CLEAR-Net – Cart-pole Learning with Enhanced Adaptive Reinforcement Network

Qilong Cheng*, Aravind Narayanan*

Electrical and Computer Engineering, University of Toronto*

UNIVERSITY OF TORONTO

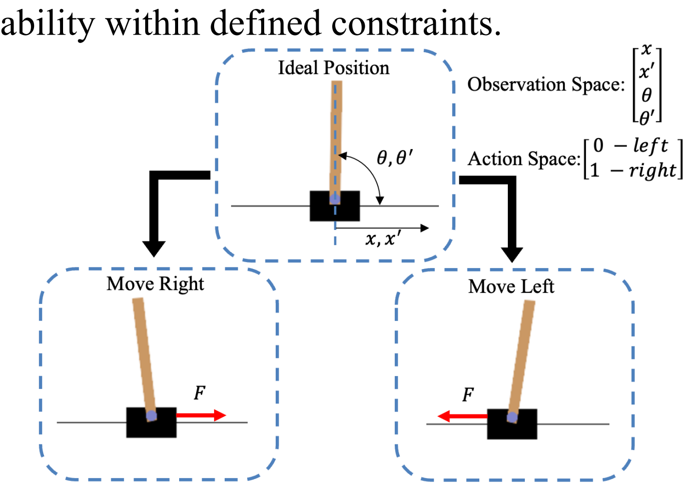
The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Abstract

This poster compares Q-learning, Deep Q-learning (DQL), and Policy Gradient methods for the cart-pole problem under noisy conditions. While DQL shows promise, it suffers from instability and catastrophic forgetting. Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) demonstrate superior stability and robustness. Our study highlights the importance of hyperparameter tuning and reward shaping, revealing that PPO and SAC are better suited for dynamic environments than traditional Q-learning.

Objectives

The cart-pole task aims to use reinforcement learning to maintain balance despite significant observational noise. The agent applies a constant force to move the cart left or right, keeping the cart within [-2.4m, +2.4m] and the pole within [-12°, +12°]. This setup challenges the agent to develop robust control policies that handle noise and maintain stability within defined constraints.



Ideal Position
θ, θ'
x, x'
Observation Space: $\begin{bmatrix} x \\ x' \\ \theta \\ \theta' \end{bmatrix}$
Action Space: $\begin{bmatrix} 0 & -left \\ 1 & -right \end{bmatrix}$
Move Right
F
Move Left
F

Figure 1: Cart-pole balancing task showing the agent's goal to maintain cart and pole within set limits using RL

Methods

1. Tabular Q-Learning:

Classical Q-learning is first implemented in the project. To solve the continuous observation space issue, each observation data is quantized to fit into different bins as shown in Figure 2. Each bin and its corresponding action have their respective action-state value. Action is drawn based on the Q-table using Epsilon Greedy.

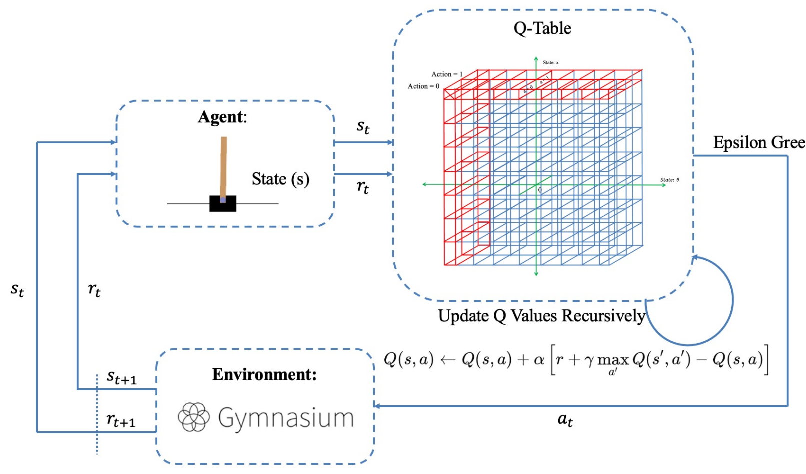


Figure 2: Overview Tabular reinforcement learning

Tabular Q-learning suffers from the curse of dimensionality, causing exponential increases in computation as states increase. Balancing performance and computational cost is crucial. Figure 3 shows comparisons between different bin sizes. We picked (8, 12, 8, 12) as our baseline for its fast convergence and stability.

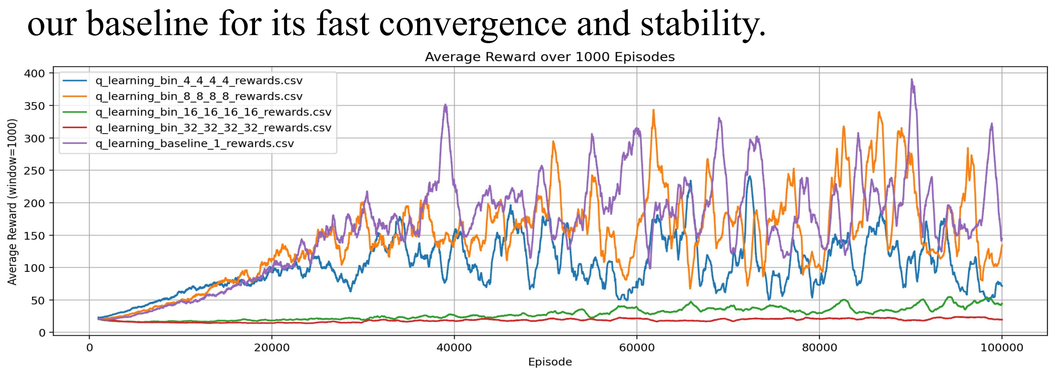


Figure 3: Performance comparison between different bin size for discretization of the observation space

2. Deep Q-Learning (DQL)

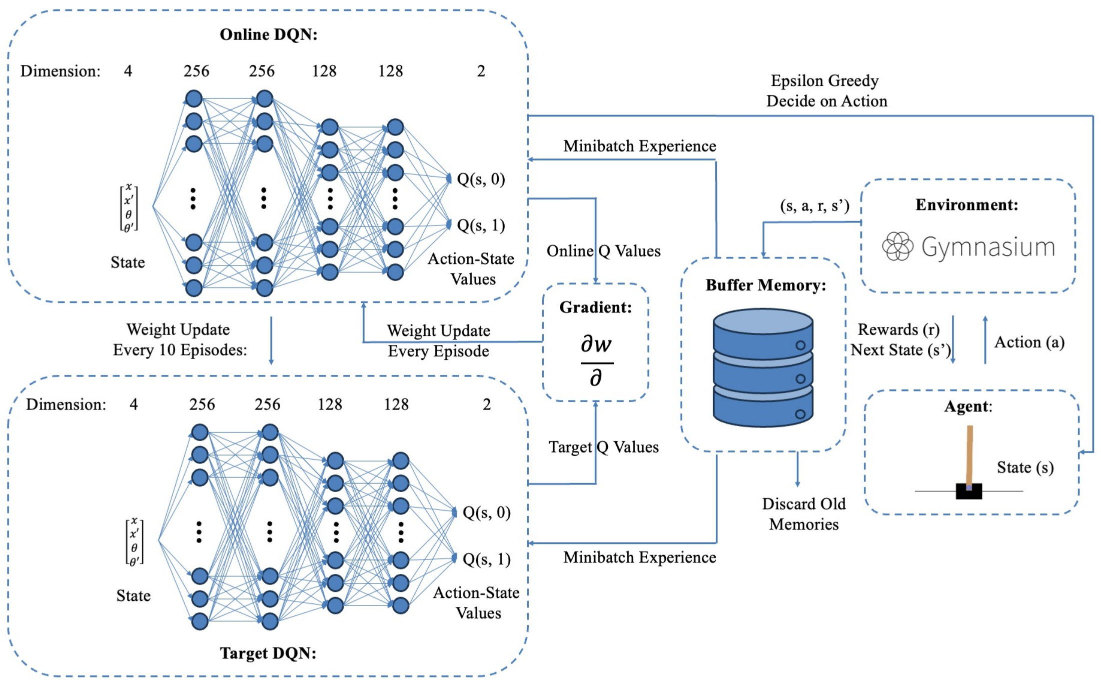


Figure 4: Overview of the DQL architecture with an additional target network and memory buffer

A big part for DQL is the hyperparameter tuning. In this project we tested different learning rate, batch size, epsilon decay rate and the noise level to be trained on. And we selected the optimal hyperparameter as our baseline:

Hyperparameters	Values
Learning Rate	0.0001
Batch Size	64
Epsilon Decay Rate	0.9995
Episodes	5000
Target Network Update Rate	10
Terminal Step Size	500

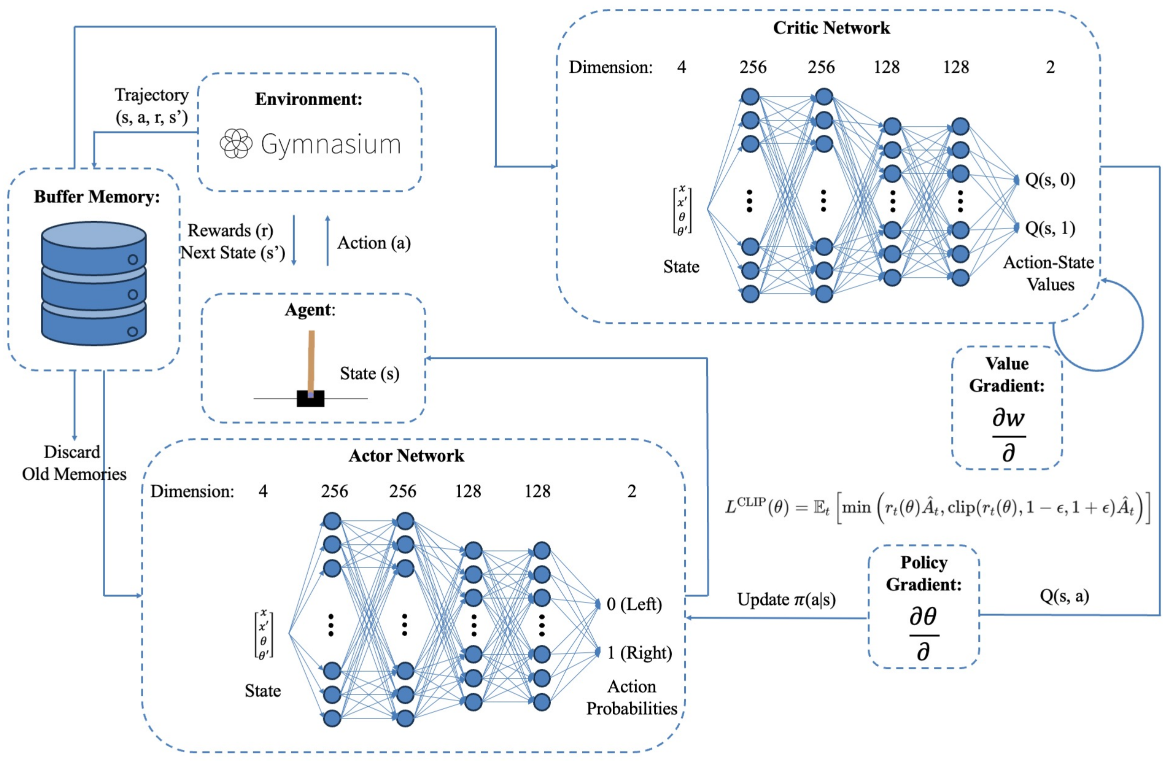
Table 1: Selected hyperparameter values for DQL

Despite impressive performance after hyperparameter tuning, the agent still suffered from drifting during testing because maintaining the cart in the middle wasn't an explicit objective. We experimented with different reward shaping and finalized a modified reward function:

$$\begin{cases} 1.0 & \text{if } |x| < 0.5 \text{ and } |\theta| < 0.05, \\ -\frac{|x|}{2.4} - \frac{|\theta|}{0.209} & \text{otherwise.} \\ -1.0 & \text{if terminated and } d < 500 \end{cases}$$

rewarding the cart for staying in the middle and the pole for staying upright, penalizing deviations from these objectives, and heavily penalizing if the total step count is below 500 at the terminal state. This reward shaping encourages faster convergence while keeping the cart centered.

3. Policy Gradient Method (PGM)

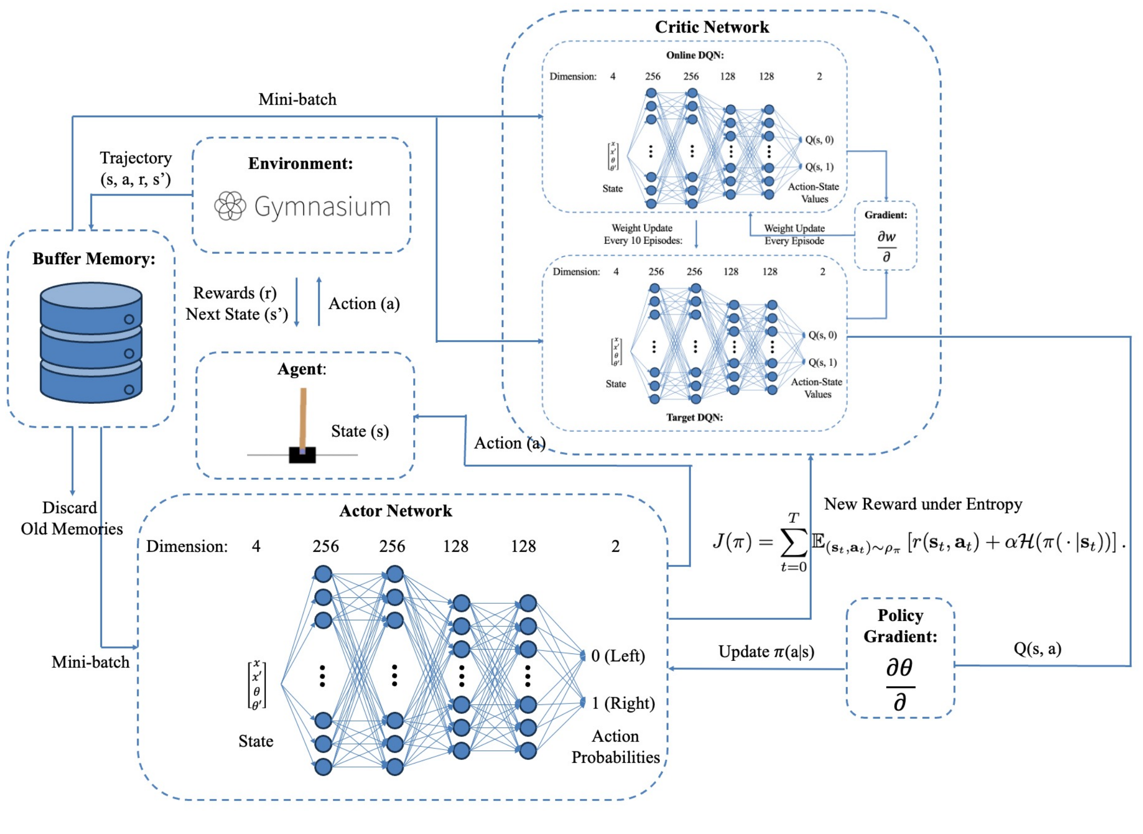


Trajectory (s, a, r, s')
Environment:
Gymnasium
Critic Network
Dimension: 4 256 256 128 128 2
State
Q(s, 0)
Q(s, 1)
Action-State Values
Buffer Memory:
Rewards (r) Next State (s')
Action (a)
Agent:
State (s)
Value Gradient:
$\frac{\partial w}{\partial}$
Discard Old Memories
Actor Network
Dimension: 4 256 256 128 128 2
State
0 (Left)
1 (Right)
Action Probabilities
$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$
Policy Gradient:
$\frac{\partial \theta}{\partial}$
Update π(a|s)
Q(s, a)

Figure 6: Overview of the Policy Gradient Method architecture

Unlike DQL, which can suffer from catastrophic forgetting, Proximal Policy Optimization (PPO) maintains a balance between exploration and exploitation by using the stochastic action space from the policy network, resulting in smoother learning curves (Figure 8). By restricting policy updates within a trust region using a clipped surrogate function, PPO provides more reliable learning, crucial for tasks like cart-pole where stable policy adjustments are essential.

4. Soft Actor-Critic (SAC)



Critic Network
Online DQN:
Dimension: 4 256 256 128 128 2
State
Q(s, 0)
Q(s, 1)
Action-State Values
Mini-batch
Trajectory (s, a, r, s')
Environment:
Gymnasium
Buffer Memory:
Rewards (r) Next State (s')
Action (a)
Weight Update Every 10 Episodes
Weight Update Every Episode
Gradient:
$\frac{\partial w}{\partial}$
Dimension: 4 256 256 128 128 2
State
Q(s, 0)
Q(s, 1)
Action-State Values
Target DQN:
Agent:
State (s)
Action (a)
Discard Old Memories
Actor Network
Dimension: 4 256 256 128 128 2
Mini-batch
State
0 (Left)
1 (Right)
Action Probabilities
New Reward under Entropy
$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right].$
Policy Gradient:
$\frac{\partial \theta}{\partial}$
Update π(a|s)
Q(s, a)

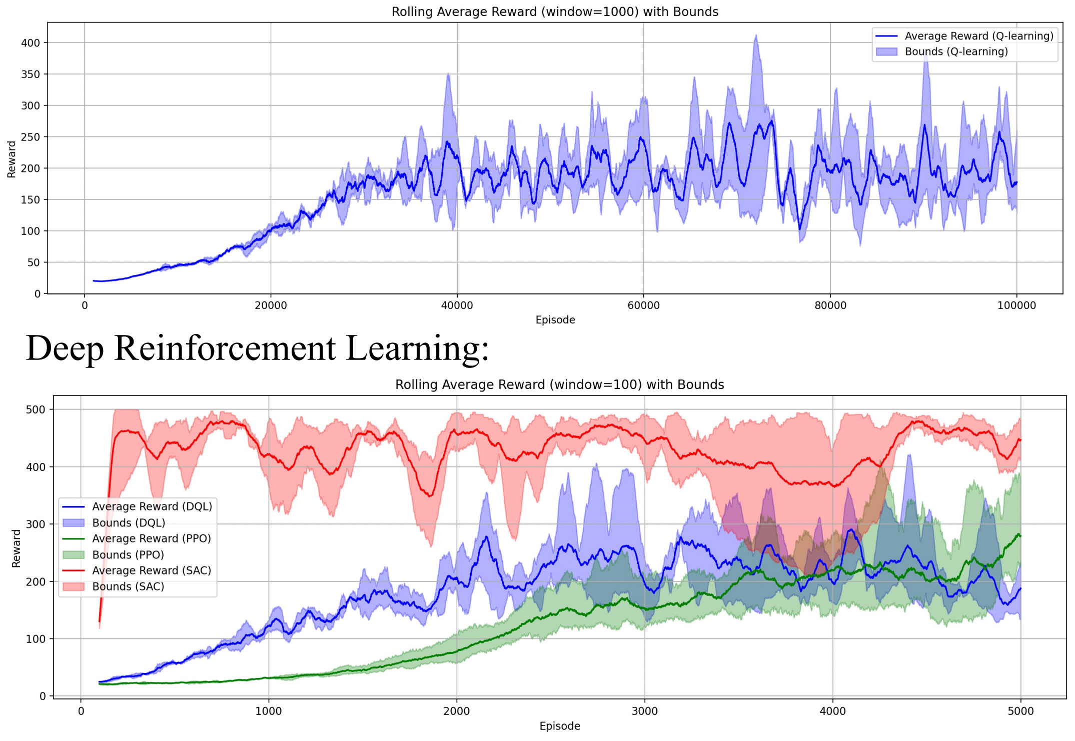
Figure 7: Overview of the Soft Actor-Critic architecture

SAC balances deterministic and stochastic policies using an additional entropy term for optimal exploration and exploitation and eliminated the need for tuning epsilon decay rate. It also uses twin Q-networks for increased stability and an action-space wrapper to convert discrete actions to continuous actions. This allows us to deploy SAC algorithm into the discrete action-space problem.

Results

Training Results

Tabular Reinforcement Learning:



Deep Reinforcement Learning:

Figure 8: Comparison between the Tabular Q-learning method versus three different Deep Reinforcement Learning (DRL) methods, DQL, PPO and SAC.

The most noticeable difference is that Deep Reinforcement Learning (DRL) algorithms require significantly fewer episodes to converge compared to the Q-learning approach. Notably, Soft Actor-Critic (SAC) converges the fastest due to its elimination of the epsilon term, leading to enhanced exploration and consistent performance. Deep Q-Learning (DQL) converges faster than Proximal Policy Optimization (PPO) but tends to suffer from "catastrophic forgetting," causing instability. While all learning-based algorithms are susceptible to overfitting, resulting in some degree of "forgetting," PPO is the most stable and robust over extended training periods.

Different Noise Level - Controller Robustness:

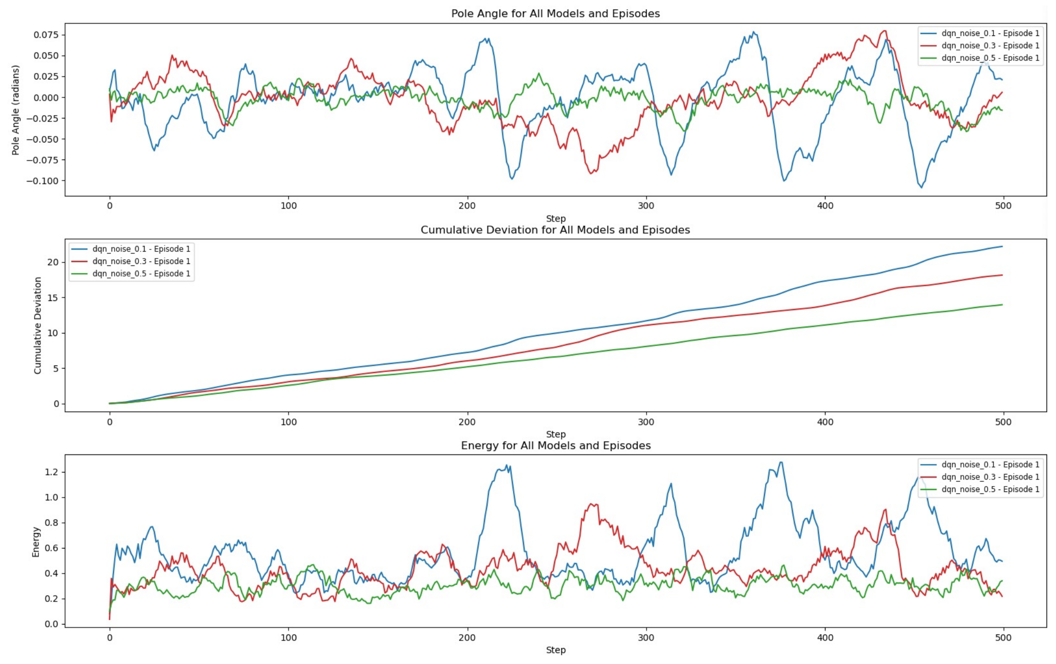


Figure 9: Robustness comparison of DQN models with varying noise levels under identical disturbances

Assessing DQN controllers trained with noise variance of 0.1, 0.3, and 0.5 showed that higher noise training (0.5) resulted in smoother pole angle variations, lower cumulative deviations, and more stable, reduced energy profiles, indicating greater robustness to real-world disturbances. PPO and SAC networks exhibited similar results.

Interestingly, models trained with higher noise performed better under similar noise conditions but worse when the noise level was reduced. Further research is needed to develop models that handle a broad range of noise levels effectively.

Conclusion

Our study investigated extensively the pros and cons of Q-learning, DQL, PPO, and SAC for the cart-pole problem. Classical Q-learning struggles with curse of dimensionality and instability, while DQL faces issues with catastrophic forgetting. PPO and SAC demonstrate superior stability and robustness, making them ideal for real-world dynamic environments. Hyperparameter tuning and reward shaping are crucial for enhancing performance, reducing required training episodes. In addition, models trained with higher noise variance tend to perform better under sudden impulsive disturbances, with faster stability convergence and lower energy requirement.

References

[1] Farama Foundation, "CartPole-v1," Gymnasium documentation. [Online]. Available: https://gymnasium.farama.org/environments/classic_control/cart_pole/. [Accessed: Aug. 5, 2024].