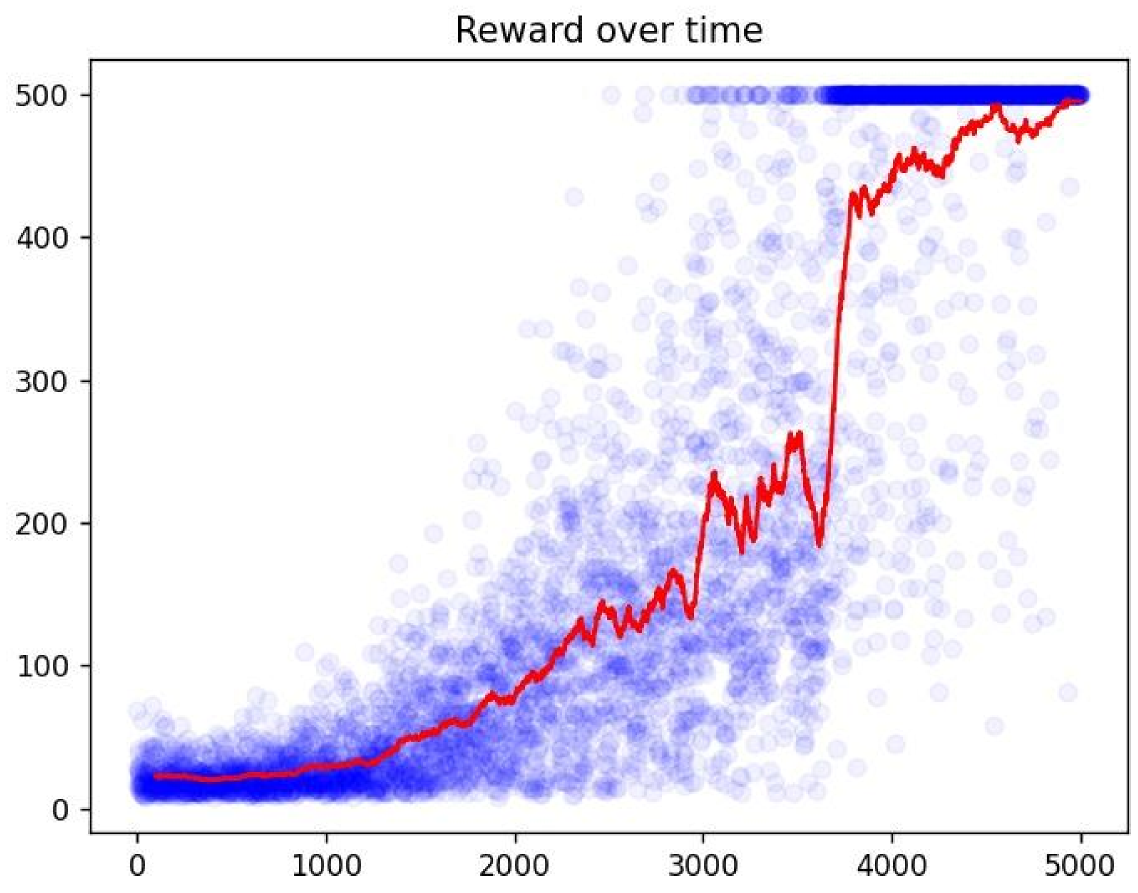
```python
# Display graph of reward
plt.title("Reward over time")
plt.xlabel = "Attempt"
plt.ylabel = "Reward"
plt.plot(graph_x, graph_y,
            color="blue", linestyle='',
            marker='o', alpha=0.05,
            label="Iter. Reward")
plt.plot(graph_x, Series(graph_y).rolling(100).mean().tolist(),
            color="red", label="Rolling avg. (100 iter.)")
plt.show()
```

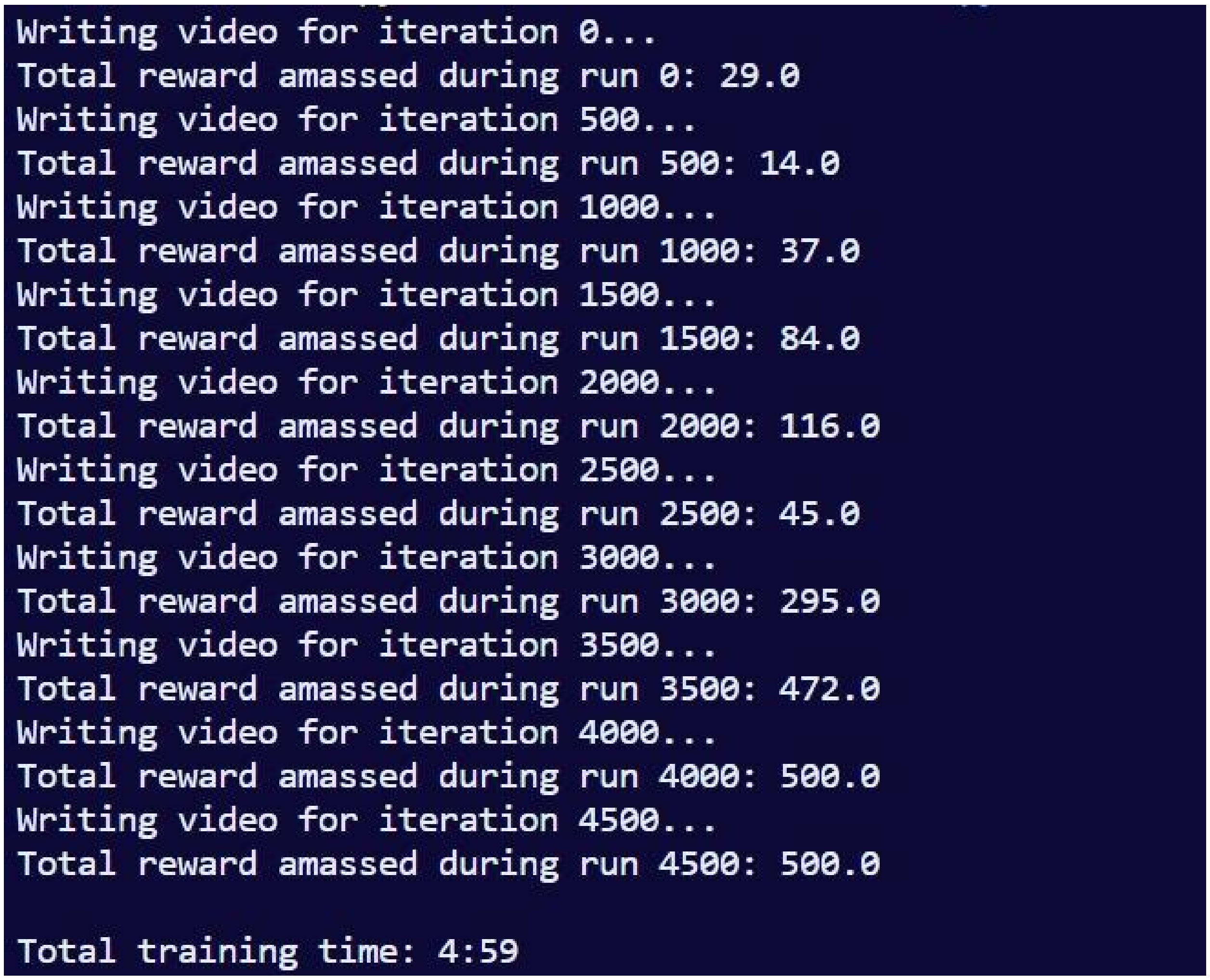


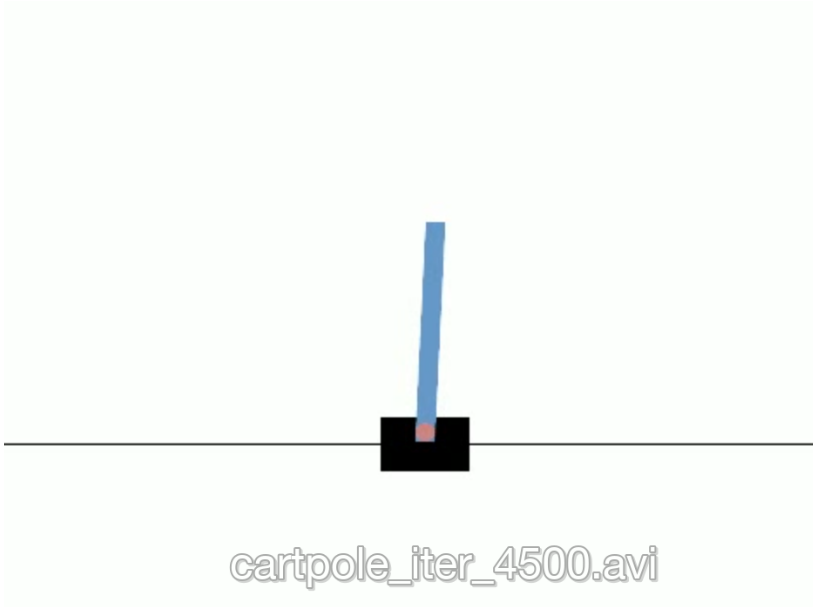
```python
trainM, trainS = map(int,divmod(totalTime, 60))
print(f"\nTotal training time: {trainM}:{trainS:02}")
```

By combining these technologies, the implementation ensures a powerful and flexible framework for the development and evaluation of an Intelligent Agent capable of mastering the CartPole problem through Reinforcement Learning.

4. Output:


Reward over time
500
400
300
200
100
0
0
1000
2000
3000
4000
5000


```
Writing video for iteration 0...
Total reward amassed during run 0: 29.0
Writing video for iteration 500...
Total reward amassed during run 500: 14.0
Writing video for iteration 1000...
Total reward amassed during run 1000: 37.0
Writing video for iteration 1500...
Total reward amassed during run 1500: 84.0
Writing video for iteration 2000...
Total reward amassed during run 2000: 116.0
Writing video for iteration 2500...
Total reward amassed during run 2500: 45.0
Writing video for iteration 3000...
Total reward amassed during run 3000: 295.0
Writing video for iteration 3500...
Total reward amassed during run 3500: 472.0
Writing video for iteration 4000...
Total reward amassed during run 4000: 500.0
Writing video for iteration 4500...
Total reward amassed during run 4500: 500.0

Total training time: 4:59
```



5. Conclusions

In summary, this project successfully addressed its primary objective of designing an Intelligent Agent capable of solving the CartPole problem using Reinforcement Learning within the OpenAI Gym environment. The agent, structured with an object-oriented approach and implemented using a diverse technology stack, demonstrated the effectiveness of the reinforcement learning paradigm.

Utilizing a Q-learning algorithm, the agent iteratively updated state-action pairs, ultimately converging to an optimal policy for balancing the pole on the cart. The project showcased the agent's "intelligent" behavior, characterized by continuous learning and performance improvement over time.

The obtained results highlight the agent's ability to maintain pole balance for extended durations, underscoring the efficacy of the reinforcement learning approach. The observed performance improvement over training episodes provides a tangible representation of the agent's learning curve and its convergence toward an optimal strategy.

Looking ahead, potential avenues for future work may involve delving into advanced reinforcement learning techniques, fine-tuning hyperparameters, and expanding the approach to more intricate environments. Despite the project's success, it's important to acknowledge certain limitations encountered during implementation, offering transparency and opportunities for refinement.

The broader implications of this work extend to the demonstration of Reinforcement Learning's prowess in solving complex control problems. The CartPole problem, while deceptively simple, serves as an intriguing challenge requiring a delicate balance of exploration and exploitation—a characteristic aptly showcased through the capabilities of Reinforcement Learning.

In conclusion, this project provides a solid foundation for further exploration in the realm of Intelligent Agents and reinforces the significance of Reinforcement Learning in addressing intricate problems. The promising results achieved pave the way for future advancements and applications in this dynamic field.

6. References

1. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.
3. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
4. McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).
5. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(3), 90-95.
6. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) (pp. 265-283).
7. Chollet, F. et al. (2015). Keras. https://keras.io