Ablation Studies on Deep Q Learning

Vraj Patel

261022581
vraj.patel@mail.mcgill.ca

Abstract

This project aims to conduct ablation studies for the DQN algorithm in two distinct environments: Cartpole from Classic Control and Lunar Lander from Box2D. By selectively removing key components of the DQN algorithm such as the Target network, memory, and exploration rate, The aim is to analyze their individual contributions to the algorithm's overall performance. Experiments employ the same neural network architecture and hyperparameters across both environments, allowed to compare and contrast the training convergence for each scenario. The results of our study could provide valuable insights into the inner workings of the DQN algorithm and inform the design of more efficient RL algorithms in the future.

1 Introduction

Understanding the contribution of individual components or modules of an RL algorithm to its overall performance can be challenging. Ablation studies, a type of experimental analysis used in RL research, help to address this issue by selectively removing particular components of the algorithm and analyzing the resulting impact on the agent's performance. This project focuses on conducting ablation studies for DQN algorithm in two different environments, Cartpole and Lunar Lander. By removing specific components such as the target network, buffer memory, and exploration policy, to nvestigate their individual contributions to the algorithm's overall performance. To ensure our results are generalizable and applicable across various environments, we trained the agent with the same architecture and hyper-parameters. We also ran independent training sessions with different random seeds to analyze the generalization issues in deep RL. By analyzing the results of our experiments, we hope to gain insights into the strengths and weaknesses of the DQN algorithm and inform the development of more efficient RL algorithms.

2 Background

2.1 Cart Pole Environment

Cartpole is one of the Classic Control environments used in reinforcement learning (RL). In this environment, a pole is attached to a cart by an un-actuated joint, and the cart moves along a frictionless track. The goal is to balance the pole by applying forces to the cart in the left and right direction. The rewards for each step taken in Cartpole are +1, and to simplify the learning process and improve convergence, we have capped the upper rewards to 500.

2.2 Lunar Lander Environment

This environment is a classic rocket trajectory optimization problem. The goal of this environment is to control a lunar lander and guide it to a safe landing on the moon's surface. The lander is subject to

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

gravitational and thrust forces, and the agent must learn to apply the right amount of thrust to achieve a successful landing. A successful landing yields a high reward, while crashing or landing too fast results in a negative reward. For faster convergence the upper limit for rewards is capped at 250.

2.3 Neural Network

The neural network consists of a first layer that takes input of size equal to the state size of the environment, a hidden layer with a size of 64, and an output layer with the same size as the number of actions in the environment, with ReLU activation function. For optimization, I've used Adam Optimizer with Mean Squared Error to calculate the loss between target and expected Q - values.

2.4 Soft updates of weights of Neural Network

It is used to update the weights of the target network in a gradual manner, rather than updating them completely at each iteration. This helps to stabilize the learning process by reducing the variance of the target Q - values, which in turn helps to reduce the variance of the updates to the policy network.

$$Q_{target} = \tau Q_{policy} + (1 - \tau) Q_{target}, \quad \tau << 1 \tag{1}$$

Here, Q_{target} and Q_{policy} are the weights of the target network and policy network.

3 Methodology

3.1 Exploration

To enable my agent to achieve the maximum rewards in both environments, I have implemented the Epsilon Decay function, which gradually reduces the exploration rate with each time step. This function balances the trade-off between exploration and exploitation during the training phase.

3.2 Training

The agent was trained through 10 independent runs, each with 10 independent seeds, and for a total of 3000 episodes. At each time step, the state-action-reward-next state transition was stored in the experience replay buffer. The training process involved two neural networks: the policy network and the target network. The agent was trained on the policy network using mini-batches of transitions sampled from the experience replay buffer. The weights from the policy network were copied to the target network at a predefined update frequency. The target network was used to set the target values for the Q values, while the policy network was used to obtain the estimate of the state-action values.

3.3 Hyperparameters

Deep Reinforcement learning algorithms are very sensitive to hyperparameters, To have a better convergence it is very important to chose the values wisely. In the Table 1 is shown the used hyperparameter values.

Table 1: Values of hyperparameters

Best hyperparameters	
Batch size	128
Episodes	3000
γ	0.99
τ	0.01
Learn step	5
Learning rate	0.001
Memory size	10^4
Fixed exploration	0.2

4 Experimental results

4.1 Training

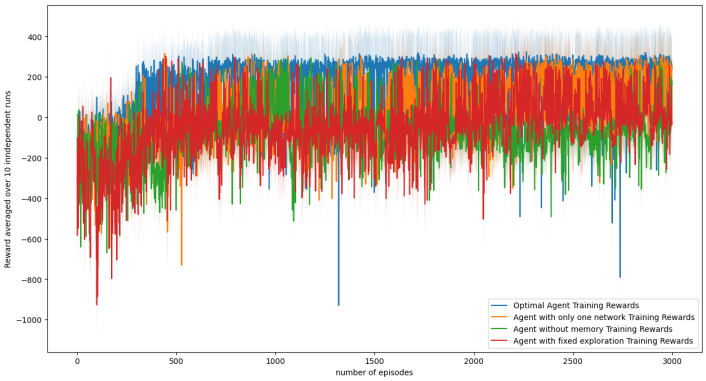

Figure 1: Lunar Lender training rewards

The training results for the Lunar Lender environment are displayed in Figure 1. Similar trends were observed in the Cart Pole environment, where the optimal agent, with both policy and target networks, converged significantly faster than other variants. The optimal agent also exhibited lower variance in rewards, in contrast to the agent without memory.

Memory plays a crucial role in the DQN algorithm, as the agent is trained on batches of transitions stored in experience replay. Interestingly, using only one network resulted in different outcomes, as the target network is required to set the target values for the Q-values of the current state. Training the agent with only one network requires more episodes to achieve the same performance as the optimal agent with both policy and target networks.

As shown in Figure 1, the policy network alone took longer to converge, but continued training ultimately led to better performance. In a fixed exploration rate was used, whereas others employed epsilon decay exploration policies. Initially, the agent requires more exploration to discover the optimal policy, but as it trains more, the exploration rate should decrease to avoid random actions and choose more optimal ones. Results show that the agent with a fixed exploration rate had higher variance than the optimal and one-network agents. Despite this, it still converged faster and more efficiently than the agent without memory.

4.2 Test

Table 2: Average test rewards for 500 episodes

Algorithm	CartPole	Lunar Lender
Optimal	500	239.31
Only policy Network	265.858	196.15
Without memory	23.078	44.73
Fixed exploration	152.218	84.53

The test results for both environments are presented in Table 2. The results clearly show that the optimal policy performs much better than the other types of policies. Even though the agent with only a policy network has a single neural network, its performance is better than that of the agent without memory and the one with fixed exploration. The results indicate that training an agent for more episodes may lead to optimal policy-level performance, thereby reducing the computational cost for calculating the loss and gradients of the neural network. The agent without memory is not able to estimate the Q-values for the state, resulting in the agent taking random actions at each time step. From Fig. 2 and 3, it can be observed that the agent with a fixed exploration rate performs worse than the agent with only one policy network. This highlights the importance of having a good exploration policy in reinforcement learning agents, as better exploration leads to better performance.

One important observation is that using the same neural network, hyperparameters, and exploration policy on different environments leads to different variances in the rewards. This suggests that having the same agent for different environments might not yield good results, as different environments require different sets of hyperparameters, neural network architecture, and exploration policies to achieve the best performance. Since the state space, reward structure, and actions are different for different environments, it is necessary to have different approaches for different environments.

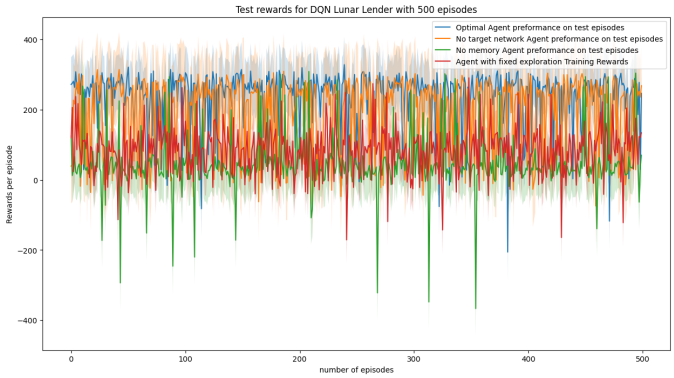

Figure 2: Lunar Lender test rewards

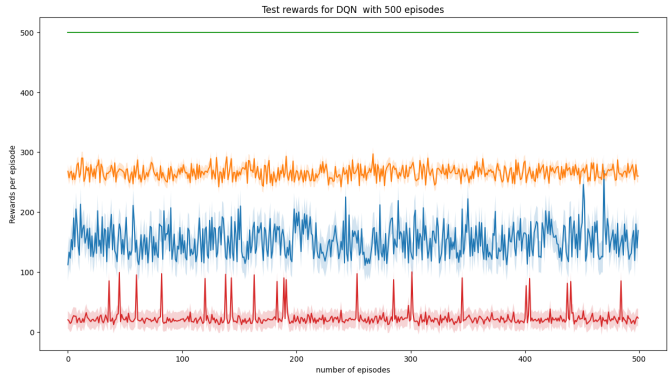

Figure 3: Cart Pole test rewards

5 Conclusion

The findings from the various Deep Q-learning algorithm variants emphasize the significance of each component for attaining optimal performance. Notably, using only a policy network can yield noteworthy outcomes, as long as there are adequate training episodes. Additionally, the exploration policy has a vital role in the agent's performance in the environment. Employing soft updates of the weight in the target network improves the stability and consistency of the results. The batch size and memory also have a substantial impact on the convergence of the Q-function, leading to faster and optimal performance. However, reproducing similar results in Deep Reinforcement Learning is challenging due to the reliance on critical parameters such as batch size, random seeds, learning rate, neural network architecture, exploration rate, learning step, and memory size. Therefore, selecting the appropriate parameter set is critical for achieving optimal performance in various environments.

6 Future work

Possible improvements can be done in many areas such as using Softmax policy for exploration, hyper parameters tuning, changing the reward structure. We can also improve the stability and performance of DQN algorithm by using Double Deep Q learning.

7 Appendix

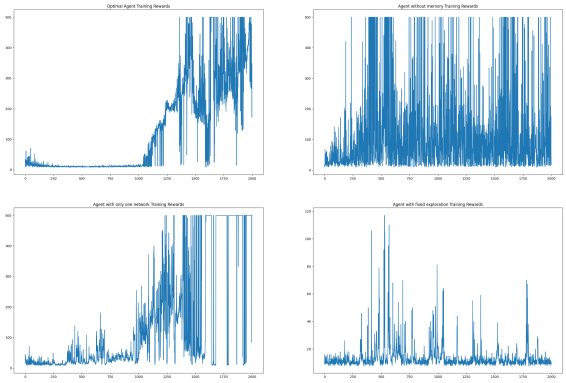

Figure 4: First 2000 episode rewards for Car Pole training for 5000 episodes

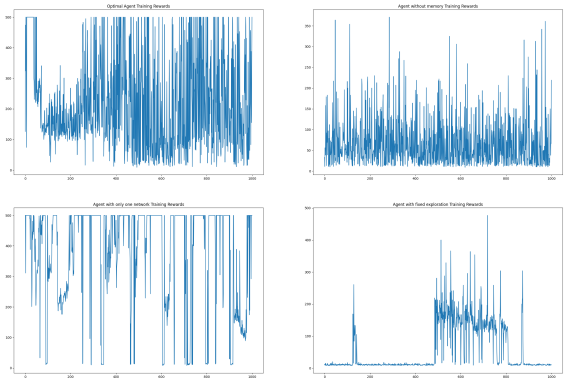

Figure 5: Overfitting during the last episodes while training Cart Pole environment for 5000 episodes

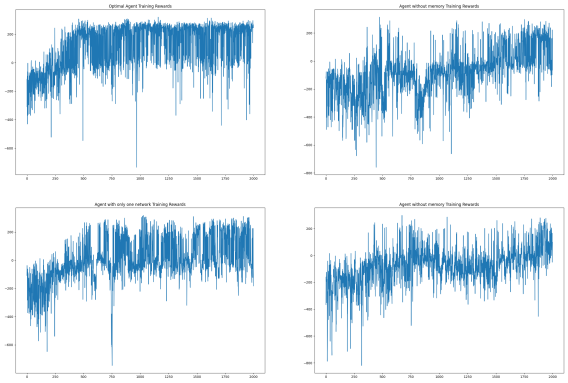

Figure 6: First 2000 episode rewards for Lunar Lender training for 5000 episodes

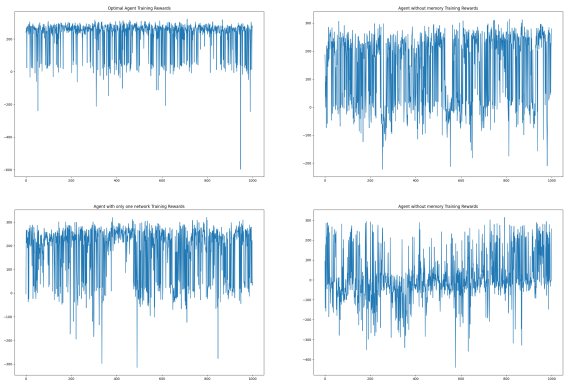

Figure 7: Overfitting during the last episodes while training Lunar Lender environment for 5000 episodes

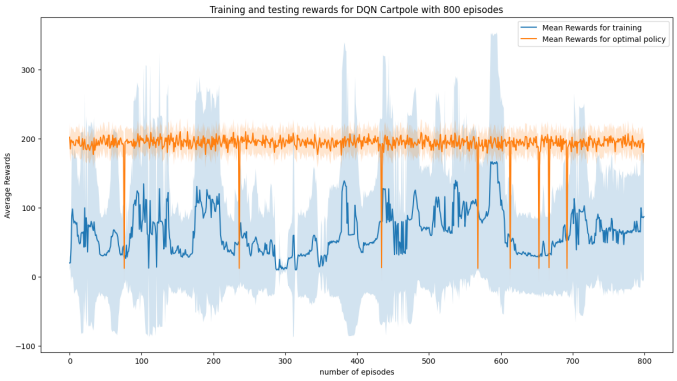

Figure 8: Without soft-updates of target network weights Cart Pole

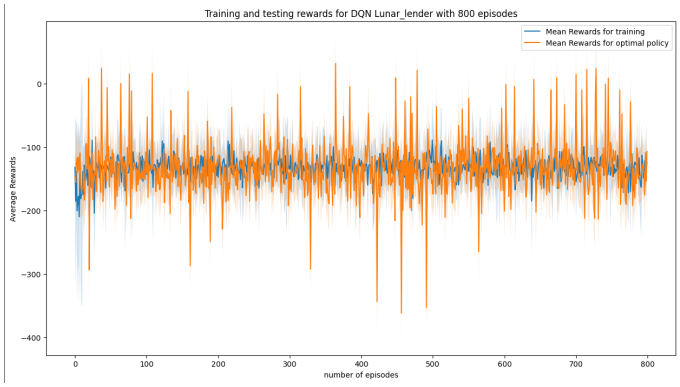

Figure 9: Without soft-updates of target network weights Lunar Lender

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[2] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D., 2018, April. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[3] Jomaa, H.S., Grabocka, J. and Schmidt-Thieme, L., 2019. Hyp-rl: Hyperparameter optimization by reinforcement learning. arXiv preprint arXiv:1906.11527.