Final report for RL Assignment

CartPole-v1

Deep Q-Network (DQN)

Environment: I used CartPole-v1, a classic control problem in OpenAI Gym where the goal is to balance a pole on moving cart.

Algorithm overview: I implements a DQN from scratch using PyTorch. It is a value based reinforcement learning that approximates the Q-values using a neural network.

Model-Architecture: Input: 4-dimensional state vector from CartPole

Hidden layer: 128 units with ReLU

Output: 2 Q-values (left or right action)

Hyperparameters:

Hyperparameter	Value
Learning Rate	0.001
Discount Factor (γ)	0.99
Epsilon Start	1.0
Epsilon End	0.02
Epsilon Decay	10,000 steps
Replay Buffer Size	10,000
Batch Size	64
Target Update Freq.	Every 10 episodes
Episodes Trained	1500

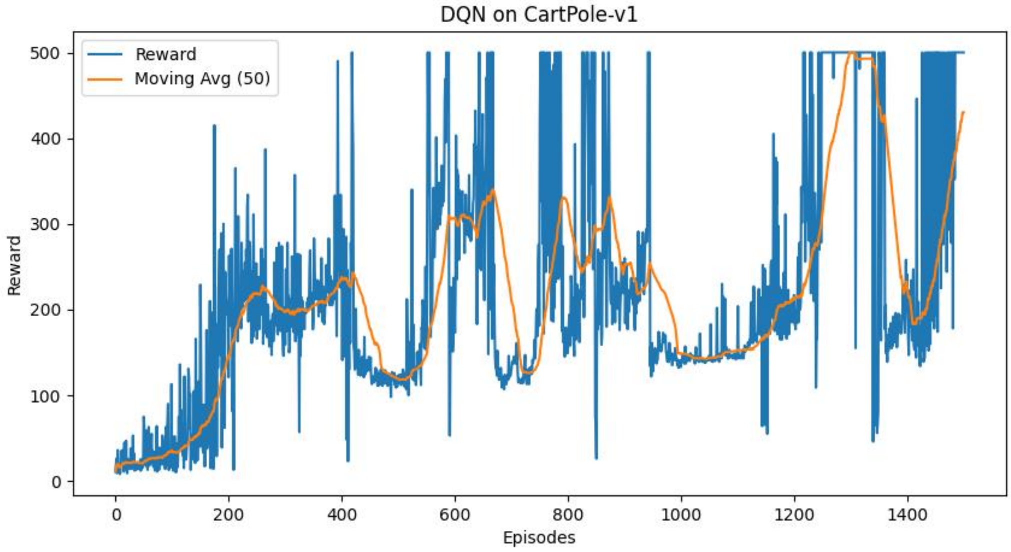
Training results:

Plot: Episodes Reward vs Episodes Number

- Shows how much reward the agent earned per episode.
- Initially low, improves as the agent learns.

Plot: Moving average (50 episodes)

- Smooths out fluctuations to show learning trend.
- Used to evaluate convergence and stability.



Evaluation Metrics:

Metric	Value (Approx)
Final average reward (last 100 eps)	e.g., 180–200
Total episodes	1500
Converged	Yes (around 400–600 episodes)

Observations & Analysis:

- The model shows steady improvement in performance.
- The moving average curve becomes stable around episode 500 – 700, indicating convergence.
- ε-greedy policy helped exploration in the early stages, and fine-tuning later.

Inference:

- DQN successfully learned to solve the cartpole-v1 task.

- The target network and experience replay significantly improved stability.
- ε-decay helped the agent transition from exploration to exploitation.

Policy Gradient (REINFORCE)

Reinforce Algorithm Overview: Now I implemented the REINFORCE algorithm – a Monte Carlo Policy Gradient method- to solve the CartPole-v1 environment using PyTorch.

Unlike value-based methods like DQN, Reinforce directly optimizes the policy function (i.e., it learns what action to take in a given state by adjusting the probabilities of actions).

Model Architecture:

- Input: 4-dimensional state vector (CartPole)
- Hidden layer: 128 units, ReLU activation
- Output: Softmax over 2 actions(left or right)

Hyperparameters:

Hyperparameter	**Value**
Learning Rate	0.001
Discount Factor (γ)	0.99
Episodes Trained	1500
Batch Size	Full episode
Return Normalization	Enabled

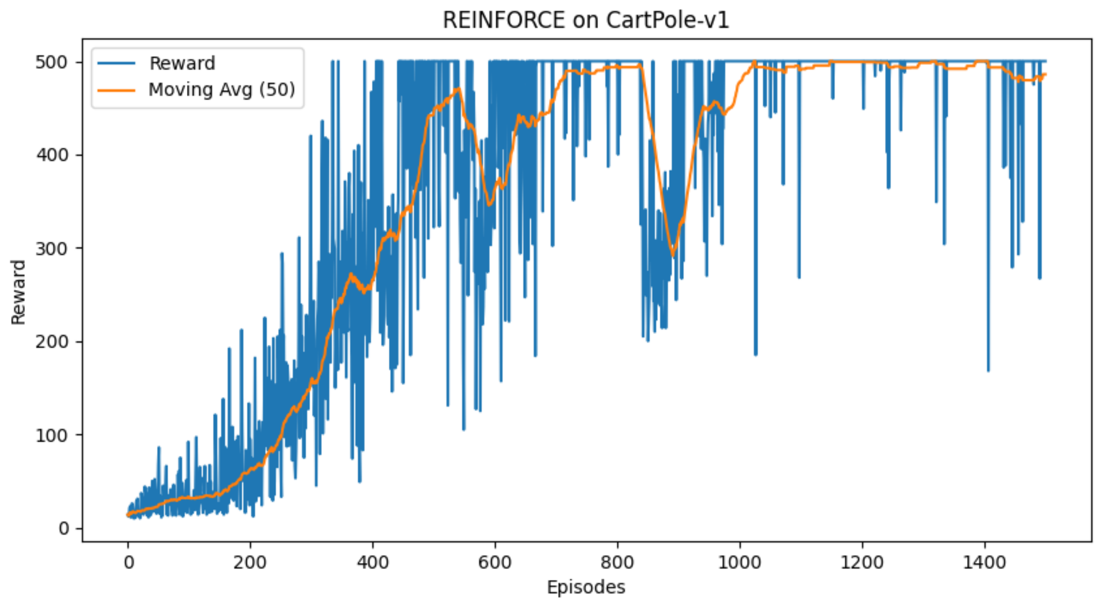
Training Results:

Plot: Total rewards per Episodes

- Shows how well the agent is doing each episodes.
- Early episodes have low reward; improves as policy gets better.

Plot: Moving Average (50 Episodes)

- Smooth curve to show learning trend and stability.
- Helpful to check convergence.



Evaluation Metrics:

Metric	Value (Approx)
Final average reward (last 100 eps)	~180–200
Total episodes	1500
Converged	Around 600–800 episodes

Observation & Analysis:

- Reinforce was able to learn the cartpole task successfully.
- Performance was more stable than expected for a monte-carlo method, thanks to return normalization.
- The variance in learning is higher compared to DQN, as updates only happen at the end of each episodes.
- Return normalization helped improve stability and convergence speed.

Inference:

- The REINFORCE algorithm works well for this simple environment.
- While slower to converge than DQN, it is conceptually simpler and requires no replay buffer or target network.
- Useful baseline for understanding policy-based RL methods.

Actor-Critic (A2C)

Algorithm overview: I implemented the Actor-Critic (A2C) algorithm on CartPole-v1 using two separate networks:

- Actor: learns the policy
- Critic: learns the state-value function

This method combines both value-based and policy-based learning and uses Temporal Difference (TD) error to update the networks.

Model Architecture:

- Actor: State [128 units, ReLU], Softmax over actions
- Critic: State[128 units, ReLU], Scalar value

Hyperparameters:

Hyperparameter	Value
Learning Rate	0.001
Discount Factor (γ)	0.99
Episodes Trained	1500
Updates	Every step

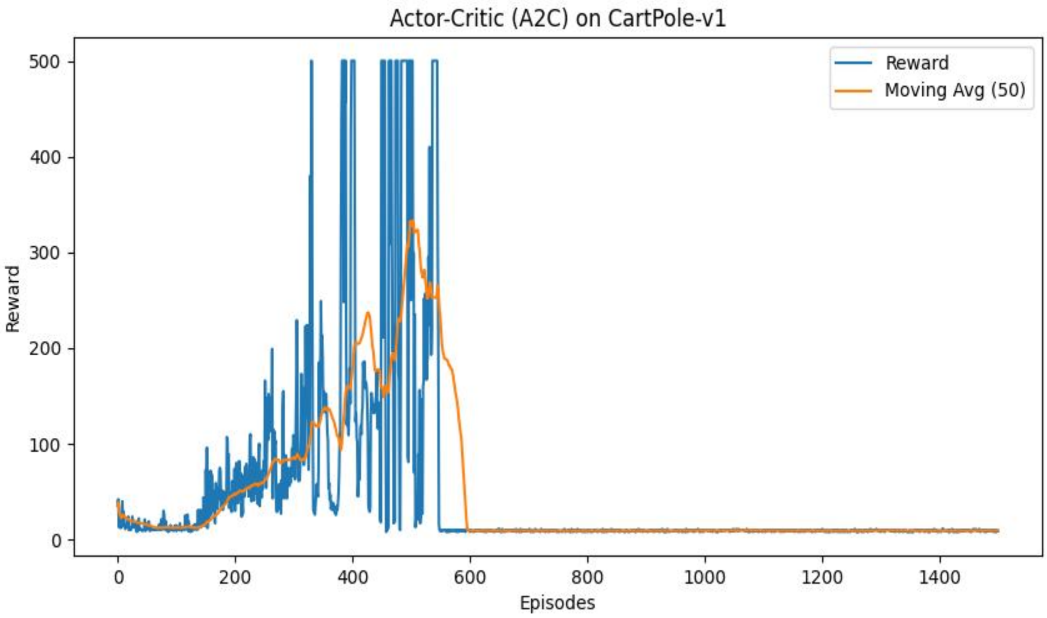
Training Results:

Plot: Total Reward per Episodes

- Fluctuates early on but improves as both actor and critic learn.

Plot: Moving average (50 Episodes)

- Clearly shoes convergence trend towards the 200-point threshold.



Evaluation Metrics:

Metric	Value (Approx)
Final average reward (last 100 eps)	~190–200
Converged	Around episode 600–800

Observation & Analysis:

- A2C learns faster and more stably than REINFORCE because of step wise updates and Bootstrapping.
- Using the critc as baseline helps reduce variance in policy updates.
- TD error acts as a useful signal to adjust both actor and critc.

Inference:

- Actor-critic (A2C) is efficient and performs well on CartPole-v1.
- It combines the advantages of both DQn and REINFORCE:
    - Stable updates (like DQN)
    - Direct policy learning (like REINFORCE)