**Assignment\_2\_Selected\_Topic**

**Environment Name:** Acrobot-v1

**Model Name:** Dueling Double Deep Q-Network (Dueling DDQN)

**Team Members:**

* Youssef Yasser (ID: 20213360)
* Youssef Mohamed (ID: 20212812)
* Karim Magdy (ID:20210304)
* Gasser Maged (ID:20212777)

**Team Members Responsibilities:**

|  |  |
| --- | --- |
| **Team Member** | **Responsibility** |
| Youssef Yasser | Designed and implemented the Dueling DQN architecture, managed hyperparameter tuning and training logic. |
| Youssef Mohamed | Focused on preprocessing the environment and reward normalization strategy. |
| Karim Magdy | Handled evaluation logic, performance visualization, and model checkpointing. |
| Gasser Maged | Documented the project, analyzed the reward trends, and contributed to reporting and presentation. |

**Observation Space:**

The observation is a 6-dimensional vector representing the current state of the Acrobot:

[cos(θ1), sin(θ1), cos(θ2), sin(θ2), θ̇1, θ̇2]

Where:

* θ1, θ2: Angles of the two pendulum links
* θ̇1, θ̇2: Angular velocities

**Action Space:**

The agent can choose from 3 discrete actions:

* -1: Apply negative torque
* 0: Apply zero torque
* 1: Apply positive torque

**State Space:**

* Angle 1: [-π, π]
* Angle 2: [-π, π]
* Angular velocity 1: clipped to [-4, 4]
* Angular velocity 2: clipped to [-9, 9]

**Model Selection – Dueling Double DQN**

We chose **Dueling Double DQN** for the following reasons:

* **Dueling DQN** separates the value and advantage functions, allowing the model to better estimate state values in scenarios where action choice has minimal effect (common in Acrobot).
* **Double DQN** addresses overestimation bias by using the policy network to select actions and the target network to evaluate them.
* This model balances stability and performance well for medium-difficulty control tasks like Acrobot-v1.

**Graph: Total Timesteps vs. Reward Performance:**

A graph with blue lines and numbers

AI-generated content may be incorrect.

**Explanation of Reward Trends vs. Total Timesteps**

|  |  |  |
| --- | --- | --- |
| Timesteps | Mean Reward | Std. Dev |
| 10,000 | -500.00 | 0.00 |
| 20,000 | -84.90 | 12.04 |
| 50,000 | -76.60 | 15.43 |
| 100,000 | -68.50 | 7.97 |
| 200,000 | -73.10 | 17.35 |

**Observations:**

* At **10,000 timesteps**, performance was poor, with no significant learning (-500 is the default min reward).
* Performance gradually improved as more training data was accumulated.
* By **100,000 timesteps**, the agent learned to swing up effectively. Here model give best Numbers
* At **200,000 timesteps**, the model Start to Overfit
* **Why Performance Changes with Timesteps:**
* Early in training, the agent explores randomly (due to high epsilon).
* Over time, the epsilon-greedy policy shifts toward exploitation, guided by improved Q-values.
* Longer training allows for better convergence of Q-values, and the use of target networks and soft updates helps stabilize this learning.