



REINFORCEMENT LEARNING'S REPORT

Reinforcement Learning Projects

Author :

BARKOUKI Oussama

Supervisor :

MR. MOHAMED NAOUM

College Year 2020 - 2021

ACKNOWLEDGMENTS

First and foremost, we thank God for providing us with the strength and courage to continue out our task despite all hurdles and problems. Pr. Mohamed Naoum, who monitored the development of this project step by step, deserves our appreciation. We appreciate his availability, participation, and invaluable guidance in helping us stay on track and accomplish such positive outcomes.

We hope that this text accurately portrays our effort and correctly shows the various aspects that we had to address.

Sincerely, Oussama Barkouki.

ABSTRACT

This report summarizes the project carried out as part of the reinforcement learning module, which represents the application of reinforcement learning algorithms.

This project aims to set up multiple reinforcements learning algorithms – QLearning, Deep-QLearning, Reinforce, Sarsa – and use them inside of multiple environment as well as track their progress throughout the training phase.

For good project management, it was first necessary to go through the understanding of the basics of each reinforcement learning algorithm before resorting to the implementation.

TABLE DES FIGURES

| | | |
|-----|------------------------------|----|
| 1.1 | Maze Game | 11 |
| 1.2 | CartPole Game | 12 |
| 2.1 | Q Function | 16 |
| 2.2 | Q Algorithm Steps | 16 |
| 2.3 | Q Function | 17 |
| 2.4 | Expected Reward | 18 |
| 2.5 | Discounted Reward | 19 |
| 3.1 | Reward Per episode | 22 |
| 3.2 | Reward Per episode | 23 |
| 3.3 | Reward Per episode | 24 |
| 3.4 | Reward Per episode | 25 |

TABLE DES MATIÈRES

| | |
|---|----------|
| Acknowledgments | 3 |
| Abstract | 4 |
| General Introduction | 8 |
| 1 General context of the project | 9 |
| 1 Introduction | 10 |
| 2 Reinforcement Learning | 10 |
| 3 Environments | 11 |
| 3.1 Maze Environment | 11 |
| 3.1.1 Action space | 12 |
| 3.1.2 Observation space | 12 |
| 3.1.3 Reward | 12 |
| 3.1.4 End condition | 12 |
| 3.2 CartPole Environment | 12 |
| 3.2.1 Action space | 13 |
| 3.2.2 Observation space | 13 |
| 3.2.3 Reward | 13 |

| | | |
|----------|--|-----------|
| 3.2.4 | End condition | 13 |
| 2 | LITERATURE REVIEW | 14 |
| 1 | Q Learning Algorithm | 15 |
| 1.1 | Q-Table | 15 |
| 1.2 | Q-function | 15 |
| 1.3 | Q Learning steps | 16 |
| 2 | Deep Q Learning Algorithm | 16 |
| 2.1 | Deep Q Learning steps | 17 |
| 3 | SARSA | 18 |
| 4 | Reinforce Algorithm | 18 |
| 4.1 | Reinforce steps | 19 |
| 3 | REALIZATION OF THE SOLUTION | 20 |
| 1 | Runtime and environment tools | 21 |
| 1.1 | Python | 21 |
| 1.1.1 | Libraries | 21 |
| 2 | Results after learning of each algorithm | 22 |
| 2.1 | Q Learning Training Reward | 22 |
| 2.2 | Deep Q Learning Training Reward | 23 |
| 2.3 | SARSA Training Reward | 24 |
| 2.4 | Reinforce Training Reward | 25 |
| 2.5 | Final Results | 25 |

GENERAL INTRODUCTION

As we get closer to a digital world, cybersecurity is becoming increasingly important. When it comes to digital security, the most difficult task is detecting unusual activities.

When we do any kind of online transaction, a large number of individuals prefer to use credit cards. Credit card credit limits can sometimes assist us in making purchases even if we do not have the funds available at the moment. Cyber attackers, on the other hand, take use of these qualities.

To address this issue, we need a system that can stop a transaction if it detects something suspicious.

This necessitates the development of a system that can track the pattern of all transactions and, if any pattern is aberrant, terminate the transaction.

This report summarizes the various stages of our project. It is divided into three sections, as follows : The first chapter covers the host organization's presentation, as well as the project's aims and objectives, as well as the strategy taken. The project's theoretical foundation and research technique are discussed in the second chapter. The project's realization is the subject of the third chapter. We'll wrap off with a personal evaluation of the project.

CHAPITRE

1

GENERAL CONTEXT OF THE PROJECT

1 Introduction

The major purpose of this chapter is to introduce reinforcement learning and explain how it differs from other learning methods. And to set the scene for the surroundings.

2 Reinforcement Learning

Reinforcement learning is the process of training machine learning models to make a series of judgments. The agent learns how to attain a goal in an unpredictable, possibly complicated environment. In reinforcement learning, an artificial intelligence is put in a game-like setting. To solve the problem, the computer uses trial and error.

To persuade the computer to do what the programmer desires, the artificial intelligence is rewarded or punished for the acts it does. Its objective is to maximize the entire return. Although the designer establishes the reward policy—that is, the game rules—he provides no tips or suggestions to the model about how to complete the game.

It is up to the model to discover out how to accomplish the job in order to maximize the reward, beginning with completely random trials and progressing to complex tactics and superhuman abilities. Reinforcement learning is presently the most effective technique to hint machine creativity by utilizing the power of search and numerous trials. Unlike humans, artificial intelligence can learn from thousands of simultaneous gameplays if a reinforcement learning algorithm is performed on a strong enough computer infrastructure.

The major problem in reinforcement learning is creating a simulation environment that is highly dependent on the job at hand. When the model is required to go superhuman in Chess, Go, or Atari games, setting up the simulation environment is rather straightforward. When it comes to creating a model capable of driving an autonomous automobile, creating a realistic simulator is critical before allowing the car to drive on the road. The model must find out how to brake or avoid a collision in a safe environment where the cost of sacrificing even a thousand automobiles is negligible. The challenging part is getting the model out of the training environment and into the actual world.

There should be no discernible distinction between machine learning, deep learning, and reinforcement learning. It is analogous to a parallelogram – rectangle – square relationship, with machine learning being the largest category and deep reinforcement learning being the narrowest. Similarly, reinforcement learning is a specialized application of machine and deep learning techniques that is aimed to address issues in a specific manner.

3 Environments

In reinforcement learning, the environment is the world in which the agent lives and interacts. The agent can interact with the environment by executing some activities, but such actions cannot impact the environment's laws or dynamics. This indicates that if humans are the agents in the earth's ecosystems, we are bound by the planet's physical rules. We may influence the environment by our activities, but we cannot change the physics of our world.

3.1 Maze Environment

A basic 2D labyrinth environment in which an agent must navigate from the top left corner to the bottom right corner in order to reach the goal. The purpose is to determine the shortest path from the beginning to the end.

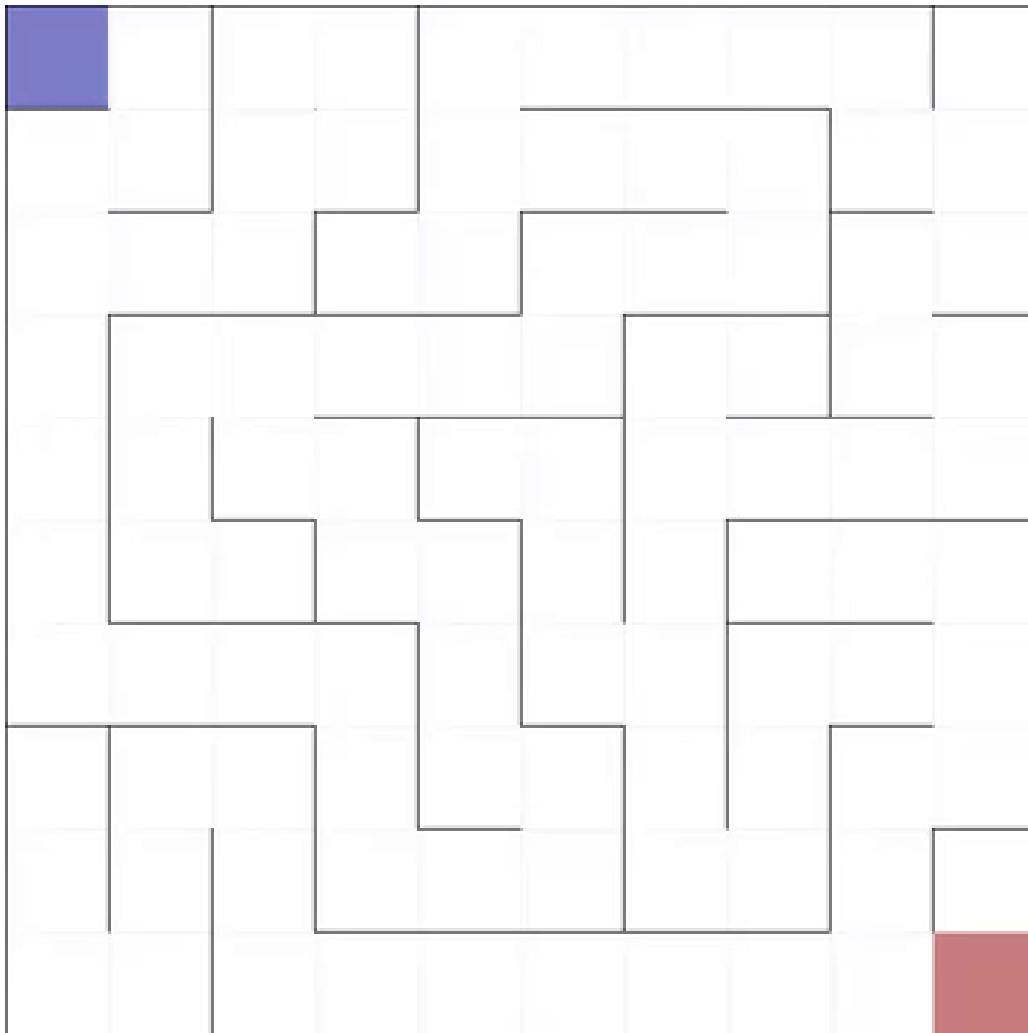


FIGURE 1.1 – Maze Game

3.1.1 Action space

The agent may only choose to go up, down, right, or left "N", "S", "E", "W". If the way is blocked, it will remain at the same the location.

3.1.2 Observation space

The agent's (x, y) coordinate is the observation space. The cell on the upper left is (0, 0).

3.1.3 Reward

When the agent achieves the goal, he receives a 1 reward. The agent earns a -0.1/ reward for each step through the maze (number of cells).

3.1.4 End condition

When the agent achieves the target, the maze is reset.

3.2 CartPole Environment

An un-actuated joint connects a pole to a cart that runs along a frictionless track. The pendulum begins upright, and the aim is to keep it from falling over by raising and decreasing the velocity of the cart.

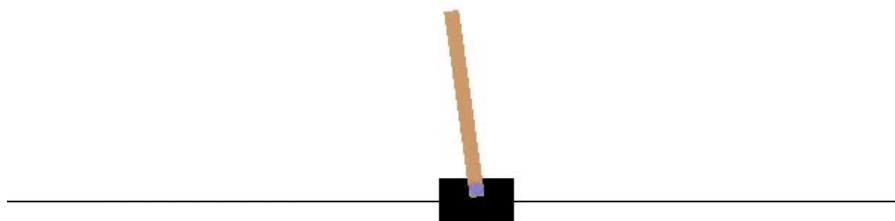


FIGURE 1.2 – CartPole Game

3.2.1 Action space

The agent has just two options : push the cart to the left or push the cart to the right.

3.2.2 Observation space

Cart Position, Cart Velocity, Pole Angle, and Pole Angular Velocity are the four continuous quantities in the agent's observation space.

The ranges of each value are listed in the table below.

| Observation | Min | Max |
|-----------------------|----------------------|--------------------|
| Cart Position | -2.4 | 2.4 |
| Cart Velocity | -Inf | Inf |
| Pole Angle | -0.209 rad (-12 deg) | 0.209 rad (12 deg) |
| Pole Angular Velocity | -Inf | Inf |

3.2.3 Reward

Reward is 1 for every step taken, including the termination step

3.2.4 End condition

There are four requirements that must be met for an episode to be declared completed.

- Pole Angle is more than 12 degrees.
- Cart Position is more than 2.4 (center of the cart reaches the edge of the display).
- Episode length is greater than 200.

CHAPITRE

2

LITERATURE REVIEW

Introduction

This chapter will provide a survey of the literature, allowing us to describe and define all of the theoretical concepts of the algorithms that will be covered in this project. We'll also go through the development tools that we used in this chapter.

1 Q Learning Algorithm

Q-learning is a reinforcement learning method that does not require a model. Q-learning is a learning algorithm that is based on values. Value-based algorithms use an equation to update the value function (particularly Bellman equation). The other kind, policy-based estimation, guesses the value function using a greedy policy generated from the most recent policy improvement.

Q-learning is an off-policy learner. That is, it learns the value of the optimum policy independently of the agent's behavior. An on-policy learner, on the other hand, learns the value of the policy being carried out by the agent, including the exploration phases, and it will discover an optimum policy that takes into consideration the exploration inherent in the policy.

1.1 Q-Table

Q Learning is based on a q table, which is a data structure used to determine the maximum predicted future rewards for each stage of action. Essentially, this chart will direct us to the optimum course of action in each state. The Q-Learning algorithm is used to learn each value of the Q-table.

1.2 Q-function

The Bellman equation is used by the Q-function, which has two inputs : state (s) and action (a).

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$$

Diagram illustrating the Q Function update equation with labels for each term:

- New $Q(s, a)$** : New Q value for the state and action (red box)
- $Q(s, a)$** : Current Q values (purple box)
- α** : Learning Rate (black box)
- $R(s, a)$** : Reward for taking an action in a state (orange box)
- γ** : Discount Rate (black box)
- $\max_{a'} Q'(s', a')$** : Maximum expected future reward (green box)
- $Q(s, a)$** : Current Q values (purple box)

FIGURE 2.1 – Q Function

1.3 Q Learning steps

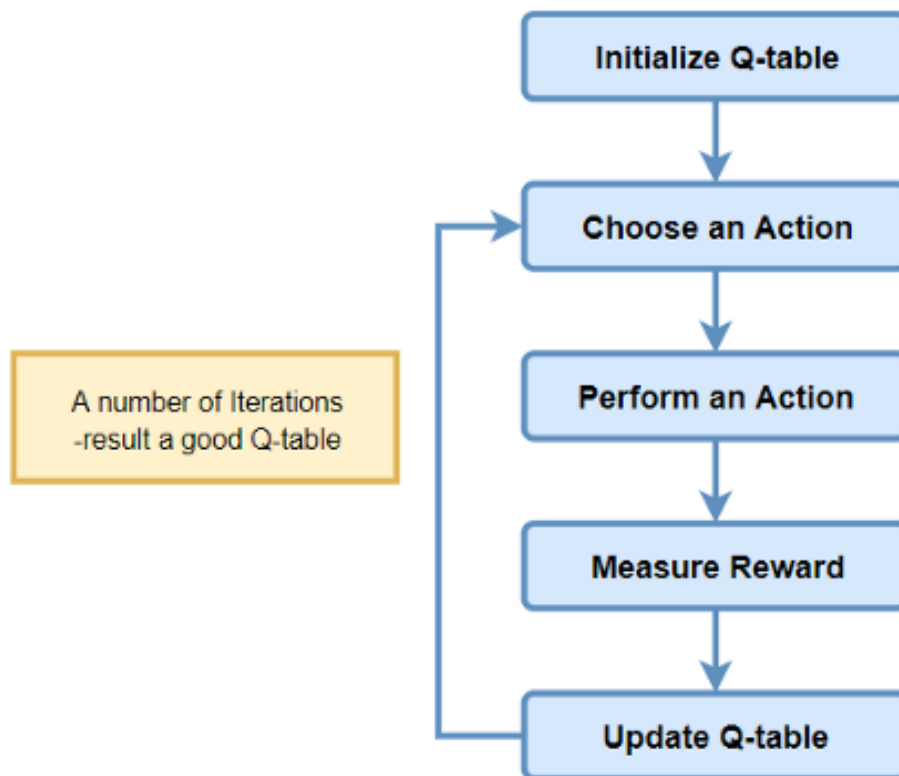


FIGURE 2.2 – Q Algorithm Steps

2 Deep Q Learning Algorithm

Deep Q Learning is quite similar to Q learning. The implementation of the Q-table is the primary distinction between Deep Q-Learning and Vanilla Q-Learning. Deep Q-Learning, crucially, substi-

tutes the traditional Q-table with a neural network. A neural network maps input states to (action, Q-value) pairings rather than mapping a state-action pair to a q-value.

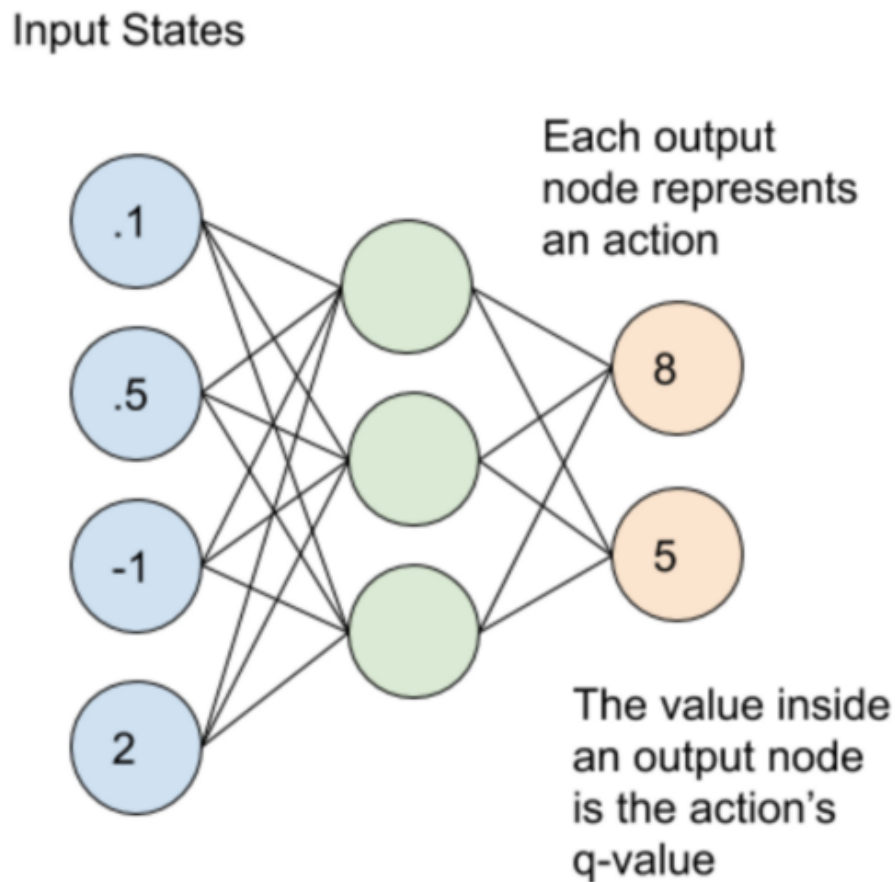


FIGURE 2.3 – Q Function

2.1 Deep Q Learning steps

- First, provide the environment's state to the agent.
- The agent uses Q-values of all possible actions for the provided state.
- Agent picks and performs an action based on Q-Value of action for gathering higher rewards.
- Observe reward and next steps.
- Stores previous experience in experience replay memory.
- Training of the networks using experience replay memory.
- Repeat steps 2-6 for each state.

3 SARSA

SARSA is an on-policy approach that, in terms of stages, is very similar to Q Learning, with the main variation being how the Q values are updated. SARSA employs the following formula :

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha (r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (2.1)$$

The update equation for SARSA in this case is determined by the present state, current action, reward earned, future state, and next action. This finding led to the learning technique's name, SARSA, which stands for State Action Reward State Action and represents the tuple (s, a, r, s', a').

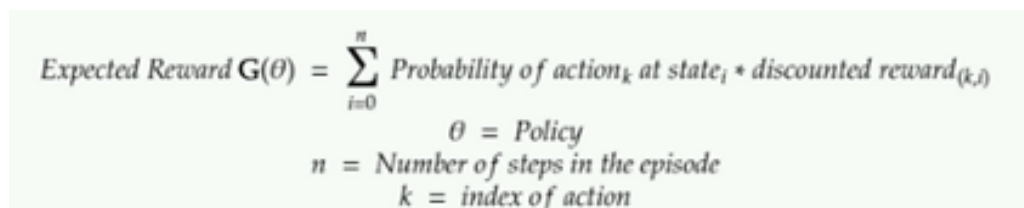
4 Reinforce Algorithm

Reinforce is a Policy Gradient algorithm, which is a subclass of Reinforcement Learning algorithms. A basic implementation of this method would be to create a Policy : a model that takes a state as input and outputs the likelihood of executing an action. A policy is simply a cheat-sheet or guide for the agent that tells it what action to perform at each stage. The policy is then iterated on and slightly changed at each stage until we get a policy that addresses the problem.

The policy is typically a Neural Network that takes the state as input and outputs a probability distribution throughout action space.

The policy's goal is to maximize the "Expected reward." Each policy increases the likelihood of an action being taken at each station of the environment.

The agent draws a sample from these probabilities and chooses an action to take in the environment. We know the total rewards the agent can receive if it follows that policy at the end of each episode. We backpropagate the reward by following the agent's journey to estimate the "Expected reward" in each stage for a particular policy.



$$\text{Expected Reward } G(\theta) = \sum_{i=0}^n \text{Probability of action}_k \text{ at state}_i * \text{discounted reward}_{(k,i)}$$

$\theta = \text{Policy}$
 $n = \text{Number of steps in the episode}$
 $k = \text{index of action}$

FIGURE 2.4 – Expected Reward

In this case, the discounted reward is the total of all the benefits received by the agent in the future, multiplied by a factor Gamma.

$$\text{Discounted Reward at } t, R(t) = r(t) + \gamma * R(t+1) = r(t) + \gamma * r(t+1) + \gamma^2 * r(t+2) + \dots + \gamma^{(T-t)} * r(T)$$

γ = Discount Factor, usually 0.9
 T = Terminal state's time step

FIGURE 2.5 – Discounted Reward

4.1 Reinforce steps

- Initialize a Random Policy (a NN that takes the state as input and returns the probability of actions)
- Use the policy to play N steps of the game — record action probabilities-from policy, reward-from environment, action — sampled by agent
- Calculate the discounted reward for each step by backpropagation
- Calculate expected reward G
- Adjust weights of Policy (back-propagate error in NN) to increase G
- Repeat from 2

CHAPITRE

3

REALIZATION OF THE SOLUTION

Introduction

This chapter will present the realization part of the solution presented previously.

1 Runtime and environment tools

1.1 Python

Python is the programming language utilized in our project for both the Machine Learning component and the API Integration. Python is a multi-paradigm, multi-platform interpreted programming language. It supports imperative structured, functional, and object-oriented programming. It offers powerful dynamic typing, as well as automated trash collection and management.

1.1.1 Libraries

Python's community provides a number of libraries that span a wide range of topics. We will utilize the following libraries for our project :

- **Open AI gym :**

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

2 Results after learning of each algorithm

2.1 Q Learning Training Reward

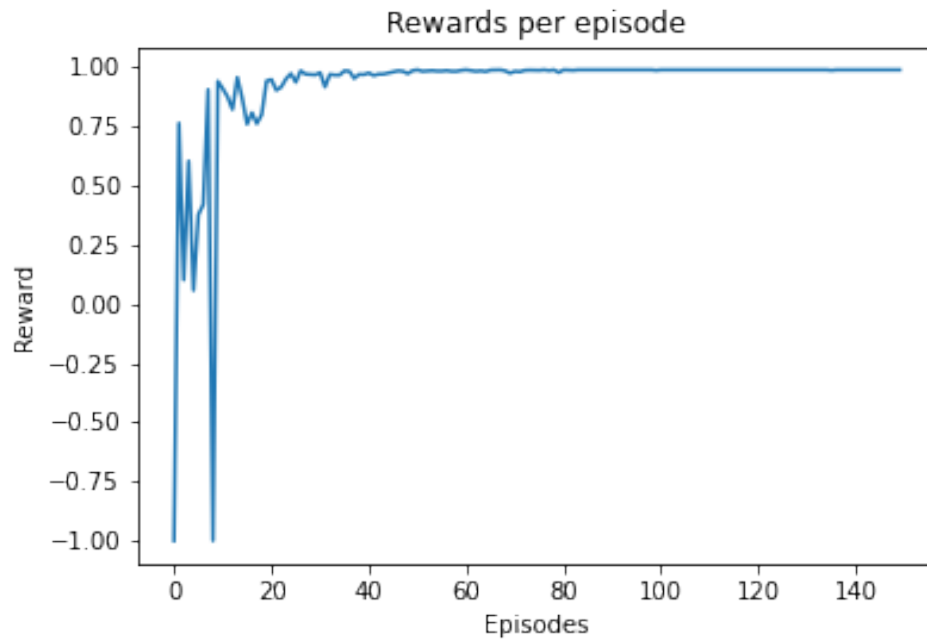


FIGURE 3.1 – Reward Per episode

2.2 Deep Q Learning Training Reward

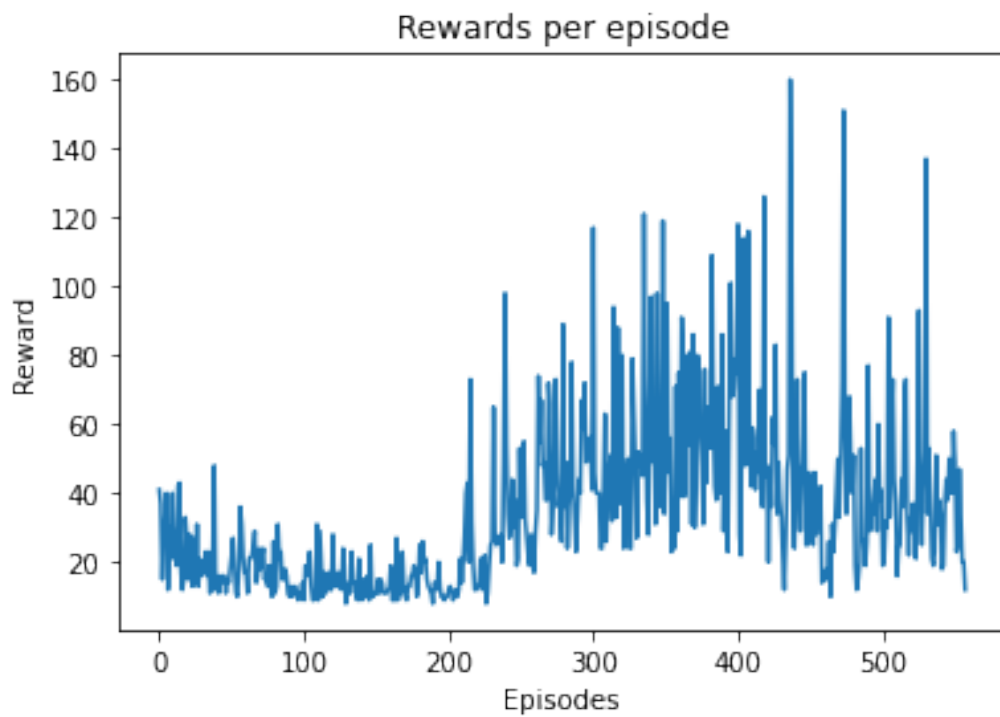


FIGURE 3.2 – Reward Per episode

2.3 SARSA Training Reward

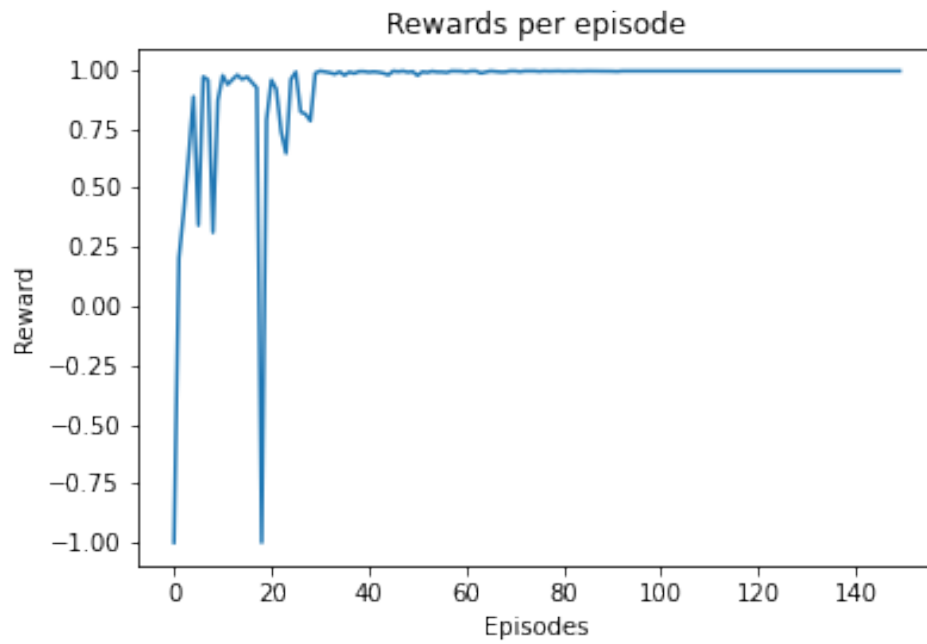


FIGURE 3.3 – Reward Per episode

2.4 Reinforce Training Reward

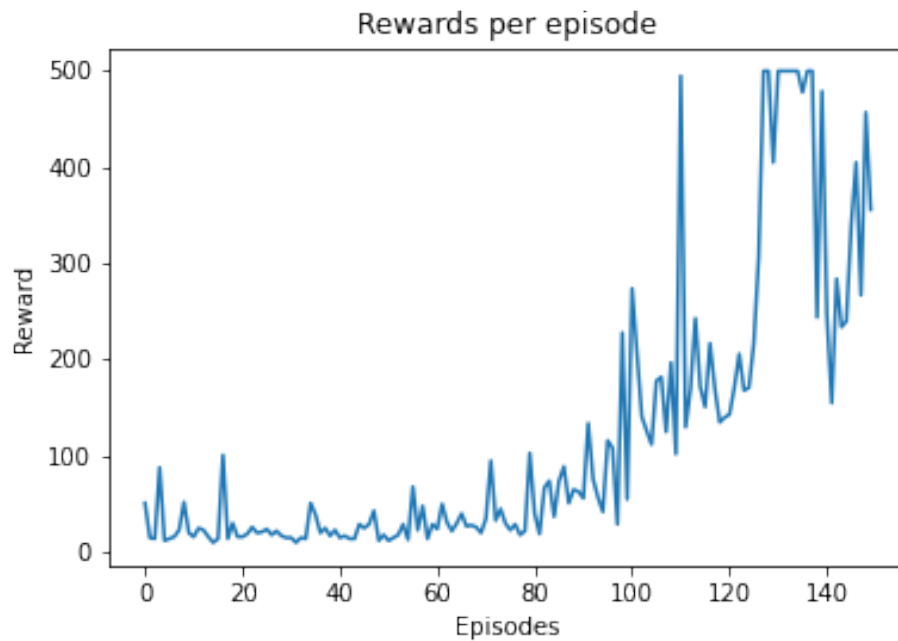


FIGURE 3.4 – Reward Per episode

2.5 Final Results

Most of the algorithms utilized were effective in learning a policy that will result in the largest reward; however, a couple of them did take a long time to complete. Overall, the project provided an opportunity to put theory into reality.

BIBLIOGRAPHIE

- [1] <https://www.eecs.tufts.edu/~mguama01/post/q-learning/>
- [2] <https://gym.openai.com/>
- [3] <https://github.com/MattChanTK/gym-maze>
- [4] <https://www.analyticsvidhya.com/blog/2020/11/reinforce-algorithm-taking-baby-steps-in-reinforcement-learning>
- [5] [https://keras.io/examples/rl/deep_qnetwork_{breakout}/](https://keras.io/examples/rl/deep_q_network_breakout/)