Snake Game using Deep Learning

Atharva Patil
Department of Computer Science
University at Buffalo
Buffalo, NY, 14221
atharvap@buffalo.edu

Karan Shah
Department of Computer Science
University at Buffalo
Buffalo, NY, 14221
kshah24@buffalo.edu

Krisha Sheth
Department of Computer Science
University at Buffalo
Buffalo, NY, 14221
krishahi@buffalo.edu

Palash Anjania
Department of Computer Science
University at Buffalo
Buffalo, NY, 14221
palashpi@buffalo.edu

Saloni Contractor
Department of Computer Science
University at Buffalo
Buffalo, NY, 14221
scontrac@buffalo.edu

Abstract

Snake is a famous game that was invented in an arcade in 1976. It derives its name from the concept where users control a line which follows a target pixel(food). As the line keeps moving, it leaves behind a trail which resembles a snake. The goal of the game is to reach the target pixel, which changes position every time you reach it, without hitting the b[1]oundary or yourself. The line keeps getting longer constantly as you reach the target pixel. Our goal in this project is to train the machine in a way that it can play the aforementioned snake game without any user interaction. For this purpose we propose a Reinforcement Learning algorithm. Few algorithms we will explore are Deep Q-Learning and Actor Critic method. In this project, we emphasize implementing the Actor-Critic model within the "Snake Classics" game. The Actor-Critic method outperforms existing models by interlinking value and policy-based reinforcement learning in a single algorithm. In the Actor-Critic method, the Actor is a policy-based model that selects the action and maps states to action probabilities, whereas the Critic is a value-based model that evaluates the state by mapping the input states to Q-values. The model can work with continuous action space problems offering a better convergence rate and reducing learning time. Further, we aim to build a comparative analysis of the Actor-Critic method with Deep Q-Learning, BFS, A* Algorithm, and Hamiltonian Cycle methods respectively. The results are shown in the github repository mentioned in the footnotes[1].

1. Introduction

In recent years, one of the highest challenges in the field of artificial intelligence has been the creation of systems capable of learning how to play classic games. The snake game is a video game genre in which the player maneuvers a growing line that eventually becomes its own principal obstacle. Blockade, an arcade game, inspired its creation in 1976. On a bounded plane, the player controls a dot, square, or object. It moves forward, leaving a trail behind it that resembles a snake. A lone player tries to consume stuff by running the snake's head into them. Because each item eaten lengthens the snake, avoiding collisions with it becomes increasingly difficult. When the snake runs into the screen boundary, another obstacle, or itself, the player loses. Non-ML techniques, Genetic Algorithms, and Reinforcement Learning are the three broad groups of approaches to an AI Snake agent. Reinforcement learning is a rapidly developing and interesting topic of artificial intelligence. At its most basic level, reinforcement learning entails an agent, an

[1] https://github.com/KaranS9271/AI_Snake_Game

environment, a set of behaviors that the agent can perform, and a reward function that rewards good behavior and punishes bad behavior. The agent adjusts its parameters as it explores the environment in order to maximize its own predicted return. The agent in the case of Snake is, of course, the snake. The environment is the NxN board (with many possible states of this environment depending on where the food and the snake are located). The environment is the NxN board (with many possible states of this environment depending on where the food and the snake are located)[1].

We present a Reinforcement Learning algorithm for this. Deep Q-Learning and the Actor Critic Method are two algorithms we'll look into. The implementation of the Actor-Critic paradigm within the "Snake Classics" game is the focus of this project. This report emphasizes building an Actor Critic method to overcome the drawbacks of traditional Deep Q-learning methodology. The Actor Critic method aims to develop a system comprising Markov Chain along with Deep Q-learning model where Deep Q-Learning model takes the all possible paths available to reach the apple and the Markov Chain rule further finds probability for all the paths derived by the DQL model to obtain the most optimized part to reach the apple (goal)[2].

By combining value and policy-based reinforcement learning into a single algorithm, the Actor-Critic technique outperforms prior models. The Actor is a policy-based model that chooses the action and maps states to action probabilities in the Actor-Critic approach, whereas the Critic is a value-based model that evaluates the state by mapping input states to Q-values. The model can solve continuous action space problems with a faster convergence rate and less time spent learning. In addition, we want to compare the Actor-Critical method against non-ML methodologies such as BFS, A* Algorithm, and Hamiltonian Cycle. The following is a breakdown of the paper's structure[12].

2. Hamiltonian Cycle

The hamiltonian cycle is one of the oldest theories in graph theory. It was invented in 1857 as a puzzle game by Irish mathematician William Rowan Hamilton. A Hamiltonian cycle or a Hamiltonian circuit is a graph cycle where every node in the graph is visited exactly once. The Hamiltonian Cycle is not a loop but for our case of the snake game it follows a loop, it starts and ends at the same vertex. The grid is treated as a graph with every cell as a node or vertex of the graph[11]. The hamiltonian cycle uses a backtracking algorithm where it back tracks the path and finds a cycle to visit all the nodes of the graph. For the snake game it follows a specific path in the grid irrespective of where the food or snack is in the grid. As it will cover all the nodes of the graph (cells in the grid) the food will be eaten eventually. It is very clear what the issue is in this algorithm. It takes the same path irrespective of the position of the food so in most cases it takes steps that are not required. This increases the number of moves taken by the snakes resulting in a lower score and more time. As it takes more time it uses more space in the memory and hence has a bad space complexity. It just does not have bad space complexity but also bad time complexity due to the extra unnecessary moves that it takes. The only advantage of this approach is that it will always most definitely complete the game. As it follows a cycle in the graph it will never get boxed or cross itself or hit the boundaries of the box, thus the snake never gets eliminated resulting in a win every time. Also it may seem that it takes the same amount of time/moves every time the game is played but that is not correct. Sometimes the snake may eat more than one food in one loop but sometimes it may be just one food in one loop and as the food is generated randomly it will be different in every game. After a lot of run throughs of the game the above statement was proved to be true. This also shows the speciality of the snake game that it has a few factors but still is very random. The hamiltonian is a very basic method to understand how game playing can be developed more. We believe it provides the basis to understand what and how the snake can be traversed through the grid for a more efficient run.

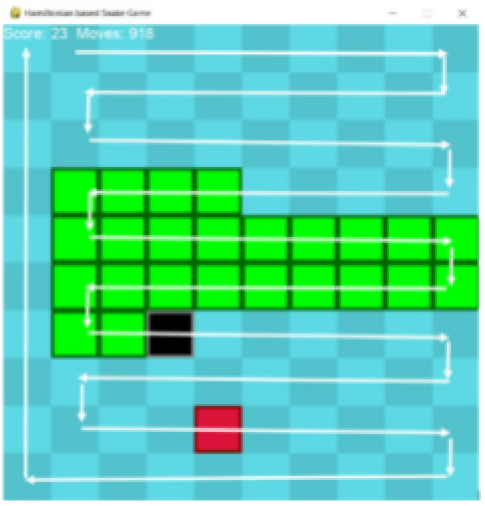

Fig. 1: Hamiltonian Path in the grid

3. Breadth First Search

Breadth First Search is an algorithm that does not require any introduction. It is one of the most common algorithms in graph theory. Breadth First Search works using a queue data structure. It is used to search all vertices in a graph in a breadth-forward algorithm. BFS works by classifying all the nodes into 2 categories, visited and unvisited. The algorithm starts with adding all the nodes to a queue and then removing them from the queue and adding them to the visited list once the node has been visited. The Breadth First Search algorithm works recursively by visiting nodes and adding them to the visited list. In our case, BFS is used because the grid where the snake game runs, or plays is treated like a graph where every cell is a node. These cells are connected like a graph with an edge from a one cell to all its neighboring cells. The Breadth First Search algorithm uses this graph to traverse through the nodes to reach the destination. The BFS algorithm finds the shortest path between the head and the food. The Breadth First Search Algorithm is accurate and makes sure that the snake finishes its game. It finds the paths even in an almost filled grid without boxing the snake. The algorithm gives a very effective final solution, i.e. helps the snake end the game at a win. The aim here was to check the empty boxes and decide which and how many empty boxes to consider so that the snake does not box itself. The BFS algorithm moves in breadth-forward way so it explores all available paths, now one might say that Depth First Search would be a better option as it will find dead end first but Depth Search first is very space expansive, it takes a lot of space and hence the Breadth First Search algorithm proves to be a better algorithm of the two. The diagram above shows the path taken by the snake using the BFS algorithm. The tail is generated after the last food is eaten and then the snake follows the following path as it is the shortest path found by the BFS algorithm.

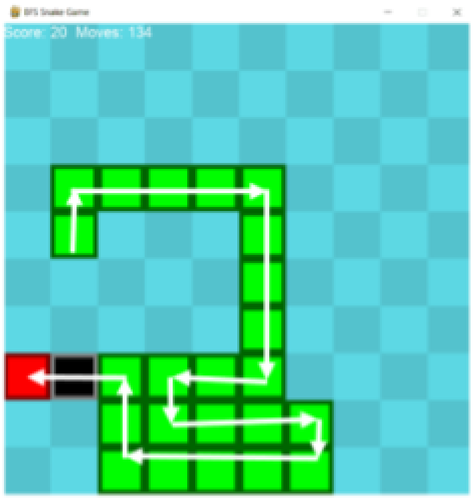

Fig. 2: path of snake in BFS

4. A-star

It is a graph algorithm which is one of the most widely used techniques in pathfinding problems. It can be said as an extension of Dijkstra's algorithm, the difference is we use the heuristic function in the A-star method. Dijkstra's algorithm finds the shortest path tree from a point to all possible goals. In the A-star algorithm we find the shortest path from a specified point to a specific goal. In our case, we wanted the snake to ideally find the best path to its reward and then move onto the next reward, hence this algorithm was suited to our problem definition[7].

The A Star algorithm works on an ‘f’ value in order to decide the path of traversal. The ‘f’ value is calculated by adding ‘h’ and ‘g’. Here ‘g’ is the cost to move from a starting point to a given point on the grid. And ‘h’ is the estimated cost from that point to the final destination. To calculate ‘h’ value we can use Manhattan, Euclidean or diagonal distances. The A-star algorithm is an optimal search algorithm in terms of heuristics. It is also the most optimal non-ML algorithm for finding the shortest path in a graph. The heuristics and design of this algorithm provides for finding solutions to complex problems. The disadvantage of this algorithm is that its accuracy and performance depends on the heuristic function used in the algorithm, using a wrong heuristic function could lead to disastrous results, sometimes even the wrong path can be taken[9]. The A-star algorithm is said to be one of the best algorithms in the path searching algorithm and rightly it has its reputation because when the A-star algorithm finds a path in the grid it moves diagonally and it is mathematically proven that a diagonal is less distance than going along the sides. The A-star algorithm gives good results to reach the food from the source, better than the previous two algorithms mentioned in this paper.

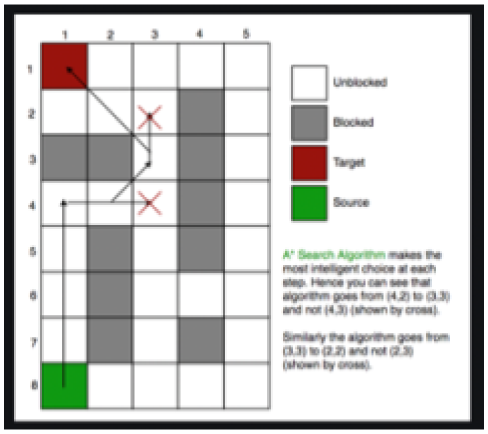
1
2
3
4
5
1
2
3
4
5
6
7
8
Unblocked
Blocked
Target
Source
A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).
Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

Fig. 3: A star algorithm working

5. Deep Q-Learning

This approach is based on the Q-learning algorithm. Deep Q-learning was introduced because of a flaw in Q-learning which is that when we increase the number of states and action pairs, then it becomes very difficult to update the Q-table and therefore Q-learning is only suitable for small environments. Also the time required to update so many states in the Q-table is enormous. To counter this disadvantage we use neural networks in Deep Q-learning. The fundamental difference between Q-learning and Deep Q-learning is that in Q-learning there is a state and action pair input which will output a Q-value and update the Q-table, while in Deep Q-learning we input only the state into the neural network and the output is a Q-value for each action. We use a variety of hyperparameters in this technique, the most important 2 parameters are: learning rate and discount factor[8]. The learning rate is used to control the rate of updating procedures. The discount has a value from 0 to 1 and implies how important is the future reward. The agent is a deep neural network representing the agent policy π(·|s) it determines what is the best action to be executed at each state. It learns by performing a number of training episodes. With every time action is taken, the environment provides a reward. Such rewards can be negative or positive based on a particular state. The goal of the training process is to learn which action maximizes the expected total reward from each state. Deep Q-Learning works by initializing all Q-values e.g. with zeros. Choose an action "action" in current states "current" based on the current best Q-value. Then perform this action "action" and observe the outcome (new state). Then measure the reward "reward". Updating Q with an updated formula that is called Bellman Equation. Repeat the above step until the learning is stabilized[10].

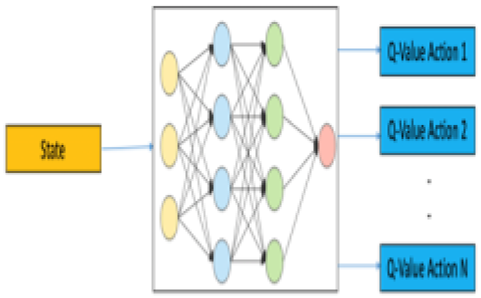

Fig. 4: Deep Q learning model

6. Actor Critic Algorithm

The Reinforcement Learning algorithms can be divided into three groups: actor-only, critic-only and actor-critic methods where actor and critic are for the policy and value functions, respectively. Policy is a function that indicates action to take in a certain state in an environment. Main goal of an agent is to find a policy that maximizes the total accumulated rewards which is called return. Value function represents this estimated return. Critic-only methods, such as Q-Learning and SARSA use state action value function. Policy in critic-only methods is by selecting greedy actions (actions for which value function gives highest return) to do this one needs optimization procedure in every state encountered to find action which leads to optimal value. Actor-only method, work with parameterized family of policies and optimize cost defined directly over parameter space of policy. Advantage of actor-only over critic-only is that it allows policy to generate actions in complete continuous space of policy[5]. It has a strong convergence property, naturally inherited from gradient descent methods. The drawback is that the estimated gradient may have a large variance which leads to slow learning. Actor-critic methods combine the advantages of actor-only and critic-only methods. Actor-only method is capable of producing continuous actions, while the large variance in policy gradient is countered by adding criticism. Critics evaluate the current policy prescribed by the actor. Actor-critic preserves convergence of policy gradient methods. There are two types of actor-critic algorithms stated in the paper: standard gradient actor-critic algorithm and natural gradient actor-critic algorithm. Standard gradient descent is useful for cost functions that have a single minimum and whose gradients are isotropic in magnitude from its minimum. But in practice, these two properties almost never hold true[6]. Non-covariance is a drawback of standard gradient descent. In natural gradient, it incorporates knowledge about the curvature of space into the gradient. When selecting an algorithm, first consider the type of control policy. If control policy produces actions in a continuous space, critic-only is no longer an option. Critic-only is used only when the controller needs to generate actions in a small, finite space. Using critic-only overcomes the problem of high variance. If the

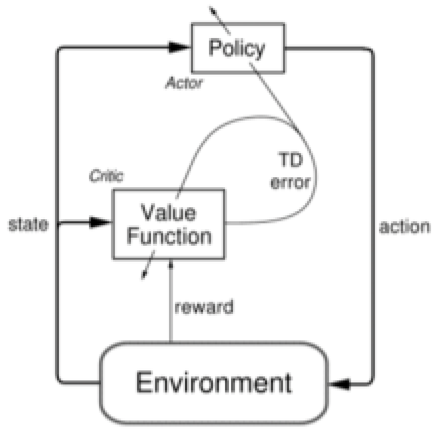

Fig. 5: Actor Critic Working

problem is modeled by stationary MDP(Markov Decision Processes) with continuous state and action space, actor-critic provides with lower variance then actor-only method. Actor-only is used when one has a fast changing non-stationary environment. Once a choice for an actor-critic has been made, choosing the right feature is another task. First parameterization for the actor is chosen then compatible features for the critic are derived.

7. Experiments and Results

Snake Game based on Hamiltonian Cycle

In the below game, when using Hamiltonian cycle to complete the game then, Hamoiltonian cycle successfully completes the entire game following the fixed part as shown in figure 6.

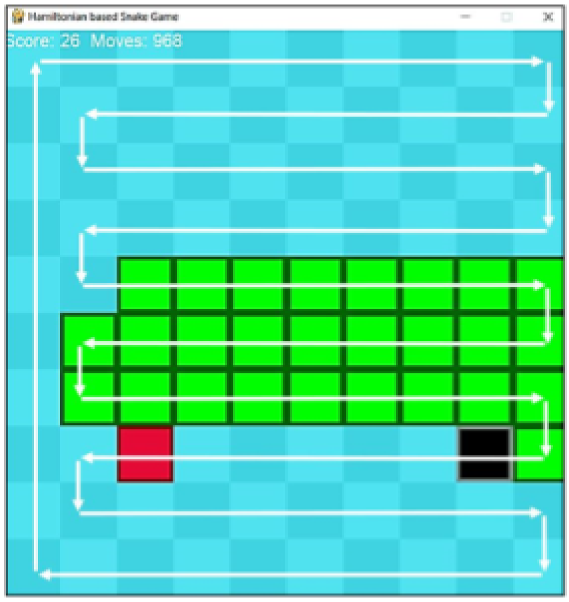

Fig. 6: Hamiltonian Cycle

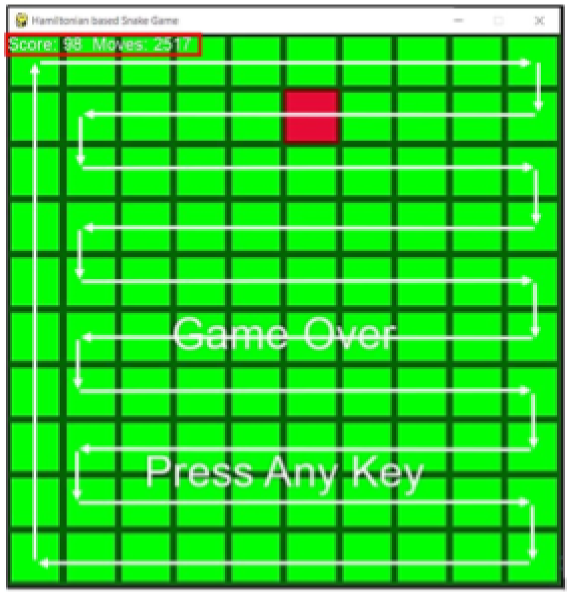

Fig. 7: Completed game in Hamiltonian

The snake completes the game using this approach because it does not make decisions based on where the food is but follows a fixed path, however the number of moves it takes in order to achieve this is considerably large. We can see in Figure 7, that the snake takes 2517 moves to complete the entire game. The major reason the snake takes so many moves is because it goes on following the same path again and again without choosing an effective part to eat the apple.

Snake Game based on BFS

Snake game using the BFS Algorithm completes the entire game. It creates multiple virtual snakes for exploring every path at every breadth. This results in consuming a lot of memory for creating many virtual snakes. Secondly, when the snake is at the same breadth the snake selects a non-optimized path and this results in the snake taking a lot of moves to complete the snake game.

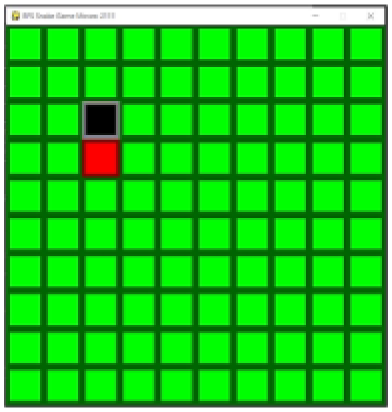

Fig. 8: Completed Game in BFS

Snake Game based on A-Star and A-Star2

The Snake game using A* algorithm aims to find the most optimum part towards the goal i.e. the apple in this scenario, but in this the snake game is not completed using this algorithm as it sometimes collides into itself or into walls while searching optimum path. Secondly as A* algorithm explores all the paths at once it consumes a lot of resources. In this algorithm the snake can move in diagonally it does not in A-star2.

The Snake game using A*2 has a different heuristics from snake game using A* algorithm. Secondly the snake in this game can't move in a diagonal way.

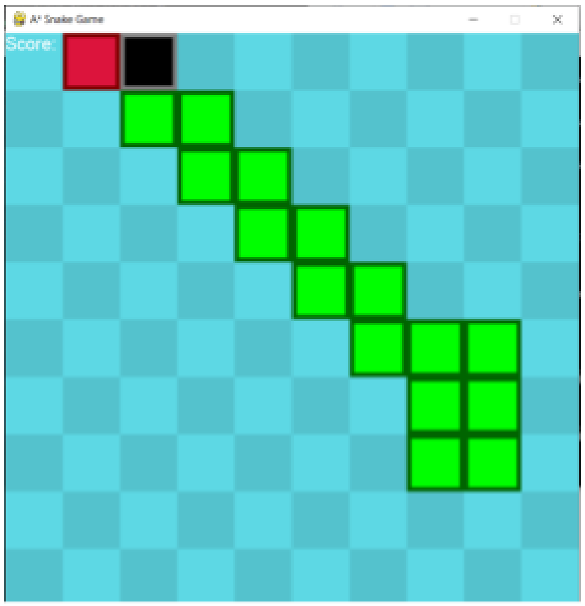

Fig. 9: A-star Snake game

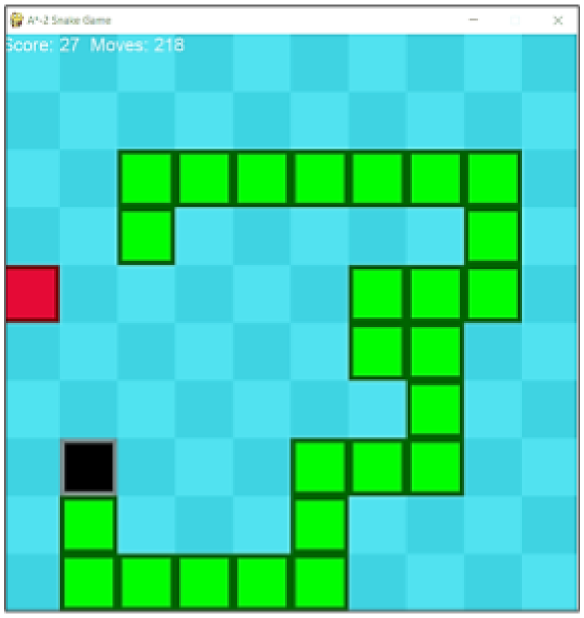

Fig. 10: A-Star2 game

Snake Game based on Deep Q-Learning

The Deep Q-Learning Reinforcement Learning Algorithm attempts to complete the entire game but as the algorithm tends to take random moves for finding the way to the apple initially, it takes a long time to train the model. Secondly, due to the random moves there is 63% chances that the snake can die either colliding with the walls or by itself. We observed that after training the model for a sufficient number of games the score starts becoming saturated and then the model doesn't show improvements. Further, the snake tends to iterate in a concentric circular direction for finding the path due to random moves taken initially.

The Model in Figure 12 shows that the model starts getting saturated after 250 game moves and then doesn't show much improvements in the score above 24.

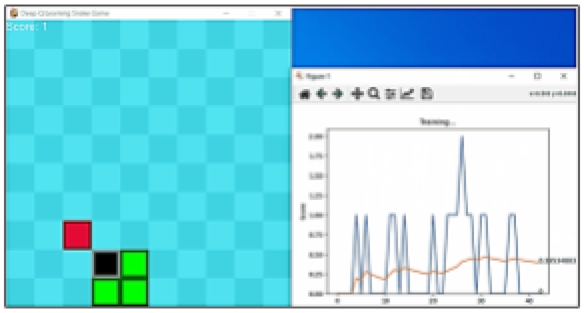

Fig. 11: Initial training of Deep Q learning

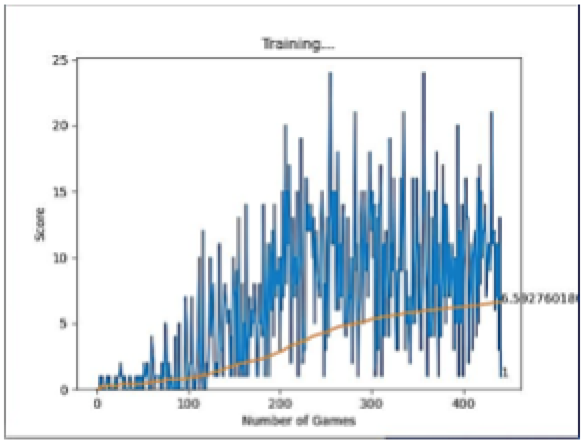

Fig. 12: Saturation in Deep Q learning model

Snake Game based on Actor Critic

The Snake Game using Actor Critic method aims to collaborate 2 algorithms i.e. the Markov chain and Deep Q-Learning algorithms. We have trained the model for 100000 examples with maximum steps to be

500 steps and learning rate equal to 0.003. The snake game using the Actor Critic method completes the entire game and is more optimized, completing the algorithm in fewer moves.
Figure 13. shows snake game with 4 * 4 grid in which the snake completes the game within 78 moves and Figure 14. Shows a Actor Critic snake game for 10 * 10 grid sized board.

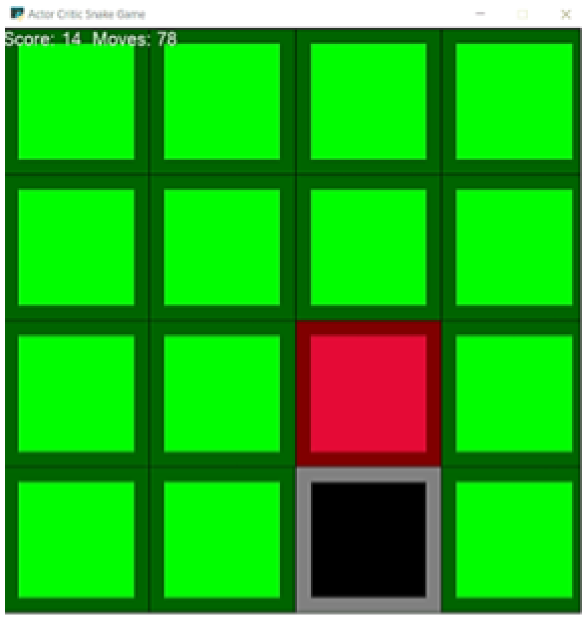

Fig. 13: Actor critic in 4x4 grid

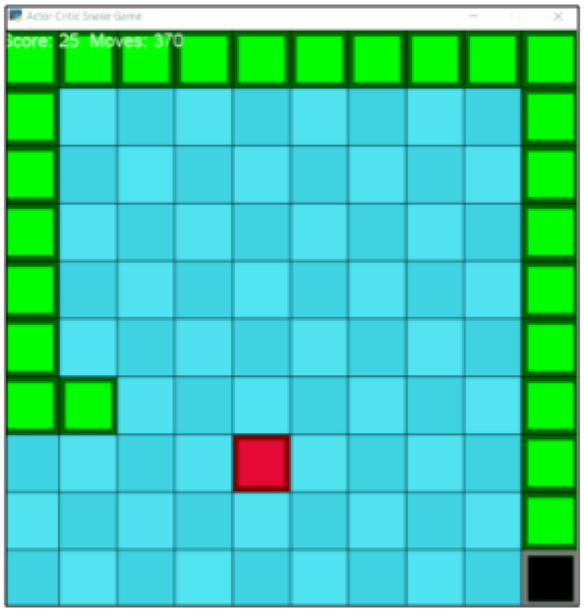

Fig. 14: Actor Critic in a 10x10 grid

8. Future Scope
The future scope of the project emphasizes on improving the policy used within the actor critic model for reducing the data and training time required for training the model. The actor critic method aims to bring both into a single model by using Deep Q-learning for finding out all possible moves available further A policy based approach to get the best moves out of all the moves found. The future scope involves building the model using Policy gradient actor critic method and comparing it with existing model. Secondly we can also build a model incorporating openCV, Google Media pipe can be used to detect the hand and then based on the movements of the fingers we can control the movement of the snake.

9. Conclusion
We can all agree that Artificial Intelligence has made a positive impact on our lives. It has dramatically transformed the technological environment, benefiting millions of individuals. In this project, we emphasize on building a perfect AI snake game that finds the shortest path to the food. Where food being the goal state, the snake gets rewarded for eating the food. Here, we have made a comparative analysis where we compare the performance of different algorithms such as Hamiltonian, BFS, A*, Deep Q-Learning and Actor Critic method respectively. We observed that Hamiltonian Cycle and BFS algorithm takes more moves but completes the game, where BFS algorithm is more resource consuming. A* Algorithm fails to complete the entire game however it can find the optimized path between the snake's current state and the food. Deep Q-Learning is a Reinforcement Learning Algorithm that tends to perform well but fails in successfully concluding the game. A lot of time is consumed in training the model. Secondly the model also doesn’t provide optimized paths as it initially takes random moves to reach the goal state.
Actor Critic Model outperforms as compared to the aforementioned algorithms as it not only takes less time but also takes less data for training the model. Secondly it takes optimized paths to read the goal state as the model comprises both Policy based as well as Value based Reinforcement Learning techniques. Hence, we conclude that the Actor Critic method surpasses most of the existing algorithms as it consists of a Deep Learning probabilistic model on top of Deep Q-Learning model that evaluates the decision taken by Deep Q-Learning model and only considers the path providing the optimal solution using probability.

10. Acknowledgement

We would like to express our gratitude to Prof. Changyou Chen for approving our project proposal and mentoring us by providing constructive input. We thank him for his competent help throughout the process. We would also like to thank the Teaching Assistants for being there to solve our doubts and provide constant encouragement throughout the process.

We'd also like to express our gratitude to the University at Buffalo's Department of Computer Science and Engineering for giving us this platform and invaluable opportunity for the development of this project.

Finally, We would like to take this time to thank everyone involved in the project for their hard work and dedication. Extending a warm thanks to Atharva Patil, Karan Shah, Krisha Sheth, Palash Anjania and Saloni Contractor for the successful development and completion of this project.

References

[1] Anton Finnson and Victor Molno. Deep Reinforcement Learning for Snake.
[2] Ruikang Zhang and Ruikai Cai. Train a snake with reinforcement learning algorithms
[3] https://www.youtube.com/watch?v=i0Pkgtbh1xw
[4] https://www.youtube.com/watch?v=w_3mmm0P0j8
[5] https://towardsdatascience.com/advantage-actor-critic-tutorial-mina2c-7a3249962fc8
[6] A survey of Actor-Critic reinforcement learning
[7] https://www.geeksforgeeks.org/a-search-algorithm
[8] https://www.geeksforgeeks.org/deep-q-learning/
[9] https://iq.opengenus.org/a-search/
[10] https://www.assemblyai.com/blog/reinforcement-learning-with-deep-q-learning-explained/
[11] Hamiltonian Cycle: Simple Definition and Example - Statistics How To
[12] https://towardsdatascience.com/training-a-snake-game-ai-a-literature-review-1cdddcd1862f