A Project Report submitted

for Artificial Intelligence (UCS411)

By

Cheshta Biala 102103545

Varun Thakur 102103547

Yatharth Gautam 102103550

Arshiya Kishore 102103565

Submitted to

Dr. Gourav Jain

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

Jan-May 2023

INTRODUCTION

About the project: Hungry Snake is like the traditional Snake Game which involves guiding a snake to collect food while avoiding running into its own tail or the boundaries of the game board. An AI trained to play Snake would involve training a machine learning algorithm to learn how to navigate the game board and collect food as efficiently as possible.One common approach to training a Snake AI involves using a reinforcement learning algorithm, where the AI is rewarded for collecting food and penalized for crashing into its own tail or the boundaries of the game board. Over time, the AI learns to optimize its behavior to maximize its reward. Overall, training a Snake AI involves a combination of game theory, machine learning, and computer programming to create an algorithm that can play the game at a high level of proficiency. With the right training and optimization, a Snake AI can become a formidable opponent for even the most skilled human players. Another approach to training a Snake AI involves using a neural network to predict the next move based on the current game state. The neural network is trained on a large dataset of game states and corresponding moves, and is then used to make predictions about the optimal move to make in a given situation.

LITERATURE SURVEY

1)

IEEE REFERENCE-

A. J. Almalki and P. Wocjan, "Exploration of Reinforcement Learning to Play Snake Game," *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2019, pp. 377-381, doi: 10.1109/CSCI49370.2019.00073.

SUMMARY-

This paper proposes a snake game with new rules defined which includes poisonous candies which have to be avoided by the snake. Using SARSA algorithm observed with reinforcement learning ,the game is coded in python.Using the SARSA algorithm the efficiency of the game increases because of the quick task performance.The function of the game is observed in reference to the agent's action and the final score.

2)

IEEE REFERENCE-

Z. Wei, D. Wang, M. Zhang, A. -H. Tan, C. Miao and Y. Zhou, "Autonomous Agents in Snake Game via Deep Reinforcement Learning," *2018 IEEE International Conference on Agents (ICA)*, Singapore, 2018, pp. 20-25, doi: 10.1109/AGENTS.2018.8460004.

SUMMARY-

This paper proposes to devise a carefully designed reward mechanism to estimate the immediate rewards based on the environmental changes directly collected from the game as effective learning signals for our autonomous agents.It uses snapshots to make agents learn how to play snake game.Deep Q-network demonstrates its ability to successfully learn complex control policies from rax pixel inputs.

3)

IEEE REFERENCE-

A. Sebastianelli, M. Tipaldi, S. L. Ullo and L. Glielmo, "A Deep Q-Learning based approach applied to the Snake game," *2021 29th Mediterranean Conference on Control and Automation (MED)*, PUGLIA, Italy, 2021, pp. 348-353, doi: 10.1109/MED51440.2021.9480232.

SUMMARY-

This paper proposes a Deep Q-Learning based approach for learning how to play the Snake game.The proposed agent uses sensor data measurements (and not image pixels), which allows the adoption of a more straight-forward (and easier to be trained) deep neural network with respect to other methods, e.g., convolution neural networks.

4)

IEEE REFERENCE-

A. Sawant, S. Shaikh and D. Sharma, "Game AI using Reinforcement Learning," *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*, Vijayawada, India, 2022, pp. 1-6, doi: 10.1109/AISP53593.2022.9760576.

SUMMARY-

This paper proposes the various AI games using Reinforcement Learning consisting of State(Current Environment), Creating Environment and the agent, Reward and action, Deep Neural Network, Algorithm, GUI(Graphical User Interface), Deep Q learning .In Reinforcement Learning we pass a reward, positive or negative, and depending on the action the system took, the algorithm needs to learn what actions can maximize the reward, and which need to be avoided. By using the states as the input, values for actions as the output, and the rewards for adjusting the weights in the right direction, the agent learns to predict the best action for a given state.

5)

IEEE REFERENCE-

M. R. R. Tushar and S. Siddique, "A Memory Efficient Deep Reinforcement Learning Approach For Snake Game Autonomous Agents," 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT), Washington DC, DC, USA, 2022, pp. 1-6, doi: 10.1109/AICT55583.2022.10013603.

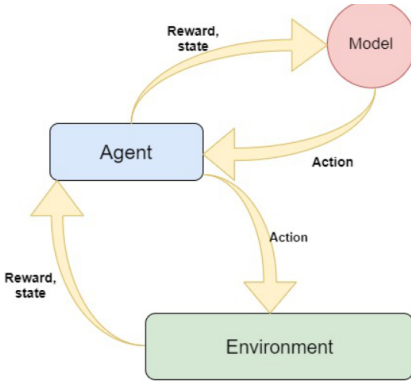
SUMMARY-

This paper proposes that better image preprocessing and constructing a better mechanism for replay buffers can reduce memory consumption on DRL algorithms during training. They have also demonstrated that using this method, the performance of the DRL agent on a lower constraint application is entirely similar, if not better. The combined method with the DQN (with some modification) algorithm to observe the method's effectiveness.

METHODOLOGY

General Working Algorithm of the Snake Game:

1. Draw the playing area with the bounding rectangle, set the counter to zero and display.
2. Draw the snake in a starting position.
3. Draw the food at a random position within the pygame window.
4. Based on AI agent actions, the snake changes its direction.
5. If the snake head is on the food, eat it, increase the score, reward the agent and again randomly place the food item on the screen.
6. If the snake collides with itself or the screen boundary, the snake dies and the game resets.
7. Go back to step 4 and continue doing the same until there is a force quit command by the user.



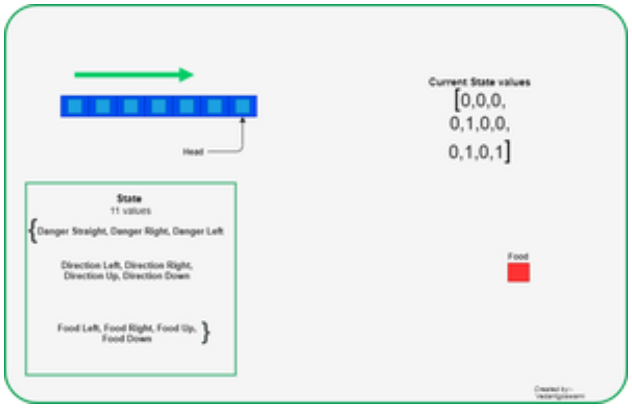
Three Modules for this project:

1. The Environment (the game that we build)
2. The Model (Reinforcement model for move prediction)
3. The Agent (Intermediary between Environment and Model)

Algorithm:

We have snake and food placed on the board randomly.

- Calculate the state of the snake using the 11 values and if any of the conditions is true, then set that value to zero else set one.

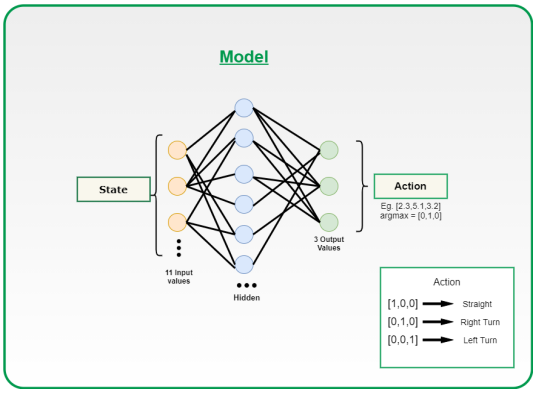


This figure shows how 11 states are defined

Based on the current Head position agent will calculate the 11 state values as described above.

- After getting these states, the agent would pass this to the model and get the next move to perform.
- After executing the next state calculate the reward. Rewards are defined as below:
    - ***Eat food : +10***
    - ***Game Over : -10***
    - ***Else : 0***
- Update the Q value and Train the Model.
- After analyzing the algorithm now we have to build the idea to proceed with coding this algorithm.

<u>The Model:</u>

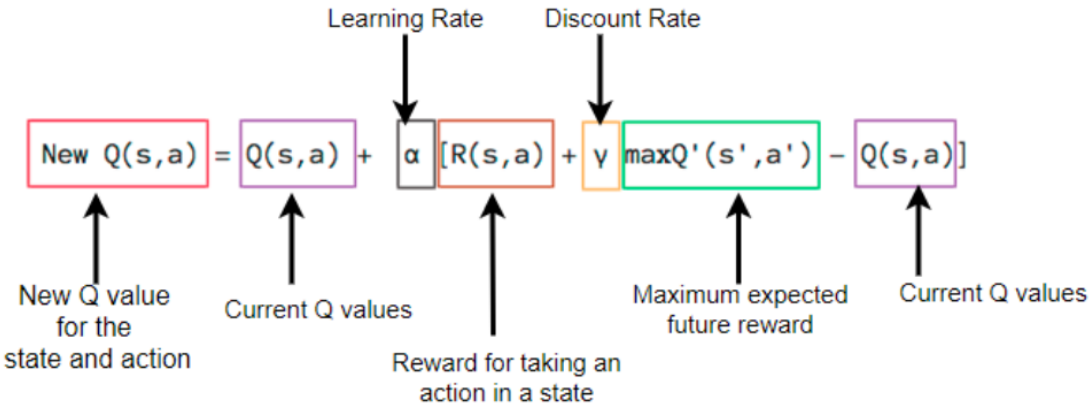


The model is designed using Pytorch or TensorFlow.

We are using a Dense neural network with an input layer of size 11 and one dense layer with 256 neurons and an output of 3 neurons. We can tweak these hyper parameters to get the best results.

HOW DOES THE MODEL WORKS ?

- The game starts, and the Q-value is randomly initialized.
- The system gets the current state s.
- Based on s, it executes an action, randomly or based on its neural network. During the first phase of the training, the system often chooses random actions to maximize exploration. Later on, the system relies more and more on its neural network.
- When the AI chooses and performs the action, the environment gives a reward. Then, the agent reaches the new state and updates its Q-value according to the Bellman equation.
- Also, for each move, it stores the original state, the action, the state reached after performing that action, the reward obtained, and whether the game ended or not. This data is later sampled to train the neural network. This operation is called Replay Memory.
- These last two operations are repeated until a certain condition is met (example: the game ends)



The heart of this project is the model that we are going to train because the correctness of the move that the snake would play will all depend on the quality of the model that has been built.

So here is the algorithm for the same.

1. Creating a class named Lineaar_Qnet for initializing the linear neural network.
2. The function forward is used to take input and pass it through the neural network and apply relu activation and give the output back.
3. The save function is used to save the trained model for future use.
4. Initialising QTrainer class
    - Setting the learning rate for the optimizer.
    - Gamma value is the discount rate used in the Bellman equation.
    - Initializing the Adam optimizer for updation of weight and biases.
    - Criterion is the Mean squared loss function.
5. Train_step function
    - As PyTorch works only on tensors, we are converting all the input to tensors.
    - As discussed above we had a short memory training then we would only pass one value of state, action, reward, move so we needed to convert them into a vector, so we had used an unsqueezed function.
    - Get the state from the model and calculate the new Q value using the below formula:

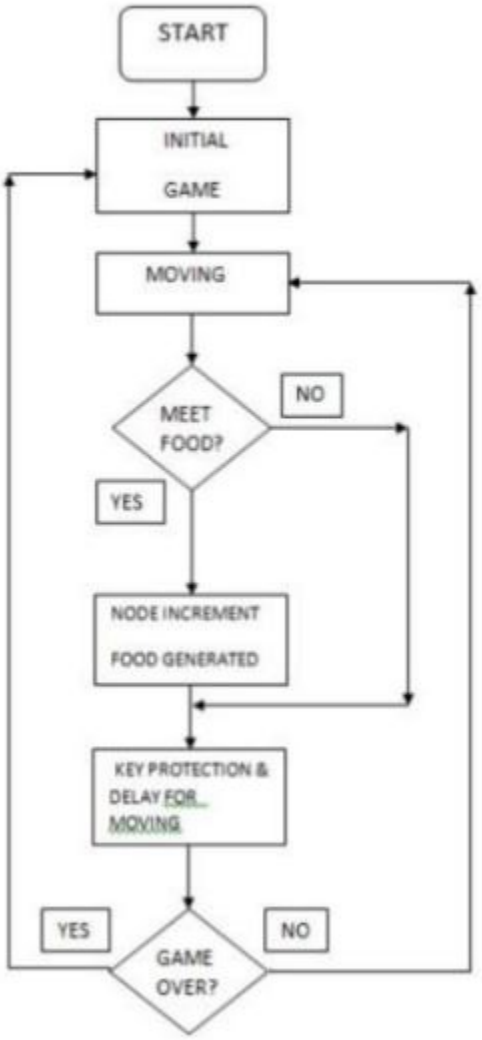
      Q_new = reward + gamma * max(next_predicted Qvalue)

    - calculate the mean squared error between the new Q value and previous Q value and backpropagate that loss for weight updation.

The Agent:

1. Get the current state of the snake from the environment.
2. Call model for getting the next state of the snake.
3. Play the step predicted by the model in the environment.
4. Store the current state, move performed, and the reward.
5. Train the model based on the move performed and the reward obtained by the Environment. (Training short memory)
6. If the game ends due to hitting a wall or body then train the model based on all the moves performed till now and reset the environment. (Training Long memory). Training in a batch size of 1000.

Flowchart:


START
INITIAL
GAME
MOVING
MEET
FOOD?
NO
YES
NODE INCREMENT
FOOD GENERATED
KEY PROTECTION &
DELAY FOR
MOVING
YES
GAME
OVER?
NO

RESULTS AND FUTURE SCOPE

The experimental results showed that the agent has overcome the baseline Deep Q-Learning Network (DQN) model and surpassed human-level performance in terms of both game scores and survival time in the Snake game.

In this project, we have used a simple application. This project will be able to be implemented in future after making some changes and modifications.

- It can be made with good graphics.
- We can add more options like Top scores and Player Profile.
- We can add a multiplayer option.
- We could add a poisonous food material and snake has to learn to avoid it.
- Finally, AI could be used to create more immersive and engaging game environments by incorporating natural language processing and computer vision technologies. For example, AI could be used to generate realistic snake behaviors, or to create dynamic game environments that adapt to the player's actions and preferences.
- We shall apply our refined DQN model to other similar application scenarios with continuous reallocated targets (such as more number of frogs) and gradually increasing constraints (such as obstacles).