# Assignment: Deep Q Learning
# Course: Reinforcement Learning,


## Abstract

With the increasing demand for examining and extracting patterns from massive amounts of data, it is critical to be able to train large models to fulfill the needs that recent advances in the machine learning are create.


## 1. Introduction

A method that can find the values of the parameters that minimize the objective function(known as optimizer). The deep Q Network is a reinforcement learning algorithm that have made the interplay between the optimization methods and machine learning one of the most important aspects of computational science. Neural networks are provide to be vital in artificial intelligence which are able to extract information and patterns from huge volume of data. of computational science. Neural networks are provide to be vital in artificial intelligence which are able to extract information and patterns from huge volume of data.
The remainder of this paper is organized as follow, in section 2 related work we discuss related work on the of artificial intelligence. Section **??** deep Q-learning explains the deep Q-learning approach and the tuned parameters. In section 5 dataset it provide information about the artificial intelligence environment. In section 6 results we provide our experimental results. Finally, in section 7 conclusion, we conclude the software outcome and provide suggestions for future implementation.

## 2. Related Work

## 3. A2c

An actor and critic network it starts the process of training with a number of actions and states. The network will produce actions that the agent will follow to obtain better reward. The actor network(target) will produce the action's. The critic network(target) receive actor's network output and with a number of state's. The critic network(target) output is the policy of the agent for each action for the next states. The policy is expressed with the bellman equation and continuous updated:

$$Q = R + \gamma * Q(s', a') \tag{1}$$

The critic network(model) expresses a policy probability of a state and action pair. The policy of the critic network(model) is compared with the policy of the critic network(target) to calculate the loss. The loss measures the policy of the current state and action critic network(model) $y$ and subtract it from the critic network(target) $\hat{y}$.

$$MSE = \frac{1}{n}\sum(y - \hat{y}) \tag{2}$$

In case of more than one critic networks(model) , the new loss is the sum of two network's values. Next this loss is backpropagated to update the weights of the network until it has reduced. The actor network(model) updates it's weights using a optimization algorithm like gradient descent that compares the action probability with the the probability that is expressed from the critic network(model), the weights of the actor network model is optimized it's output.

$$\bigtriangledown_\phi J(\phi) = N^{-1}\sum \bigtriangledown_\alpha Q_{\theta_1(s,\alpha)|_{\alpha=\pi_{\phi(s)}}} \bigtriangledown_\phi \pi_\phi(s) \tag{3}$$

At last the neural network(target) updates it's weight from the neural network(model). So during the process of training the network transfer their weights to the neural network(target).

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \tag{4}$$

$$\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i' \tag{5}$$

## 4. Evolution Strategy

Evolution strategy is part of reinforcement learning. The approach of evolution stategy use an objective function that it tests the performance of the population. The evolution strategy optimize a mathematical function. In reinforcement learning the reward is stochastic and in each episode the algorithm generate a number a population that consists of a number of offsprings. In each episode estimates the population with the obtained reward.

$$\theta(t+1) = \theta(t) + \eta \frac{1}{N\sigma} \sum_{n=1}^{N} F_n \epsilon_n \tag{6}$$

The algorithm of evolution strategy has the options of learning rate $\eta$, the standard deviation $\sigma$ and the initial policy parameter of $\theta$ that it's space is the network weights. The algorithm modify each offspring of the population($\epsilon$). Since $\epsilon_n$ is a random space, that is weighted by the reward $F_n$ value, high reward represents an update of the space to the convergence. Instead of using the reward directly $F_n$, it standardizes the reward before applying the update, this means that subtracts the mean reward and divide it by the standard deviation so that has an outcome to accept only positive rewards.

$$F_n = \frac{\sum^{R}}{N} + \sqrt{\frac{\sum(R_i - \frac{\sum^{R}}{N})}{N}} \tag{7}$$

$$\theta(t+1) = \theta(t) + \sigma * \epsilon_n \tag{8}$$

The new parameter is evaluated from the fitness function. The function optimizes the outcome, as the randomness has been added to the offspring the outcome does not converge always to the solution. The process repeated until the population size has been tested. As the evolution strategy update is an approximation of the gradient descent.

## 5. Dataset

## 6. Result

## 7. Conclusion

## References