

# AI DRIVEN SNAKE GAME USING DEEP Q - LEARNING



## TEAM MEMBERS:

SHRIYA KANNOJ – 11507709.

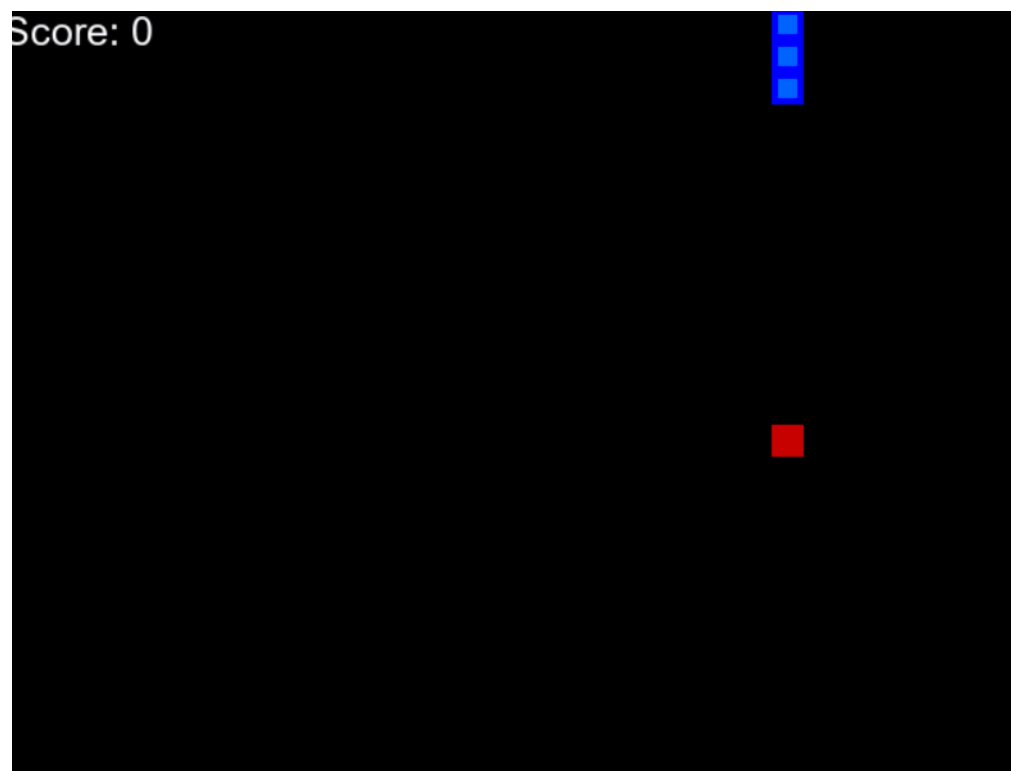
GITHUB: <https://github.com/SHRIYAKANNOJ/SDAI-PROJECT>

## VIDEO LINK:

## INTRODUCTION:

Snake game using deep Q-Learning is based on the reinforcement learning which basically instructs the snake to eat the apple within the boundaries and not to collide with the boundaries. The snake game is the most popular mobile game since the first-hand phone brand Nokia has included this game in their games list. It has been a very familiar and popular game since the 90's. In this game the main objective of the player is to score the maximum with still keeping the snake alive. The snake dies or we lose the game when the snake hits its own tail or the wall. Making the code for the snake game is very challenging for me as I am just a beginner and learning python. This project inspired me to not only learn the python but also made the learning fun and interesting. We will be using the PYGAME to create this game and we use this PYGAME open library to collect data and which is basically designed for the games like snake game. This PYGAME has all the required libraries for video and graphics for the snake game already inbuilt so, it would make the code a little short and easy to understand. The reinforcement learning is added to the game after the basic steps are implemented.

The GIF below is a basic example of snake eating the apple within the boundary in the snake game.



## **GOALS AND OBJECTIVES:**

### **MOTIVATION:**

The snake game is a very popular mobile game in which the player can have control on the snake which is inside the closed boundary and must eat the apple to increase the score. When the snake eats the apple, the snake increases in length and the score of the player increases. The reason for me to design this game using the PYGAME and the DEEP Q – Learning is to outsmart the human beings and score the maximum score possible using the AI by training the snake to eat the apple by not bumping into its own tail and the boundaries. The main objective of this project to develop a snake game which is a perfect snake game which scores the maximum score than a human being.

### **SIGNIFICANCE:**

This game is invented in the 1970's which we have seen first seen in the Nokia mobile phones. The snake game was introduced to people for having fun and also have a better understanding and prediction in the games which will activate the neurons in brain cells in the past. As the time progressed very much and the snake game is still one of the most popular games because it is not only simple to understand and play but also interesting to reach high score in this simple game. When the Nokia phone was famous, the screen of the mobile was so small and the buttons were hard to press but still the game was very interesting to play. Even after there are many versions of the game available now, the old version of the snake with no face is very interesting for me to play. This snake game has always been the classic game for me.

### **OBJECTIVES:**

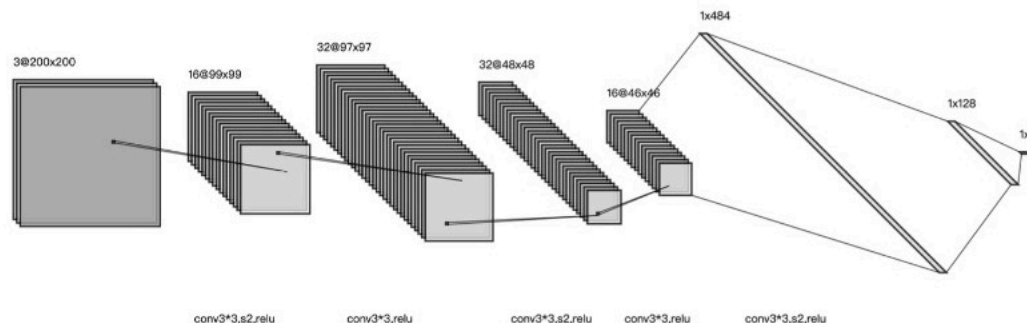
The main reason for me to develop this project is to create a professional player who can score better and compete with the human. To have the high score, the AI bot must be able to determine the best path for the snake to eat the apple avoiding the hurdles. The basic programming in this snake game is done using the PYGAME with the python. Once the game is done programming, the deep learning stuff is done again. The score can be achieved by studying few objectives and implementing them.

The main objectives are as follows:

- To develop a snake game and be able to play it.
- Introducing the AI bot to the snake game.
- Develop and compare the AI bot VS human agent in the game.
- Study different algorithms used by AI bot and its results.

## THE RULES:

- The snake tries to eat as many apples as possible, but the steps are finite.
- Snake must not bite its own tail to not kill it-self. Snake must avoid its own Collision with tail to score more and continue the game.
- snake grows by 1 unit when eating the apple. The length growth is seen immediately after the snake eats the apple.
- There are four directions for the snake to choose for the next step: left, right, top and bottom. When the snake grows in length, however there mayn't be few directions available to move.
- The game board dimensions are fixed in the size and are said as boundaries /wall. When the snake collides with this boundary it loses the game. Snake must avoid this boundary to win the game.
- After the snake eats the apple, there will be another apple placed randomly within the boundary for the snake to eat as the next target.

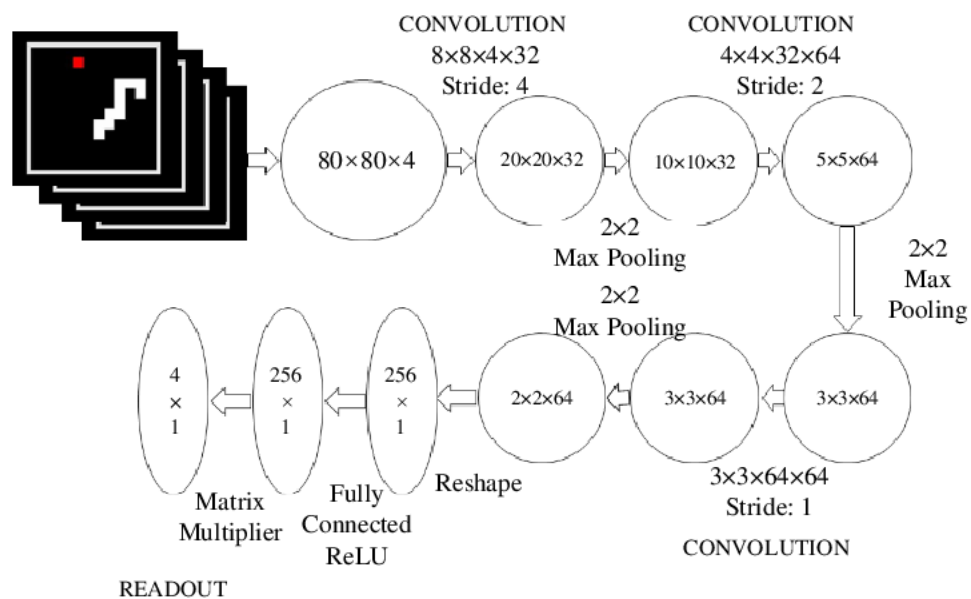


CNN

## MY MODEL:

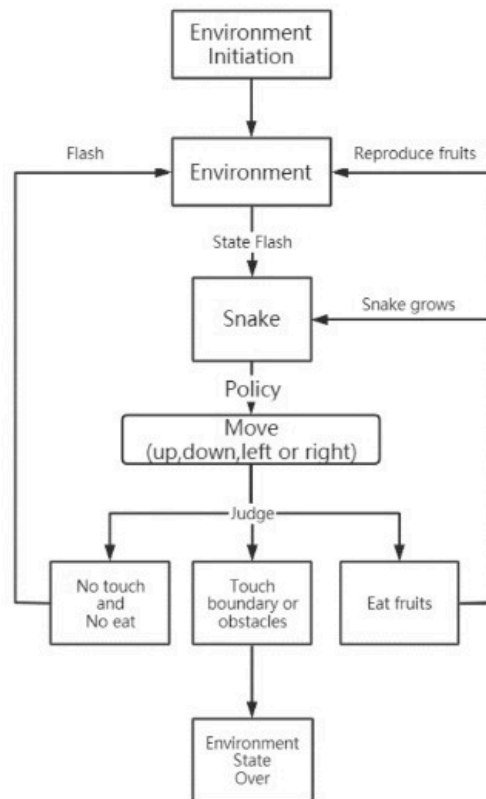
### ARCHITECTURE DIAGRAM:

In this project, we used the reinforcement learning which is very easy to understand and implement. The agent basically will learn about the changes by the interacting with its environment and then it forms different states and rewards for it. We have the cycle of this learning and then exploring in continuation forever and this learning never stops.



## WORKFLOW DIAGRAM:

As from the project, we have seen different states and the layers for the learning process. The below diagram shows the workflow for the project.



### Environment and Agent

From the diagram we can see that the Environment is being learnt by the source and the output is then sent to the output is sent to the snake and the policy is then sent to the move if the move is up left down or right the output is sent to the a judge the judge will decide if the snake touches the border or if it is eating the apple or if it is going through any obstacles in between then the output is then analyzed and sent to the environment state over. This process goes through in a cycle if the snake does not touch or does not eat the apple then the output is flashed to the environment and then the environment then sends the output to the snake and the cycle goes on and if the output

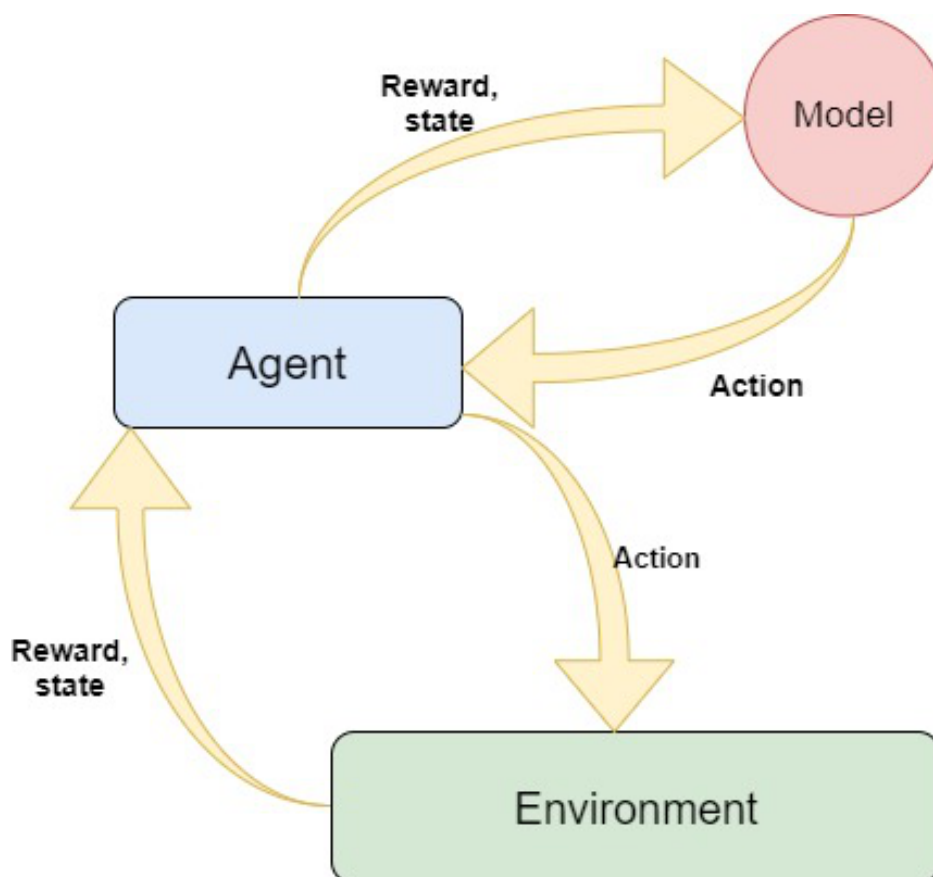
from the judge is the snake eating the apple then they output goes to the snake going again to eat the another apple or it will either reduce the number of fruits from this environment for the snake to flow within the boundary and eat the apple.

## **FEATURES:**

### **THE PREREQUISITE FOR THIS PROJECT ARE:**

- Reinforcement Learning
- Deep Learning (Dense Neural Network)
- Pygame

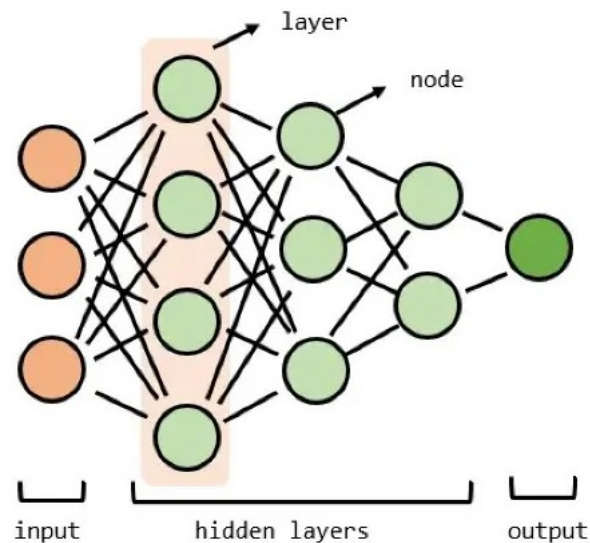
For this project, we must construct three modules:



The Environment (the game that we build)

The Model (Reinforcement model for move prediction)

The Agent (Intermediary between Environment and Model)

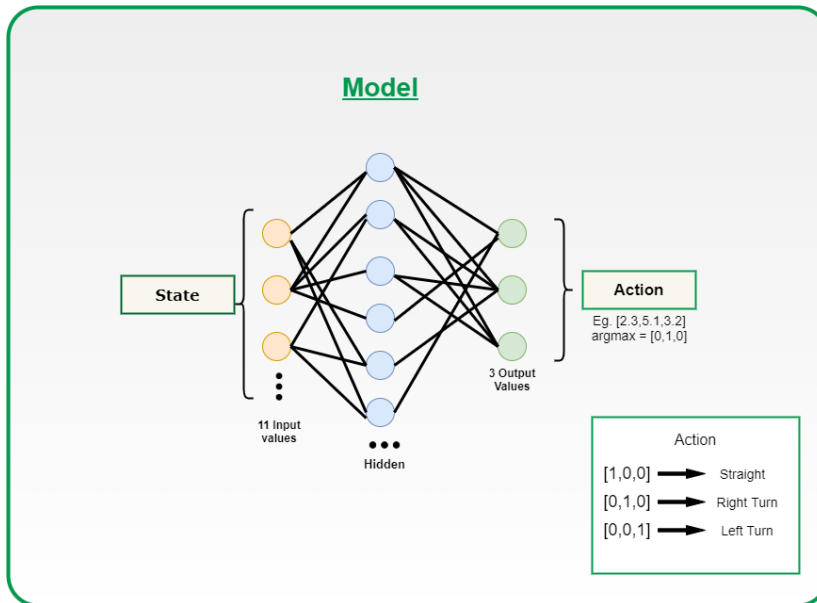


We've randomly put snakes and food on the board.

- Calculate the snake's state using the 11 values, and if any of the requirements is met, set that value to zero; otherwise, set one.
- The 11 state variables indicated above will be calculated based on the current Head position.
- After obtaining these states, the agent would transmit them to the model, which would then conduct the following action.
- Calculate the reward after performing the next state. The following are the rewards:  
+10 for eating food  
-10 for game over  
Else: 0
- Update the Q value (to be mentioned later) and Form the Model.
- After understanding the method, we must now develop a concept for how to proceed with coding this program.



We're utilizing a Dense neural network with an 11-layer input, one dense layer with 256 neurons, and three neurons as output. You may fine-tune these hyper settings to achieve the best results.



## RELATED WORK (BACKGROUND):

When it comes to the Q – learning I have started learning about the deep Q – learning in this course and in this project itself. I have done some tossing the coin code to understand better for me. This code helped me to have a very brief understanding about the reinforcement learning and the Q – learning.

The reinforcement learning is basically the training of the agent to have the better outcome and I have trained the snake to make it a perfect snake by running it 100's of times.

## DATASET:

For creating the basic design of the snake, we use the function `draw.rect()` which draws a rectangle and we can give the snake our desired color and size with the help of variables in the PYGAME. For the movement of the snake, we use key events from the key down class from the PYGAME. The key events that we use are `K_DOWN`, `K_UP`, `K_RIGHT`, and `K_LEFT` for moving the snake down, up, right, and left. We use the if condition statement to satisfy the condition that if the snake hits the boundaries of the screen, then the game is over. For

displaying the food at a random point for every iteration we use the random function and color the random position with a specific color. The initial length of the snake is 1 and it increments by 1 whenever the snake eats the fruit. For displaying the score, we use a function that displays the length of the snake decreased by 1. Finally, the score from different algorithms can be compared using pandas' data frames. We used the PYGAME library which already has the inbuilt libraries for the graphics and video pop up screen for the snake game.

## **FEATURES:**

- Snake will try to eat as many apples as possible, but the steps are finite.
- Snake must not bite its own tail to not kill it-self. Snake must avoid its own Collision with tail to score more and continue the game.
- snake grows by 1 unit when eating the apple. The length growth is seen immediately after the snake eats the apple.
- There are four directions for the snake to choose for the next step: left, right, top and bottom. When the snake grows in length, however there mayn't be few directions available to move.
- The game board dimensions are fixed in the size and are said as boundaries /wall. When the snake collides with this boundary it loses the game. Snake must avoid this boundary to win the game.
- After the snake eats the apple, there will be another apple placed randomly within the boundary for the snake to eat as the next target.
- Calculate the snake's state using the 11 values, and if any of the requirements is met, set that value to zero; otherwise, set one.
- The 11 state variables indicated above will be calculated based on the current Head position.
- After obtaining these states, the agent would transmit them to the model, which would then conduct the following action.
- Calculate the reward after performing the next state. The following are the rewards:
  - +10 for eating food
  - 10 for game over
  - Else: 0

- Update the Q value (to be mentioned later) and Form the Model.
- After understanding the method, we must now develop a concept for how to proceed with coding this program.

<b>Actions</b>	
Snake moves up	0
Snake moves right	1
Snake moves down	2
Snake moves left	3
<b>Rewards</b>	
Snake eats an apple	10
Snake comes closer to the apple	1
Snake goes away from the apple	-1
Snake dies (hits his body or the wall)	-100
<b>State</b>	
Apple is above the snake	0 or 1
Apple is on the right of the snake	0 or 1
Apple is below the snake	0 or 1
Apple is on the left of the snake	0 or 1
Obstacle directly above the snake	0 or 1
Obstacle directly on the right	0 or 1
Obstacle directly below the snake	0 or 1
Obstacle directly on the left	0 or 1
Snake direction == up	0 or 1
Snake direction == right	0 or 1
Snake direction == down	0 or 1
Snake direction == left	0 or 1

We are using the deep Q – learning in this project which has the dense neural network with **11** input layers and one dense layer with **256** neurons having the **3**-neuron output layer. We can adjust these parameters to our choice for the better results.

<b>Rewards</b>	
Snake eats an apple	10
Snake comes closer to the apple	1
Snake goes away from the apple	-1
Snake dies (hits his body or the wall)	-100

## ANALYSIS:

I have tried using many different ways and environment execution methods, but I have analyzed that it is better for me to have a separate environment for the execution, and it is better to install all the required modules and start the game.

The training for the snakes a little longer than I expected, and I wanted to have the perfect score soon. So, I used the GPU for the better and fast training of the snake to eat the apple.

## IMPLEMENTATION:

- When the game starts, the value of the Q is just randomly initialized.
- After the q value is initialized, the system receives the current state which is S.
- Now based on the value of the S, the steps for the code are proceeded and then executed in a loop till the snake keeps eating the apple perfectly by its own with no human actions.

---

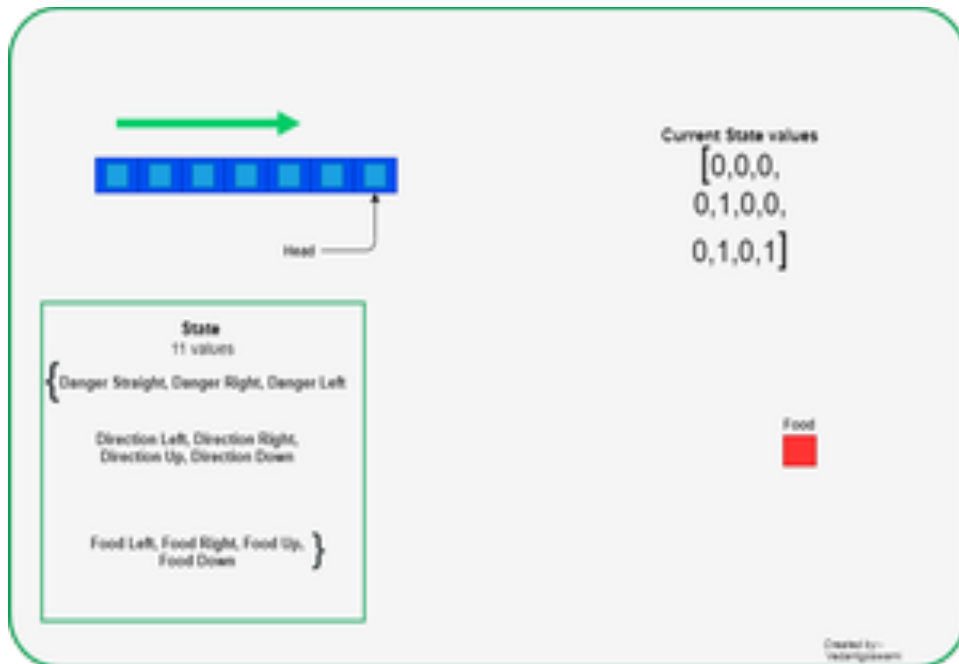
### Algorithm 1 PPO with Clipped Objective

**Input:** initial policy parameters  $\theta_0$ , clipping threshold  $\epsilon$

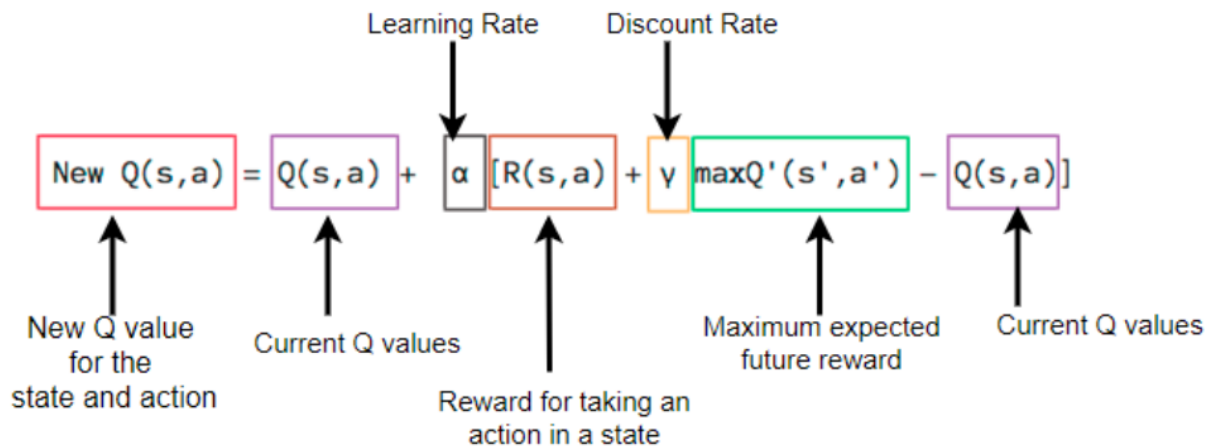
---

- 1: some description
- 2: **for**  $k = 0, 1, 2, 3 \dots$  **do**
- 3:   Getting partial trajectories of Snake  $D_K$  on policy  $\pi_k = \pi(\theta_k)$
- 4:   Estimate advantages  $\hat{A}_t^{\pi_k}$  using advantage estimation algorithm
- 5:   Compute policy update
$$\theta_{k+1} = (L_{\theta_k}^{CLIP}(\theta))$$
- 6:   by taking K steps of minibatch SGD (via Adam)

- Based on the neural network, the next step for the snake is chosen. At the beginning of the game, the snake takes some random action for it to familiarize the environment and move accordingly.



- Once the environment is familiar, the snake starts to take the professional actions and steps.
- When the AI selects the path, that means the AI is well trained for the better actions as the apple is put in the random spots each time and the AI cannot predict the apple spot.
- After the snake eats one apple the value of current state S and Q value is updated according to the bellman equation that we studied in the class.
- After every move of the snake, the value of the original state and the apple spot is updated, and the score is seen on the screen.
- Depending on the original state and the score the AI decides whether the game is ended or it still is in a way to eat the apple.
- For this execution, I have created different classes and names them accordingly for better understanding.



## RESULTS:

I was not successful at the beginning as I am just a beginner in learning the python and deep learning is on a next level of concept for me. I have started working on this project little by little understanding the classes, datasets, pygame, the concept of the agent-environment-model and all the work done by me is taken from a lot of references for better understanding for me. I was successful after learning all the basic stuff and not jumping into the code at the beginning. I have tried some basic code related to the deep learning and then used the implementation that code for my project. So, I basically have done the code in many different parts and assembled them to make the code run successfully. The results for the game at the beginning were like the below and I have changed the colors of the snake, apple and background for my choices later in the project just to keep the output interesting to look at every time I run the code.

python Window zoom Sun Nov 20 10:52 PM

Align and justify text x Login Instagram x (10) WhatsApp x My Drive - Google x SDAI - Google Drive x Downloads/ x snakegamebfs - Ju x

localhost:8889/notebooks/Downloads/snakegamebfs.ipynb

Jupyter snakegamebfs Last Checkpoint: 43 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

+

## AI BASED SNAKE GAME USING Q - LEARNING.

Snake game using deep learning is based on the reinforcement learning. In the snake game the snake eats the apple within the boundaries and not colliding with its own tail.

The rules for the snake game are:

- > The snake will try to eat the apple but the steps for the snake to eat the apple are limited.
- > The snake must not collide with its own tail and boundaries to keep itself alive.
- > The snake grows by 1 unit after eating the apple and then the games becomes even more complex to play.
- > There are 4 directions in the snake game which are left, right, top and bottom.
- > The boundaries are fixed and finite for the snake to move and achieve high score.

I have mounted my drive with the colab.


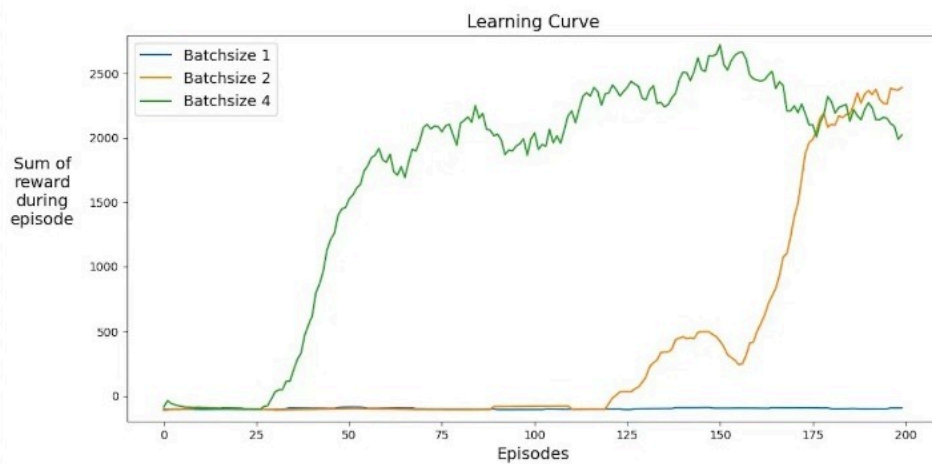
Install the pygame for the graphics and video libraries for the game which are already available in the pygame.

import videodriver for a window screen to popup for the snake game.

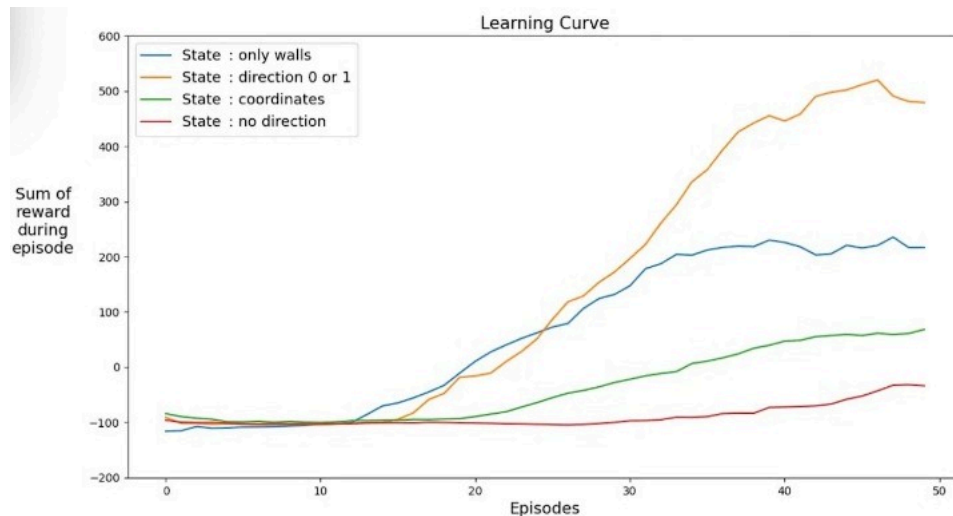
This is the code for the snake game and all the classes defined accordingly.

```
In [1]: !pip install pygame
```

Requirement already satisfied: pygame in /Users/manasakilaru/opt/anaconda3/lib/python3.9/site-packages (2.1.2)

This is the output curve for uh the X axis having the episodes and the Y axis having the sum of reward during the episodes and this is the learning curve for the documentation and the datasets that we've used for the snake game



They bully their bold learning curve shows that the episodes are on the X axis and the sum of reward during episodes is on the Y axis and we can see the different states in the curve and the blue state curve states the only walls yellow state curve tells the direction of the snake is it zero or one and the green state tells the coordinates and when it comes to the red state it tells about the snake we fit is having no direction change and we can see that there are a lot of directional changes for the snake but the zero or one detection is the major change in the curve that we have seen from the above graph.

## PROJECT MANAGEMENT

### IMPLEMENTATION STATUS REPORT

- Work completed:

I have finished mostly around 85% of the work and the code but I would still like to work on the speed of the snake and scoring the maximum with the speed.

- Description

The work was done in the duration of two and half months and I have tried my best to understand the concept and not just jump to the code. This project gave me a better understanding of the reinforcement learning and the usage of different layers to the best use in the snake game code and functionality.

- Responsibility (Task, Person)

SHRIYA KANNOJ – 11507709:



I am a single person team and I have worked as a single person on the code and description to achieve the code and iutput.

- Contributions (members/percentage)

SHRIYA KANNOJ 11507709:

Code – 100%.

Explanation – 100%.

- Work to be completed

I would like to try to make the snake more speed to eat the apple and have a faster movement than the snake output that I have received now.

- Description

This can be achieved by having as many layers in the environment as possible, but I was just working for the perfect snake and perfect score in the increment -1. I would like to implement the same code with a better speed movement in the snake.

- Responsibility (Task, Person)

SHRIYA KANNOJ 11507709:

Code-100%

Description – 100%

- Issues/Concerns

I am very concerned about increasing the layers as the code changes, but I would still be happy to work on the code again if that makes me learn new things about my own code. The code of this project has always made me nervous because there many classes that I have defined and am working on.

## REFERENCE:

- <https://www.geeksforgeeks.org/ai-driven-snake-game-using-deep-q-learning/>
- <https://github.com/vedantgoswami/SnakeGameAI>
- S. Sharma, S. Mishra, N. Deodhar, A. Katageri and P. Sagar, "Solving The Classic Snake Game Using AI," 2019 IEEE Pune Section International Conference (PuneCon), 2019, pp. 1-4, doi: 10.1109/PuneCon46936.2019.9105796.
- Automated Snake Game Solvers via AI Search Algorithms Shu Kong, 80888472, skong2@uci.edu Joan Aguilar Mayans, 87286425, joana1@uci.edu,
- Comparison of Searching Algorithms in AI Against Human-Agent in Snake Game.  
Naga Sai Dattu Appaji
- <https://github.com/kmeng01/pygame-snake>
- <https://github.com/Karthikeyanc2/Autonomous-snake-game-with-A-star-algorithm-in-PYTHON/blob/master/snakeself.py>