DUELING NETWORK ARCHITECTURES FOR DEEP RL

Nityam Pareek
Shantanu Chaudhari
Aryan Lath

MOTIVATION

- In many states, the choice of action has little to no effect on the outcome. Eg : Where to steer on empty road in car game.

- Standard Q-Networks estimate the value of every action in every state, which is inefficient when many actions have similar value.

- We want to learn :
  - How valuable is being in a particular state?
  - How much better is an action compared to others?

Q-LEARNING RECAP

Algorithm 1 Deep Q-Network (DQN)

Initialize replay buffer $\mathcal{D}$, Q-network θ, target network $\theta^- \leftarrow \theta$
for each episode **do**
    Initialize state s
    while not terminal **do**
        Select action a: ε-greedy: random or $\arg\max_a Q(s, a; \theta)$
        Execute a, observe r, s', store (s, a, r, s') in $\mathcal{D}$
        Sample mini-batch $(s_j, a_j, r_j, s'_j) \sim \mathcal{D}$
        Compute target:

$$y_j = r_j + \gamma \max_{a'} Q(s'_j, a'; \theta^-)$$

        Update θ using loss:

$$L(\theta) = \left(y_j - Q(s_j, a_j; \theta)\right)^2$$

        Every C steps: $\theta^- \leftarrow \theta$
        $s \leftarrow s'$
    end while
end for

PROBLEM WITH VANILLA DQN

- Vanilla Deep Q-Networks predict one scalar per action.
- The same feature representation is used for all Q(s, a) values.
- No mechanism to separate state evaluation and action ranking.
- **The Goal** : Separate estimation of state value and advantage to improve learning efficiency.

DUELING NETWORKS

Core Idea

- Decompose Q-Values into two streams :

$$Q(s,a) = V(s) + A(s,a)$$

- Where :
  - V(s) - Value Function (how good is the state)
  - A(s, a) - Advantage Function (how much better is action a over others)

- Problem : This decomposition is not unique, i.e. we can add/subtract constants between V and A with no change in Q. This ambiguity degrades training performance.

DUELING NETWORKS

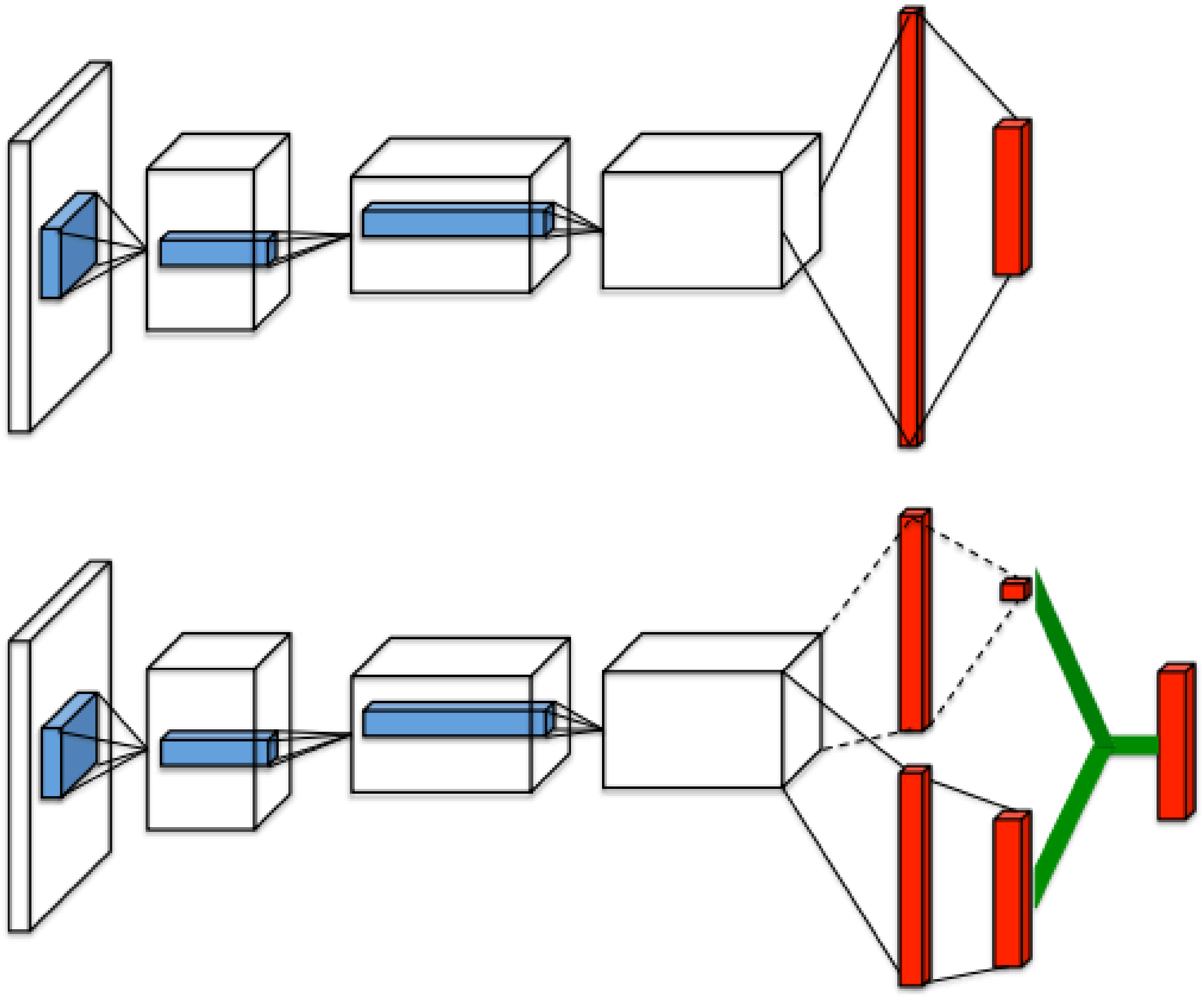
Solving the Ambiguity

- Use normalized advantage instead of naive sum :
  - Max Normalized : $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a'} A(s, a'; \theta, \alpha) \right)$
  - Mean Normalized : $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha) \right)$
- In experiments, mean normalized sum is used as it increases training stability.

DUELING NETWORKS

Architecture

1. Shared Feature Layer : CNN or ANN depending on usage.
2. Two Separate Fully Connected Streams :
   a. Value Stream : Outputs scalar V(s)
   b. Advantage Stream : Outputs vector A(s, a)
3. Aggregation layer : Compute Q(s, a)

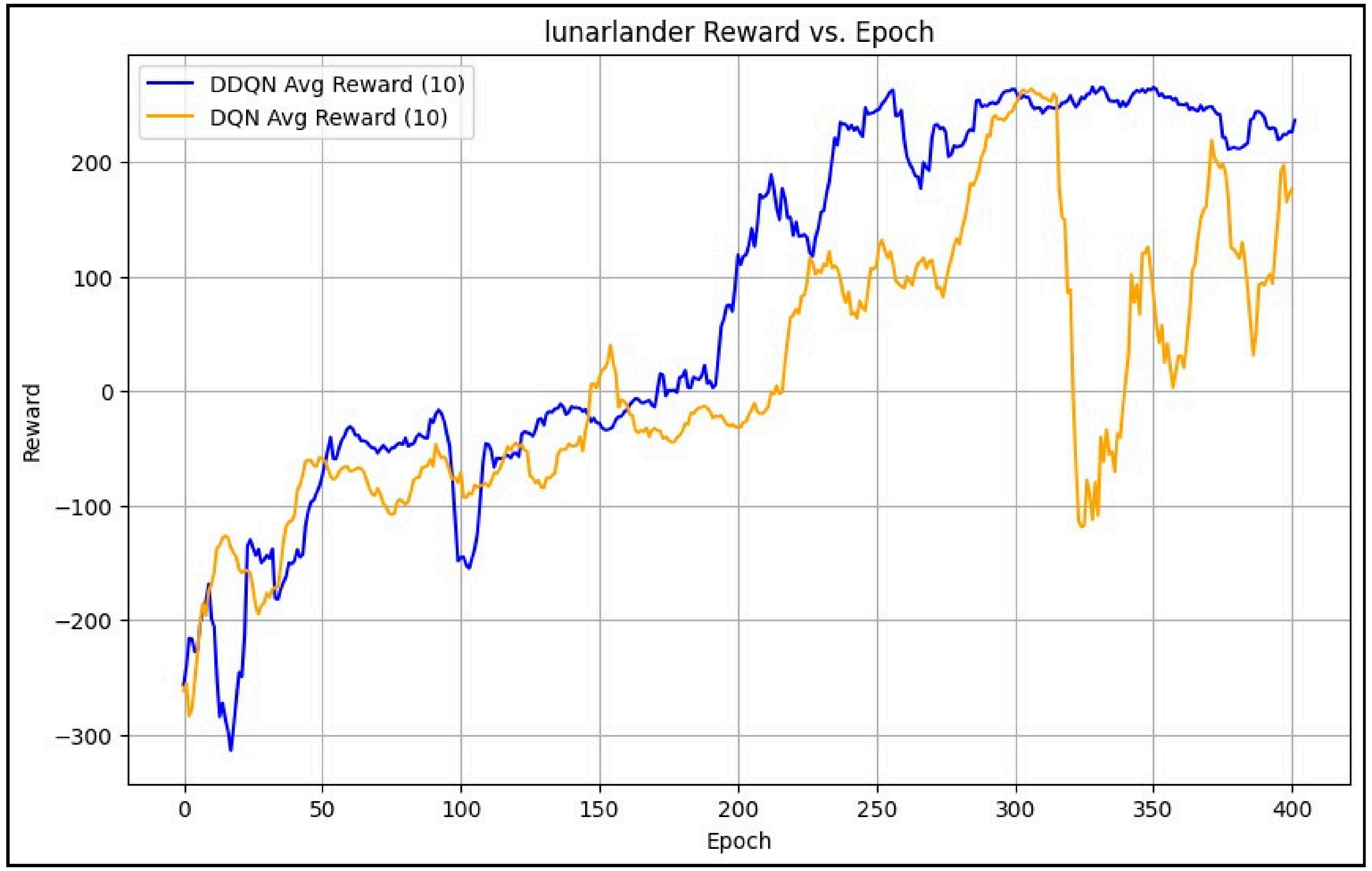


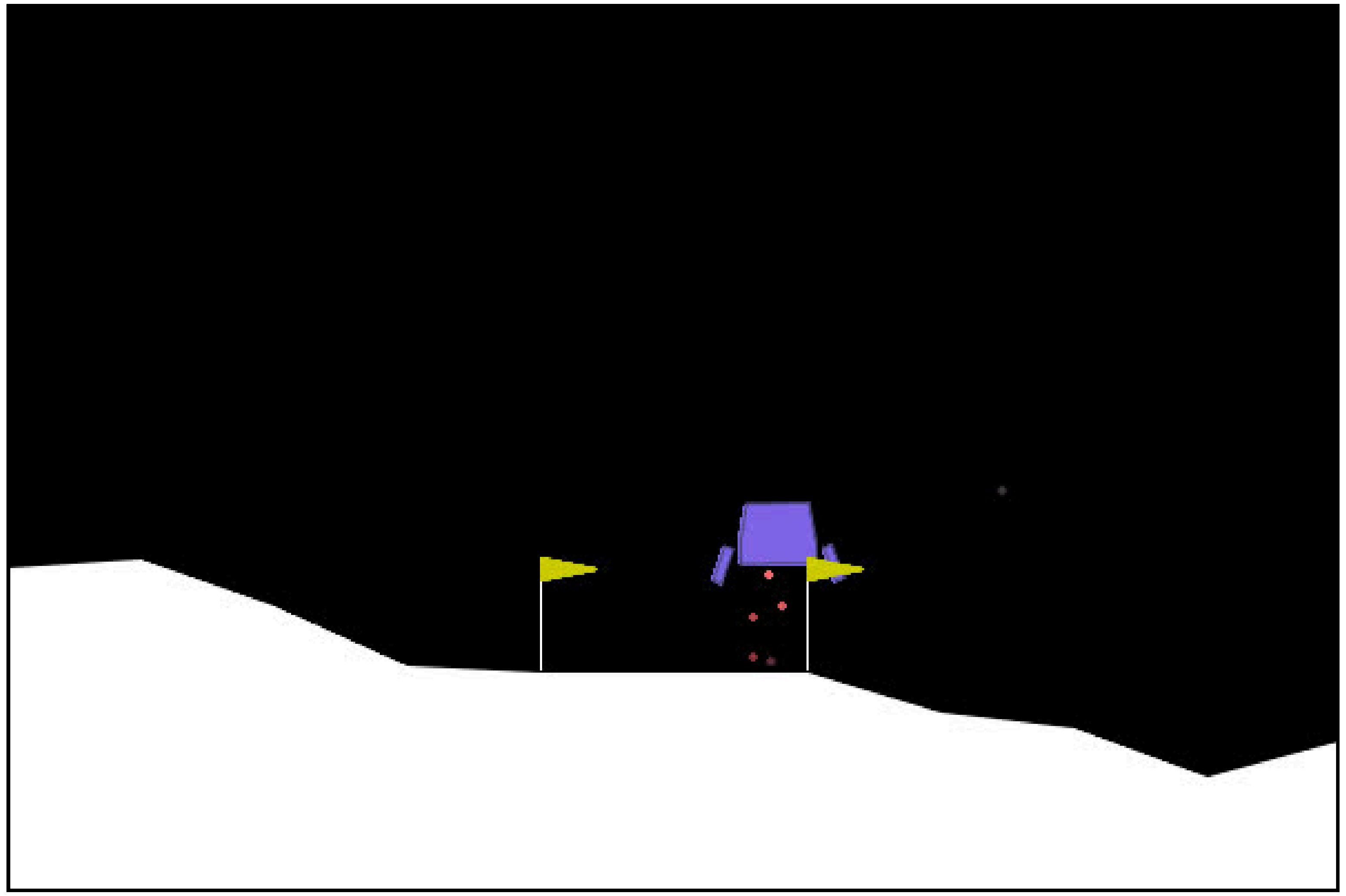
EXPERIMENT SETUP

- **Baseline** : Vanilla Q-Network
- **Environments** : LunarLander-v3, CartPole-v1, Pong-v5 (Atari)
- **Models** : ANN(LunarLander-v3, CartPole-v1), CNN+ANN(Pong-v5)
- **Loss** : Mean Squared Error (MSE) Loss
- **Visualization** : Reward vs Episode visualized after smoothing with a moving average filter of length 10 (for better clarity)

RESULTS : LUNAR LANDER

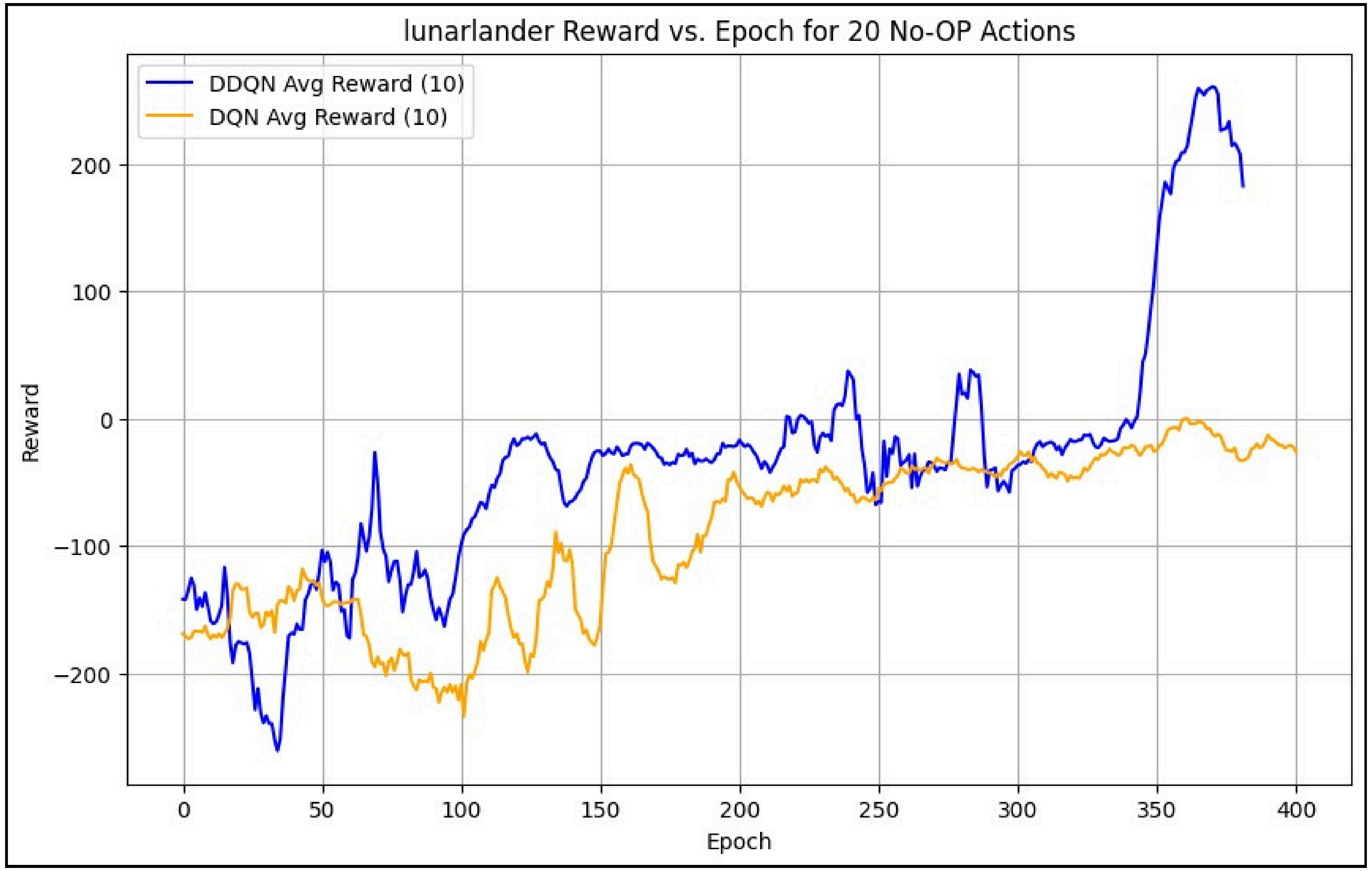
Default Action Space



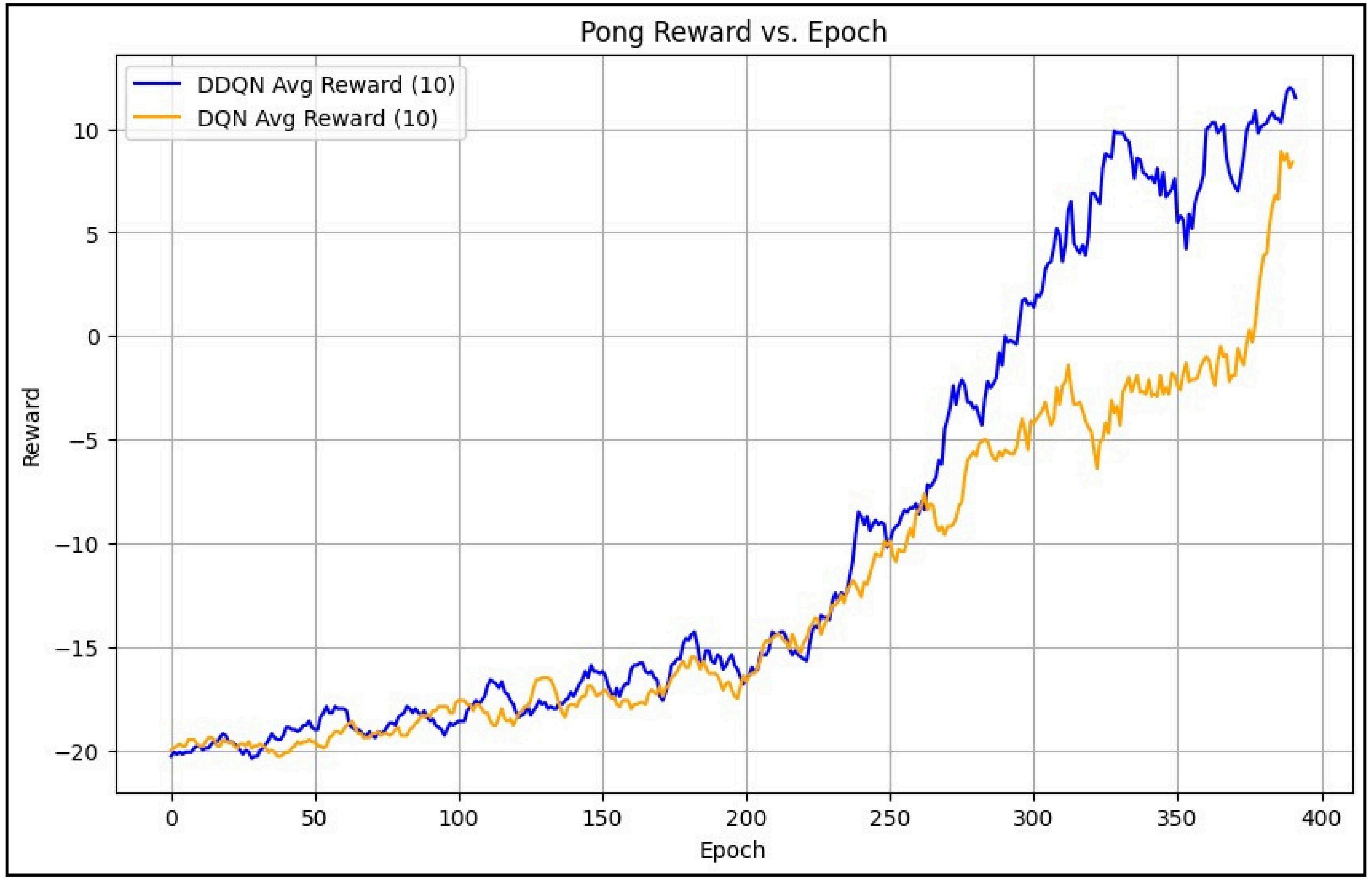


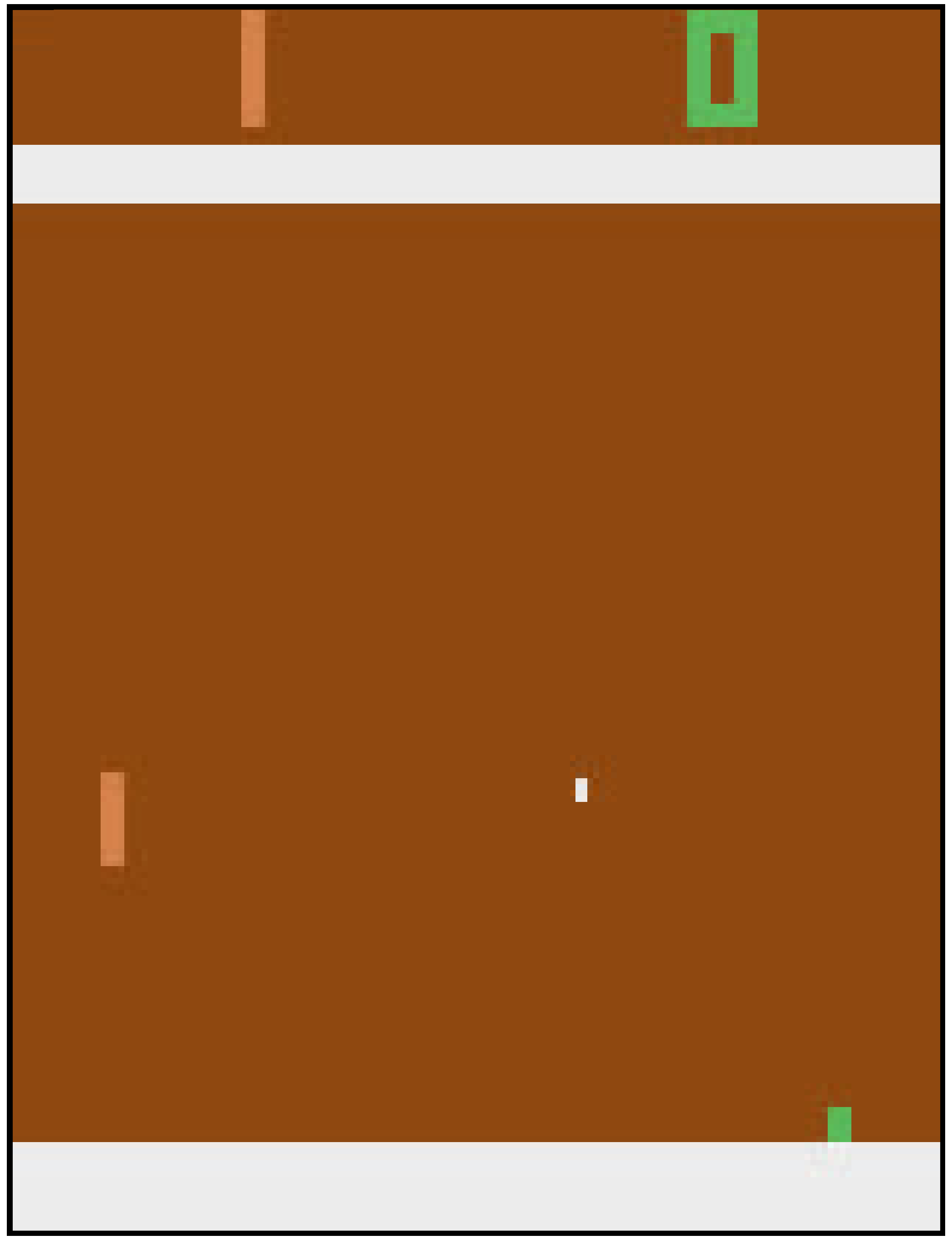
RESULTS : LUNAR LANDER

Modified Action Space with 20 Added No-Ops

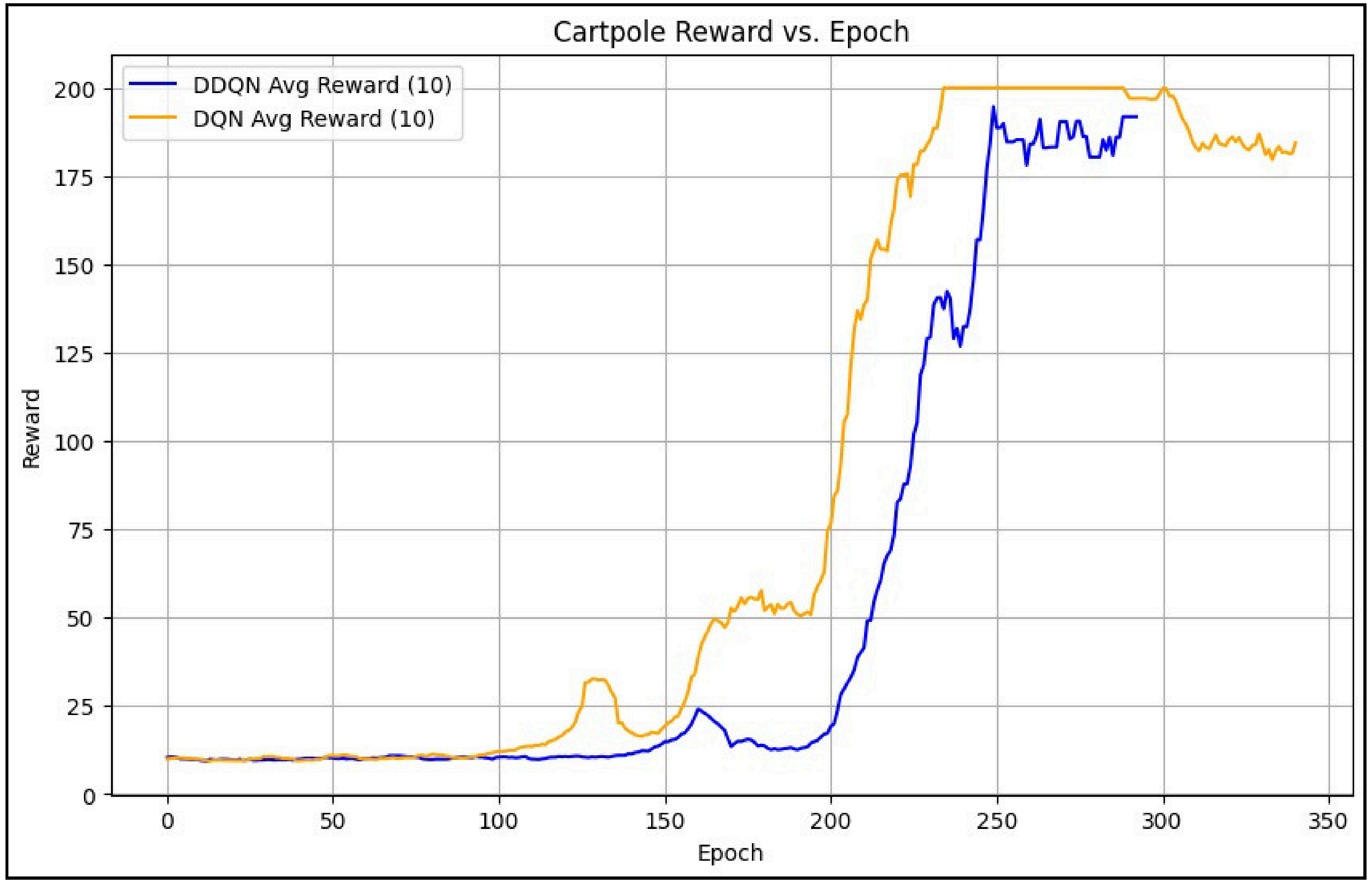


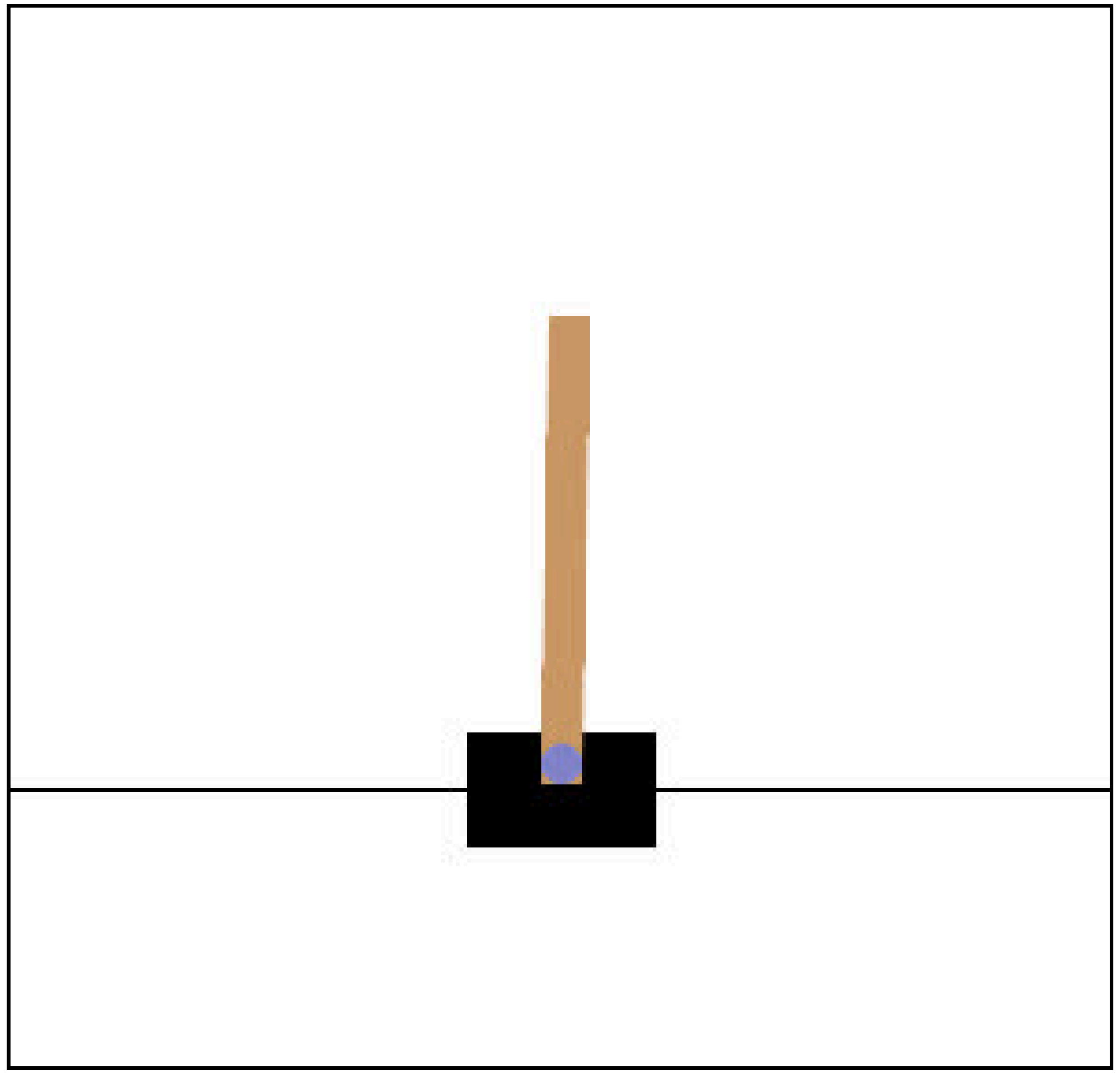
RESULTS : PONG





RESULTS : CARTPOLE





OBSERVATIONS

1. Dueling DQN offers superior performance in complex environments like LunarLander and Pong as compared to Vanilla DQN.
2. Even when the action space is increased, the Dueling DQN is able to maintain its performance because of the decoupling between value and advantage functions whereas Vanilla DQN performs poorly.
3. In simple environments like CartPole, there is not much benefit in decoupling value and advantage functions, so the Vanilla DQN offers good performance even with a simple architecture.

THANK YOU!