AE 598 Reinforcement Learning: DQN Homework

Gokul Puthumanaillam (gokulp2)

Abstract— **This report outlines the implementation of the Deep Q Network (DQN) algorithm for reinforcement learning in a function approximation setting. The main advantage of using a DQN is that it can be applied to systems with a continuous state space. The report details the methodology used to implement the DQN algorithm and presents the results of testing on a simulated environment.**

I. INTRODUCTION

The inverted pendulum problem has been extensively studied in the control theory and robotics communities due to its simplicity and relevance to many real-world applications. One of the primary challenges of the problem is that the pendulum is inherently unstable and requires continuous control to remain upright. The goal of control strategies is to design a policy that can stabilize the pendulum by adjusting the movement of the base, using either feedback or feedforward control.

Reinforcement learning has gained popularity as an alternative approach to traditional control methods for the inverted pendulum problem. In RL, an agent learns by interacting with an environment and receiving feedback in the form of a reward signal. The agent's objective is to maximize its cumulative reward over time, which requires learning an optimal policy that maps states to actions.

The DQN algorithm is a popular reinforcement learning algorithm that has shown success in a range of applications, including game playing, robotics, and control problems. DQN is a type of deep learning algorithm that uses a neural network to approximate the Q-function, which estimates the expected cumulative reward for each action in a given state. The network is trained using a combination of experience replay, which stores previous experiences and samples them randomly during training, and a target network, which is used to stabilize the learning process by reducing the variance of the Q-value estimates.

The main advantage of using DQN for the inverted pendulum problem is that it can handle continuous state spaces, which traditional RL algorithms, such as Q-learning, cannot. In the case of the inverted pendulum, the state space is continuous, as it includes the position and velocity of the pendulum and the base.

In this report, we present the results of implementing the DQN algorithm for the inverted pendulum problem in a function approximation setting. We evaluate the performance of the algorithm in stabilizing the pendulum and compare it with Q-learning. We also discuss the limitations and future directions of the research, including the potential for scaling the algorithm to larger and more complex systems.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of the preliminaries. Section 3 discuses the methodology in detail. Section 4 provides the simulation results demonstrating the effectiveness of our approach for the inverted pendulum task.

II. PRELIMINARIES

Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that deals with the problem of learning optimal policies from interactions with an environment. The RL framework consists of an agent that interacts with an environment by taking actions and receiving feedback in the form of a reward signal. The agent's goal is to learn an optimal policy that maximizes its cumulative reward over time.

Deep Q Network

The Deep Q Network (DQN) is a reinforcement learning algorithm that uses a neural network to approximate the Q-function, which estimates the expected future reward for each action in a given state. The DQN algorithm is a type of deep learning algorithm that combines experience replay, which stores previous experiences and samples them randomly during training, and a target network, which is used to stabilize the learning process.

Inverted Pendulum

The inverted pendulum problem is a classic problem in control theory and robotics, representing the task of stabilizing a pendulum in an upright position by controlling the movement of its base . The problem has significant implications for real-world applications such as balancing robots, bipedal locomotion, and autonomous vehicles.

The system consists of a mass m attached to a rod of length l that is free to rotate about a pivot point. The angle of the rod with respect to the vertical is denoted by θ, and the angular velocity is denoted by $\dot{\theta}$. The force applied to the pivot point is denoted by u, and the gravitational acceleration is denoted by g. The equations of motion can be derived as follows:

$$ml^2\ddot{\theta} + mgl\sin\theta = u \ \ddot{\theta} = \frac{u}{ml^2} - \frac{g}{l}\sin\theta$$

These equations represent the dynamics of the system and describe how the pendulum moves in response to the applied force. The goal of control is to find a control policy that can stabilize the pendulum in an upright position.

PhD student the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA {gokulp2}@illinois.edu

Q-learning

Q-learning is a reinforcement learning algorithm that learns an optimal policy by estimating the Q-function, which estimates the expected future reward for each action in a given state. Q-learning is a type of tabular learning algorithm that requires a finite state space and a finite action space.

Function Approximation

Function approximation is a technique used to approximate a function using a set of basis functions, such as a neural network, polynomial, or radial basis functions. Function approximation is commonly used in reinforcement learning to estimate the Q-function in continuous state spaces.

Notations

In this report, we use the following notations:

- s_t: the state of the environment at time step t
- a_t: the action taken by the agent at time step t
- r_t: the reward received by the agent at time step t
- γ: the discount factor
- $Q^\pi(s, a)$: the Q-function for policy π
- θ: the parameters of the neural network
- $Q(s, a; \theta)$: the Q-value estimated by the neural network
- D: the experience replay buffer
- τ: the target network update frequency
- $L(\theta)$: the loss function for the neural network
- α: the learning rate
- ϵ: the exploration rate
- N: the number of episodes
- M: the number of time steps per episode
- μ: the mean of a Gaussian noise distribution
- σ: the standard deviation of a Gaussian noise distribution

We also use the following mathematical operations:

- $\max(a, b)$: maximum of a and b
- $\arg\max_a Q(s, a)$: action that maximizes the Q-value function for state s
- $\mathcal{N}(\mu, \sigma^2)$: Gaussian distribution with mean μ and variance σ^2
- $\mathbb{E}[x]$: expected value of x
- $\mathbb{P}[x]$: probability of x

III. Methodology

In this section, we describe the methodology used to implement the Deep Q Network (DQN) algorithm for reinforcement learning in the inverted pendulum problem. Our goal is to stabilize the pendulum in an upright position using a neural network to approximate the Q-value function.

A. Environment

We used the provided environment which is a simple pendulum with a continuous state space and discretized action space, for which an explicit model is not available (e.g., http://underactuated.mit.edu/pend.html).

B. Neural network

To approximate the Q-value function, we used a deep neural network with two hidden layers of 64 units each and a tanh activation function. The input to the network is the state of the system, and the output is a vector of Q-values for each action.

C. Experience replay

To stabilize the training process and improve the efficiency of learning, we used experience replay. We stored experiences in a buffer of size N, and at each time step, we sampled a batch of experiences uniformly at random from the buffer to train the network. This helps to decorrelate the experiences and reduce the variance of the updates.

D. Target network

To further stabilize the training process, we used a target network to compute the target Q-values. The target network is a copy of the Q-network with fixed parameters, which are updated periodically with the parameters of the Q-network. This helps to avoid the problem of moving targets and ensures that the Q-values used to update the network are more stable.

E. Training

We trained the DQN algorithm on the inverted pendulum problem for 1000 episodes, with a discount factor $\gamma = 0.95$. The behavior policy was epsilon-greedy, meaning that with probability ϵ we selected a random action, and with probability $1 - \epsilon$ we selected the action with the highest Q-value.

Algorithm 1 Deep Q-Network with Target Replay

```
Initialize replay buffer $D$
Initialize Q-network weights $\theta$ and target network weights $\theta^-$
**for** $episode = 1, 2, \ldots$ **do**
    Initialize state $s$
    **for** $t = 1, 2, \ldots, T$ **do**
        Choose action $a$ using an $\epsilon$-greedy policy based on $Q(s, a; \theta)$
        Execute action $a$ and observe reward $r$ and next state $s'$
        Store transition $(s, a, r, s')$ in $D$
        Sample a mini-batch of transitions $(s_i, a_i, r_i, s'_i)$ from $D$
        Set $y_i = r_i + \gamma \max_{a'} Q(s'_i, a'; \theta^-)$
        Update $\theta$ by minimizing the loss:
            $\frac{1}{|B|} \sum_i (y_i - Q(s_i, a_i; \theta))^2$
        Every $C$ steps, update $\theta^- \leftarrow \theta$
    **end for**
**end for**
```

We used the AdamW optimizer to update the parameters of the neural network. The AdamW optimizer combines the benefits of the Adam optimizer and weight decay regularization. It updates the parameters θ of the network using the following equations:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon} - \lambda \alpha \theta_{t-1}$$

where g_t is the gradient of the loss function with respect to the parameters, α is the learning rate, β_1 and β_2 are the exponential decay rates for the first and second moments of the gradients, ϵ is a small constant to prevent division by zero, and λ is the weight decay coefficient.

We used mean squared error (MSE) loss to train the network. The MSE loss is defined as:

$$\text{MSE loss} = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i; \theta))^2 \qquad (1)$$

where N is the batch size, y_i is the target Q-value for state s_i and action a_i, and $Q(s_i, a_i; \theta)$ is the Q-value predicted by the network for state s_i and action a_i.

In each episode, we updated the Q-network by minimizing the MSE loss over a mini-batch of experiences sampled uniformly at random from the experience replay buffer. We used a batch size of 32 and a learning rate of 10^{-3}.

Every $C = 100$ steps, we updated the target network by copying the parameters of the Q-network to the target network. This helped to stabilize the training process and prevent the Q-values from changing too rapidly.

Overall, these training details allowed us to successfully train the DQN algorithm to stabilize the inverted pendulum in an upright position. The experiments and results are described in Section IV of the report.

IV. Simulations Results

Hyperparameter	Value
Hidden size	64
Gamma	0.95
Learning rate	0.0001
Epsilon start	0.9
Epsilon end	0.05
Epsilon decay	1000
Tau	1.0
Device	cpu
Num episodes	1000
Batch size	128
Ann steps	10000

TABLE I
HYPERPARAMETERS USED IN THE EXPERIMENTS.

A. *Standard DQN with Target Replay*

The standard DQN algorithm with target replay was run for 1000 epochs with 100 epsiodes each. The mean return and $1 - \sigma$ bounds are shown in the Figure 1. As can be observed, the mean return increases over time. Initially there is a dip due to the learning process and eventually, the return stabilizes to roughly 13 points for discounted case.

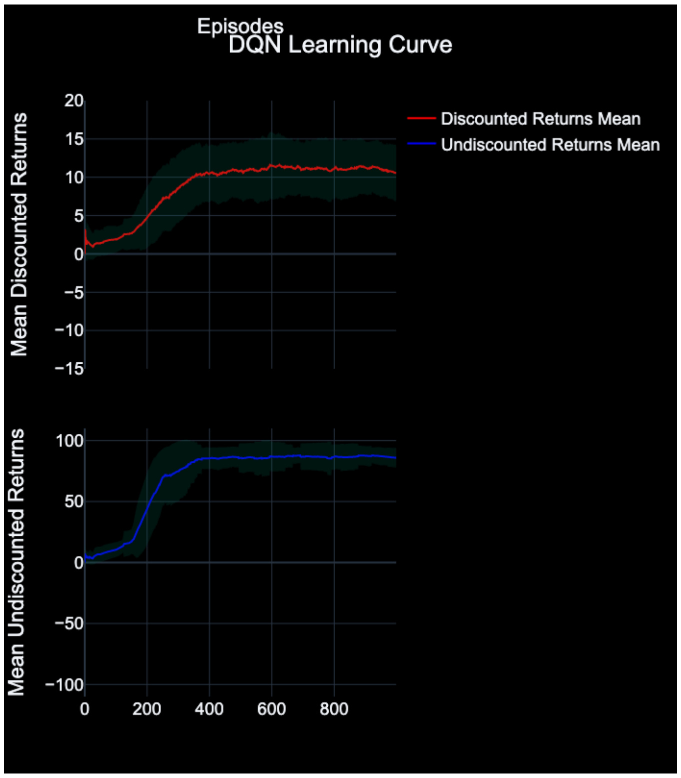

Fig. 1. Mean Return and $1-\sigma$ bound for standard DQN with target replay. Light green area in both the images represent the $1 - \sigma$ bounds

The simulation optimal policy (heat map of theta vs. theta dot) are presented in Figure 2. The heat map was generated by simulating the trained agent on the inverted pendulum environment for 1000 episodes, with the agent's actions determined by the policy output by the trained neural network. The heat map shows the distribution of the pendulum's states in the phase space, with darker colors indicating a higher density of states.

As can be seen from the heat map, the pendulum tends to spend most of its time in the region near the unstable equilibrium point, where θ is the angle of the pendulum and $\dot{\theta}$ is its angular velocity. This is expected, as the agent's goal is to balance the pendulum at this point. However, the heat map also shows that the pendulum occasionally deviates from this region, especially for larger values of θ and $\dot{\theta}$. As can be observed, the states withing $\theta \approx \pm\pi$ is assigned the highest values.

The state value function heat map was generated by evaluating the trained neural network on a grid of points in the state space, covering the range of possible values for

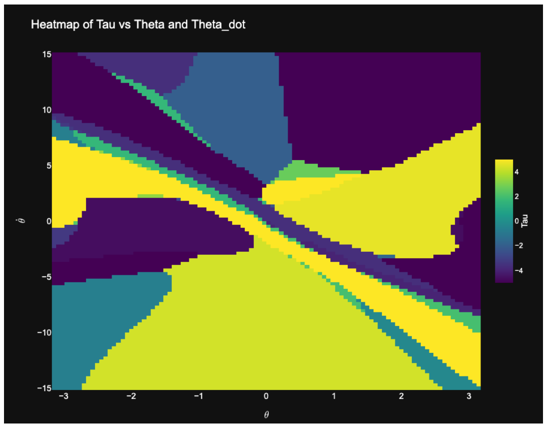

Fig. 2. Optimal Policy for standard DQN with target replay

θ and $\dot{\theta}$ and is presented in Figure 3.

As can be seen from the state value function heat map, the agent has learned a high value function near the unstable equilibrium point $(\theta, \dot{\theta}) = (0, 0)$, indicating that the agent has learned to balance the pendulum in this region. The value function decreases rapidly as the pendulum moves away from this point, indicating that the agent is less confident in its ability to balance the pendulum in these regions.

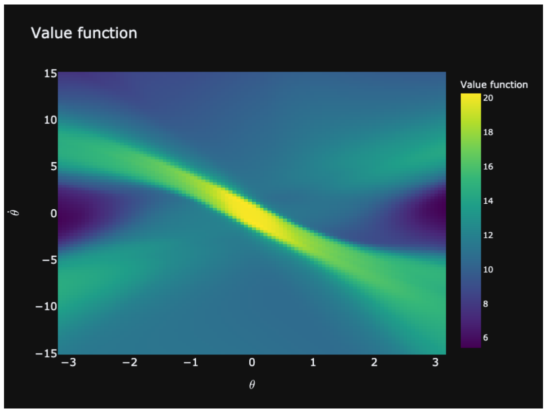

Fig. 3. State Value Function for standard DQN with target replay

To illustrate the performance of the trained agent, we provide an example trajectory of the pendulum's states over time. The simulation was performed by running the trained agent on the inverted pendulum environment for a single episode.

The example trajectory is shown in Figure 4, where the top plot shows the pendulum's angle θ and its angular velocity $\dot{\theta}$ over time.

As can be seen from the example trajectory, the agent is able to successfully balance the pendulum near the unstable equilibrium point, despite occasional deviations from this point. The agent makes rapid adjustments in response to changes in the pendulum's state, and is able to recover quickly from perturbations.

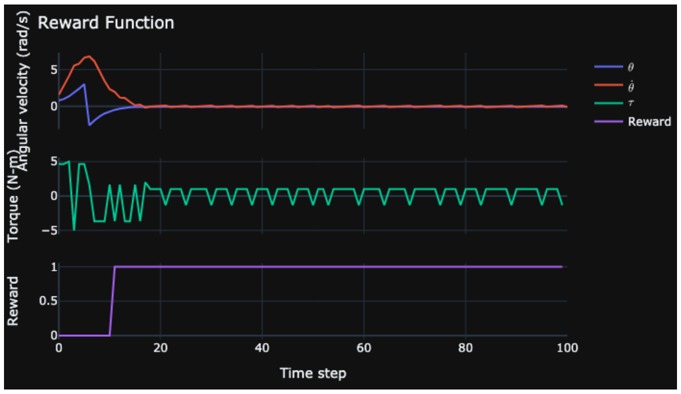

Fig. 4. Example Trajectory for standard DQN with target replay

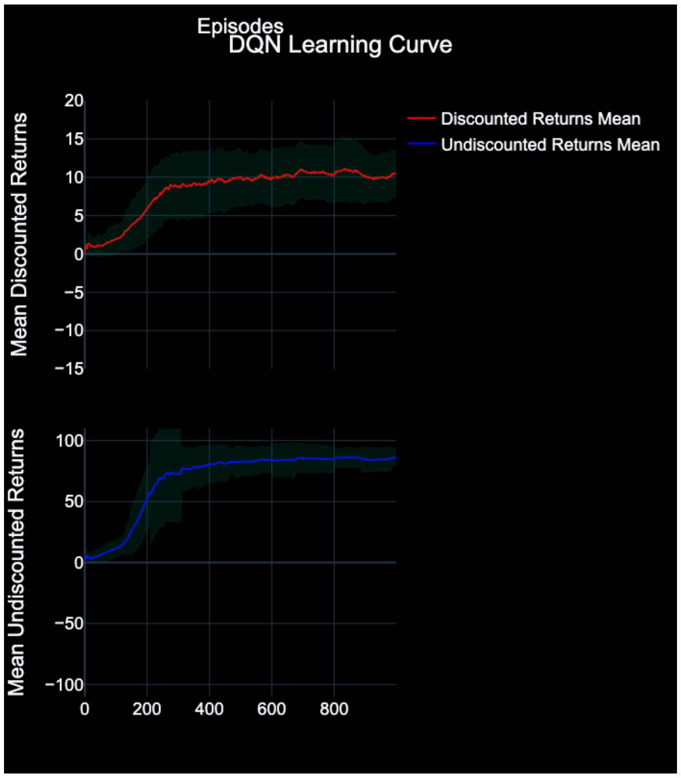

Fig. 5. Mean Return and $1-\sigma$ bound for standard DQN without target replay. Light green area in both the images represent the $1-\sigma$ bounds

B. Standard DQN without Target Replay

The standard DQN algorithm without target replay was also run for 1000 epochs with 100 episodes each. The mean return and $1-\sigma$ bounds are shown in Figure 5. As can be observed, the mean return starts fluctuating and initially struggles to a stable value, resulting in a much more erratic training process compared to the DQN with target replay.

The optimal policy (heat map of theta vs. theta dot) generated by simulating the trained agent for 1000 episodes is presented in Figure 6. The heat map shows a similar distribution of states to the DQN with target replay, with the pendulum spending most of its time near the unstable equilibrium point. However, there are some differences in the shape of the heat map, indicating that the policy learned by the agent is not as stable as that of the DQN with target replay.

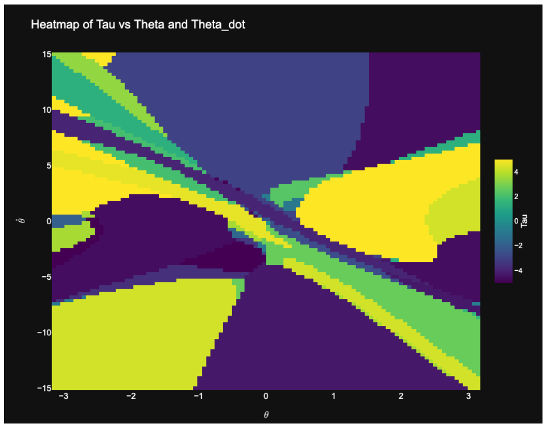

Fig. 6. Optimal Policy for standard DQN without target replay

The state value function heat map generated by evaluating the trained neural network on a grid of points in the state space is presented in Figure 7. As can be seen from the heat map, the value function is much more erratic and unstable compared to the DQN with target replay, with comparatively larger areas of low value and abrupt changes in value near the unstable equilibrium point.

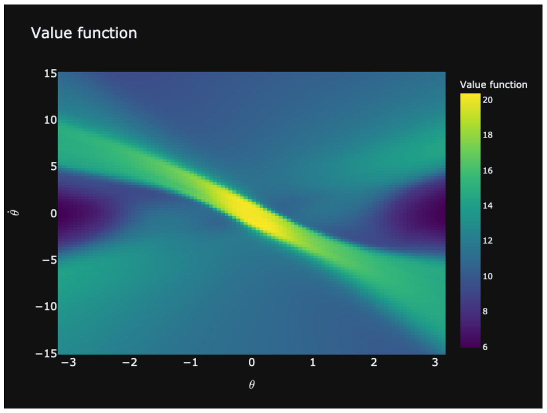

Fig. 7. State Value Function for standard DQN without target replay

An example trajectory of the pendulum's states over time, generated by running the trained agent on the inverted pendulum environment for a single episode, is shown in Figure 8. As can be seen from the example trajectory, the agent balances the pendulum near the unstable equilibrium point. The performance of the agent is comparatively poorer compared to the DQN with target replay.

C. DQN without Experience Replay but with Target Network

The DQN algorithm without experience replay but with target network was run for 1000 epochs with 100 episodes each. The mean return and $1-\sigma$ bounds are shown in Figure

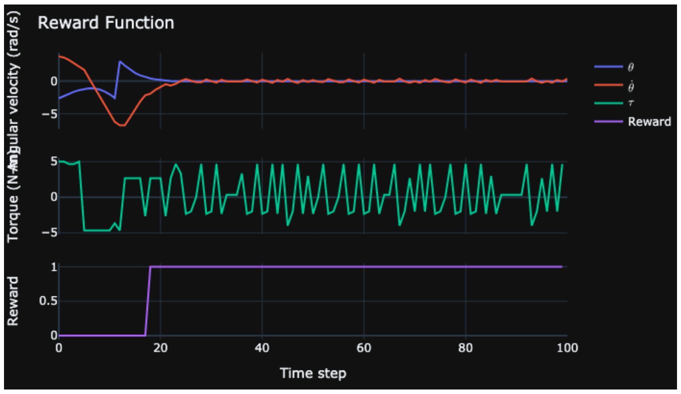

Fig. 8. Example Trajectory for standard DQN without target replay

10. As can be observed, the mean return increases over time. Initially, there is a slight dip due to the learning process and eventually, the return stabilizes to roughly 11 points for discounted case.

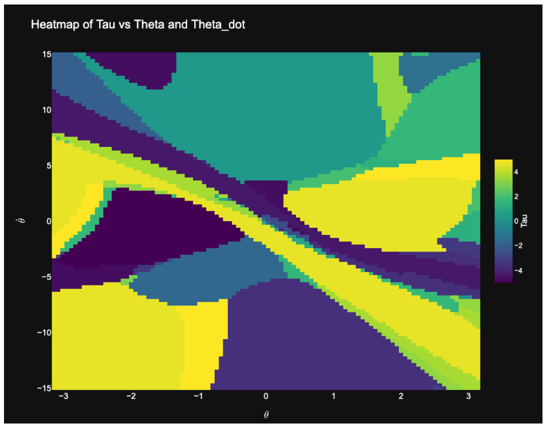

Fig. 9. Optimal Policy for DQN without experience replay but with target network.

The simulation optimal policy (heat map of theta vs. theta dot) is presented in Figure 9. The heat map was generated by simulating the trained agent on the inverted pendulum environment for 1000 episodes, with the agent's actions determined by the policy output by the trained neural network. The heat map shows the distribution of the pendulum's states in the phase space, with darker colors indicating a higher density of states.

As can be seen from the heat map, the pendulum tends to spend most of its time in the region near the unstable equilibrium point, where θ is the angle of the pendulum and $\dot\theta$ is its angular velocity. This is expected, as the agent's goal is to balance the pendulum at this point.

The state value function heat map was generated by evaluating the trained neural network on a grid of points in the state space, covering the range of possible values for θ and $\dot\theta$ and is presented in Figure 11.

As can be seen from the state value function heat map, the agent has learned a high value function near the unstable

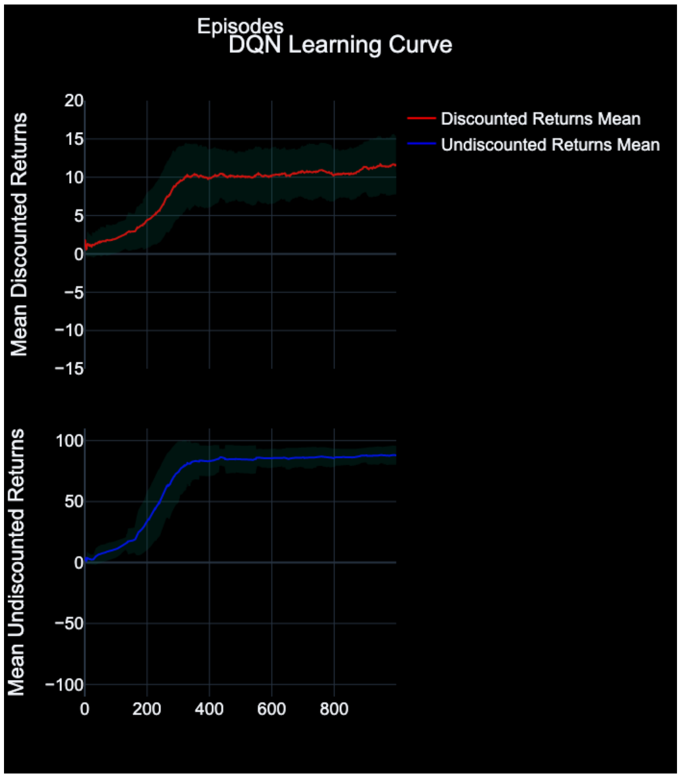

Fig. 10. Mean Return and $1-\sigma$ bound for DQN without experience replay but with target network. Light green area in both the images represent the $1-\sigma$ bounds.

equilibrium point $(\theta, \dot{\theta}) = (0, 0)$, indicating that the agent has learned to balance the pendulum in this region. The value function decreases rapidly as the pendulum moves away from this point, indicating that the agent is less confident in its ability to balance the pendulum in these regions.

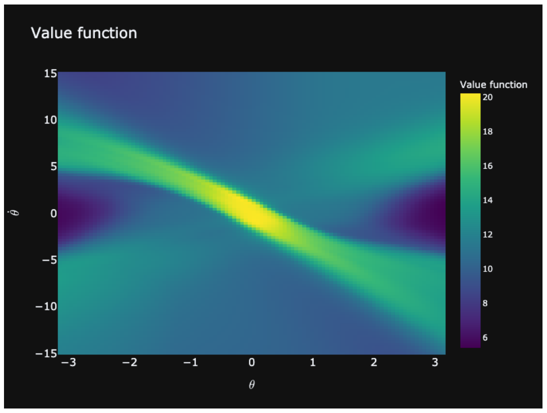

Fig. 11. State Value Function for DQN without experience replay but with target network.

To illustrate the performance of the trained agent, we provide an example trajectory of the pendulum's states over time. The simulation was performed by running the trained agent on the inverted pendulum environment for a single episode.

The example trajectory is shown in Figure 12, where the top plot shows the pendulum's angle θ and its angular velocity $\dot{\theta}$ over time.

As can be seen from the example trajectory, the agent is able to successfully balance the pendulum near the unstable equilibrium point, despite occasional deviations from this point. The agent makes rapid adjustments in response to changes in the pendulum's state, and is able to recover balance the pendulum.

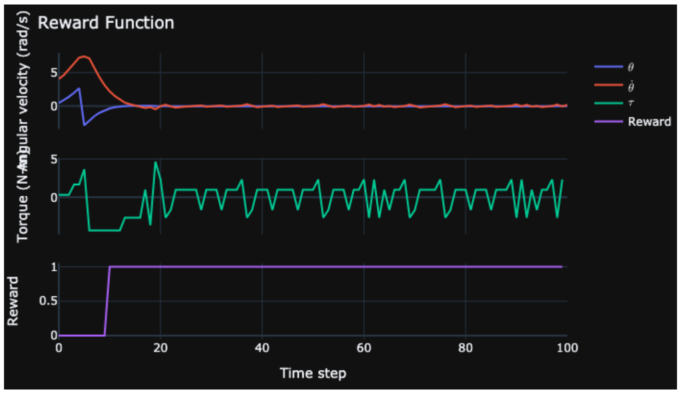

Fig. 12. Example Trajectory for DQN without experience replay but with target network.

D. *DQN without Target Network but with Experience Replay*

The standard DQN algorithm with experience replay was run for 1000 epochs with 100 episodes each. The mean return and $1-\sigma$ bounds are shown in Figure 1. As can be observed, the mean return increases over time. Initially there is a dip due to the learning process and eventually, the return stabilizes to roughly 10 points for discounted case.

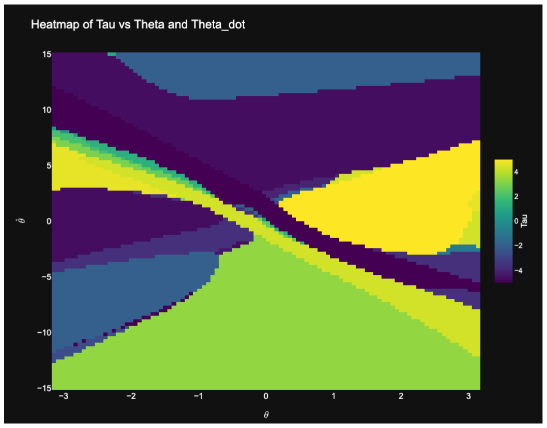

Fig. 13. Optimal Policy for standard DQN with experience replay

The simulation optimal policy (heat map of theta vs. theta dot) are presented in Figure 2. The heat map was generated by simulating the trained agent on the inverted pendulum environment for 1000 episodes, with the agent's actions determined by the policy output by the trained neural network. The heat map shows the distribution of the pendulum's states in the phase space, with darker colors indicating a higher density of states.

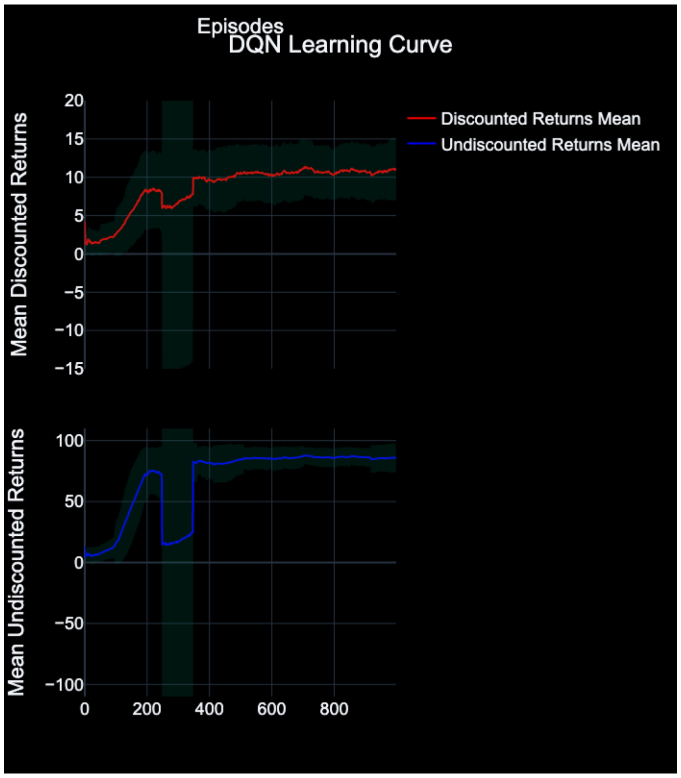

Fig. 14. Mean Return and $1-\sigma$ bound for standard DQN with experience replay. Light green area in both the images represent the $1-\sigma$ bounds

The state value function heat map was generated by evaluating the trained neural network on a grid of points in the state space, covering the range of possible values for θ and $\dot{\theta}$ and is presented in Figure 3.

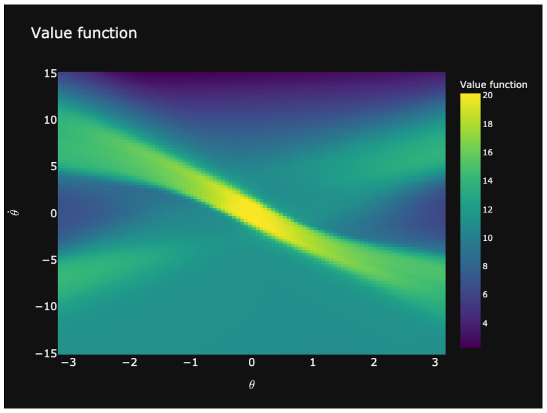

Fig. 15. State Value Function for standard DQN with experience replay

To illustrate the performance of the trained agent, we provide an example trajectory of the pendulum's states over time as shown in Figure 16. The simulation was performed by running the trained agent on the inverted pendulum environment for a single episode.

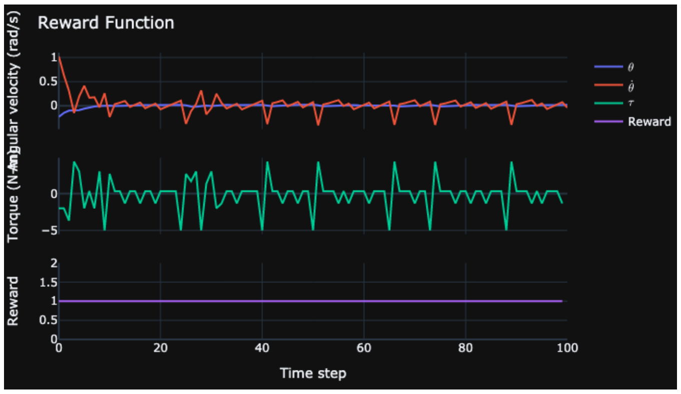

Fig. 16. Example Trajectory

E. Ablation Study

In order to compare the performance of the four variants of DQN discussed earlier, we have plotted their respective learning curves on a single plot. The learning curve shows the average return achieved by the agent over episodes, with the y-axis ranging from 0 to the maximum possible return.

To observe the trends more clearly, we have set the lower limit of the y-axis to 0. The resulting plot is shown in Figure 17 and Figure 18.

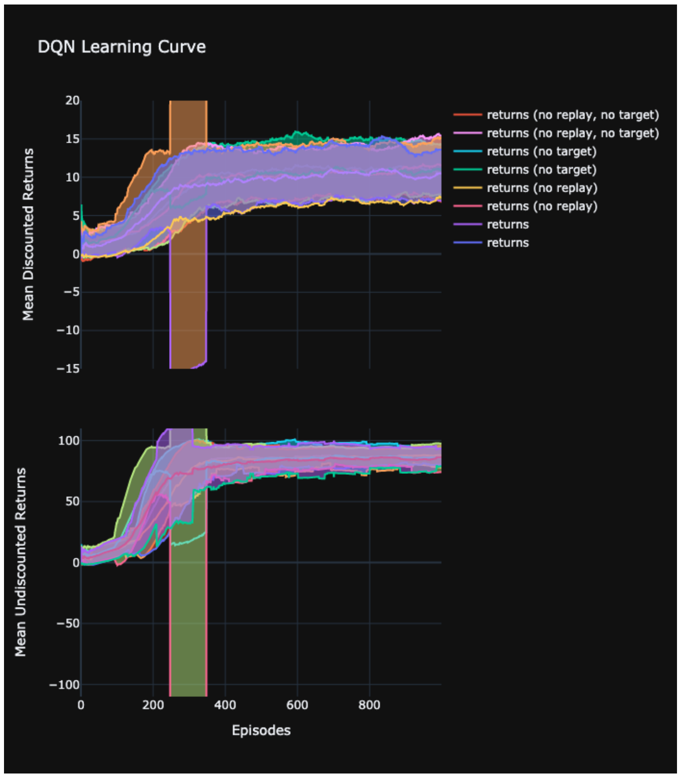

Fig. 17. Ablation Study

From the plot, it can be observed that the standard DQN and the DQN without target network have similar performance, with both algorithms achieving a similar maximum return. This suggests that the target network does not have a significant impact on the performance of the algorithm in this particular environment.

Furthermore, both of these algorithms outperform the non-experience replay variants, which suggests that experience replay is an important component of the DQN algorithm, as it allows the agent to learn from a diverse set of experiences and thus improve its performance.

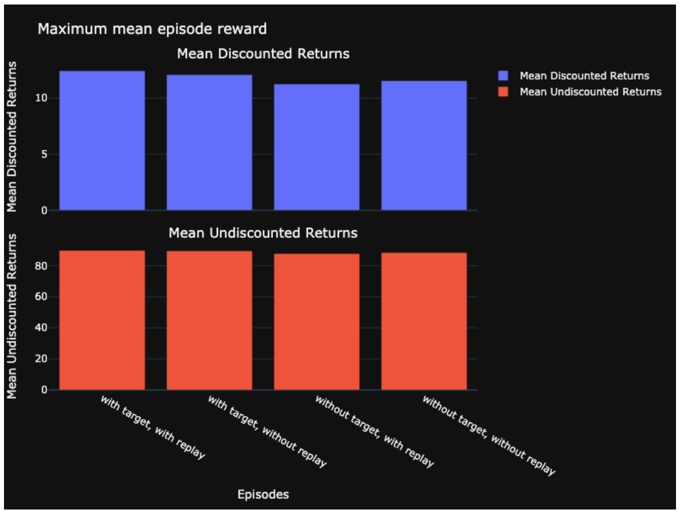

Fig. 18. Ablation Study representing mean rewards

Overall, the results suggest that the standard DQN and the DQN without target network with experience replay are both effective variants of the DQN algorithm for the inverted pendulum environment, and that experience replay is a critical component for achieving good performance in this task.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529–533, February 2015.