A white background with red text

AI-generated content may be incorrect.

Reinforcement Learning

**Environment:** *CartPole-v1***Model Used:** *Deep Q-Network (DQN)*

**Team Members:**

* Esraa Mohammed 20213796
* Gehad Osama 20212167
* Nouran Hossam 20213804
* Maryam Ayman 20213997

**Team Members’ Responsibilities**

|  |  |
| --- | --- |
| **Team Member** | **Responsibility** |
| Esraa Mohammed | Custom evaluation function |
| Gehad Osama | Plot performance |
| Nouran Hossam | Train and evaluate models |
| Maryam Ayman | Retrain and save best model |

Table of Contents

[**Environment Description: CartPole-v1** 4](#_Toc195733602)

[**Observation Space:** 4](#_Toc195733603)

[**Action Space:** 4](#_Toc195733604)

[**State Space:** 4](#_Toc195733605)

[**Why DQN?** 5](#_Toc195733606)

[**Performance Graph** 5](#_Toc195733607)

[**Effect of Changing Total Timesteps on Performance** 6](#_Toc195733608)

[**Summary** 8](#_Toc195733609)

# **Environment Description: CartPole-v1**

## **Observation Space:**

The agent receives a 4-dimensional continuous observation vector:

1. Cart position
2. Cart velocity
3. Pole angle
4. Pole angular velocity

These values give the agent an understanding of how the pole is moving and its relation to the cart.

## **Action Space:**

The action space is **discrete with 2 actions**:

* 0: Move the cart to the left
* 1: Move the cart to the right

## **State Space:**

While the observation space is continuous, the **state space is theoretically infinite**, as each value can take any number within a range. However, the environment only remains active until the pole falls beyond a certain angle or the cart moves too far from the center.

# **Why DQN?**

We chose the Deep Q-Network (DQN) algorithm because it is one of the most widely used reinforcement learning models for environments with discrete action spaces. DQN combines Q-learning with deep neural networks, allowing it to handle complex environments where the state space is too large to represent with traditional Q-tables. The CartPole-v1 environment is a classic control problem that fits well with DQN's capabilities.

# **Performance Graph**

A graph showing a blue line

AI-generated content may be incorrect.

The following graph shows how the mean reward changes with the number of trainings timesteps. In general, models trained with more timesteps performed better.

The shaded area around the line shows the standard deviation to indicate how stable the performance was.

# **Effect of Changing Total Timesteps on Performance**

Changing the number of trainings timesteps had a significant impact on the model's performance. With a small number of timesteps (e.g., 1000), the agent did not have enough time to learn effective strategies, resulting in low mean rewards. As the timesteps increased to 5000, 10000, and beyond, the model was able to better understand the environment and improve its decision-making process. The highest rewards were observed at the highest training timestep (50000), demonstrating the benefit of extended training time.

| **Timesteps** | **Mean Reward** | **Std Dev** |
| --- | --- | --- |
| 1000 | ~14 | ±2.4 |
| 5000 | ~45 | ±10 |
| 10000 | ~120 | ±18 |
| 20000 | ~200 | ±5 |
| 50000 | ~500 | ±0 |
|  |  |  |

**Analysis:**

* Initially, the model performed poorly (low mean reward), especially with fewer timesteps like 1000 or 5000.
* As timesteps increased, the model had more opportunity to explore and learn the environment, significantly improving performance.
* At **50,000 timesteps**, the model consistently solved the environment (maximum reward of 500).

# **Summary**

In this project, we applied reinforcement learning using the DQN algorithm on the CartPole-v1 environment. The results showed that the model improved with more training timesteps. This experiment demonstrates how DQN can be used effectively for control tasks.