Assignment 2: Deep Q-Network Implementation

Sachin Kulkarni - UBIT : sk429

1 The DQN network on 3 environments

1.1 Introduction

Abstract

This assignment involved the implementation of the Deep Q network algorithm from scratch. The algorithm was then used to solve three different environments. The drone environment which was from the assignment 1 , the cartpole environment and Frozen lake environment.

A Deep Q-Network is basically an extension of Q-learning, a reinforcement learning algorithm, where a deep neural network is used to approximate the Q-value function. This approach allows Q-learning to scale to high-dimensional state spaces.

The Bellman equation for Q-learning is:

$$Q^*(s,a) = r + \gamma \max_{a'} Q^*(s',a')$$

In Deep Q-Networks (DQN), the loss function is defined as:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[(y - Q(s,a;\theta))^2\right]$$

where the target Q-value is:

$$y = r + \gamma \max_{a'} Q(s',a';\theta^-)$$

The gradient update for optimizing the neural network parameters θ is:

$$\nabla_\theta L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[(y - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta)\right]$$

where: - θ are the parameters of the Q-network, - θ^- are the parameters of the target network, updated periodically, - D is the experience replay buffer.

1.2 Benefits of Using experience replay in DQN

- The main benefit of using the replay buffer in the deep q networks is that it breaks the temporal correlations between experiences. The idea here is that the algorithm samples random episodes which are not correlated. As we know that while working with neural networks, for a model to learn better from the data, it should be independent of corelation between the data samples. Hence, replay buffer helps to store the experiences and then we can sample random experiences to train out network. It also smooths the learning distribution by mixing old and new experiences.

  The size of buffer influences how well the network is generalized. The is if the replay buffer size is small then the network has chance of overfitting where as if the buffer size is large it will generalize better.

1.3 Introducing the target network

- The benefit of using saperate target network In Deep Q-Networks is used to compute the target Q-values, while the main Q-network is updated during training. If the target values are computed using the same network that is being updated, the training process can oscillate or diverge, hence 2 separate networks are needed. The target network parameters are periodically updated by copying the main network parameters. This will eliminate moving target problem. We know that in neural networks the target value should be fixed to learn better. So by using two networks we make the q values of the target fixed for certain number of iterations and then again update the network with new learned weights.

1.4 Representing the Q function as q(s,w)

- This is a game changer introduced by the deep mind.The Q-function is represented as where a deep neural network approximates the Q-values instead of using a traditional Q-table. This approach offers several key benefits. Firstly, it enables function approximation in large or continuous state spaces, overcoming the limitations of tabular Q-learning, which becomes infeasible as the number of states and actions increases. This representation also allows handling high-dimensional inputs, such as raw pixel images from Atari games, by leveraging deep neural networks to learn hierarchical features. The Generalization to unseen states is improved, as the network learns patterns rather than memorizing specific state action values. The whole benefit of using neural network as it is the system used to generalize anything. Hence, representing the Q-function as in Deep Q-Networks significantly enhances scalability, stability, and efficiency, making deep reinforcement learning applicable to high-dimensional and complex decision-making problems.

2 Environment Specifications

2.1 CartPole-v1

- The goal of the agent is to balance a pole, which is attached to a cart, in an upright position for as long as possible.
- The agent here is a model that controls the cart's movements by applying forces to the left or right.
- **State**: The state space consists of four continuous value variables:

  Cart Position: The position of the cart along a horizontal axis.

  Cart Velocity: The velocity of the cart.

  Pole Angle: The angle of the pole relative to the vertical position.

  Pole Velocity: The angular velocity of the pole.
- **Actions**: The agent can take two discrete actions:

  Move the cart to the left.

  Move the cart to the right.
- **Reward**: The agent receives a reward of +1 for every time step the pole remains upright. The goal is to maximize the total reward by keeping the pole balanced
- The episode ends when the pole's angle exceeds a certain threshold or if the cart moves too far off the track, which is often defined by a position boundary. The task is relatively simple but becomes progressively harder with more complex control strategies or for extended time limits. In CartPole-v1, the episode terminates after 500 steps, and the agent is expected to balance the pole for this duration.

2.2 FrozenLake-v1

- The FrozenLake-v1 environment is a slippery surface environment. This slripness brings the scholastic behavior. The agent must learn an optimal policy to reach a goal while avoiding pitfalls.
- The state space consists of 16 discrete states (for a 4x4 grid), where each state corresponds to a unique position on the grid.The states are typically encoded as a one-hot vector, that means the agent's position is represented as an index in a 16-dimensional vector.
- The reward formulation for this purpose i have given +10 if the agent reaches the goal (G) and 0.01 for all other moves including stepping on a frozen tile or slipping.No penalty for falling into a hole, but the episode terminates immediately.

2.3 AutoDrone Grid World

- The autonomous drone delivery system where the agent (drone) interacts with the environment (the grid world) to pick up the object from the warehouse and deliver it to the desired delivery point. The grid world has obstacles like no fly zone which needs to be avoided while planning the path.
- I have modeled this deterministic environment as below: State : The position of the drone in the grid space along with information whether it is carrying the package or not.
- Action: There are 6 possible actions a drone can take. Up, down, left, right, pickup, drop
- Reward: -1 for each step it takes, -100 if the drone goes to no fly zone cell, +25 if the drone picks up the package, +100 if the drone successfully drops the package to the goal position. -10 if the drone either performs drop or pickup in unnecessary situations.
- Objective: The drone must navigate from the warehouse to the delivery location while avoiding no fly zones, picking up the package, and delivering it.

$$\mathcal{R} = \begin{cases} -100 & \text{no-fly zone} \\ 25 & \text{pickup} \\ 100 & \text{delivery} \end{cases} \tag{1}$$

3 Training Plots of 3 Environments using DQN

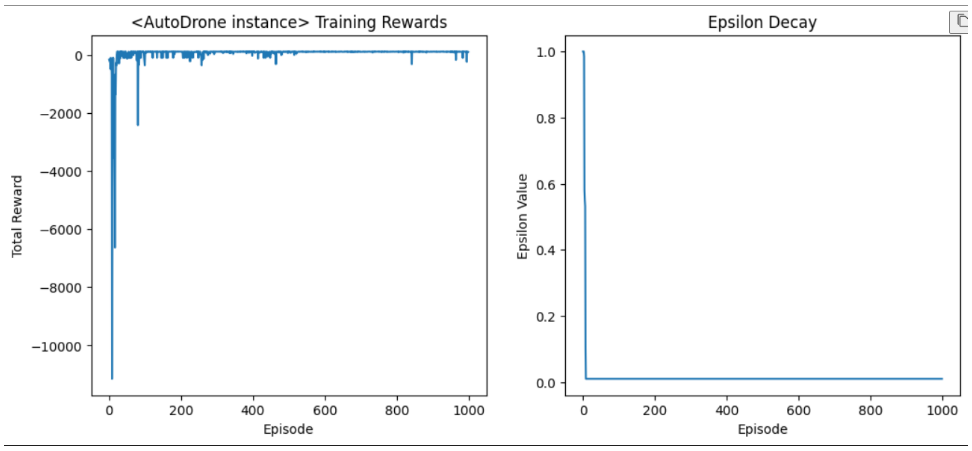

Figure 1: Reward progression and epsilon decay in AutoDrone environment

- The reward starts at a very low value in the initial episodes, indicating that the agent was struggling to reach its goal. However, as training progresses, the reward quickly improves and stabilizes around zero, meaning the agent has learned an optimal policy.
- The epsilon value drops steeply within the first few episodes, showing that the agent quickly transitions from exploration to exploitation.
- The agent successfully learned to navigate the grid-based environment and improved its performance quickly. The steep improvement in rewards suggests that the environment is relatively simple for the agent to master once it figures out the optimal path

- CartPole is a more dynamic environment where small errors can lead to failure, causing the reward to fluctuate.The reward increases steadily at first, showing that the agent is improving in balancing the pole. However, there are significant fluctuations even in later episodes, where rewards drop suddenly.

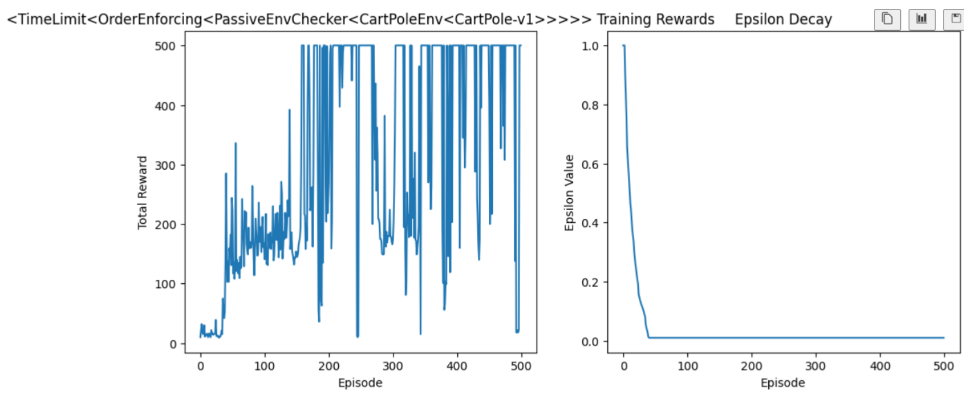

Figure 2: Reward progression and epsilon decay in CartPole environment

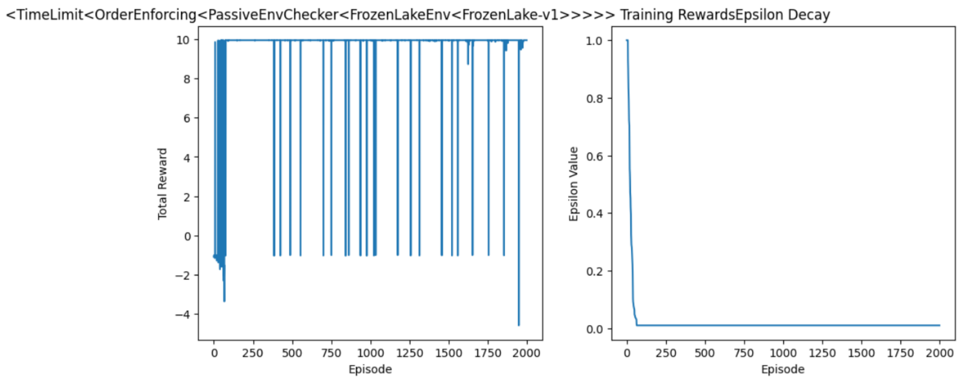

Figure 3: Reward progression and epsilon decay in FrozenLake environment

- The epsilon value decreases more gradually compared to the Drone Gridworld environment, meaning the agent explores for a longer period before committing to exploitation

- Unlike the previous two environments, Frozen Lake introduces stochasticity, meaning that even with an optimal policy, the agent can still fail due to slipping. The occasional reward drops are a sign that randomness plays a role in success. The agent eventually learns an optimal path, but the inherent uncertainty makes it harder to achieve consistent performance.The reward starts low, with negative values in the beginning. Over time, the rewards improve, but there are still noticeable drops even after 1000+ episodes. The agent sometimes fails due to slipping on ice.
- The epsilon decay pattern is similar to the other environments, but the agent takes longer to stabilize due to the randomness in the environment

4 Policy Evaluation

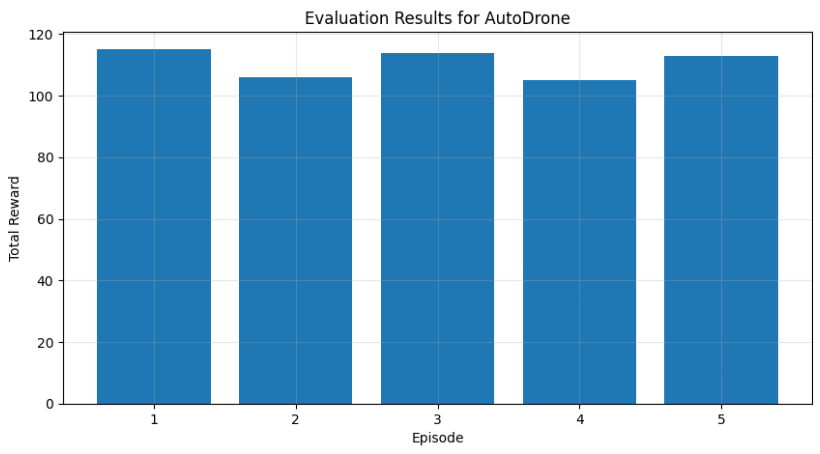

Figure 4: The Evaluation results of 5 run on grid world environment

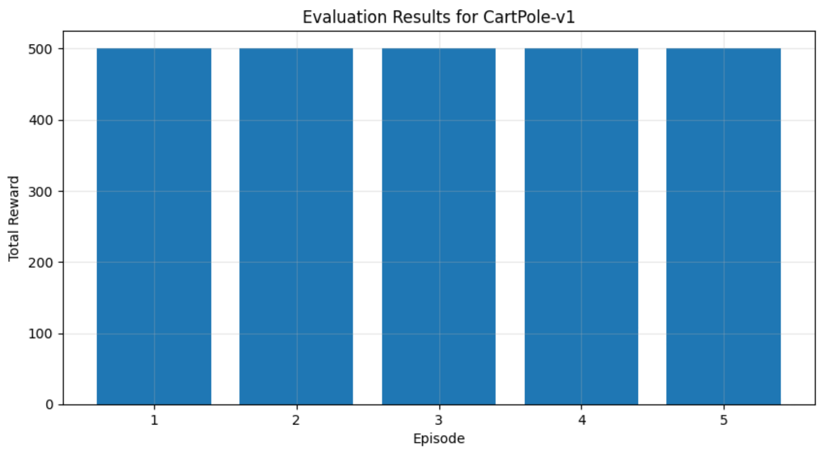

Figure 5: The Evaluation results of 5 run on cartpole environment

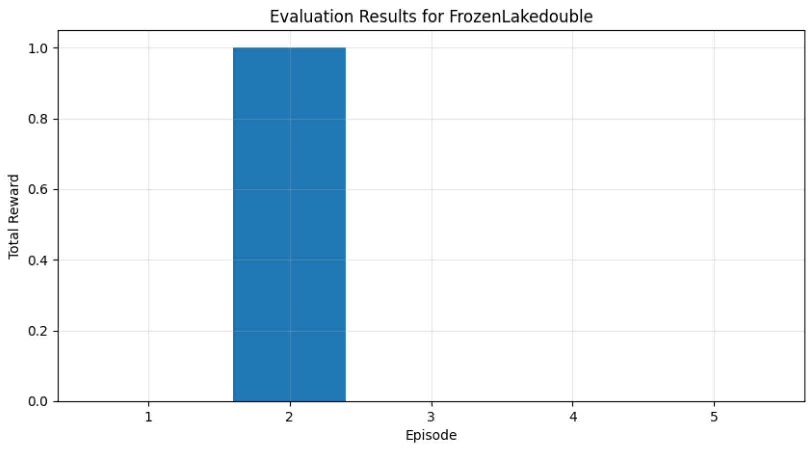

Figure 6: The Evaluation results of 5 run on Frozen Lake environment

5 Double DQN on 3 environments

5.1 Introduction

As discussed in above sections, DQN have literally revolutionized reinforcement learning by enabling agents to learn complex policies from high dimensional state spaces by integrating neural network to learn policy because we all know that neural networks are better at learning generalized functions. However, vanilla DQN suffers from overestimation bias, leading to suboptimal policies. To solve this issue, we implemented Double DQN, which decouples action selection and value estimation to provide more stable learning. Double DQN improves upon DQN by using the main network for action selection and the target network for value estimation, reducing overestimation bias. The update rule is modified as follows:

$$y_t = r_t + \gamma Q_{\text{target}}(s_{t+1}, \arg\max_a Q_{\text{online}}(s_{t+1}, a; \theta_t); \theta^-)$$

Where:

- y_t is the target for the Q-value update.
- r_t is the reward at time t.
- γ is the discount factor.
- $Q_{\text{target}}(s_{t+1}, \arg\max_a Q_{\text{online}}(s_{t+1}, a; \theta_t); \theta^-)$ is the Q-value of the next state s_{t+1} according to the target network, where $\arg\max_a Q_{\text{online}}(s_{t+1}, a; \theta_t)$ selects the action with the highest Q-value from the online network at the next state.
- θ_t represents the parameters of the online network.
- θ^- represents the parameters of the target network.

The main improvement of double DQN over vanilla DQN is that it reduces the overestimation of Q-values by ensuring that the action chosen by the main network is evaluated with a separate stable target network. This leads to more stable and accurate learning in the long run. As the agent is less likely to incorrectly estimate the value of actions because it may suffer from overfitting the Q-values.

5.2 Implementing The Double DQN on envs

Below are the plots that are generated to show the variation in the reward episode that we got by running the Double DQN agent on grid world, cartpole and Frozen lake environments.

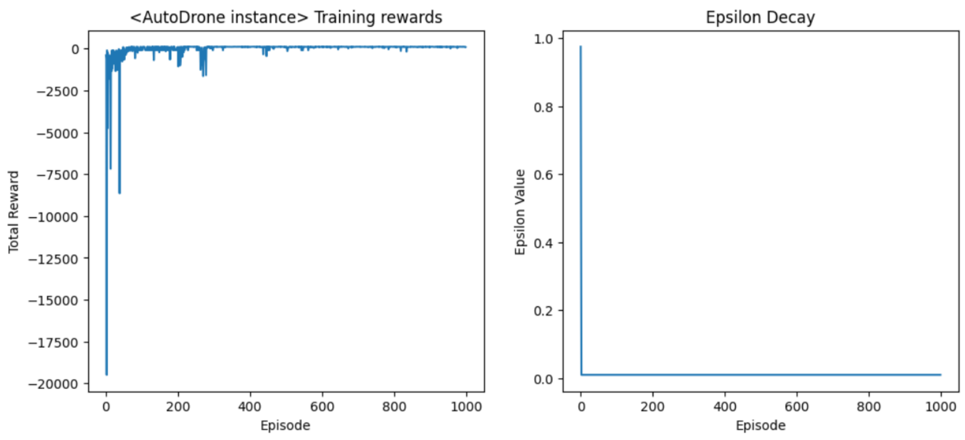

Figure 7: Reward progression and epsilon decay in Drone environment with Double DQN

- For the drone environment we can see that the agent gives negative reward initally but converges to a optimal policy lateron and preforms good

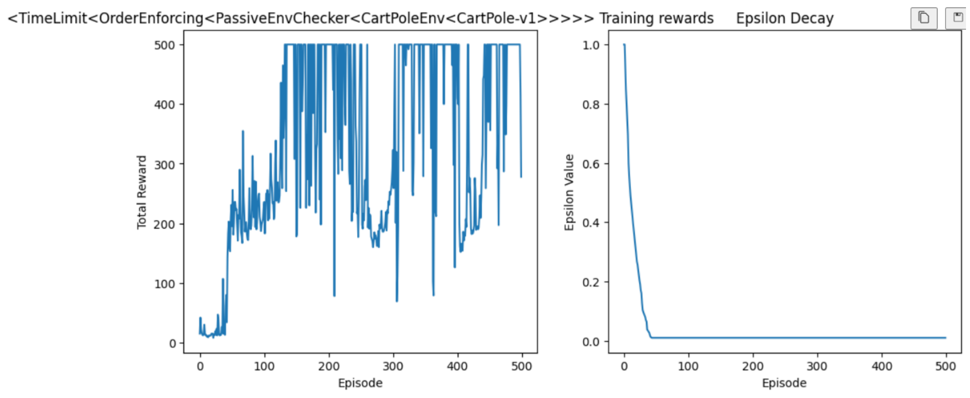

Figure 8: Reward progression and epsilon decay in CartPole environment with Double DQN

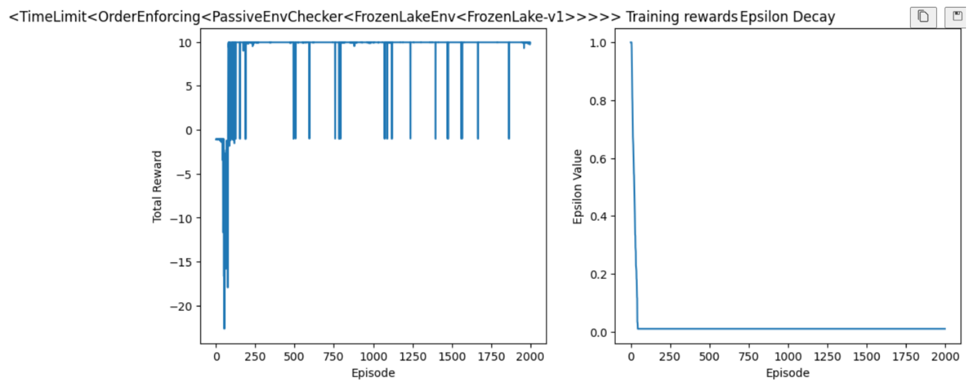

Figure 9: Reward progression and epsilon decay in Frozen Lake environment with Double DQN

- For the cartpole environment the agent initially struggles but gradually improves, achieving consistently high rewards. The agent initially struggles but gradually improves, achieving consistently high rewards
- For the frozen lake environment, the agent starts with negative rewards, indicating poor performance in the early episodes and later on it improves significantly, reaching mostly stable rewards. we can also see some sudden drops in rewards suggest occasional failures in reaching the goal due to exploration or stochastic nature of the environment.

6 Comparing DQN and Double DQN

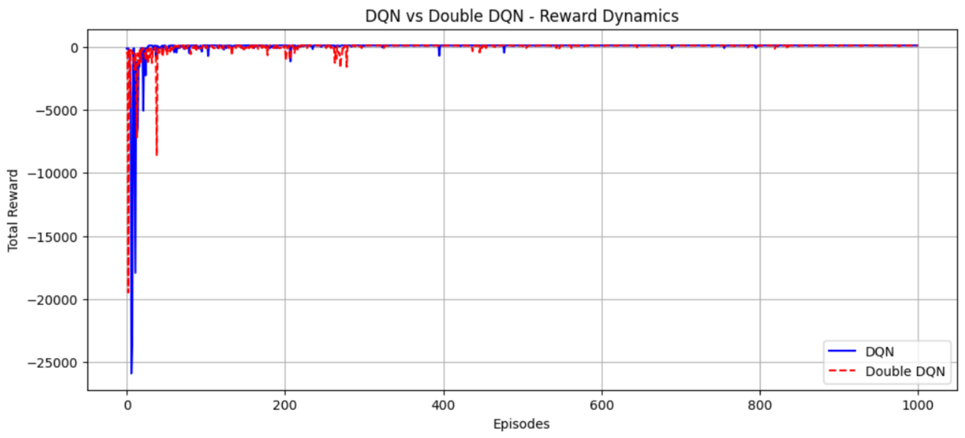

Figure 10: Comparing DQN with Double DQN on Drone environment

- We can clearly see that both DQN and Double DQN converge to stable rewards, but Double DQN exhibits fewer fluctuations. This tells us that simple environments with deterministic state transitions, DQN may still work well, but Double DQN reduces overestimation bias, leading to slightly more stable learning.

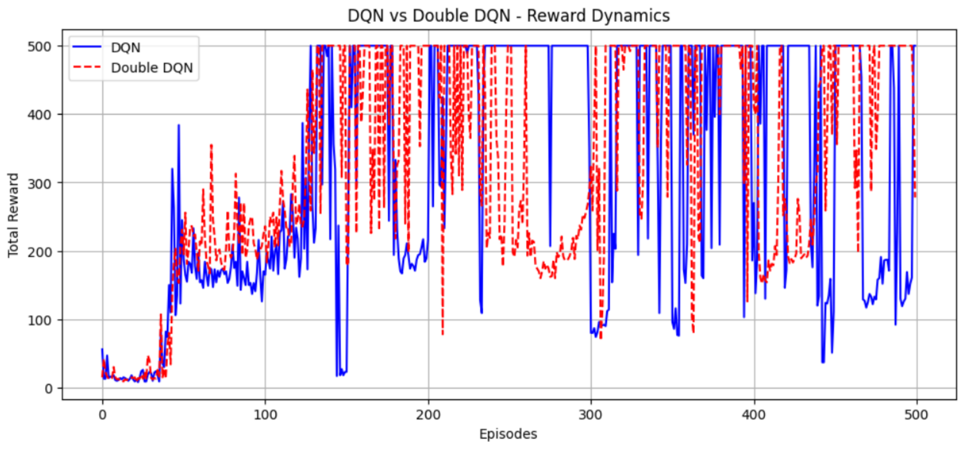

Figure 11: Comparing DQN with Double DQN on CartPole environment

- Both the agents are almost same, the total reward increases over episodes for both algorithms, but Double DQN has more fluctuations. CartPole-v1 is a more dynamic control task requiring precise balancing, we still need to tune hyperparameters more precisely according to me.

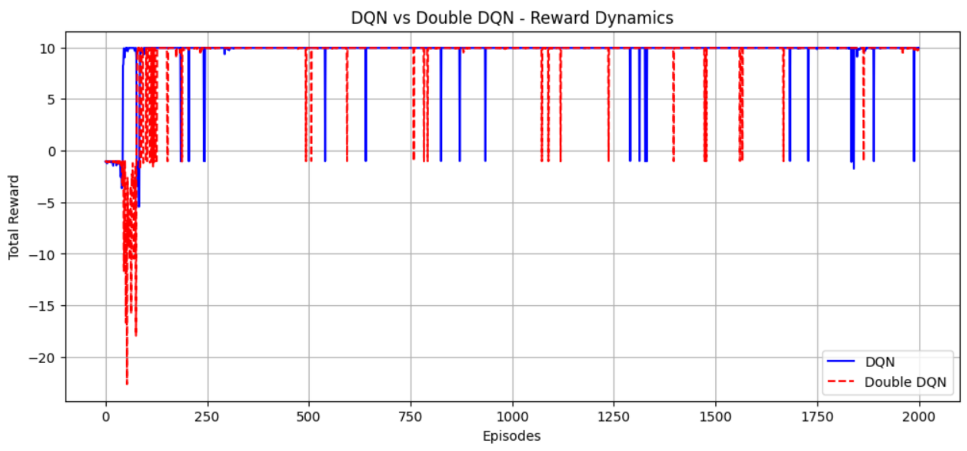

Figure 12: Comparing DQN with Double DQN on Frozen Lake environment

- In the frozen lake complex environment, we can see that learning curve starts with a steep drop and then stabilizes for both algorithms, The double DQN takes more time to converge but once it learned the optimal policy then it make few mistakes compared to DQN. In complex environments, standard DQN struggles with Q-value overestimation, leading to suboptimal policies. Double DQN, while still fluctuating, reduces this issue, resulting in better performance over long training periods.

By seeing all the above comparision we can say that DQN struggles with overestimations, leading to unstable learning in complex environments where as Double DQN improves stability but may still require tuning for optimal performance. The performance gap widens as complexity increases, making Double DQN preferable for larger environments.

7 Evaluation of Double DQN

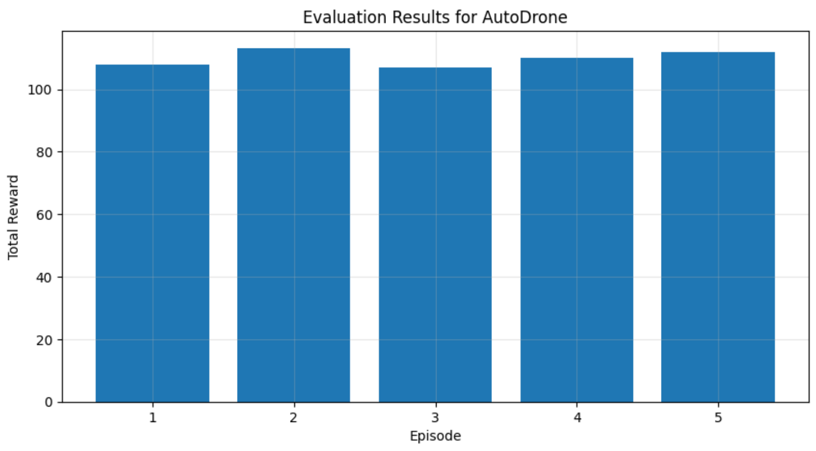

Figure 13: The Evaluation results of 5 run on grid world environment for Double DQN

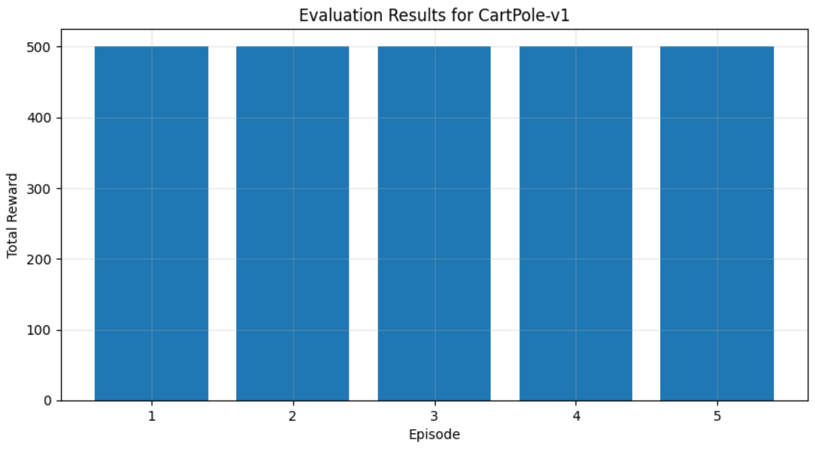

Figure 14: The Evaluation results of 5 run on cartpole environment for Double DQN

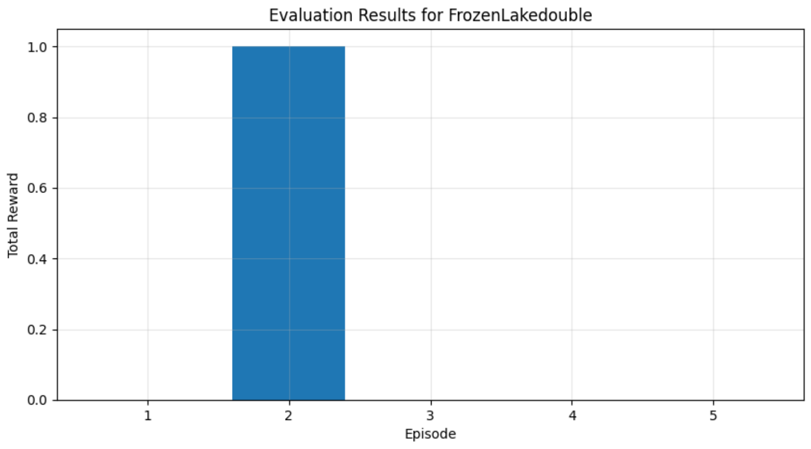

Figure 15: The Evaluation results of 5 run on Frozen lake environment for Double DQN

8 References

1. Mnih, V. et al. (2015). *Human-level control through deep RL.* Nature, 518(7540), 529-533.
2. Overleaf (LaTex based online document generator)- a free tool for creating professional reports
3. Lecture slides
4. Sutton, R.S. & Barto, A.G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.
5. Gymnasium Documentation. `https://gymnasium.farama.org/`

9 Contribution

- Sachin Kulkarni sk429 : 90 percent
- Rutuja : 10 percent