Car Racing Game

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering in

Artificial Intelligence and Data Science

by

Roll No: 04 Name Sarthak Bansod

Roll No: 16 Name Ashish Gupta

Roll No: 19 Name Nimisha Jain

Roll No: 54 Name Rohan Singh

under the guidance of

Mrs. Sangeeta Oswal



**Department of Artificial Intelligence and Data Science**

**CERTIFICATE**

This is to certify that **Sarthak Bansod, Ashish Gupta, Rohan Singh, Nimisha Jain of** Second Year of Artificial Intelligence and Data Science studying under the University of Mumbai have satisfactorily presented the Mini Project entitled **Car Racing Game** as a part of the MINI-PROJECT for Semester-IV under the guidance of **Mrs Sangeeta Oswal** in the year 2021-2022.

Date: 4 May 2022

(Name and sign) (Name and sign)

Head of Department Supervisor/Guide

**Department of Artificial Intelligence and Data Science**

**DECLARATION**

We, ***Sarthak Bansod, Ashish Gupta, Rohan Singh, Nimisha Jain*** from ***D6AD***, declare that this project represents our ideas in our own words without plagiarism and wherever others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our project work.

We declare that we have maintained a minimum 75% attendance, as per the University of Mumbai norms.

We understand that any violation of the above will be cause for disciplinary action by the Institute.

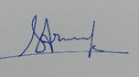
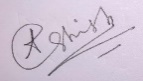
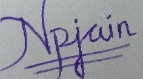
Yours Faithfully

1.Sarthak Bansod

2.Ashish Gupta

3.Nimisha Jain

4.Rohan Singh

(Name & Signature of Students with Date)

**Acknowledgement**

We have taken a lot of effort into this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to Mrs. Sangeeta Oswal ma'am for their guidance and supervision. We would like to thank her for providing the necessary information and resources for this project.

We would like to express our gratitude towards our parents & our friends for their kind co-operation and encouragement which help us a lot in completing this project.

Thank you to all the people who have willingly helped us out with their abilities.

Table of Contents

**Abstract**

**List of Tables**

**List of Figures**

**1. Introduction**

1.1. Introduction

1.2. Problem Statement

1.3. Objectives

1.4. Scope

**2. Literature Survey**

2.1 Literature/Techniques studied

2.2 Papers/Findings

**3. Analysis and Design**

3.1 Analysis of the system

3.2 Proposed Solutions

3.3 Design of the proposed system

**4. Results and Discussion**

**5. Conclusion and Future Work**

**References**

Introduction:

The Car Racing-vo environment task learns from pixels. Each state contains 9216 = 96x96 pixels. For each frame there is a fixed reward of -0.1 and for every track tile visit there is reward of +1000/N where N is the total number of tiles in the track in that particular frame. For example, if you have finished in 732 frames, your reward is 1000 - 0.1\*732 = 926.8 points. [3] An episode finishes when the car visits all the tiles. We choose OpenAI Gym platform since it provides somewhat closer to the real-world RL applications where actions are continuous and state spaces are generally high dimensional.

Problem Statement:

Car Racing game using Reinforcement Learning with OpenAI Gym

Objectives:

The objective of this study was to explore a real-world application of machine learning. Using positive reinforcement to incentivize the vehicle to stay on the desired path is similar to what is being developed for autonomous vehicles.

This can be applied in other fields such as natural language processing and reinforcement learning for computer games

Scope:

This can further be used to develop a computer car racing game

Despite the recent breakthroughs in AI and extraordinary efforts of many auto-making companies, fully autonomous cars are still out of

reach except for some trial programs. Engineers and AI scientist have been developing prototypes to make self-driving more efficient and to outperform humans. Driving is one of the most complicated activities when it comes to teaching an AI agent. In order to teach the car to self-drive, a simulator can be a reliable option, which is a synthetic environment that can imitate the real world.

Literature/Techniques

OpenAI GYM:

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It makes no assumptions about the structure of your agent, and is compatible with any numerical computation library, such as TensorFlow or Theano.

The [gym](https://github.com/openai/gym) library is a collection of test problems — environments — that you can use to work out your reinforcement learning algorithms. These environments have a shared interface, allowing you to write general algorithms.

Deep Learning:

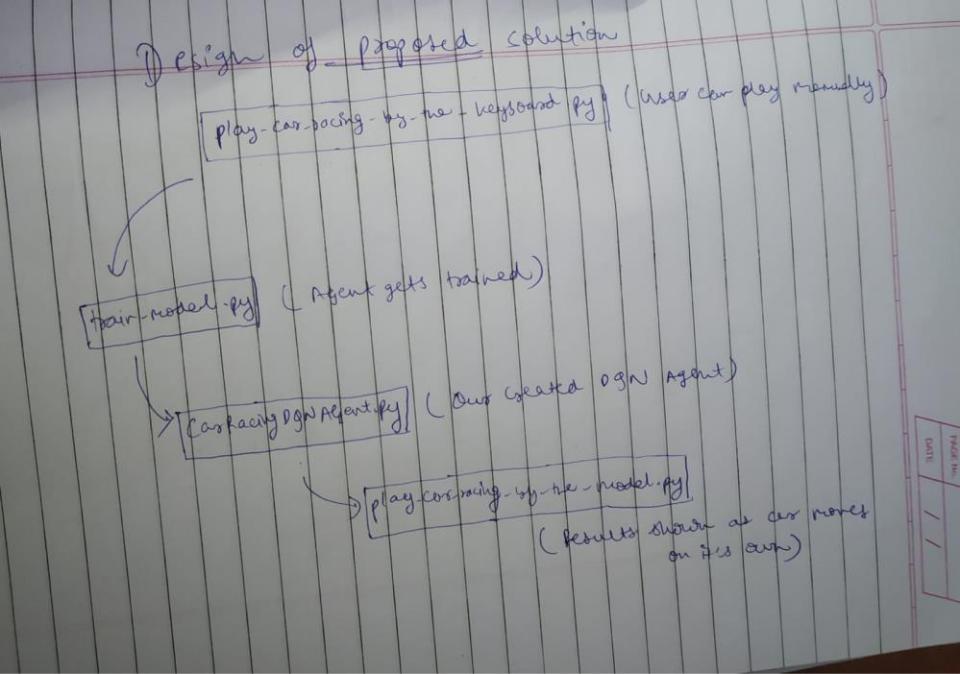
Deep learning is a subset of [machine learning,](https://www.ibm.com/cloud/learn/machine-learning) which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many [artificial intelligence (AI)](https://www.ibm.com/cloud/learn/what-is-artificial-intelligence) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Papers/Findings:

[Introduction: Reinforcement Learning with OpenAI Gym | by ASHISH RANA | Towards Data Science](https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2) [Applying a Deep Q Network for OpenAI’s Car Racing Game | by Ali Fakhry | Towards Data Science](https://towardsdatascience.com/applying-a-deep-q-network-for-openais-car-racing-game-a642daf58fc9)

1. **Analysis and Design:**
   * Analysis of System: We’ve created a Reinforcement learning model/agent which learns/trains from playing the game. Basically, we’ve to maintain the vehicle (in our case, a car) in a specified path, so that the car doesn’t leaves the track. The more we maintain the car in that path, the more accurate the agent will be trained. In this way, we can create an agent which can be used in a different game as a part which plays on its own.
   * Proposed Solutions: For making the agent more and more accurate, we have used the reinforcement approach of Machine Learning, as it fits right according to our need. We have created the agent using TensorFlow library of python along with libraries such NumPy, pandas, etc. We’ll be making use of OpenAI Gym environment, as it fulfils our car racing need. The more accurately a user plays, the more accurate the agent will be.
   * Design of the proposed solution:



**4. Results and Discussion:**

Implementation Details

We have implemented the whole project through the following file structure:

* train\_model.py: The training program.
* common\_functions.py: Some functions that will be used in multiple programs will be put in here.
* CarRacingDQNAgent.py: The core DQN class. Anything related to the model is placed in here.
  + play\_car\_racing\_by\_the\_model.py: The program for playing CarRacing by the model.
  + play\_car\_racing\_with\_keyboard.py: The program for playing CarRacing with the keyboard.
  + save/: The default folder to save the trained model.

Deep Q Learning/Deep Q Network (DQN) is just a variation of Q Learning. It makes the neural network act like the Q table in Q Learning thus avoiding creating an unrealistic huge Q table containing Q values for every state and action.

Q value is the expected rewards given by taking the specific action during the specific state.

In a more mathematical saying, Q value can be written as:

> Q(s,a) = r(s,a) + γ(maxQ(s',A))

* `s` is the current state
* `s'` is the next future state
* `a` is the particular action
* `A` is the action space
* `Q(s,a)` is the Q value given by taking the action `a` during the state `s`
* `r(s,a)` is the rewards given by taking the action `a` during the state `s`
* `maxQ(s',A)` is the maximum Q value given by taking any action in the action space `A` during the state `s'`
* `γ` is the discount rate that will discount the future Q value because the future Q value is less important

The Q value given the state `s` and the action `a` is the sum of the rewards given the state `s` and the action `a` and the maximum Q value given any action in the action space and the next state `s'` multiplied by the discount rate.

Therefore, we should always choose the action with the highest Q value to maximize our rewards.

To play the game with your keyboard, execute the following command.

python play\_car\_racing\_with\_keyboard.py

* Control the steering wheel by using the `left` and `right` key.
* Control the gas by using the `space` key.
* Control the break by using the `shift` key.

After having the DQN model trained, let's see how well did the model learned about playing CarRacing.

python play\_car\_racing\_by\_the\_model.py -m save/trial\_XXX.h5 [-e 1]

* `-m` The path to the trained model.
* `-e` The number of episodes should the model play.

The Deep Q Network (DQN) takes 3 consecutive top views of the current state of the 2d car racing game as the input and outputs the Q value for each action.

* Note that the input shape is 96x96x3. The last dimension "3" doesn't mean "RGB" in colors but the 3 consecutive top views of the current state(96x96). The top view image has only one dimension because it is a grayscale image.

Color doesn't matter much in this game, so we take advantage of the original color dimension by storing the top view image stack

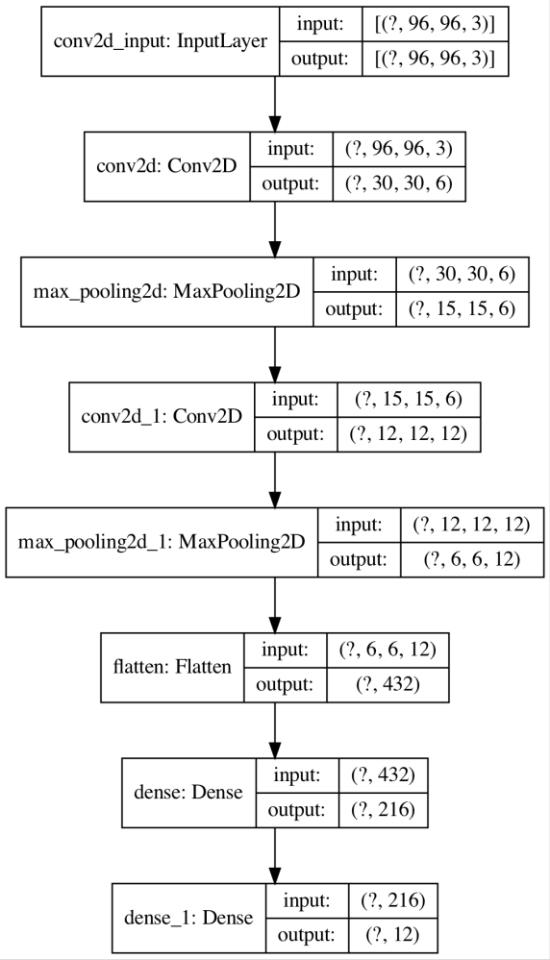
* Convolutional layers are used for capturing features from the image.
* Max pooling layers are used for preserving important features meanwhile discarding unnecessary information from the network in order to keep the network small.
* Dense layers are the other terms for fully connected layers in Keras.
* The output shape represents the Q value of the 12 actions. These actions are:

3 states of the steering wheel (left, straight, right), 2 states of the gas (full gas, release gas), and 2 states of the break (giving 20% break, release break).

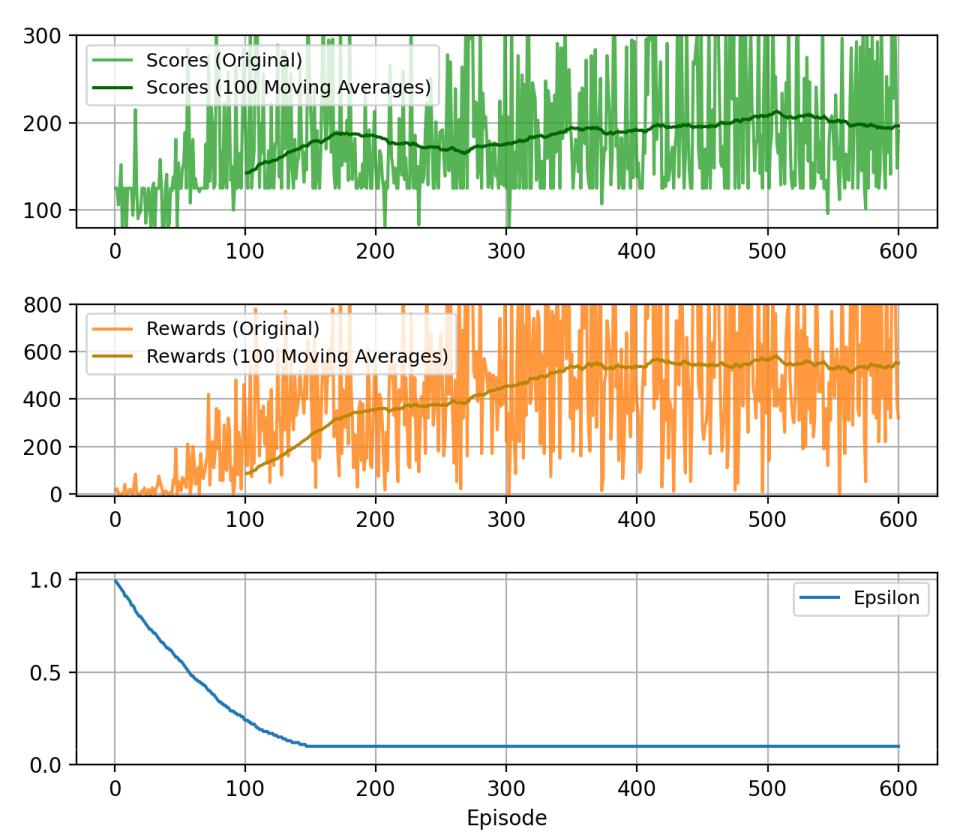
* The `? ` dimension is used for batch input.

Working Model/Output :

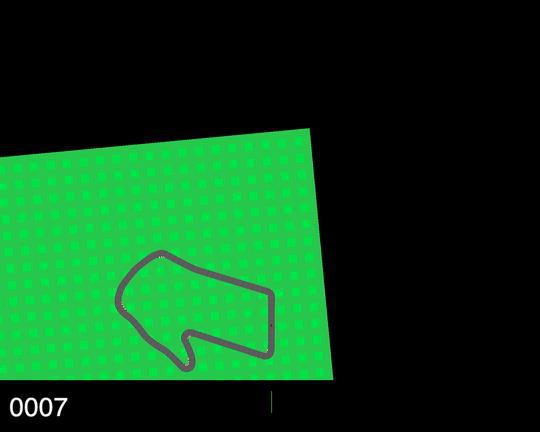
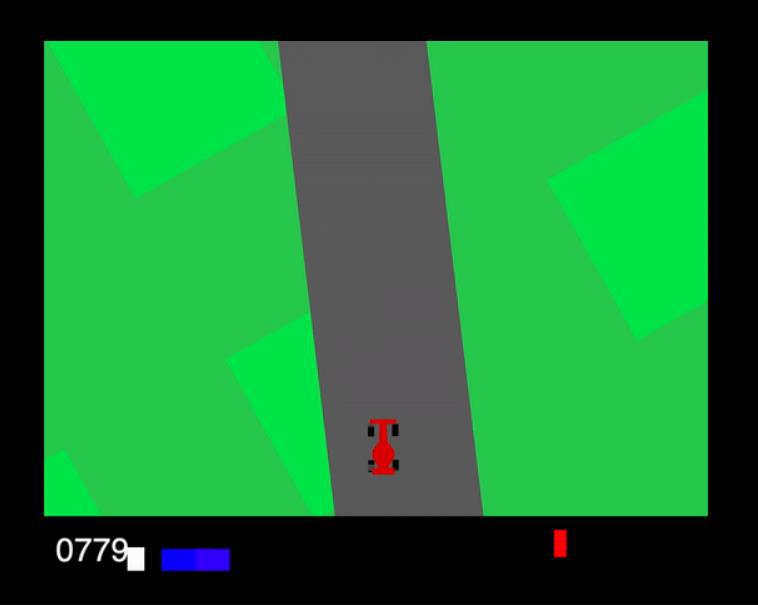
* Model Architecture:



* Training Results:



Game Screenshots:



Output Analysis:

* + As you can see, when we hit ‘python

play\_car\_racing\_with\_keyboard.py’, it lets user play normally.

* + After that we train the agent using ‘python train\_model.py’.
  + The DQN agent trains itself and we save the model in ‘save’ directory.
  + Now, when we hit ‘python play\_car\_racing\_with\_model.py’, we can see how well our agent has been trained, as its self-drives through the lane.

1. **Conclusion and Future Work:**
   * So, this is how we have implemented Reinforcement learning with a Car Racing game.
   * In future, we would like to integrate the Agent we have trained in an Actual game where a user plays against computer or Self-trained agent.
   * Our agent can be used in a game where we have multiple cars running in lane, where the car which goes out of the lane, gets eliminated.
   * There, we can have a user playing against our well-trained agent.

* So, this is how we would like to continue this project with exciting future applications.