AUTONOMOUS VIRTUAL VEHICLE USING REINFORCEMENT LEARNING

BY

SADMAN SAUMIK ISLAM
ID: 161-15-7221
AND

SAMIA BINTA ALAM
ID: 161-15-7457

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Subroto Nag Pinku
Lecturer
Department of CSE
Daffodil International University

Co-Supervised By

Md. Ferdouse Ahmed Foysal
Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

DECEMBER 2019

APPROVAL

This Project titled **"Autonomous Virtual Vehicle Using Reinforcement Learning"**, submitted by Sadman Saumik Islam, ID No: 161-15-7221 and Samia Binta Alam, ID No: 161-15-7457 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 5th December, 2019.

BOARD OF EXAMINERS

Dr. Syed Akhter Hossain — **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Saiful Islam — **Internal Examiner**
Senior Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Shaon Bhatta Shuvo — **Internal Examiner**
Senior Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Dewan Md. Farid — **External Examiner**
Associate Professor
Department of Computer Science and Engineering
United International University

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Subroto Nag Pinku, Lecturer, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

28.11.19

Subroto Nag Pinku
Lecturer
Department of CSE
Daffodil International University

Co-Supervised by:

28.11.19

Md. Ferdouse Ahmed Foysal
Lecturer
Department of CSE
Daffodil International University

Submitted by:

28.11.19

Sadman Saumik Islam
ID: -161-15-7221
Department of CSE
Daffodil International University

28.11.19

Samia Binta Alam
ID: -161-15-7457
Department of CSE
Daffodil International University

ii

©Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Subroto Nag Pinku**, **Lecturer**, Department of CSE, Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of Machine Learning to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to Subroto Nag Pinku, Md. Ferdous Ahmed Foysal, and Dr. Syed Akhter Hossain**,** Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

Artificial intelligence plays a vital role in self driving vehicle. The direction in which the automotive industry is headed it is expected that autonomous vehicles capable of driving without human supervision will be released to market in the next decade. The development of such intelligence is fairly in its early stages. The problem of creating such intelligence is that the real-world environment is ever-changing. To solve this problem the autonomous vehicle needs to have such intelligence that it can cope with the changing real-world environment. We though can achieve such intelligence by simulating an autonomous self-driving agent in a virtual 2D platform where it will be able to follow the track on its own. Using supervised learning to solve this problem will not be an efficient approach to this problem because no matter how much training and testing is done, it will not be able to keep up with the dynamic real-world environments. Therefore, we are proposing a novel approach for creating autonomous AI powered virtual vehicle using Reinforcement Learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The field of autonomous vehicles have seen a rapid development in the recent years due to the evolution of sensors, wireless communications and most importantly artificial intelligence. The main objective imposed on an autonomous vehicle is to drive long distances with safety and security. The development of autonomous vehicles can be traced back to the early 1920. Recently, the development of autonomous vehicles has become the main focus of the automotive industries. Organizations like Tesla have made tremendous progress with their autonomous car.

This research is focuses on teaching a 2D vehicle to drive by itself in a virtual environment by using deep reinforcement learning [1]. Deep Q network (DQN) [2] was used for training the virtual vehicle agent to learn to drive on the track and the follow the track accordingly.

With insufficient hardware power and in a short time the virtual car agent was able learn to drive on the track without going off track and complete a whole lap of the track in an appropriate gaming environment that represents real world driving tracks.

1.2 Motivation

Human Error causes 94% of the road accidents which results in countless loss of valuable life every day. This research work is devoted to create such intelligence that can make a car be able to drive itself which will result in much less road accidents if not none at all.

This sort of intelligence is only possible to acquire by using machine learning. A flawless artificial intelligence for self-driving car is not yet possible but using deep reinforcement learning we can get as close as to a perfect intelligence for autonomous cars.

The intelligence developed in this work if applied to a real-world car and with sufficient training should be able to make the car able to drive itself on the road without any human interference.

1.3 Rationale of the Study

Deep reinforcement learning uses the concept of deep learning [3] combined with the principles of reinforcement learning algorithms which can be applied to video games and robotics that gives the optimal solution to the problem at hand. Deep reinforcement learning is a model free approach which means it can be applied on any environment with some minor environmental changes to the model. This is possible because deep reinforcement learning generally uses the raw sensors or image signals as inputs which leverages not only the benefits of reinforcement learning algorithms but also convolution neural networks. To teach the virtual car to drive autonomously we have used the renowned Q learning [2] algorithm from reinforcement learning combined with a simple convolution neural network to create a Deep Q Network (DQN). Q learning algorithm uses a reward and punishment system to teach the agent to solve the game. The Q algorithm dose not only take decisions based on current rewards but also takes in consideration the possible future rewards which can be achieved if an action is taken.

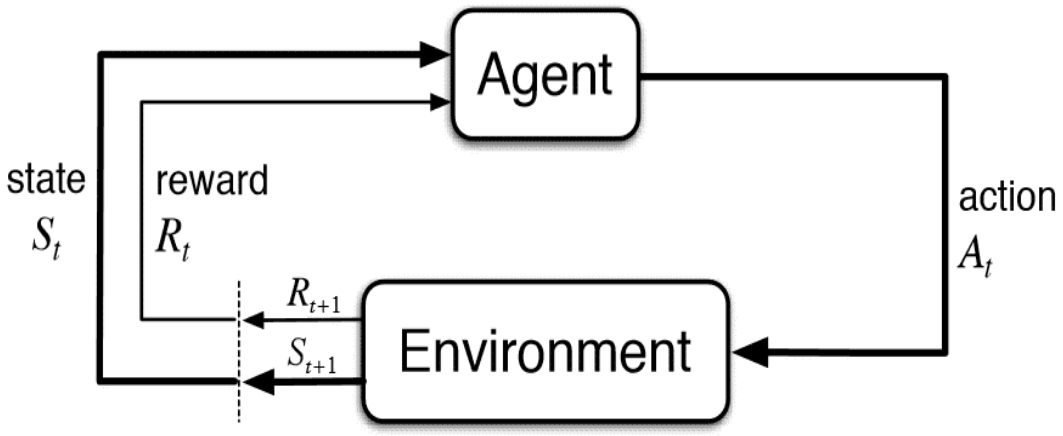


Figure 1.3.1: Q-learning Algorithm.

The DQN was applied on the "CarRacing-v0" environment from OpenAI Gym [4]. The game is completed when the car is able to go around a lap of the track and gets the

highest reward. The positive reward is given when the car visits a new tile of the track and it is given a negative reward or punishment if it goes off the track.

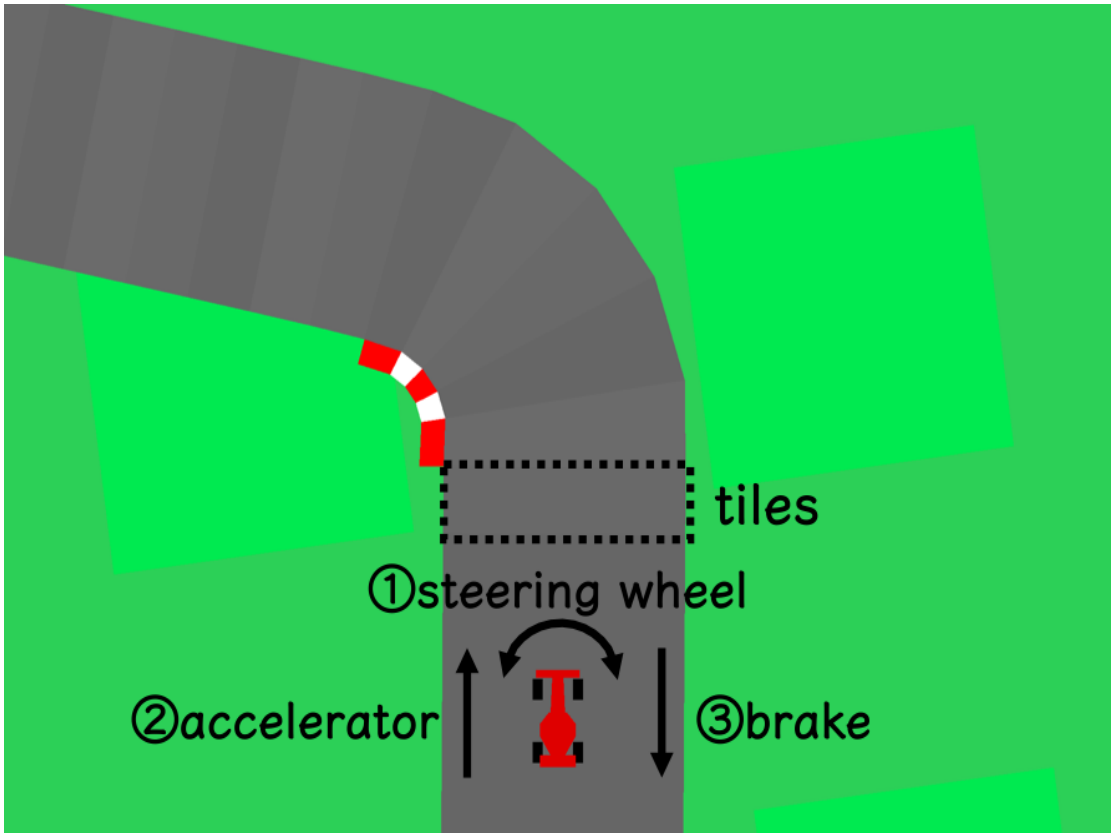


Figure 1.3.2: "CarRacing-v0" environment from OpenAI Gym.

"CarRacing-v0" features true speed, four ABS sensors, steering wheel position and also gyroscope. Anti-lock Braking System (ABS) prevents the wheels from locking up during the braking which stops the car from skidding. Gyroscope helps maintaining the orientation of the car agent in the environment. These features make the game behave almost as same as a real-world car. Therefore, this environment is perfect for applying the DQN to work on and the results generated from it is as close as it gets if it was applied on an actual car.

Another main reason to apply the DQN in this game environment is that the Q learning algorithm learns by the process of trial and error. The car goes off the track many times to learn that it is the wrong thing to do. It is not efficient to apply the DQN directly to real car as it does not give us the luxury of trial and error learning. The game gives us the perfect environment where the virtual car can learn from its errors and previously

acquired experiences. Also, the frames from the game screen are optimal input for the convolution neural network.

1.4 Research Questions

- How can we decrease the road accidents?
- What approach should we take to create artificial intelligence for car?
- What type of environment should be used so that it could give results that can be correlated with real-life situations?
- What is the best reinforcement algorithm to train a virtual self-driving vehicle?
- How efficient is the DQN for tackling the situations that the virtual car agent will face in the environment?
- How much of training dose the agent needs?

1.5 Expected Output

This work is dedicated to acquire such intelligence that a virtual 2D car can drive itself by using deep reinforcement learning. The deep q network makes the virtual car agent learn through the trial and error process. DQN searches for the optimal policy for any given possible state in the environment for the agent to take. As the main objective for Q learning algorithm is to get maximum reward by choosing the optimal policy, the virtual car agent will try to get the maximum reward by following the track without going off-track. The game is completed when the agent can drive around a full lap of the track in the game. With sufficient training the agent will be able to follow the track by itself without needing any interference from a human user. After the training the agent should be able acquire a policy which has the perfect balance between exploration and exploitation. After accruing the experience from the virtual car agent, this then can be applied to the remote controlled (RC) cars to train it further in the real-world environments to teach the RC cars to drive by themselves without needing any assistance from humans. The RC cars will represent the virtual car agent from the gaming environment. Finally, after sufficient training of the RC cars that intelligence can be applied to the real cars to achieve autonomous self-driving cars.

1.6 Report Layout

In this paper, we have tried to split up this entire paper into five chapters for the particular implementation and extended explanation to make the concept more convenient in term of understanding. The first chapter has been ornamented with the basic understanding and rationale of the study to portrait the expected outcome and the motivation behind the research to understand easily. After the first chapter in the second chapter has been made with the background study for this particular research with relevant related works regarding this matter and the highlights for the summary of our research also the challenges and the different types of errors faced during the research and training process of the virtual agent. The third chapter is based on the research methodology taken for out specific work. We have tried to explain elaborately the stages of data collection and processing of the data for training purpose. Then the statistical analogy for the research and also the reequipments for the work. For the fourth chapter, we have broadly explained the final implementation of the research correlating with the menologies explained in the previous chapter to finish the task at hand successfully. Finally, a short but descriptive summary of our research work and all the studies regarding the autonomous virtual cars and the implication for future studies has been described with accurate and rightful explanation to exploit the findings of the research work and technique for further usability in the sector of artificial intelligence for autonomous cars.

CHAPTER 2

BACKGROUND

2.1 Introduction

Road safety is has become a major concern specially after the 21st century due to the fact that a massive expansion in automobile industry and ever-increasing cars on the road. The main reason behind the road accident is human errors. Which results in countless valuable lives lost and many more left effected due the circumstances. The concept of using artificial intelligence for vehicles is not something new. The aircraft sector has long adapted to the autopilot system which uses artificial intelligence that has resulted in significant amount of safety. Up to this date the safest way to travel is by air which is the credit of being able to use artificial intelligence commercially in aircrafts. There has been already a lot research on autonomous cars and countless more happening right this moment but none has been able to provide a perfect model that could be used commercially.

2.2 Related Works

There have many researches in the field of self-driving autonomous cars. Kim et al. [5] proposed a system where deep learning was used to take control of the steering wheel. They approached the problem in two stages. In the first stage they trained a convolution network with images of steering wheel with correlating expected output for the angel of the steering wheel. The second stage was testing the model on three dataset for 16 hours of driving with tremendous success.

Another very good approach was proposed to achieve the autonomous AI for car by Bojarski et al. [6] where they used raw images acquired form the front camera of a car and used it to train a convolution neural network to take control of the steering wheel. This end to end deep learning approach proved to be powerful solution.

Both of the approaches targeted supervised learning for acquiring a model for the car but no matter how much training is done by this process it will be able to keep up with

the constantly changing and sophisticated world environment and will be faced with situations where the model fails to provide any decision.

For getting better model that is able to explore an environment without having any prior knowledge about the it should be considered. Reinforcement learning algorithm are best suited for exploring unknown environments and gather knowledge to solve them effectively. A very good example for this type of work was done by Mnih et al. [7] where they used deep reinforcement learning to play famous Atari games like super Mario. The algorithm was able to find the optimal solution to play the game and get highest reward.

2.3 Research Summary

This work features reinforcement algorithm combined with a convolution neural network to solve the CarRacing-v0 environment from OpenAI Gym. We have used the Q learning algorithm for our particular problem. This algorithm works on the basis of reward and punishment where the agent is rewarded for right steps taken and given punishment for wrong steps taken. The environment of the CarRacing-v0 game is so much sophisticated that normal Q learning algorithm is not able to keep up with. For that reason, we combined for Q learning algorithm with a simple convolution neural network to create a Deep Q Network (DQN). The DQN takes the raw pixel data from the frames of the game and outputs the possible q values for a state action pair. Finally, the after 7 hours of training the DQN was able to provide a model that made the virtual car agent able to drive on the track for a full lap.

2.4 Scope of the Problem

In the previous works the main focus was on controlling the just the steering wheel of the work. Also, the model they acquired was by using supervised learning. The CarRacing-v0 environment provides us with and environment that has real world features. Moreover, the virtual car agent not only has steering controls but also it features gas and brake which enables the virtual car to accelerate and decelerate. This gave the car 5 possible actions (turn left, turn right, accelerate, brake, do nothing). By

applying our DQN we were to receive the q values for any possible state taking these five actions. Exploring the environment, the virtual car takes actions that offer the best q values to receive maximum rewards and also considering future rewards.

2.5 Challenges

This research deals with real-time screen images from the game's environment. The hardware needed to cope with this computational work is very costly. Our laptop was able to keep up with the required hardware but barely. The training process needed a significant amount of time and any error resulting in restarting the training and spent a long time for its completion. Another main challenge was to learn about the deep learning and reinforcement learning. As we had no prior knowledge before this project, it was very difficult for us to learn everything from scratch. That turned out be very troublesome as we had to spend a lot of time studying rather than implementing the research. That is why it proved to be extremely difficult for us to make any alteration in our implementation.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

Reinforcement Learning (RL) [1] is subsector of machine learning that is focused of how a virtual agent seeks out to take necessary actions in an environment that will result in some sort of maximum cumulative reward. Along with supervised learning and unsupervised learning, reinforcement learning is one the three basic machine learning approaches. The reinforcement algorithm that is used to implement our research work is Q learning.

Q learning

Q-learning [2] is a model-free reinforcement learning algorithm. The main goal for the Q learning algorithm is to learn the optimal policy that tells the virtual agent to take what actions under any particular circumstances. It can handle problems without needing adaption by using stochastic transitions and rewards. Q learning algorithm finds the best policy that is the optimal one keeping in mind to maximize the value of rewards for any and all successive states for any finite Markov decision process.

Markov decision process

Markov decision process (MDP) deals with state and action pair (S, A) sets. S denotes the all possible state in and environment and A denotes all the possible actions for a particular state. The process in each step is at some state S with possible actions A to reach the next state denoted by S' depending on some reward denoted by R. The probability that the process moves to the next state is defined in the function P(S, S') where action A at state S in t time will lead to action S' in t+1 time.

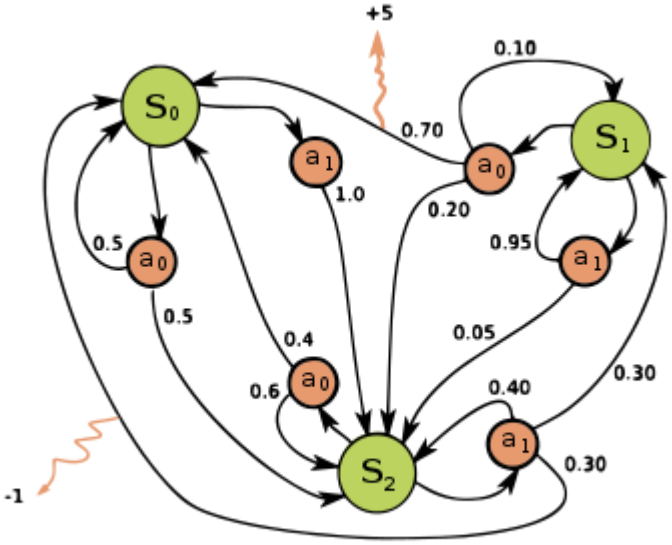


Figure 3.1.1: Example of MDP.

Bellman equation for Q learning

Bellman equation states that given a particular state that an agent is by taking the best possible action what type of future rewards can the agent expect. The Bellman equation is a recursive process to get expected outcome. For example, the expected reward for being in a particular state s and following some fixed policy has the Bellman equation, this equation describes the expected reward for taking the action prescribed by some policy. Using the equation, the q values of each explored state and action pair is updated. The Bellman equation for calculating q values are given below:

$$Q^{new}(s_t, a_t) = (1 - \alpha)\ .\ Q(s_t, a_t) + \alpha\ .(r_t\ +\ \gamma\ .\ maxQ(s_{t+1}, a)\ ) \dots\dots\dots (1)$$

From equation (1), the $Q(s_t, a_t)$ denotes the older q value for the given state action pair. α denotes the learning rate $(0 < \alpha \leq 1)$. The reward for the for action a_t is denoted by r_t for moving to state s_t.The discount factor is represented by γ. The estimation for the optimal future values is represented by $maxQ(s_{t+1}, a)$. An episode of the algorithm ends when state s_{t+1} is a final or terminal state. However, Q-learning can also learn in non-episodic tasks. If the discount factor is lower than 1, the action values are finite even if the problem can contain infinite loops.

Learning Rate

The learning determines to what extend the newly acquired information should the model overwrite on the previous information. For example, if the learning rate is set to 0 then the agent learns nothing from its new experiences. If the learning rate is set to 1 then the agent only considers the most recent information from experience, this is optimal for deterministic environments. For stochastic environment the learning rate is set at a constant between 0 and 1 in order to get the optimal solution.

Discount factor

The discount factor explains the important of future rewards. It is possible that an action for a given state might not give the best reward but if that action taken the subsequent actions give the best possible reward. If the discount factor is set to 0 then the agent will only consider the current rewards without thinking for long term reward. Then again, if it is set to 1 then the agent might take so many random steps that it might never reach a terminal state. For this reason, the discount factor is set a high value at the start of the process and then decayed with each training episode by constant decay value.

Exploration vs. Exploitation

Exploration means the virtual agent takes steps that he had not explored before in the environment. Exploitation means the agent takes the step that resulted in best reward from previous experiences. If the agent only exploits the environment by taking only the steps that gave best reward during previous exploration than there might be actions that would give fewer current rewards but give the maximum reward in future states. This gives an agent which does not find the optimal solution. That why in the beginning of training the agent explores more and with time in future the agents tend more towards exploitation. It is possible that an action that does not give the best reward among the possible actions but it will lead to future actions that have more rewards than if the action with the best reward. The action with the best reward can also result in bad rewards for future actions.

Deep Q Network (DQN)

The DQN combines the Q learning algorithm with a neural network which in most cases happen to be some kind of convolution neural network. The convolution filters of the CNN try to mimic the effects of repetitive fields. A nonlinear function approximator such as a neural network is used to represent Q is not suitable for reinforcement learning as it is unstable. This instability comes from the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and the data distribution, and the correlations between Q and the target values.

The technique that is used to solve this problem is experience replay or simple called replay memory. Replay memory is a mechanism that uses randomly sampled prior actions rather than taking the most recent action to procced forward. This smoothens the changes in the data distribution and also removes the correlation in the observation. Iterative update adjusts Q towards target values that are only periodically updated, further reducing correlations with the target.

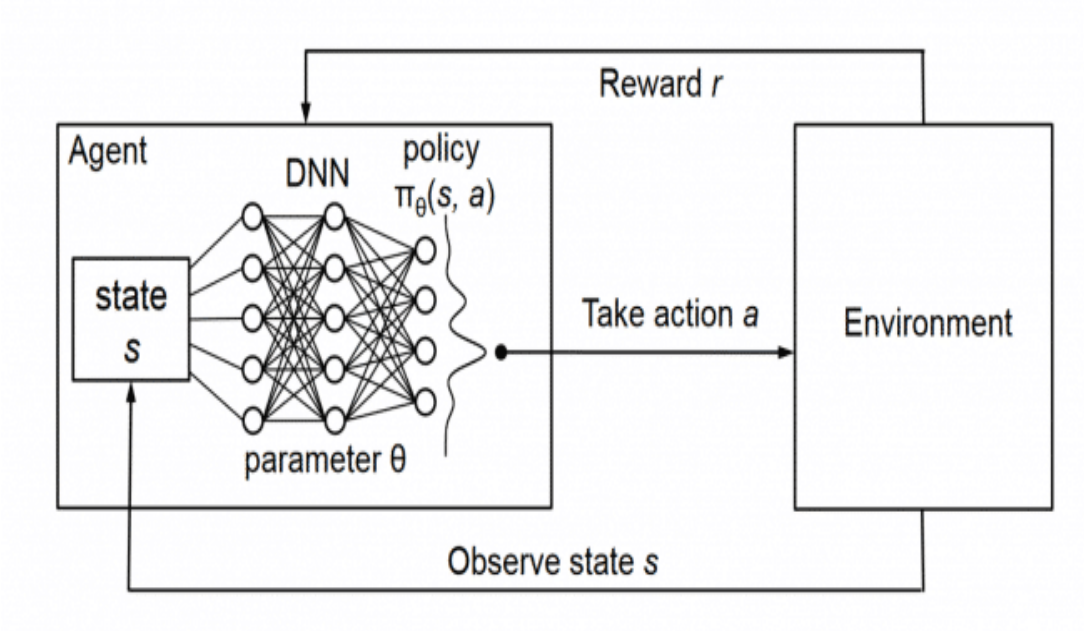


Figure 3.1.2: Deep Q network for Q values.

The DQN takes the game states as input and gives the q values for the possible state action pair. Then the Q algorithm choses the best actions possible according to its policy

and then it is applied to the environment to reach to the next state. The state for the taken action then becomes the new input for the deep neural network. This action is repeated until the terminal state is reached. During the iterations the q values for an observed state is updated and also the weights in the network is also updated. To calculate the q values, we know that future rewards of possible future rewards are taken in the equation. In order the to get the future rewards of actions the DQN needs to take to forward passes when calculating the q values. The first forward pass calculates the reward for current action and the second forward pass calculates the rewards for the future rewards for a given action. Using this this two forward passes the q value for a action state pair is calculated.

The calculation of loss is a very complicated process as there is no dataset to compare the q values. To calculate the loss a clone network of the main network is used which is called the target network.

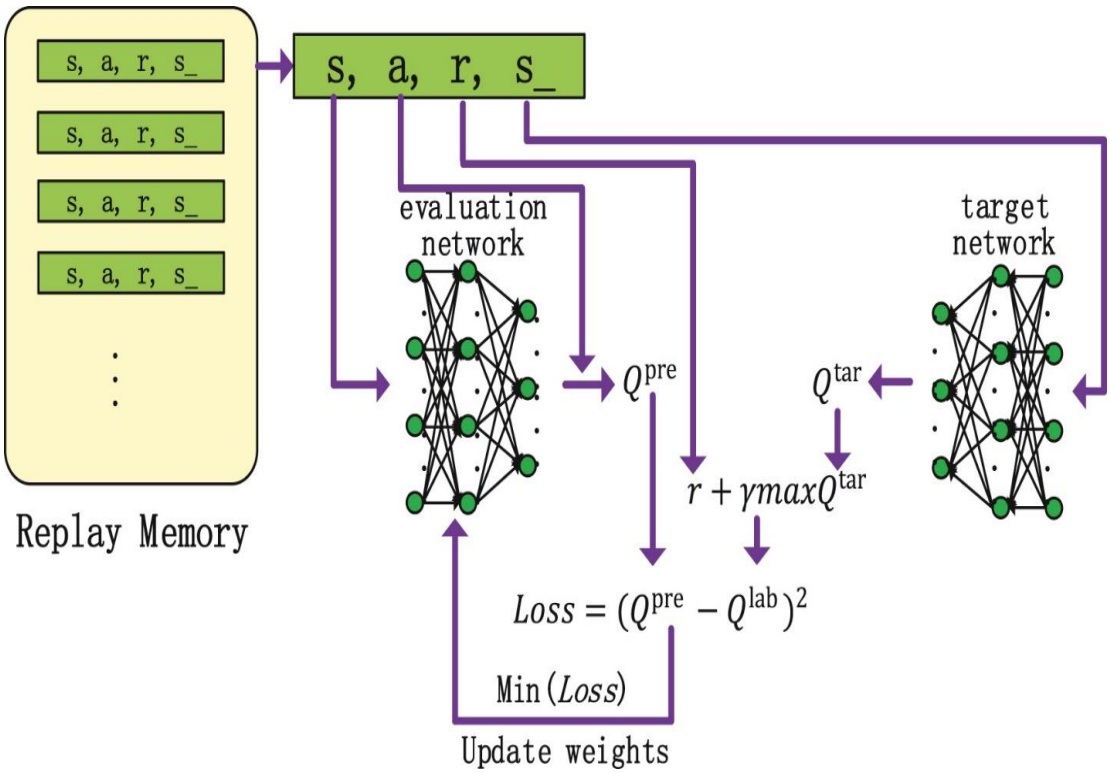


Figure 3.1.3: Loss calculation and weight updates.

Though the target network is just the clone of the main network which is called evolution network or policy network but there is very significant difference between them. The weights in the evolution network is updated every iteration where is the

weights in the target network is updated after N numbers of iterations. The q values are acquired from both the evolution network and the target network and the loss is calculated from the difference. Then gradient decent is applied and the weights in the network is updated.

Q learning vs. Deep Q Learning

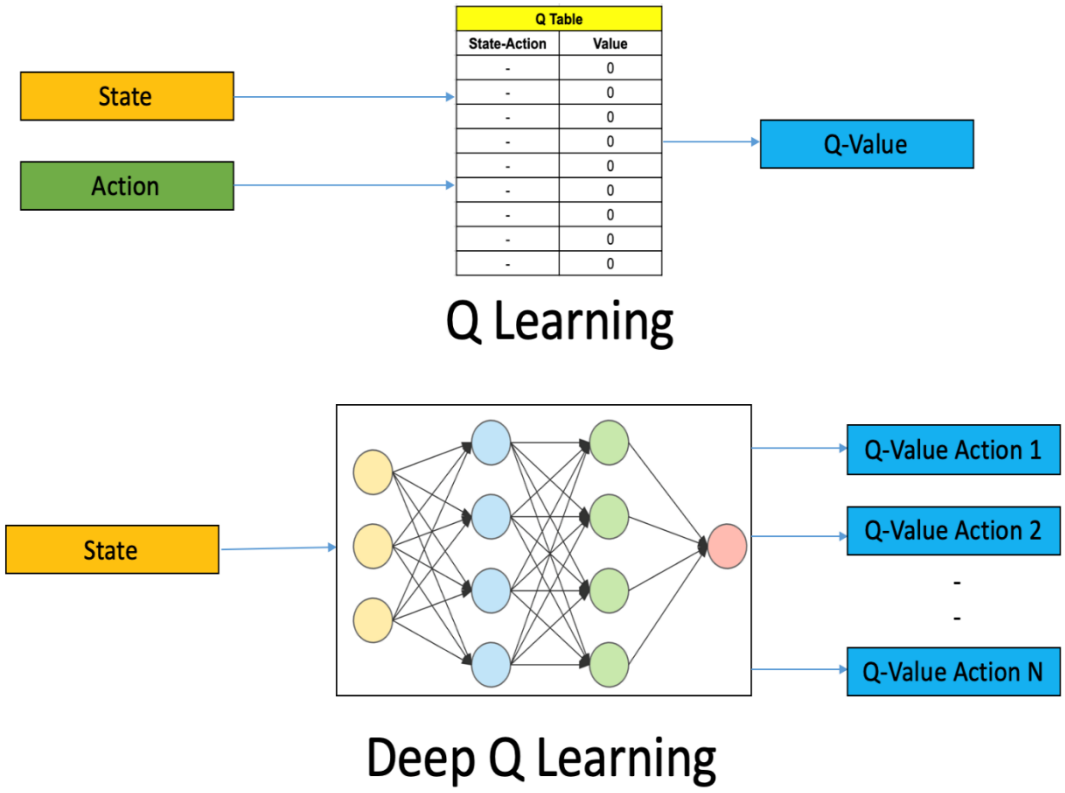


Figure 3.1.4: Q Leaning vs. Deep Q Learning.

The Q Learning algorithm generates a Q table for the environment which contains the q values for every possible state action pair possible in the environment. Now this a very good approach if the environment is deterministic and not so complicated. This algorithm is not suitable for the stochastic environment which are very sophisticated. For stochastic environment it is impossible to generate a q table of q values for every possible state action pair. This where the neural network comes in to the play. In the Deep Q Learning the neural network is used to calculate the q values for any given state. The network takes the current state as the input and gives the q values for all the possible actions for the given state as the output.

3.2 Research Subject and Instrumentation

The goal of our project is to create a model that could be applied to real-world car for achieving autonomous self-driving car. To achieve this, we choose the CarRacing-v0 gaming environment by OpenAI gym to run our reinforcement learning algorithm. We used a DQN to take raw pixel values from the game screen and output q values for the five possible actions for a given state. We have tried to make the virtual car in the game to go around the track without going off track.

Instrumentation

- Python
- Tensorflow
- OpenAI Gym
- Numpy
- Pyglet

3.3 Data Collection Procedure

For q learning algorithm to work we need q values as the data. The q values were acquired by feeding the convolution network 4 fames from the game as input and the network gave q values for the five possible actions for a given state. The frame size of the game is represented as $96 \times 96 \times 3$ grid of RGB values. The RGB values contributes very little to the training such that if it is not considered it has very little effect almost none. So, the frame is first converted to gray scale then feed to the network as inputs which results in a lot less computation for the computer to handle.

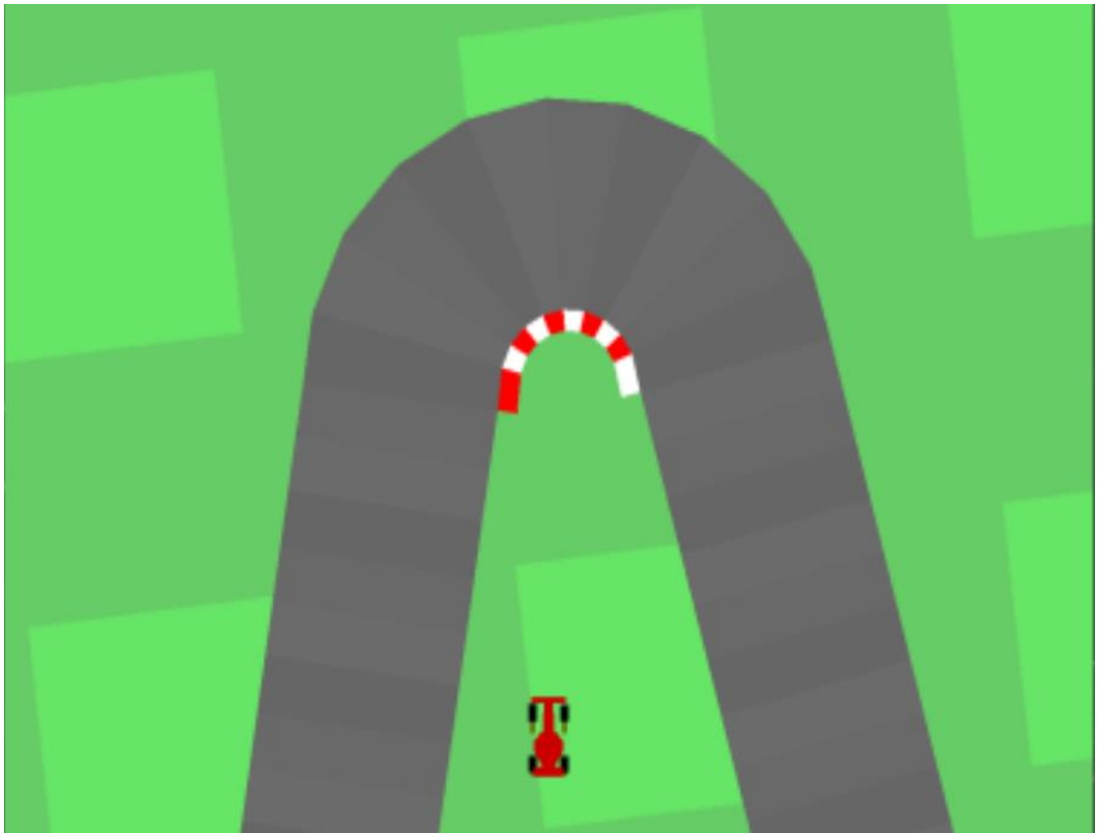


Figure 3.3.1: Game screen before processing.

Figure 3.3.2: Game screen after processing.

Convolutional neural network

Convolutional neural network [8] layers are particularly well-suited for image recognition and feature extraction, so it was natural to use them as the first layers in a neural network. We used TensorFlow's Conv2d [9] to assemble a network with the following architecture

- 8 7x7 filters Convolution layer using ReLU activation.
- 2x2 filters Pooling layer.
- 16 3x3 filters Convolution layer using ReLU activation.
- 2x2 filters Pooling layer.
- Flattening layer.
- Dense layer of 256 nodes, linear activation.
- Output Q values.

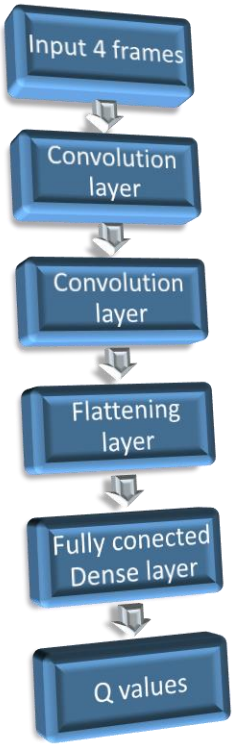


Figure 3.3.3: Convolution network structure.

3.4 Statistical Analysis

Each episode for the training process consisted of 1000 iterations. For each episode highest reward gained by the agent, number of track tile generated and average reward for last 100 episodes were calculated.

TABLE 3.4: REWARDS, TILES and AVG. REWARDS FOR THE FIRST 10 EPISODES.

Episode no.	Tiles	Max Reward	Avg. Reward
1	328	34.55	33.55
2	302	232.23	133.39
3	323	182.61	149.79
4	350	172.25	155.39
5	300	308.03	185.92
6	256	366.66	216.05
7	276	85.54	197.32
8	280	32.62	176.79
9	279	439.57	205.99
10	297	349.32	220.32

Episode

Episode is the length of the simulation at end of which the system ends in a terminal state. Consider a video game, the time your player is alive is an episode. Episodic tasks in RL means that the game ends at a terminal stage or after some amount of time.

Track Tiles

For each episode when the game is reset the track is generated randomly with different track length. The tiles represent a fixed length of the track which is very small as shown in the figure 1.3.2. These titles add up to make the full track.

Reward

The reward for this game is for each frame 0.1 and for every tile visited in the track the reward is 1000/N where N is the total number of tiles visited on the track. If the agent goes far off the track in the playfield then it is given -100 reward and the car dies and resets.

3.5 Implementation Requirements

To implement this project, we needed the following software

- Python
- OpenAI Gym
- Tensorflow

This project implements the DQN reinforcement learning agent similar to Human-level control through deep reinforcement learning but the convolution network used here is much smaller and simpler. We have used python and TensorFlow to implement the DQN on the OpenAI Gyms gaming environment. Python is a high-level interpreter and general-purpose use programing library. Different libraries like numpy, matplotlib etc. is very helpful for matrix calculation and plotting graphs. TensorFlow is main used for writing the structures of neural networks. It is an open source software developed by Google for dataflow and differentiable programming. OpenAI Gym is offering a broad range of gaming environments based on python's piglet library. The gaming environments are best suited for applying different types of reinforcement learning algorithms

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Introduction

The main result that we are concerned about is the maximum rewards the virtual car agent can achieve. The higher rewards indicate that the agent was able to visit most of the track without going off-track. During the training process the agent chooses between exploitation and exploration where exploiting the experience it had gained from prior training to take the action with the highest q value or it explores the actions it had not taken before in the hope of future rewards. Taking the action with highest q value might not be the optimal policy for all situations as some actions with lower q value will lead to actions that have the best q values. The maximum rewards are achieved by taking the optimal policy between exploitation and exploration. Finally, after a sufficient amount of training our agent was able to come up with an acceptable solution for solving the game.

4.2 Experimental Results

Though the calculation of our research work is very complex but the goal is very simple to maximize the reward for each episode. The reward is given to the agent when the car agent visits a new title in the track and also for each frame skipped. The negative rearward is given when the car agent goes too far off the track and the game is then ended. The goal of the agent is to go one full lap around the track and collect rewards. The reward is given to the agent per episode and the reward is calculated for the first 100 episodes for the purpose of analysis. The following figures represent the rewards for the first 100 episode and also the running average reward.

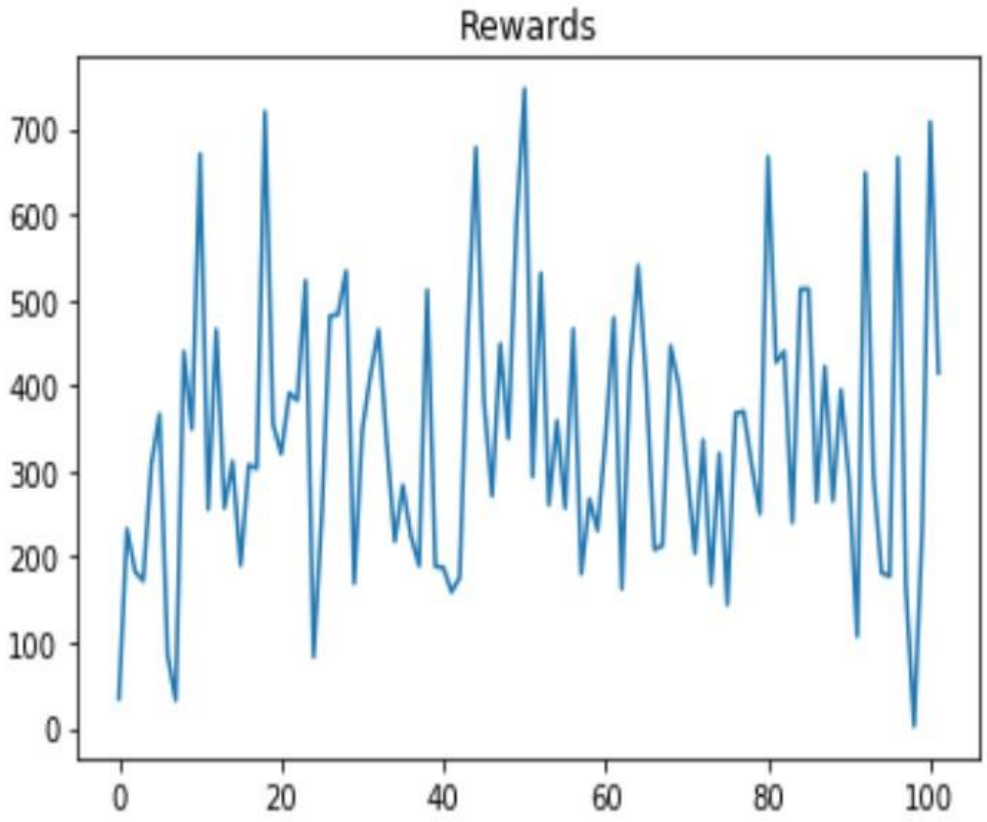


Figure 4.2.1: Reward for first 100 episodes.

From figure 4.2.1 we can see the performance of our Deep Q Network. The graph represents the first 100 episode with its rewards. At the start of the training the exploration rate is kept a lot higher so the agent takes a lot of risk exploring the environment in order to find the optimal policy. In the first hundred episodes we can observe that the car was able to receive over 700 reward points quite a few times. The DQN was able to adopt a very good policy that helped the car receive rewards around 250 points on an almost constant basis. For the first 20 episode or so the rewards were very low most of the episodes the agent was getting less than 100 for reward. That means the agent was not able to go a full lap around the track. Most of the times it was going too far off track which resulted in termination of the episode. Also, in the earlier stages of the training the agent was mostly going off track so while it was going the distance but as it was not visiting the titles on the track so it was actually receiving no rewards for it. The episodes it got high rewards was due to the fact that it was able to drive on the track most of the time.

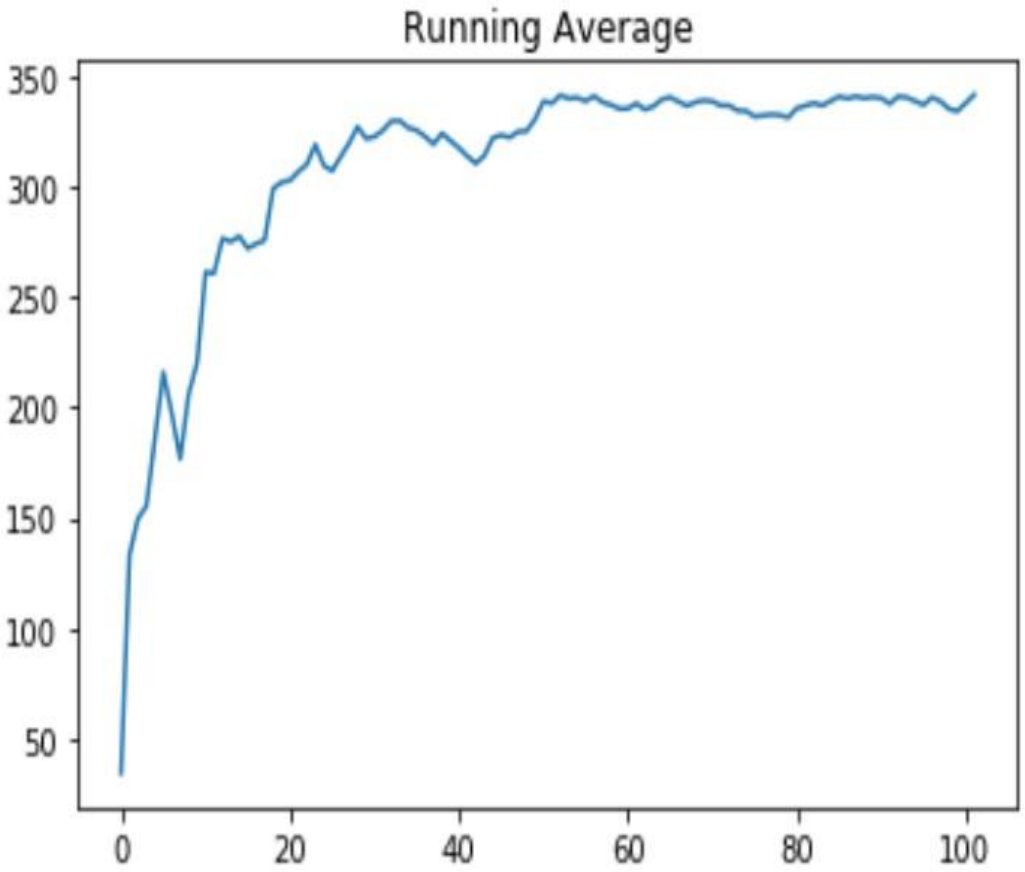


Figure 4.2.2: Average reward for 100 episodes.

From the figure 4.2.2 we can see the representation of the average reward acquired by the agent for the first 100 episodes. At the beginning of the training process the agent adapts the policy to explore the environment rather than taking actions based on experience. So, for the first 15 or so episode the reward average fluctuates a lot. From episode number 20 the agent has figured out a suitable policy that has started to give it on average just over 300 reward points. After the first 20 episodes the agent was getting sometimes over 700 which increased the average reward per episode substantially. The developing policy of the agent was able to collect very good results fairly in the early stages of the training. From 60 episodes and after it can be seen in the figure 4.2.2 that there are very minor fluctuations in the graph. This average reward per episode shows the efficiency of the Deep Q Network.

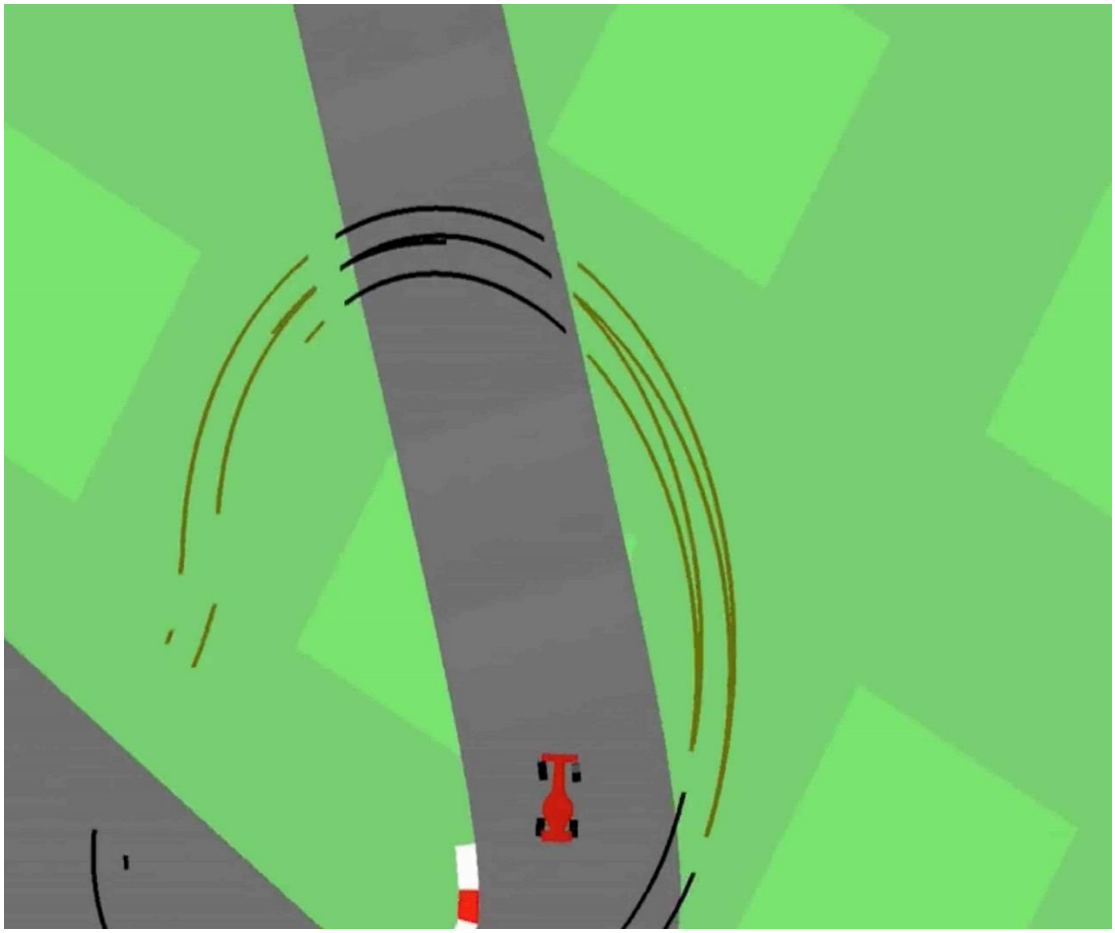


Figure 4.2.3: Training image.

In the figure 4.2.3 we can see the car agent training and trying to explore the world. From the tire marks from the screen we can observe that the car has gone off the track even drove around in loops before it come back to the track. During this time the car was not getting any rewards. As it is stated that the car only receives rewards for the new titles visited on the track. The car was visiting the titles on the track but titles that it was visiting were already explored and the reward for it was already given to the agent. The car had gone a long way on from the starting point of the track but did not receive any rewards as it skipped the track and did not visit any title on the track. The only reward that it was getting is the reward for each frame which is very low compared to the reward that it receives when it visits a new tile. This why the reward for the first few episodes were very low as it was not on the track while and driving and the rewards in the later episodes increased hugely as it was driving on the track most of the time.

4.3 Descriptive Analysis

The Deep Q Network uses a simple convolution network along with the q learning algorithm to generate the q values. Then the from the generated q values we subtract the targeted q value to calculate the loss. According to the loss the neural network then updates the weights of the connections in the network by performing gradient decent. The work process is given in steps.

- Initialize the policy network with random weights.
- Clone the policy network, and call it the target network.
- For each episode:
  - Initialize the starting state.
  - For each time step:
    1. Select an action via exploration or exploitation
    2. Execute selected action in an emulator.
    3. Observe reward and next state.
    4. Store experience in replay memory.
    5. Sample random batch from replay memory.
    6. Preprocess states from batch.
    7. Pass batch of preprocessed states to policy network.
    8. Calculate loss between output Q-values and target Q-values.
    9. Requires a pass to the target network for the next state
    10. Gradient descent updates weights in the policy network to minimize loss.
    11. After x time steps, weights in the target network are updated to the weights in the policy network.

To make use of the DQN to its fullest we used four frames from the game screen as the input for the network as the information from a static frame is of no use for the network. We used the greedy epsilon decay function to choose between the exploration and exploitation. The initial value for the epsilon was set to 0.99 and then decayed at the rate of 0.01 for each episode and minimum epsilon value was set at 0.01.

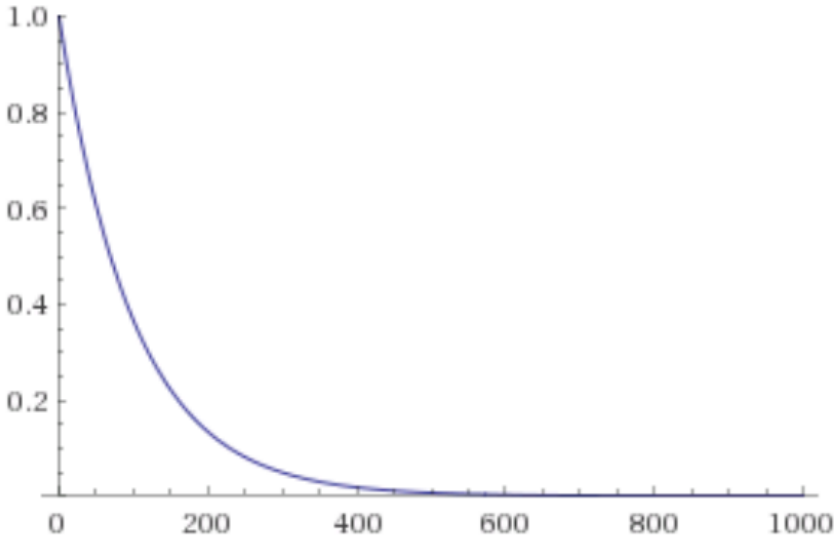


Figure 4.3.1: Epsilon decay.

For each time of taking a decision for making an action via exploration or exploitation the greedy approach compares the epsilon value with a number randomly generated in between 0 to 1 and if the number is greater than the epsilon value then the agent choses to take action via exploitation and if the value randomly generated is smaller than the epsilon value than the agent takes the action via exploration. At at the beginning of the training the epsilon value is very high and the agent takes most action based on exploration but after 200 or more episodes the agent starts to take action based on prior knowledge while exploring the environment also. With the evolution of the policy as the training goes on the agent finds the perfect balance between weather to explore or to take action based on prior knowledge. In the final stages of the training the agent takes actions based on the prior knowledge almost all the time. As the value of the epsilon becomes very small which favors the virtual agent to take actions only based on previous knowledge.

Our agent was trained for 1500 episodes by only using CPU as we did not have any dedicated graphics in our computer. The training process took more than 6 hours to complete. After the training the model was saved and then used to play the game. The end result was very desirable as the agent was able to drive around the track for a full lap with only making a few mistakes sometime. The car was able to get a reward of over 700 points almost all the time in the final stage of training. After the training was complete in the test runs the agent always collected rewards more than 800.

4.4 Summary

The game is considered to be solve if the agent is able to gain 900 reward points. Even after playing the game manually it is very hard to get score more than 850 reward points. Our virtual agent was able to get rewards around 820 consistently after the training. With limited hardware resource the training process proved to be very time consuming for our implements work. Even after training the agent for over 1000 episode the virtual agent still does some occasional mistakes. It was very difficult to change the structure of the neural network or try other types of network to solve the game. The trained agent still cuts some corners in the turns as shown in the figure 4.4.1 but is able to follow the track very efficiently.

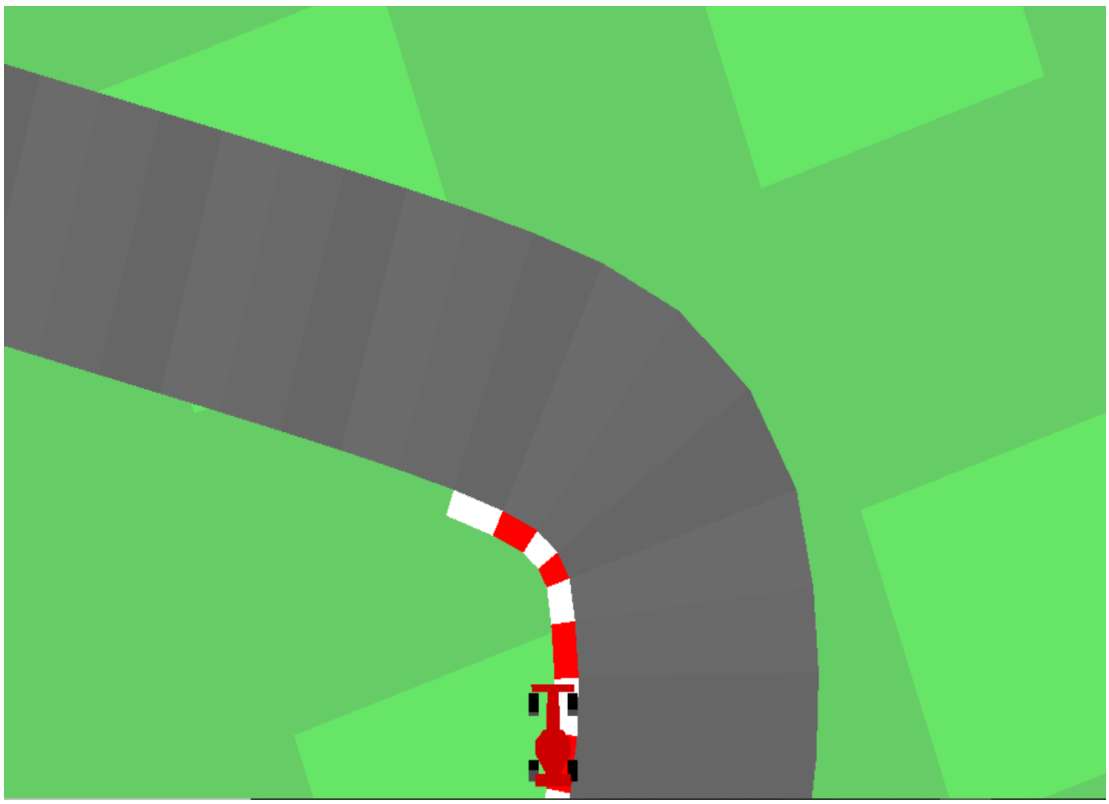


Figure 4.4.1: Agent cutting corners in turns.

As we had no prior knowledge of anything regarding our research, the results from our implementation was more than satisfactory for us. We believe if we are able apply some minor environment related tweaks and play around with the neural network a little bit

more than the we should be able to eradicate the minor flaws the agent still has. Then we should be able to use the model on a remote-controlled car to do training and testing in the real-world environment. The main goal of our research was to solve the CarRacing-v0 game by using the DQN and with the results in hand it can be said that we were able to achieve our goals

CHAPTER 5

SUMMARY, CONCLUSION, RECOMMENDATION AND IMPLICATION FOR FUTURE RESEARCH

5.1 Summary of the Study

The field of autonomous AI vehicles is growing vastly. The progress in this sector in the recent years have been remarkable the least to say. There are many challenges to make the AI for the autonomous vehicle perfect so that it could eradicate the road accidents that is a burning question in the automobile sector. Our approach in this research has resulted in positive findings regarding this matter. Though it is not the perfect model that could be applied to real-world cars yet but with sufficient amount of training and developing the model can be a bright solution for achieving autonomous cars.

5.2 Conclusions

Both of us had no prior knowledge about machine learning prior to this research. Unfortunately, most of our time was spent to learn about the basics of machine learning, specifically reinforcement learning. We spent significant amount of time learning about the use of TensorFlow, and trying to implement the algorithm combined with the neural network. We would have loved to give this time to think more about our methodology to achieve more convincing results. Another major drawback was our computer hardware as neural networks take so much of computational power to train, most of our time was spent trying to train our neural networks, we did not have the time to do more experiments by changing the shape of our neural network or use a different type of network to get better results. This forced us to do a lot of our own research on DQNs and be able to somewhat successfully implement it with limited resources. If we could have had more resources we could have managed to train and test more methodology smoothly and more time efficiently. Though, if it were that easy then we would have not had this much excitement and enthusiasm implementing this research. In spite of our results not being able to live up to our expectations to the fullest, we learned a lot in the process.

5.3 Recommendations

As we have mentioned in our related works that a lot of research has been done regarding this matter but there is yet to be a major breakthrough. After dedicating a significant amount of time behind the background studies for the research we could acknowledge a suitable way to address the implementation of our research. After a lot of hard work and studying about the neural networks and reinforcement learning algorithms, we are at a point where we can say that we could achieve our research goal. To achieve our goals, a tremendous amount of guidance was needed for guiding us through the right path of research. We have faced a lot of problems while both in our study and implementation of our research work throughout the whole journey. We needed to spent a huge amount of time understanding the broad area of deep learning, neural networks and their implementations. It would have been a much more efficient and desirable result if we were to have prior knowledge about deep learning and Q learning algorithms. Throughout our research, our co-supervisor Md. Ferdous Ahmed Foysal and our supervisor Subroto Nag Pinku sir have been massively supportive towards reaching our goal and guiding us through it.

5.4 Implication for Further Study

The future of automobile industry lies in the artificial intelligence enabled cars which can roam the roads with maximum amount of safety measures taken and cover large amount of distances without any errors. It can be said in the next decade, we will be introduced with commercially available autonomous cars for general publics to use. Since 2015 there have been significant amount of development in the field of deep learning with the introduction of technologies like Tensorflow, Keras, Theano etc. for research purposes. As much as we have seen the advancement in the field of autonomous AI cars with the introduction of deep learning and reinforcement learning, there are huge amount of problems that is yet to be solved to really achieve a true flawless experience of having autonomous cars. The policy acquired by our agent could can be improved more by making some little tweaks in the model which could be used for the training of real-world vehicles.

REFERENCES

[1] Reinforcement learning - Wikipedia, available at https://en.wikipedia.org/wiki/Reinforcement_ learning, last accessed on 01-11-2019 at 12:00 PM.

[2] Q-learning - Wikipedia , available at https://en.wikipedia.org/wiki/Q-learning, last accessed on 01-11-2019 at 12:00 PM.

[3] Deep learning - Wikipedia. , available at https://en.wikipedia.org/wiki/Deep_learning, last accessed on 01-11-2019 at 12:00 PM.

[4] Gym , available at https://gym.openai.com/envs/CarRacing-v0/, last accessed on 01-11-2019 at 12:00 PM.

[5] J. Kim and J. Canny, "Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention," in The IEEE International Conference on Computer Vision (ICCV), pp. 2961-2969, Oct. 2017.

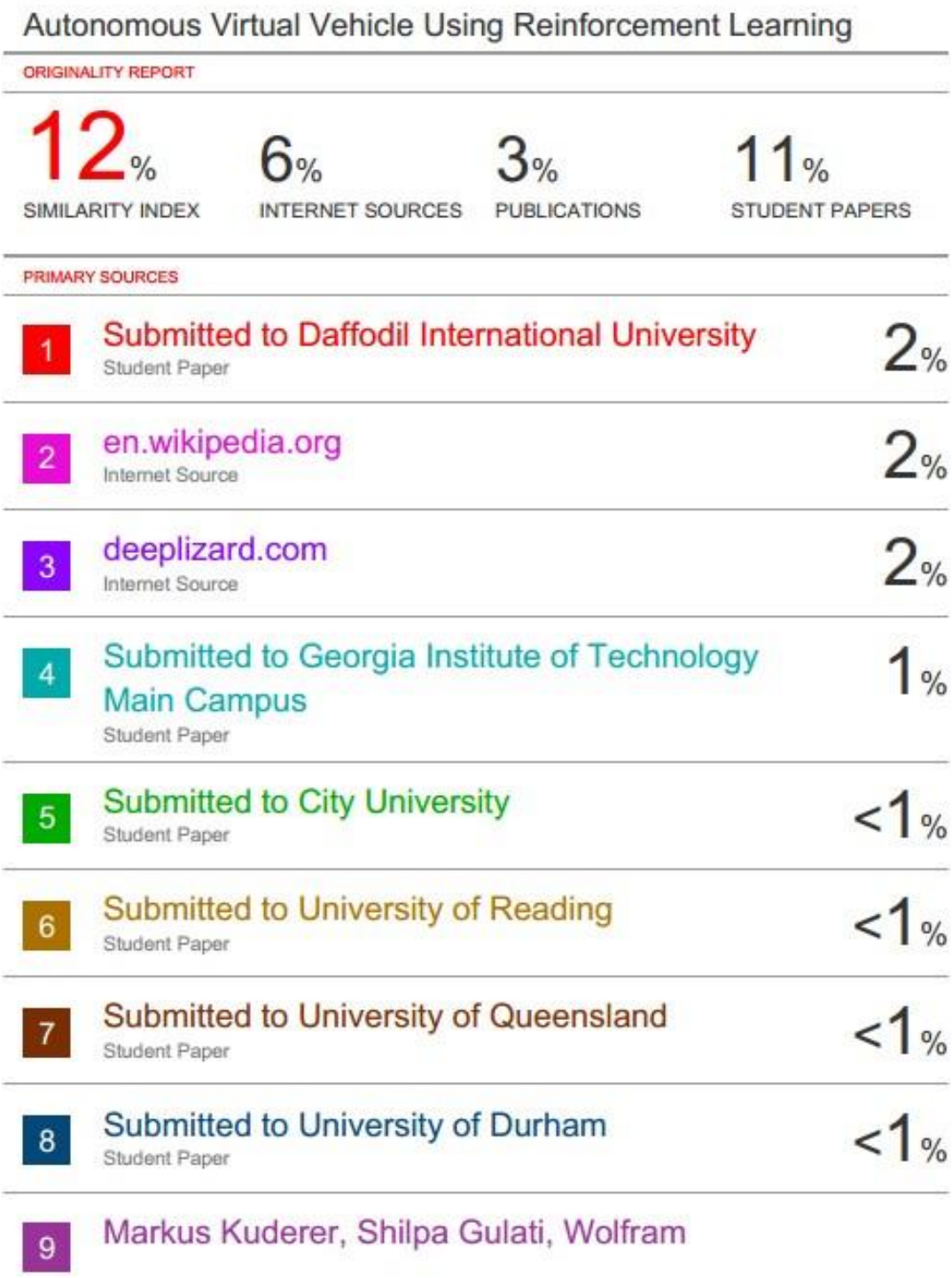
[6] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, "End to end learning for self-driving cars", CoRR, vol. abs/1604.07316, April 2016.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529, Feb. 2015.

[8] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. , available at https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, last accessed on 01-11-2019 at 12:00 PM..

[9] TensorFlow Core, available at https://www.tensorflow.org/tutorials, last accessed on 01-11-2019 at 12:00 PM.

Plagiarism Report



Autonomous Virtual Vehicle Using Reinforcement Learning

ORIGINALITY REPORT

12%	6%	3%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	2%
2	en.wikipedia.org Internet Source	2%
3	deeplizard.com Internet Source	2%
4	Submitted to Georgia Institute of Technology Main Campus Student Paper	1%
5	Submitted to City University Student Paper	<1%
6	Submitted to University of Reading Student Paper	<1%
7	Submitted to University of Queensland Student Paper	<1%
8	Submitted to University of Durham Student Paper	<1%
9	Markus Kuderer, Shilpa Gulati, Wolfram	