# CSCI-397 Assignment 7 Report

1. Algorithm Choice: **Double DQN Learning**
2. Algorithm Research

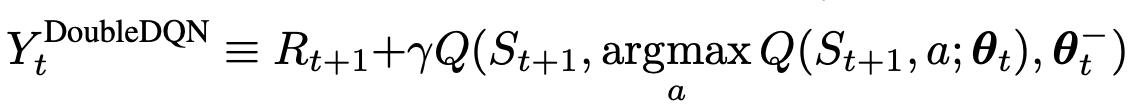
The Double DQN algorithm appears to have first surfaced in the community through a paper by Hado van Hasselt, Arthur Guez, and David Silver who are part of the Google Deepmind team. The goal of the paper is to combine the components of Deep Q Networks and Double Q learning to limit overestimation bias in the Q-learning process with deep neural networks. This paper shows overestimations are more common and severe in practice, that double Q-learning is effective at mitigating this effect at scale, and then puts forward Double DQN to further build upon Double Q-learning as it demonstrated better policy formation and results with the Atari 2600 domain.

1. Algorithm Conceptual Explanation:

(Explanation, problem it solves, type of learning, strengths and weaknesses, mathematical foundation)

Double DQN Learning is a variant of the standard DQN algorithm that builds upon the vanilla algorithm with the intention of limiting overestimation, stabilizing, and increasing reliability in the learning process. Regular DQN combines regular Q learning with deep neural networks which approximates the Q-function to which the agent uses for action selection which then receives a reward, and the network is tuned to minimize the difference between predicted and target Q-values. Double DQN aims to limit overestimation bias in regular DQN learning by decoupling the action selection from the Q-value estimation in the online and target networks. Rather than having action selection and Q-value estimation done in the target network, the action selection will occur in the online network while the target network estimates the Q-value of the action. This algorithm uses the model free temporal difference learning model that learns directly from the agent’s actions. With its decoupling of action selection and value estimation, it helps produce more stable learning processes with less overestimation bias and can be applied to a wide range of environments. However, this algorithm can still have a high degree of computational complexity which can pose some efficiency issues, especially with large environments which may require many interactions. This form of learning remains sensitive to hyper parameter adjustments as well.

Mathematical Foundation

[[1]](#endnote-1)

The equation included above is the error for Double DQN. As you might notice, it is essentially the same equation implemented in Double Q-learning which includes the value estimation of the greedy policy according to the weights of the current values, but with Double DQN learning, it performs the evaluation for the current greedy policy with the set of weights from the target network. This change to the equation is noticeably small and recognized in the paper as they state:

This version of Double DQN is perhaps the minimal possi- ble change to DQN towards Double Q-learning. The goal is to get most of the benefit of Double Q-learning, while keeping the rest of the DQN algorithm intact for a fair comparison, and with minimal computational overhead. (4)

1. Code Analysis:

The Double DQN attributes were first apparent in three specific functions which are included below. First the act( self, state): and td\_target(self, reward, next\_state, done): function shows that the online network is responsible for evaluating the epsilon greedy policy and calculating Q-values for the state, while the target network is used to estimate the values for the state based on the action selected by the online network. This where the distinction of double DQN is apparent as we can see the online network choose actions and the target network’s role in estimating their value from which a TD target is then calculated for each value in the batch.

A screen shot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

Learning Loop:

A screen shot of a computer program

Description automatically generated

The learning loop is straight forward as it first starts with checking that enough steps have been taken to accumulate enough experiences in the replay memory. From there, the agent will then sample batches of experiences from replay memory for evaluation. Each learning iteration involves calculating the td\_estimate with the online network and the td\_target. The td\_target is where the double DQN aspect of the algorithm is implemented as it involves using the online network for the best action in the next state and then the target network to estimate the related Q-value which is then adjusted by the immediate reward and discounted by gamma. The loop then calls upon the update\_Q\_online function to compute the loss between the TD estimates and the TD targets and uses back propagation to adjust the networks’ weights. Finally, in the initial if-conditions provide the structure for the target and online networks to be synced every 1e4 steps. The initialization of the agent includes learning parameters which are used in the conditions and learning and are named ‘burnin’, gamma, learn\_every, and sync\_every. ‘Burnin’ controls the amount of experiences accumulated before learning begins, while gamma is the rate at which rewards are discounted, and learn\_every and sync\_every are controls for the learning rates and for when the target and online networks are synced.

1. Van Hasselt, H., Guez, A., & Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. arXiv preprint arXiv:1509.06461. Retrieved from https://arxiv.org/pdf/1509.06461.pdf [↑](#endnote-ref-1)