CS1110: Artificial Intelligence and Machine Learning

SUBWAY RACER USING ARTIFICIAL INTELLIGENCE

PREPARED BY

Ankur Soni (2018BTECHCSE125)

Harshit Singhal(2018BTECHCSE107)

Kunal Sharma(2018BTECHCSE109)

Shivi Sharma(2018BTECHcse109)

FACULTY SUPERVISOR

DR. ALOK KUMAR

Department of Engineering (IET)
JK Lakshmipat University Jaipur

DECEMBER 2020

ACKNOWLEDGEMENT

Firstly, we would like to express our gratitude to our course supervisor **Dr Alok Kumar**. Without his constant support and advice, we would not be able to execute this report flawlessly. He also allowed us to contact him freely regarding any problem in the project from the very first beginning of this work until it is done. His guidance and support paved the soft way for us to complete the project in the given time. We have no hesitation to say that, without his constant guidance, we would probably fail to complete the project.

Also, we would like to extend our thankfulness to our parents for their constant support during this pandemic when we were working from home.

Sincerely
Ankur Soni
Harshit Singhal
Kunal Sharma
Shivi Sharma

ABSTRACT

This report presents a car dodging game called Subway Racer, in which the computer protects the Red car from dodging with the Blue cars running in three adjoined lanes. The objective is to provide the viewer with a challenging and enjoyable experience. The programs are designed to access the track and object information and are simulated for collision detection. The project revolves around the identification of objects or cars. The unique car is the one which is driven through Artificial intelligence by our program. To achieve the said objective, the game comprises Artificial Intelligence (AI) techniques. This report gives an overview to the reader with a brief history of AI techniques that are used in games, their uses, and the specific techniques used in the formation of these games – Subway Racer. Keeping cars on the track and henceforth avoiding the collision is the objective around which the project revolves. The algorithm is accurate and robust to cope up with the fast-changing responses. The objective of the project is to develop an efficient solution for the problem considering machine learning algorithms and artificial intelligence algorithms-based problem-solving approach. The project is implementation is followed by a detailed research in the respective field. The subway racer is a result of abstraction programming along with automatic user interface to mimic the process of natural driving.

CONTENTS

ACKNOWLEDGEMENTS	*2*
ABSTRACT	*3*
LIST OF FIGURES	*4*
CHAPTER 1: INTRODUCTION	**5**
CHAPTER 2: ABOUT THE PROJECT	**6**
CHAPTER 3: BACKGROUND LITERATURE	**7**
CHAPTER 4: TECHNOLOGY USED	**8**
CHAPTER 5: IMPLEMENTATION	**9**
CHAPTER 6: RESULTS	**13**
CHAPTER 7: CONCLUSION	**14**
CHAPTER 8: FUTURE SCOPE	**15**

CHAPTER 1- INTRODUCTION:

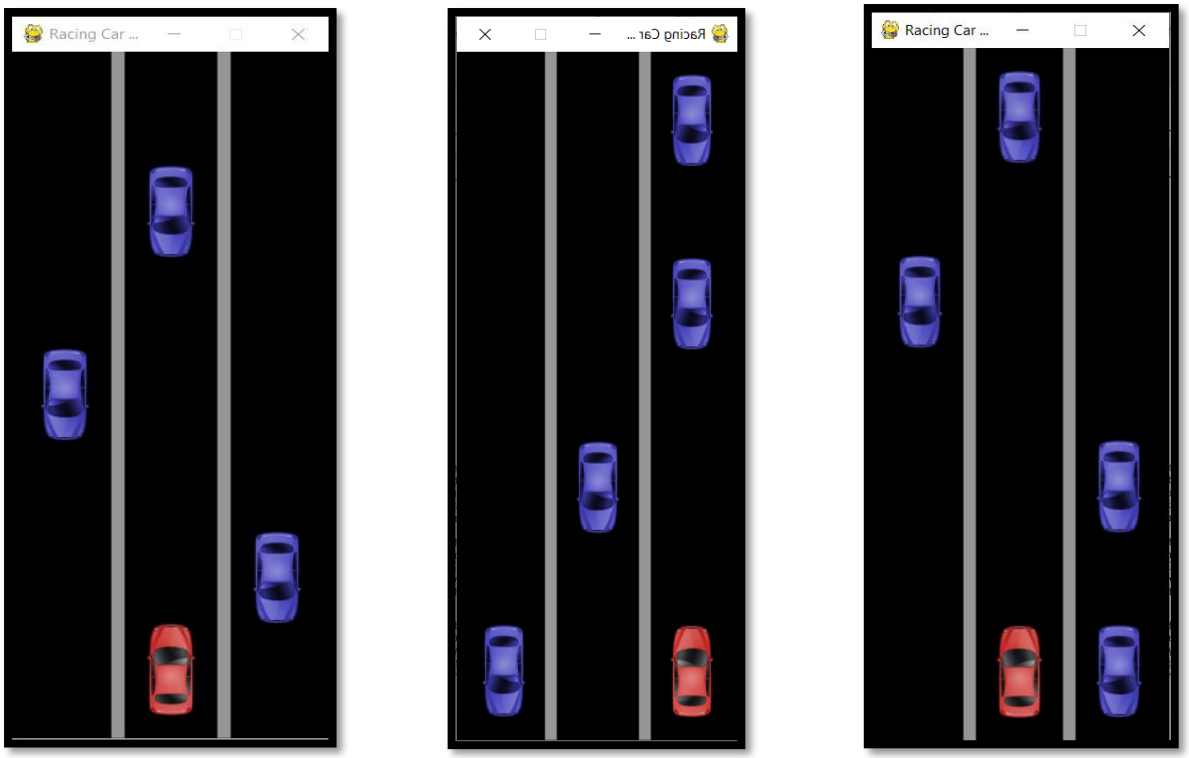
Games have become an intrinsic part of everyone's life. Traditional games often include artificial conflicts having a quantifiable outcome with rules and regulations. These generally include puzzles or such challenges that provide gamers to solve real-world issues. But these traditional games are now getting replaced by electronic games especially by the upcoming gaming generation which majorly consists of "digital natives".

A similar transition has been witnessed by the Artificial Intelligence Community from the 'classical AI games' to the contemporary AI techniques that are being adopted in electronic games. The difference between the two lies in their objectives, wherein the objective of classical AI games is the quantifiable outcome, the objective of contemporary games is to provide the player an enjoyable experience with no quantifiable outcome. The Subway Racer game is a computer-controlled video game that is made with a view to providing an enjoyable experience to the viewer.

CHAPTER 2- ABOUT THE PROJECT:

The project is about implementing a 2D Car Racing game environment using Pygame with possible play mode, autopilot mode and AI mode and implementing a Neural Network to play the game using Keras with categorical cross-entropy loss function. In order to avoid obstacles and collisions from random dynamic obstacles, we have applied Artificial Intelligence. The project is basically a GUI application where we have a terrain road distributed in 3-lanes which are straight segments, named track. One of the programmes is designed to access the track and object information while another one is simulated for collision detection. A uniquely identifying car is made to move automatically in forwarding direction whereas rest all the similar cars are identified to be moving in a downward direction. The project revolves around the identification of objects or cars. The unique car is the one which is driven through Artificial intelligence by our program. Opponent cars are similar as of the unique car, but differs in color and moving direction, and are also driven with Artificial intelligence. Keeping cars on the track and henceforth avoiding the collision is the objective around which the project revolves. The algorithm is accurate and robust to cope up with the fast-changing responses.

Our project can be accessed through github at :- https://github.com/jklu-jaipur/SUBWAY-RACER--A-BOT-GAME



(FIGURE 2.1- INGAME IMAGES)

CHAPTER 3- BACKGROUND LITERATURE:

There has been significant work that focuses on developing racing games using artificial intelligence, these games are played between humans and players that are trained by artificial intelligence generally referred to as bots. The role of bots in the game was introduced so that a human need not to play with any other human, instead, he or she can play with the players that are controlled by computers through artificial intelligence. Not only car stimulating or racing games but also games like pubg , counter strike, and Call of Duty are some of the games that are supported with bots. The most suitable and common bot game example is Pacman, the computer-controlled ghost in the games can move towards the player control character. Racing games that have been created to date are mainly human versus bot games, hey many games provide simple racing between the given two in the Maps of different tracks and curves. All the games are based on winning strategies.

Some card games are based on a strategy of object avoiding, in these types of game, bot control cars that run through the track ignoring and avoiding obstacles come into their pathway. These types of games are not basically winning or losing types of games, these types of games are based on how much you can survive, as long as you are racing through the tracks avoiding obstacles and objects in the pathways you will be in playing mode, as soon as you hit any obstacle your game ends.

CHAPTER 4 - TECHNOLOGY USED:

3.1 Software Details

- Anaconda Distribution (v5.1)
- Python (3.6.5)
- Packages Used:

    1. Pygame
    2. Pandas (0.22.1)
    3. NumPy (1.14.2)
    4. Keras
    5. TensorFlow

3.2 Hardware Details

- **Operating System (OS):** Windows 10, Linux, 64-bit macOS.
- **System architecture:** 64-bit or 32 bit with Linux or Windows.
- **CPU:** Intel Core 2 Quad CPU Q6600 @ 2.40GHz or greater.
- **RAM:** 4 GB or greater.

CHAPTER 5- IMPLEMENTATION:

The very first requirement for making this game is to install pygame, pandas, keras and tensorflow in your system.

>>> pip install pygame
>>> pip install pandas
>>> pip install keras
>>> pip install tensorflow

Classification Model

1. Import all the necessary libraries.
2. Read the data from the csv.
3. Remove all the redundant columns.
4. CSV has 5 columns:

    - First column - My_position,
    - Second column - Left_line,
    - Third column - Mid_line,
    - Fourth column - Right_line,
    - Fifth column – Action

    My_position and Action have integer values from [0 to 2]
    Left_line, Middle_line and Right_line have integer values from [0 to 8]

	My_position	Left_line	Mid_line	Right_line	Action
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000
mean	0.941700	3.502200	3.327200	3.418050	1.000100
std	0.786977	2.645646	2.603282	2.618764	0.447895
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	3.000000	3.000000	3.000000	1.000000
75%	2.000000	7.000000	6.000000	6.000000	1.000000
max	2.000000	7.000000	7.000000	7.000000	2.000000

From the table above we can assume the following:

Mean values of all columns are acceptable

- For My_position and Action columns the mean is expected to be around 1
- For Left_line, Mid_line, Right_line columns the mean is expected to be around 3.5

Std values of all columns are acceptable, although Action column seems a bit off

- For My_position and Action columns the std is expected to be around 1
- For Left_line, Mid_line, Right_line columns the std is expected to be around 2 - 3

Min values of all columns are correct

- For My_position and Action columns the min has to be equal to 0
- For Left_line, Mid_line, Right_line columns the min has to be equal to 0

25%, 50%, 75% values of all columns are acceptable

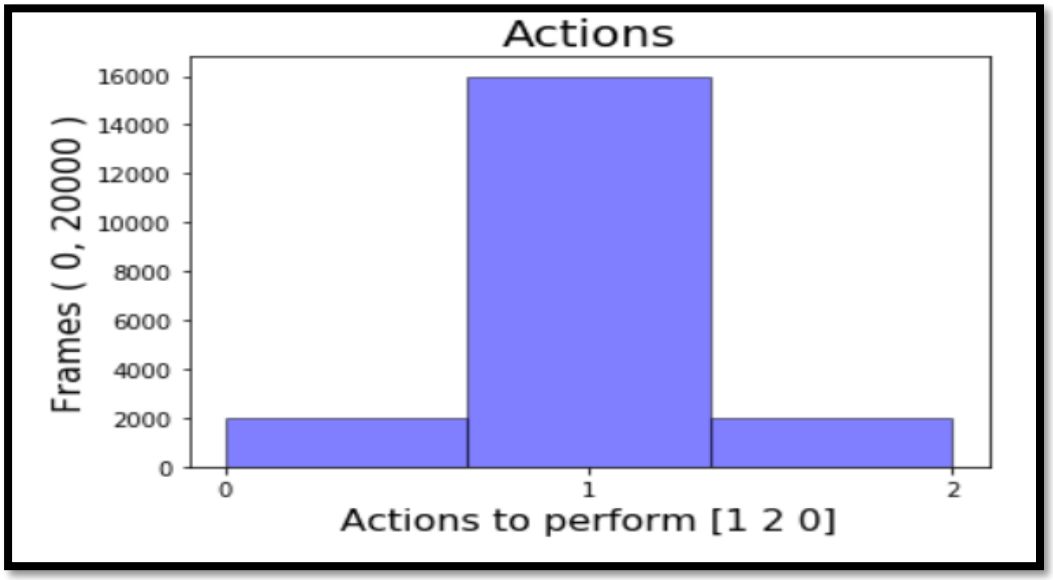
- For My_position and Action columns the values are expected to be 0, 1, 2 respectively
- For Left_line, Mid_line, Right_line columns the values are expected to be 1-2, 3-4, 5-6 respectively

Max values of all columns are correct

- For My_position and Action columns the max has to be equal to 2
- For Left_line, Mid_line, Right_line columns the max has to be equal to 7

5. Build up a histogram of Actions column to find out why std is small
6. Action types are as follows: Left - 0, Stay - 1, Right -2
7. Set a title for histogram
8. Find out amount of unique actions
9. Set X axis to have only values of the actions that exist in the column
10. Show actions that can be performed on the X axis.
11. Show total amount of values the column has
12. Set the histogram with above properties

13. Show the histogram



(FIGURE 5.1- GRAPHICAL REPRESENTATION)

From the histogram above we can see that most of the actions are 1 which means stay.

This makes sense, since every 2 frames a new car appears and when it does it has to choose only 1 road line to appear at. Every batch 6 of cells (3 lines in 2 steps) had only 1 car on it, which let my car stand most of the time.
If we train our model with this data, it will assume that standing is the best choice in most of the cases. There are several ways to solve this problem. Some of them are listed below:

- We need to train some more data saved in another file and add from that to this file new rows with actions other than 1 to this dataset.
- We need to modify the car appearing function in the game that will make sure cars do not appear after one another.
- Remove some of the results from this data frame where action is 1.

14. We took the last choice and remove some portion of data that has action 1.
15. Find out the portion to be removed from ~16k down to ~2k. It makes 12.5%.
16. We took random number between 0 and 125 and if it is greater than *13 and* removed the Action.
17. After reducing Action column with specific value 1 by 87.5% we have normalized our dataset. This sort of action leads to a better performance during training the model.
18. Now it is time to separate input data and labels and split it into train and test datasets.
19. The neural network is trained in Main.py.
20. Import all the necessary libraries.
21. Set clock to keep track of frames.
22. Set runtime speed of clock to 30frames/second.

In `constants.py` file in `constants` folder, we have set `FRAME_RATE` to influence how often the game frame will change in pygame, `DATA_ROWS` to limit number of rows to be collected into a csv file, `ACTION_PERFORM_RATE` to decide how frequently your car will move (Note: The new car will appear twice as rare as ACTION_PERFORM_RATE) to make sure there is always a way out for your car during the game. Images of cars are into variables `MY_CAR_ICON` and `ENEMY_CAR_ICON`. The background of the screen as well as the white vertical lines: `BLACK = (0, 0, 0) `. stands for the colour of background and `GREY = (150, 150, 150) ` stands for vertical lines that separate the road lines. Remove cars that are out of map boundaries and filter out not active cars.

CHAPTER 6- RESULTS:

Development of Subway Racer was easier with different AI techniques and programming tools that we have used in our project to create user interface of the game keeping the performance factor the main priority in our minds. Besides using traditional AI techniques, the AI techniques and model that we have used resulted in developing the game in more attractive and efficient manner. The red/unique car is shown changing the track paths as soon as the collision condition is detected due to other colored similar cars, I.e., obstacles. The artificial intelligence algorithm results the car to automatically move and change the path. Also, the other similar cars moving in opposite direction can be seen following a systematic way of no two obstacles colliding among themselves and providing a smooth working. Hence the efficient working of model due to proper implementation of algorithm provides an enjoyable experience to the users.

CHAPTER 7- CONCLUSION:

We have concluded that implementing a 2D Car Racing controller environment using Pygame with possible play mode, autopilot mode and AI mode and implementing a neural network to play the game using Keras with cross-entropy loss function has made the things easier to achieve the complex task of object detection and overtaking the lane in the racing game. Also, lane changing maneuvers are complex and hard to implement because dynamic objects in a racing game which is a rapid changing environment and affects performance factor of the game so using AI techniques we have achieved lane maneuvering performance up to the mark with less effort and more efficiently . Which also has increased the overall game performance. The algorithms designed for detective path are intensive for CPU and processing and consumes more time. henceforth, we have proposed a solution to overcome the racing game problem of collision with random obstacles. The proper implementation tools have provided better development of the project. The project concludes that with more inputs of learning model data and implementation algorithm, it is possible to emulate human player activities using AI-controlled system.

CHAPTER 8- FUTURE SCOPE:

1. The game SUBWAY RACER was purely based on racing and entertainment factors but adding a mode of driver stimulation would add educational factors into the game. By making some changes into the game like adding a braking system, car handling controls, different angles of camera and mainly a road map for this particular mode will work perfectly. Through these changes the racer/user can easily adapt the stimulation and can get a basic idea of how to drive. Adding a traffic rule system will also make it more educational.

2. Addition of a point system will make the game more interesting and attractive where user can get their score at the end of the game which will include how much time the user took, how many obstacles the user successfully crossed and driver rating for the user satisfaction.

3. Adding different tracks of different graphics will make the game more user attractive and will play a major role in holding the interest of the user into this game.

4. Adding vehicle options other than cars like motorbikes, monster trucks, ATV's will play a key role in grabbing user attention and making the game more appealing.