Exploring Gymnasium Environments with Reinforcement Learning

CSCI 166, Fall 2024

Dr. David Ruby

Izaiah Benavides, Charith Kondapally, Mateo Nuñez

December 2, 2024

Table of Contents

DOMAIN SECTION AND SETUP..... 3

MODEL DESIGN AND IMPLEMENTATION..... 4

HYPERPARAMETER TUNING AND EXPERIMENTATION..... 5

RESULTS ANALYSIS AND EVALUATION..... 6

Deep Q-Learning..... 9

NOVEL CONTRIBUTIONS AND FUTURE WORK..... 11

REFERENCES..... 12

The project's GitHub location is on the following link:

https://github.com/mnunezr1/CSCI-166-Project/tree/main

The Google Collab link is the following:

https://colab.research.google.com/drive/1bk0Pas9utFW7jvYfMB-tvPF4FVhIyplX?usp=sharing

DOMAIN SECTION AND SETUP

In this project, our team chose 3 separate learning environments from Gymnasium. Frozen Lake for Q-Learning, Mountain Car Continuous for Approx. Q-Learning, and Lunar Landing for DQN. All of these domains can apply to reinforcement learning.

Frozen Lake is an environment that presents a straightforward, discrete-state problem, ideal for understanding the basics of Q-Learning. The agent must get to a goal, avoiding holes that could lead to failure. The state space is represented by grid positions, and the action space includes four possible moves: left, right, up, and down. The reward structure is binary, offering a reward of +1 for reaching the goal and 0 otherwise.

For Approximate Q-Learning, we selected the Mountain Car Continuous environment, which involves a car that must drive up a steep hill but lacks the power to do so directly. Instead, the car must build momentum by moving back and forth. The state space in this environment includes the car's position and velocity. The reward structure includes a negative reward for each time step to encourage efficiency.

The Lunar Lander is a classic problem that provides a rich and challenging test for reinforcement learning algorithms, particularly those involving continuous action spaces and complex dynamics. The state space in the Lunar Lander environment includes a variety of parameters: the position of the lander (x and y coordinates), its velocity (x and y components), its angle and angular velocity, and two boolean flags indicating whether each leg is in contact with the ground. This combination of continuous and discrete elements makes it an ideal candidate for exploring Deep Q-learning techniques. The action space in Lunar Lander is discrete, consisting of four possible actions: do nothing, fire left engine, fire main engine, and fire right engine. The reward structure is designed to incentivize soft landings on a specified landing pad. Specifically, the agent receives a reward of +100 for successfully landing, a reward of -100 for crashing, and a small negative reward (-1) for

each frame to discourage unnecessary movements and fuel wastage. The goal of the problem is to land the lunar module softly on the designated landing pad without crashing.

MODEL DESIGN AND IMPLEMENTATION

The project involved experimenting through different reinforcement learning models: starting with Q-Learning, moving to Approximate Q-Learning, and finally implementing Deep Q-Learning for more complex scenarios.

Q-Learning: For Q-Learning, we implemented a model-free algorithm that learns the value of state-action pairs directly. This initial phase allowed us to understand the fundamentals of reinforcement learning, including how state-action pairs are updated using the Bellman equation. In the Frozen Lake environment, we used the slippery version, which added stochastic transitions to the agent's movement, increasing task complexity. The Q-Learning algorithm was implemented with a discrete state-action space, allowing us to construct and update a Q-table. To balance exploration and exploitation, we used Epsilon-Greedy action selection, where the agent occasionally chose a random action (exploration) while primarily selecting actions that maximize its expected reward (exploitation). This approach enabled the agent to explore the environment while converging toward an optimal policy.

Approximate Q-Learning: As we progressed to larger and continuous state spaces, such as those encountered in the Mountain Car Continuous environment, we transitioned to Approximate Q-Learning using linear function approximation. This approach allowed us to handle the impracticality of maintaining a full Q-table by approximating the Q-values using a linear combination of features. For both environments, we designed a feature set that captured the essential aspects of the state, such as position, velocity, angle, and other relevant dynamics. To balance exploration and exploitation, we implemented Epsilon-Greedy action selection, where the agent occasionally chose random actions (exploration) while primarily selecting actions that maximize expected rewards (exploitation). The Approximate Q-Learning model enabled us to better generalize from seen to unseen states, improving learning efficiency and performance in these more complex environments.

Deep Q-Learning: For the most advanced stage of our project, we implemented Deep Q-Learning (DQN) using neural networks. This approach is particularly suited for environments with high-dimensional state spaces, like the Lunar Lander, where the complexity of the state space makes traditional Q-Learning and linear approximations infeasible. Our DQN architecture consisted of several fully connected layers with ReLU activation functions. The input to the network was the state vector, and the output was a set of Q-values corresponding to the possible actions. We used the Adam optimizer for training, with a mean squared error loss function. Key hyperparameters included the learning rate, discount factor, and epsilon for the epsilon-greedy exploration strategy. Implementing DQN involved several challenges, such as handling the instability of training deep networks in reinforcement learning contexts. We employed techniques like experience replay and target networks to stabilize the learning process. Experience replay allows the agent to store past experiences and sample from them randomly during training, breaking the correlation between consecutive updates and leading to more stable learning. Target networks, which are updated less frequently than the main network, help to mitigate oscillations and divergence in Q-value estimates

HYPERPARAMETER TUNING AND EXPERIMENTATION

Throughout the experimentation process, we documented the impact of each hyperparameter on the model's performance. This documentation included notes on the configurations tested and their observed effects on learning dynamics and overall performance. By recording these details, we identified the most effective hyperparameter settings and gained insights into how different parameters influenced the agent's behavior. The tuning was performed in the Frozen Lake environment due to its simplicity and efficiency; other environments took significantly longer to compile and lacked the clear reward structure of Frozen Lake, making it ideal for initial experimentation. After determining the optimal settings, we applied the tuned parameters to other models to validate their effectiveness. Below are the original hyperparameters and their values. We focused on tuning the learning rate (α) and discount factor (γ), keeping all other parameters consistent throughout the tuning process.

Original Hyperparameters and Values:

```
alpha = 0.01  # Learning rate
gamma = 0.99  # Discount factor
epsilon = 1.0  # Exploration rate for Q_Learning
apx_epsilon = 0.1 #For Approx. Q_Learning
epsilon_min = 0.01 #For Q_Learning
epsilon_decay = 0.995 #For Q_Learning
num_training_episodes = 10000
num_evaluation_episodes = 1000
max_steps = 300 #For Approximate Q_learning and Q_Learning
```

Tuning Alpha (α):

Experiments were conducted on the Frozen Lake Environment with α values of 0.001, 0.01, and 0.1. At α=0.001 the maximum average reward did not exceed 0.09, with episode lengths stabilizing between 7 and 9, occasionally reaching a maximum of 12. For α=0.01, rewards oscillated, with the maximum average reward not exceeding 0.07. Episode lengths were more consistent, averaging between 6 and 6.8. The best performance was observed at α=0.1, where average rewards steadily converged to 1.0, and episode lengths consistently ranged from 35 to 40. Based on these results, α=0.1 was selected as the optimal value for learning rate.

Tuning Gamma (γ):

Experiments were conducted with γ values of 0.95, 0.99, and 0.995. At γ=0.95, average rewards oscillated heavily between 0.06 and 0.11, with episode lengths ranging from 8 to 10. For γ=0.99, rewards oscillated between 0.04 and 0.08, with episode lengths increasing slightly to a range of 9 to 11. At γ=0.995, rewards oscillated between 0.03 and 0.07, while episode lengths ranged from 11 to 14. Due to the more favorable reward trends and episode lengths at γ=0.95, it was chosen as the optimal discount factor.

These observations demonstrate how careful tuning of hyperparameters significantly impacts both learning efficiency and performance.

RESULTS ANALYSIS AND EVALUATION

Q-Learning

For the Q-Learning implementation in the Frozen Lake environment, we analyzed the agent's learning progression over 10,000 episodes. The left side of Figure 1 shows the last frame of the Frozen Lake environment and the learned Q-values for each state. The left subfigure displays the final position of the agent in the Frozen Lake grid, where it reached the goal in approximately 50% of the runs within the last 1,000 episodes. The right subfigure visualizes the learned Q-values, with arrows indicating the best action to take in each state according to the learned policy. The color gradient indicates the magnitude of the Q-values, with darker shades representing higher values. This visualization reveals that the agent consistently learned to favor actions leading to the goal, particularly in states closer to the goal.

In Figure 2, the left subfigure presents the average rewards per episode over time. The x-axis represents the number of episodes (from 0 to 10,000), while the y-axis shows the average rewards. Initially, the average rewards were around 0.1, reflecting the agent's exploratory phase and frequent failures. However, as learning progressed, we observed a steady increase in the average rewards, reaching around 0.7 by the 7,000th episode and stabilizing close to 0.8 towards the end. This upward trend in rewards indicates that the agent gradually improved its policy, achieving the goal more consistently as episodes progressed.

The right subfigure in Figure 2 illustrates the average episode lengths over time. The x-axis represents the number of episodes (from 0 to 10,000), while the y-axis displays the average episode lengths. Initially, the episode lengths increased gradually, corresponding to the agent's extensive exploration and frequent retries. As the agent's policy improved, the average episode lengths decreased significantly, dropping to around 40 steps by the 7,000th episode and stabilizing between 35 and 40 steps towards the end. This reduction in episode lengths demonstrates the agent's increasing proficiency in navigating the Frozen Lake environment,

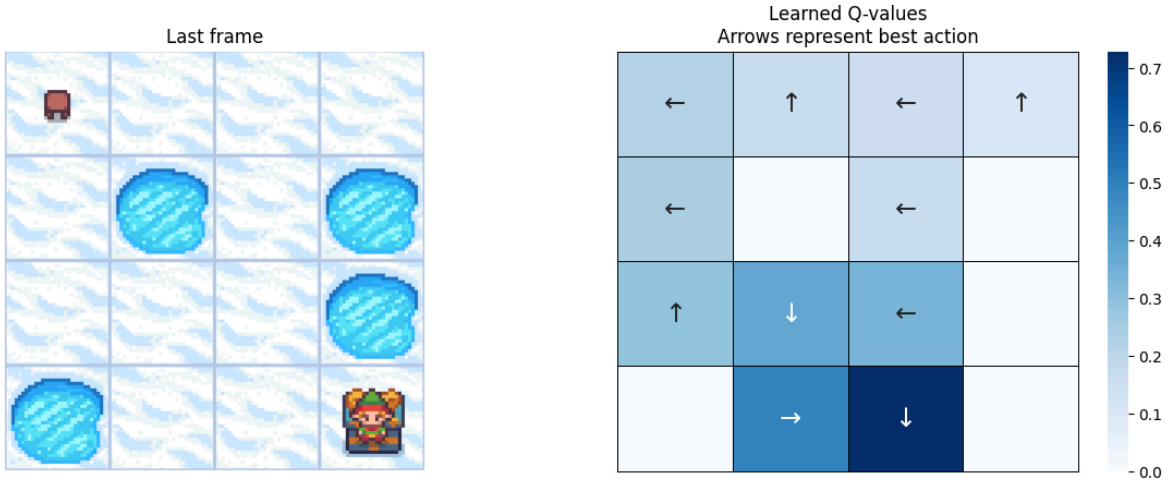


Figure 1: Last Frame and Learned Policy for the Frozen Lake Environment using Q-Learning

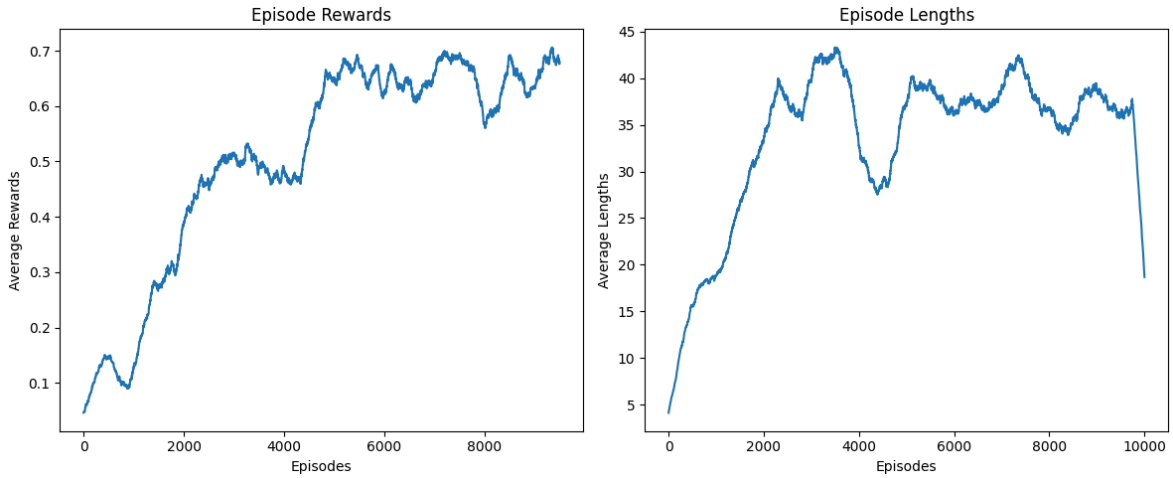


Figure 2: Average Rewards and Lengths per Episode for the Frozen Lake Environment using Q-Learning

Approximate Q Learning

For the Approximate Q-Learning implementation in the Mountain Car Continuous environment, we observed significant learning progression over 10,000 episodes, as illustrated in the provided figures. The first part of Figure 3 displays the policy mapping for the Mountain Car Continuous environment. The x-axis represents the position ranging from -1.25 to 0.50, and the y-axis represents the velocity ranging from -0.08 to 0.08. The policy mapping is visualized with a grid of red lines, indicating the direction of the optimal actions learned by the agent. This visualization highlights the regions where the agent has learned to apply force effectively to build momentum and reach the goal. The right part of the figure shows the last frame of the Mountain Car environment, where the car is positioned on the hill with a flag at the top, indicating the goal.

The left graph in Figure 4 presents the average rewards per episode over time. The x-axis represents the episode number (from 0 to 10,000), and the y-axis shows the average reward. Initially, the average rewards were around -22, reflecting the agent's struggle to reach the goal efficiently. However, as learning progressed, we observed fluctuations in the rewards, with a general upward trend. By the 5,000th episode, the average rewards improved to approximately -15, indicating that the agent was learning to perform better and reach the goal more consistently. Towards the end of the training, the average rewards stabilized around -25.

The right graph in Figure 4 illustrates the average episode lengths over time. The x-axis represents the episode number (from 0 to 10,000), and the y-axis displays the average length of episodes. Initially, the episode lengths were around 290 steps, as the agent struggled to build the necessary momentum to reach the goal. Over time, there were noticeable fluctuations in episode lengths, but a general decreasing trend was observed. By the 3,000th episode, the average episode length was reduced to about 288 steps. Towards the end of the training, the episode lengths fluctuating, ending around 296 steps. The small range of episode lengths indicates the agent's improved efficiency in solving the problem, effectively applying learned policies to navigate the continuous state space.

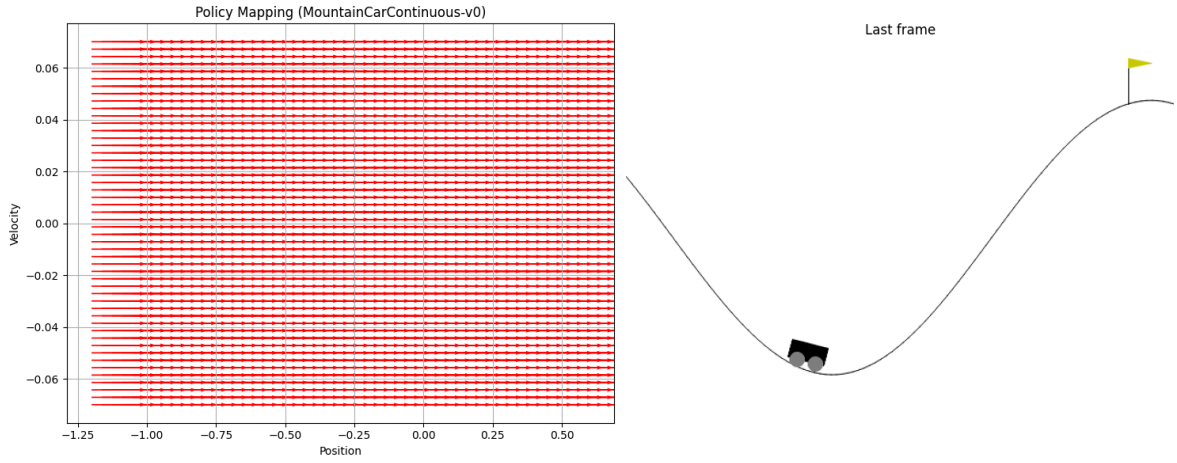


Figure 3: Last Frame and Learned Policy for the Mountain Car Continuous Environment using Approximate Q-Learning

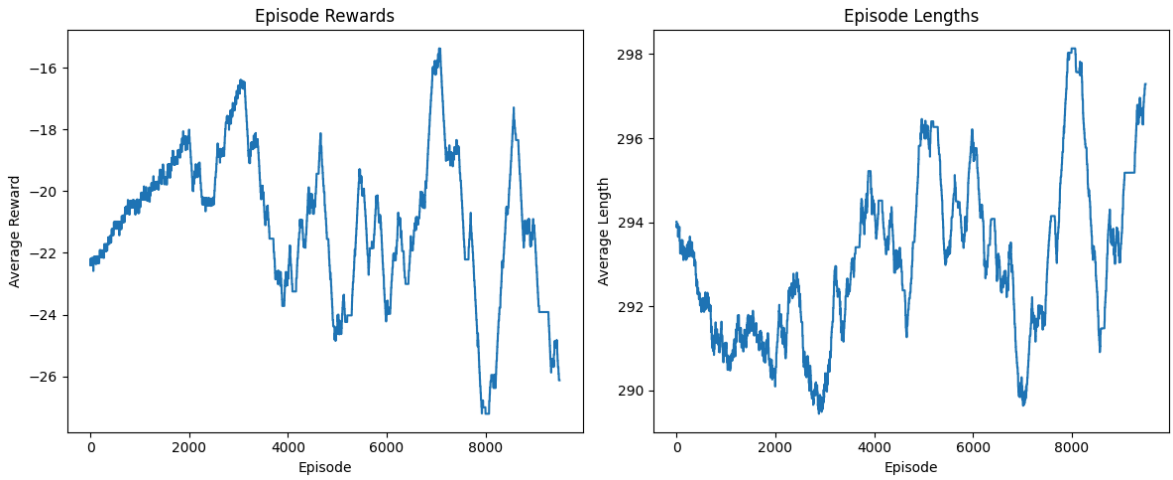


Figure 4: Average Rewards and Lengths per Episode for the Mountain Car Continuous Environment using Approximate Q-Learning

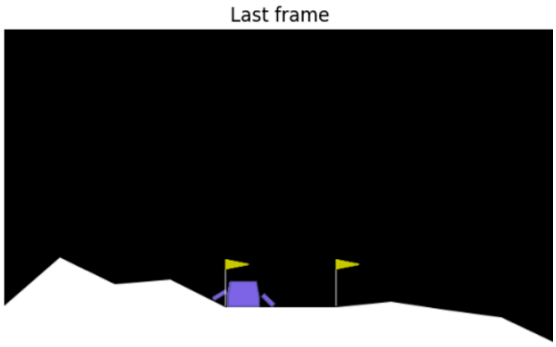
Deep Q-Learning

For the Deep Q-Learning implementation in the Lunar Lander environment, we observed the agent's learning progression over 10,000 episodes, as depicted in the provided figure. The last frame of the Lunar Lander environment shows the agent's final position between two flags, with the average reward over 1,000 episodes being -131.05. This indicates that the agent did not achieve optimal performance, as higher rewards are preferable in this environment.

In Graph 2, the left graph in the figure, titled "Episode Rewards," presents the average rewards per episode over time. The x-axis represents the episode number (from 0 to 10,000), and the y-axis shows the average reward. The graph reveals that the average reward fluctuates significantly, with values generally ranging between -250 and -100. Initially, the rewards were quite low, reflecting the agent's struggle to learn effective policies. Over time, the agent's performance improved slightly, but the rewards remained inconsistent. The fluctuations in rewards suggest that the agent may not have learned a stable policy and is struggling to achieve consistently high rewards.

The right graph in the figure, titled "Episode Lengths," illustrates the average length of each episode over time. The x-axis represents the episode number (from 0 to 10,000), and the y-axis displays the average length of episodes. Initially, the episode lengths were long, indicating that the agent was exploring the environment extensively. As training progressed, the episode lengths decreased sharply, suggesting that the agent quickly learned to terminate episodes faster. However, the episode lengths then stabilized with some fluctuations around 70 to 75 steps. This stabilization suggests that while the agent became more efficient at ending episodes, it did not necessarily translate into improved performance, as indicated by the fluctuating rewards.

Several factors could explain the suboptimal performance of the agent in the Lunar Lander environment. First, given the complexity of the task. Additionally, the hyperparameters used for training the Deep Q-Learning algorithm may have not been optimal, affecting the agent's ability to learn effectively. Another possible reason for the inconsistent performance is the balance between exploration and exploitation; the agent may have struggled to explore adequately while exploiting learned policies, leading to fluctuating rewards.




```
Testing the trained agent...

Average Reward over 1000 episodes: -131.05
```

Figure 5: Last Frame and Average Reward for the Lunar Lander Environment using Deep Q-Learning

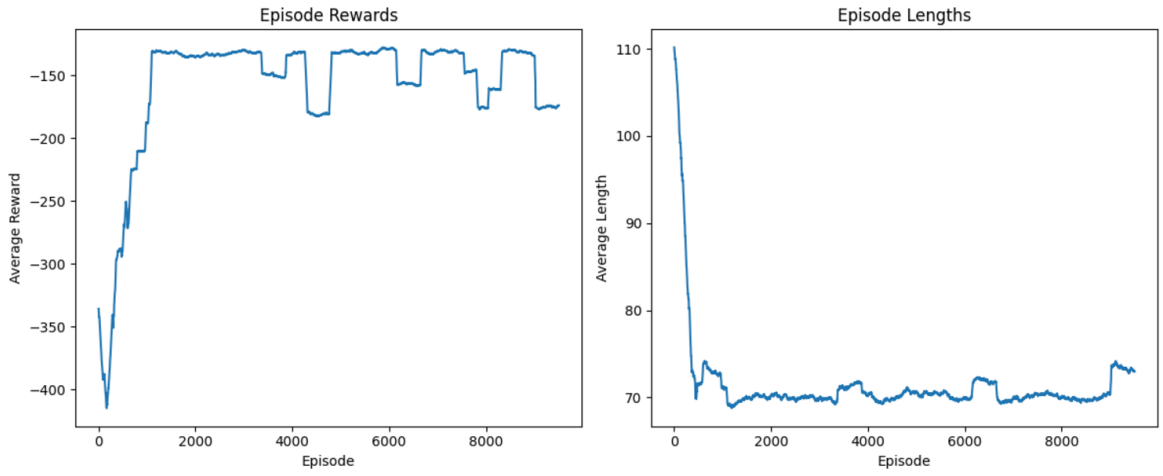


Figure 6: Average Rewards and Lengths per Episode for the Lunar Lander Environment using Deep Q-Learning

NOVEL CONTRIBUTIONS AND FUTURE WORK

A key area for future work is the further tuning of hyperparameters for each model. During our experimentation, we identified that the performance of the agents could be significantly enhanced by optimizing hyperparameters such as the learning rate, discount factor, and exploration strategies. In particular, for the Q-Learning model in the Frozen Lake environment, fine-tuning the discount factor and learning rate could lead to more efficient exploration and faster convergence. For the Approximate Q-Learning model in the Mountain Car Continuous environment, exploring different feature sets and learning rates for the linear function approximation could improve the agent's ability to generalize from seen to unseen states. Similarly, for the Deep Q-Learning model in the Lunar Lander environment, further experimentation with network architectures, learning rates, and experience replay parameters could stabilize training and enhance performance, potentially addressing the instability and suboptimal rewards observed.

Another additional possibility for Deep Q-learning is to explore more advanced deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to capture more complex patterns in the state space. Additionally, incorporating techniques like prioritized experience replay or double Q-learning could further enhance the stability and efficiency of the learning process. Another promising direction for future research is the application of transfer learning, where the agent can leverage knowledge gained from one environment to improve performance in a different but related environment. This approach could significantly reduce the training time and improve the generalization capabilities of the agent in combination with finding optimal alpha and gamma values.

The main contribution and analysis lay in conducting experimentations to identify the optimal alpha (learning rate) and gamma (discount factor) values for each model. This was done by running the model until an acceptable solution to the problem was found.

REFERENCES

Klimov, Oleg. “Gymnasium Documentation.” Farama.org, 2022, gymnasium.farama.org/environments/box2d/lunar_lander/. Accessed Dec. 2, 2024

“Gymnasium Documentation.” Farama.org, 2024, gymnasium.farama.org/environments/classic_control/mountain_car_continuous/. Accessed Dec. 2, 2024.

“Gymnasium Documentation.” *Frozen Lake - Gymnasium Documentation*, gymnasium.farama.org/environments/toy_text/frozen_lake/. Accessed Dec 2,. 2024.

GitHub Repository: https://github.com/mnunezr1/CSCI-166-Project/tree/main

Google Collab: https://colab.research.google.com/drive/1bk0Pas9utFW7jvYfMB-tvPF4FVhIyplX?usp=sharing