A Comparative documentation of the accuracy of the DQN Algorithm at different timestamps for the Lunar Landing Module.

[1]Aryaman. M
[1]Student, Dept. of Artificial Intelligence and Machine Learning,DSATM, Bangalore
E-mail: aryaman.misraa@gmail.com

1.0 Introduction

The main purpose of this study is to help decide a suitable timestamp for training the DQN model for Reusable Lunar Landing Module using reinforcement learning.
The DQN model was selected since it was averaging a greater score as compared to the other readily available models to train reinforcement learning models (A2C, ACER, DDPG, HER) while training the model.

2.0 DQN

DQN, also referred to as Deep Q Network, was developed by DeepMind in 2015. DQN enhances Q-learning with deep neural networks using a technique called experience replay.

2.1. Q-Learning

Q-learning is a model-free, off-policy reinforcement learning method that finds the best of action, given the current state of the agent, depending on the position of the agent in the environment, it decides the next action to be taken.
Q-learning is based on Q-function, a Q-function is also called a state-action

function, represented as *Q(s,a),* where s is the state and a is the action.

Q-function for a policy π, represented by Q^{π} (s,a) measures the expected return or the discounted sum of rewards from state 's' by taking an action 'a' first and then following policy π thereafter. The optimal function Q*(s,a) determines the maximum return obtained starting at state 's' by taking action 'a' and following policy π thereafter.
Optimal Q-functions obey the Bellman optimality equation, which means that the maximum return from state *s* and action *a* is the sum of the immediate reward *r* and the return (discounted by γ) obtained by following the optimal policy thereafter until the end of the episode (i.e., the maximum reward from the next state *s′*).

2.2. Deep Q-Learning

It is impractical for us to represent Q-functions as a table containing each combination of 's' and 'a', therefore we train a function approximator. In this case it is a neural network with parameters θ to estimate the Q-value represented as Q(s,a;θ) ~ Q*(s,a) which is the optimal function.

To put it in simple terms, DQN models produce better scores since the model dictates the agent to take actions that will help in obtaining higher scores.

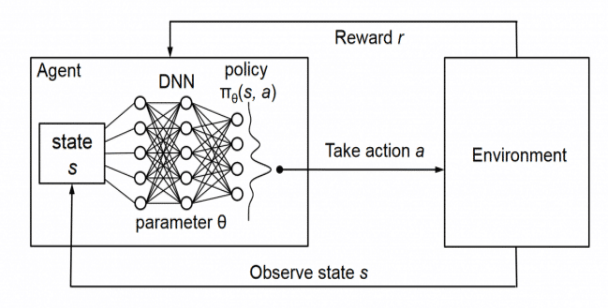


Fig. 1.0- Graphical representation of working of DQN.

3.0 Methodology

The methodology for training the model was to train it with an increment of 50,000 timestamp, since increment in values less than 50,000 did not produce a significant result in the performance of the model.

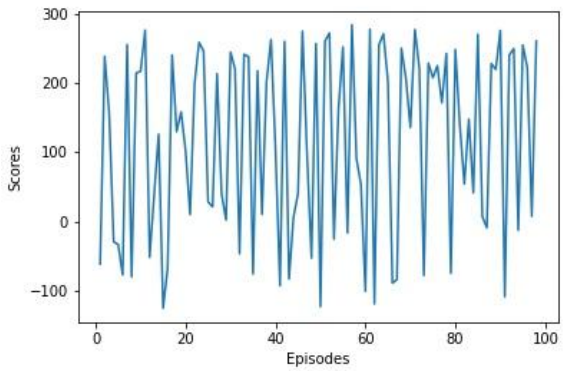
After the model was trained for 5,00,000 timestamp, the change in its scores was minimal. In an effort to see how the model behaves in a case of overtraining, the model was trained until 40,00,000 timestamp.

The policy used was LnMlp or the Multi Layer Perceptron with layer normalisation.

The model has been tested for 100 episodes for each timestamp it has been trained for and their results have been tabulated graphically.

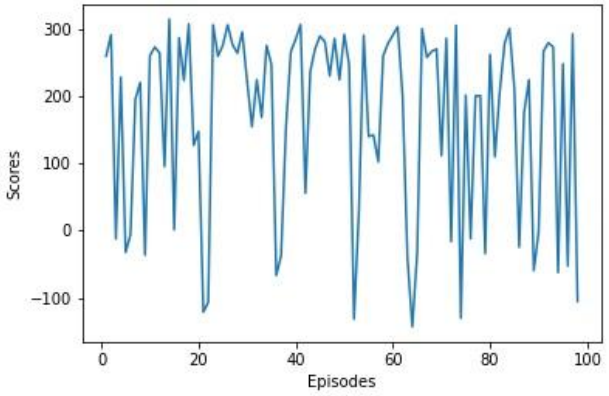
4.0 Results

4.1 DQN with timestamp - 50000:



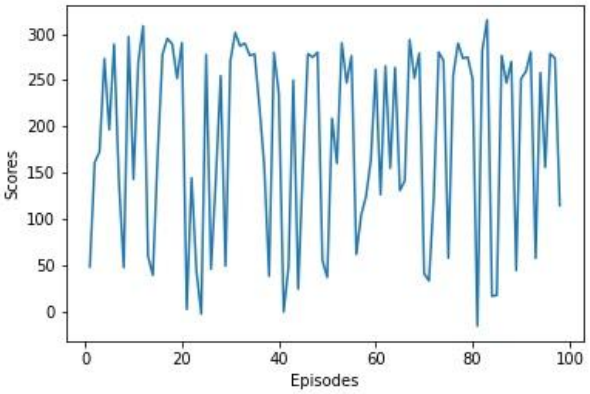
The score of the model ranged between -150 and +290. The performance of the model is extremely fluctuant and cannot be relied upon to provide reliable results.

4.2 DQN with timestamp - 100000:



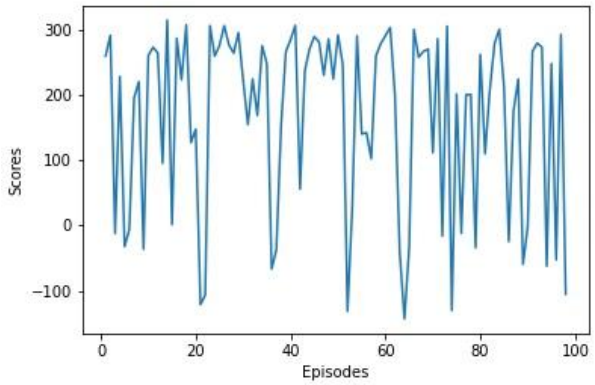
The score of the model ranged between -150 and +310. The performance of the model was extremely turbulent and unreliable.

4.3 DQN with timestamp - 150000:



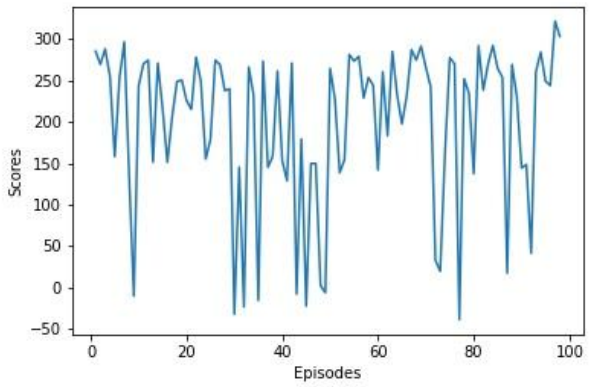
The score of the model ranged between -50 and +315. The performance is turbulent and unreliable, oscillating between high and low scores.

4.4 DQN with timestamp - 200000:



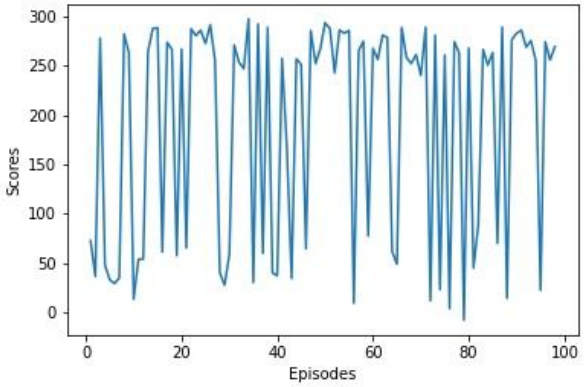
The score of the model ranged between -150 and +300. The performance is still turbulent.

4.5 DQN with timestamp - 250000:



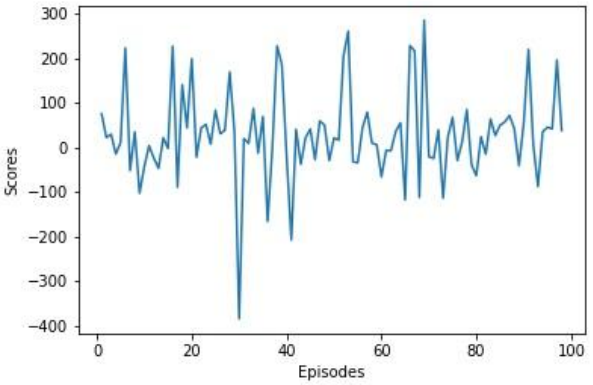
The score of the model ranged between -45 and +325. The turbulence of the model has decreased and the model is fluctuating between +290 and +150, occasionally going below +150.

4.6 DQN with timestamp - 300000:



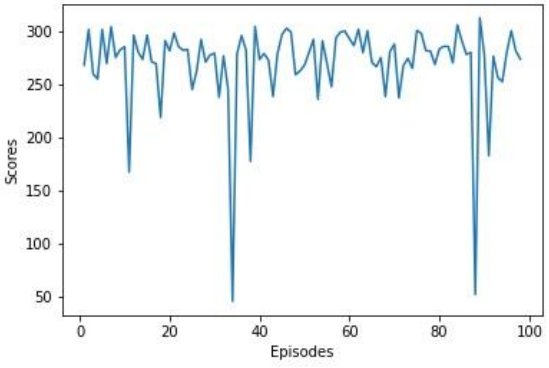
The score of the model ranged between +290 and 0. The model became turbulent again. Every high value is generally followed by a value in the lower end.

4.7 DQN with timestamp - 350000:



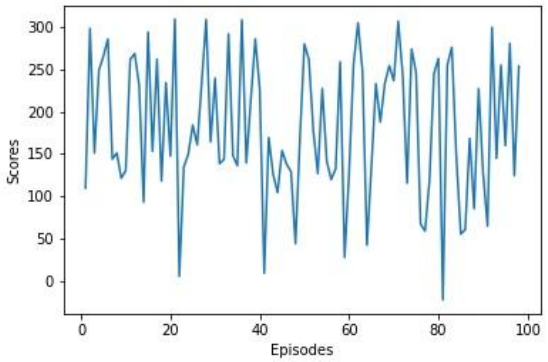
The score of the model ranged between -390 and 280. The model appears to be stabilising without having extreme drops in performance.

4.8 DQN with timestamp - 400000:



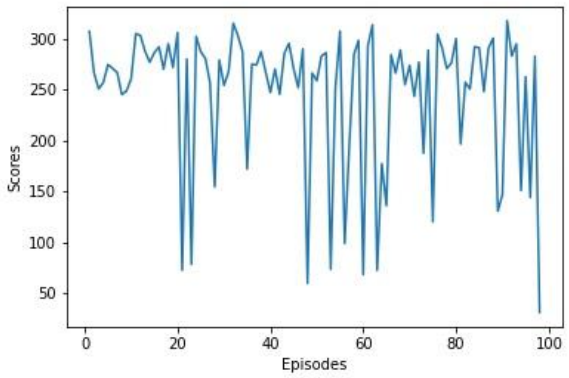
The score of the model ranged between +50 and 310. The model appears to be stable with a very reduced drop in performance as compared to the other models.

4.9 DQN with timestamp - 1000000:



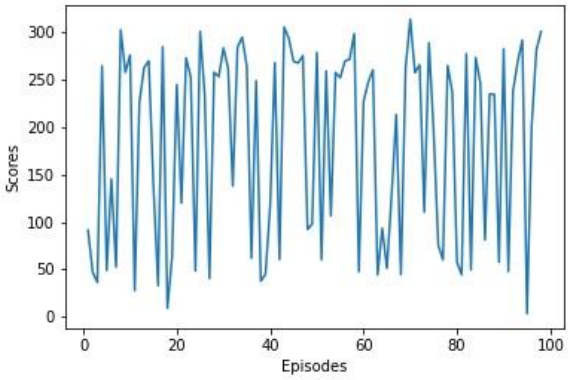
The score of the model ranged between -50 and +310. The model is turbulent but the score is mostly within +150 and occasionally goes below the level.

4.10 DQN with timestamp - 2000000:



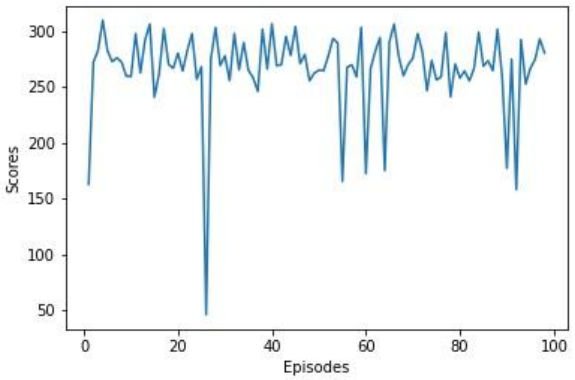
The score of the model ranged between +45 and +320. The model is, for the most part, stable and producing scores above +200. There are instances where the performance drops.

4.8 DQN with timestamp - 3000000:



The score of the model ranged between 0 and +320. The model is highly turbulent and does not produce reliable scores.

4.8 DQN with timestamp - 4000000:



The score of the model ranged between +50 and +320. The model is providing extremely reliable performance with one case of a poor result and an occasional drop off in performance, which recovers immediately.

5.0 Conclusion

This documentation aims to provide readers with a comparative between the performance of the DQN model using LnMlP for various timestamps of training.
The results above indicate that the DQN model is stable at 350000 providing

consistent results while the model trained for 400000 timestamp provides the highest score as compared to the other models with an increment of 50000.
The model with the highest score is the model trained for 4000000 timestamp, but this result is very similar to the model trained for 400000 timestamp and therefore the extra training is not recommended.

5.1 Limitations

The scores obtained for every test for a specific training timestamp of the model provides a variable result.

6.0 References

[1] Introduction to RL and Deep Q Networks (https://www.tensorflow.org/agents/tutorials/0_intro_rl)

[2] Introduction to Experience Replay for Off-Policy Deep Reinforcement Learning (https://towardsdatascience.com/a-technical-introduction-to-experience-replay-for-off-policy-deep-reinforcement-learning-9812bc920a96)

[3] Bellman Equation (https://www.geeksforgeeks.org/bellman-equation/)

[4] DQN and LnMlP Documentation (https://stable-baselines.readthedocs.io/en/master/modules/dqn.html)