The Snake Game using AI. A Detailed Study.

Kenneth Xavier Dsilva
ID: 10600644
10600644@mydbs.ie

Handup Date: 12/08/2022

1 Literature Review.

The birth of Artificial Intelligence was in a workshop held on the campus of Darthmouth College in 1956 (Bruderer 2016). However, many believe that it began with the idea of Alan Turing in the late 1940s where he was determined to build a machine to solve the problems caused by machines.

Reinforcement learning on the other hand has no such relevant history clearly mentioning its addition to the field of AI. Some do say that the introduction of Markov Decision Process is the starting point for Reinforcement Learning and its application(Kaelbling, Littman, and Moore 1996).

The introduction of Reinforcement learning in games (Szita 2012) has had a huge impact in the gaming and development communities. From training agents to play and defeat world-class chess players like Garry Kasparov and the AI AlphaGo defeating the worlds top 5 players. It hasn't just pushed the boundaries of the aspects of gaming, but has redefined the entire aspect of gaming as well. In this report, I present a minor contribution in the works and advancements of AI designed games. My proposed method stems from pre-trained models, however implementing a mathematical model to improve my agent to play the all time classic game of Snake.

2 Our Approach.

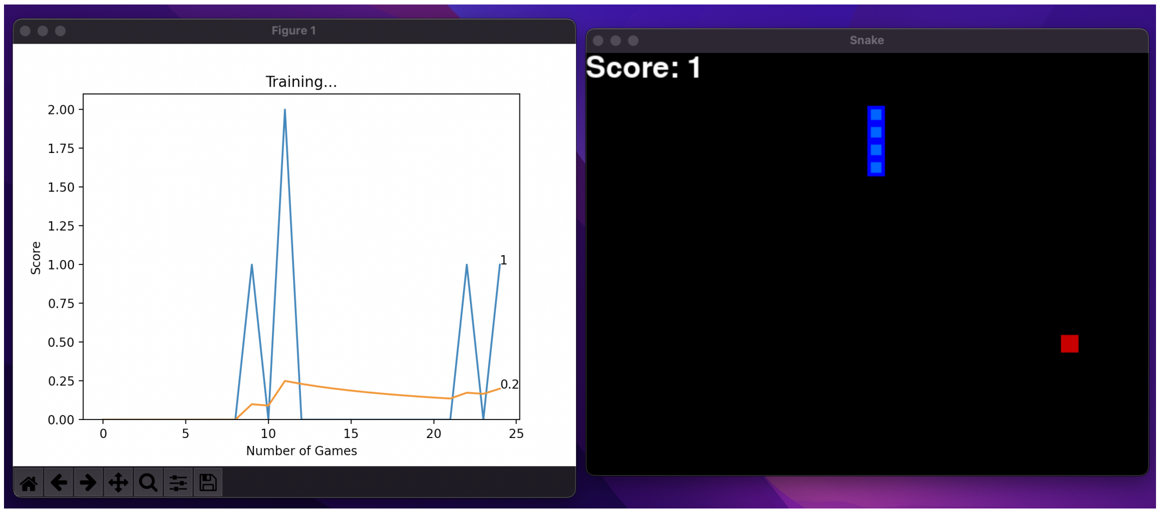
There are numerous repositories on GitHub and even more papers available online on this topic of training an AI Agent to play the snake game, so this implementation being an original work isn't the main idea. However, after a deep research I can conclude that the addition of a Mathematical problem solving model along with RNN and LSTM i can state that my approach is fairly new and innovative.

My approach i have divided in 3 major parts, a pre-implemented Base Model where a basic training for the snake game to learn and understand its environment is carried out. This output is then pickled and saved as I do not want to loose over 3 hours of training data progress. This is followed by a mathematical model implementation of the Hamiltonian Cycle where the snake needs not learn the environment or make decisions but just follow the Hamiltonian path and easily win the game, however without taking any shortcuts or decisions it takes the snake to complete the game in over 50 minutes or more compared to a human player. Hence, the final and the Model which introduces RNN and LSTM to make decisions. to cut between Hamiltonian cycles and also find the shortest paths to the food.

The following sections describe it in detail.

2.1 Base Model.

The base model was adopted by Vedant Goswami who has a descriptive post on the Geeks for Geeks platform (Goswami 2022). Adopting his idea I have created an Agent that trains to take necessary steps from past experiences of the State it was in and the Reward it received upon the actions it took in the past. The GUI is shown below.



GUI for Base Model With Training.

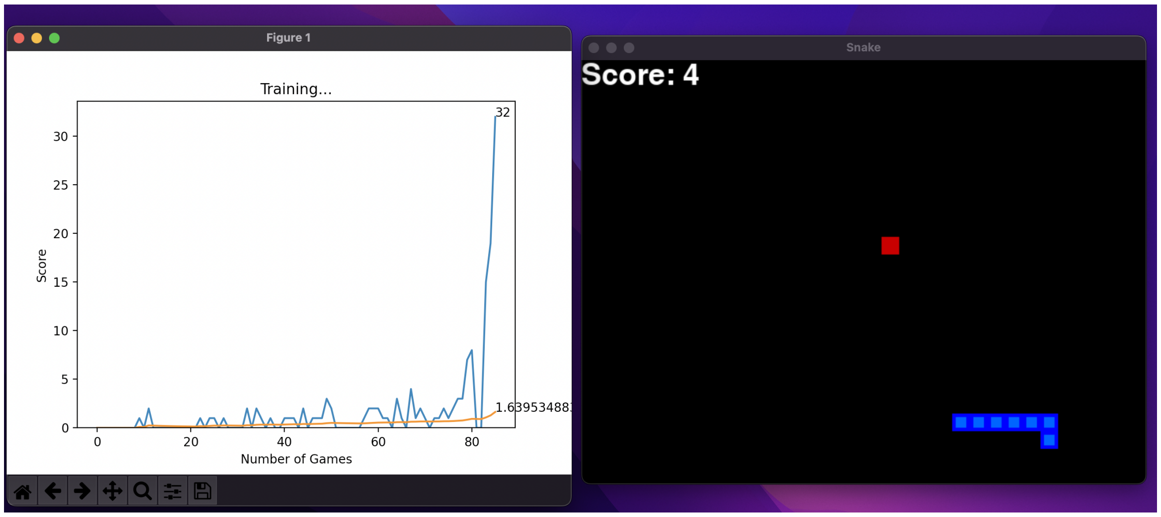
We defined the moves the agent can take along with the danger moves when the snake is close to the edge of the play area. The rewards distributed are simple a +10 for every food consumed and -10 for every death.

The problem of an infinite loop of the snake without progress in a certain direction is taken care by a play_timmer funtion where if the snake doesn't consume its food for 10 seconds it ends the game with a penalty of a kill reward(-10) for the agent hence avoiding any un-necessary loops or steps.

However, there are major flaws to this algorithm:

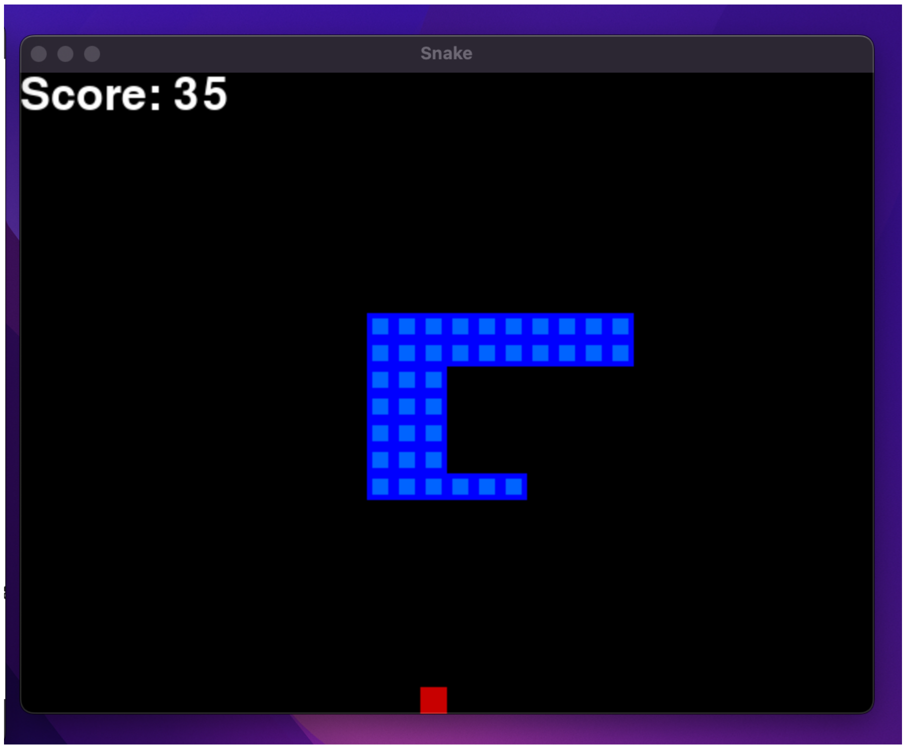
1. No Shortest Path Finding Adjustment: Although the Snake avoids the unnecessary moves and infinite loops, finding the shortest path to the food is not possible as no setup to do so is present.

2. Training time: The time taken to train the snake is too high, it takes around 25 games for the snake to just reach the food and get a score above 5 consecutive wins and 80+ games to reach a score above 30 with an average game score barely above 1 win per game!



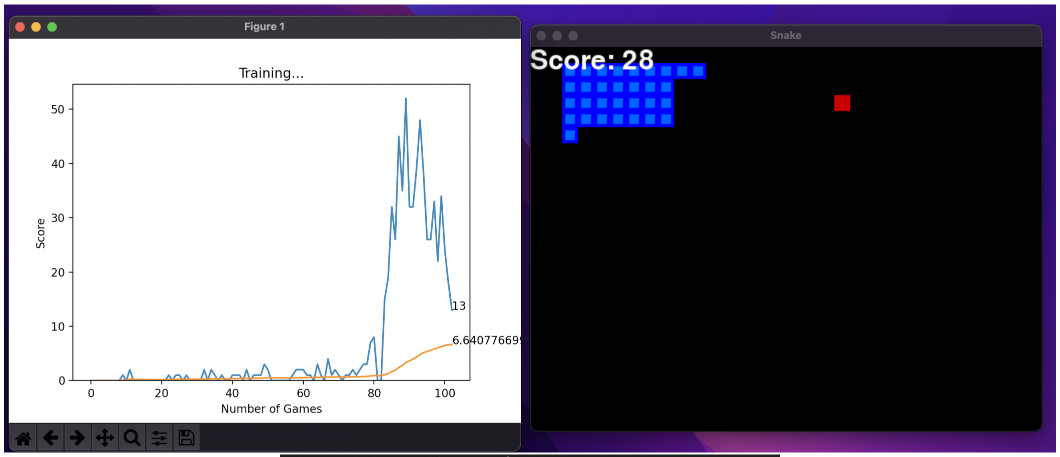
83 Games to reach the 1st high score of 32 Points.

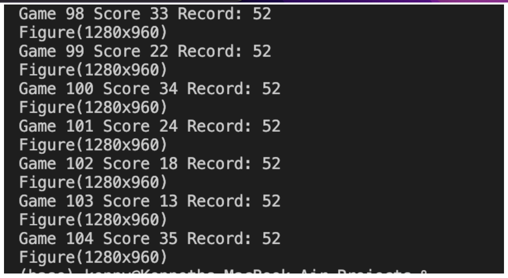
3. Large snake Self Consumption: The greatest issue of post training the snake to grow is that it ends up eating or trapping itself. Even though the penalty of -10 is induced, it becomes convenient for the snake to eat itself after gaining multiple positive rewards.



Self Consumption as the Snake increases in size.

4. Can Never grow until the end of the map: Professional players win the entire game by starting off using the shortest path approach, however as the snake increases in size they move onto the longest path approach so as to avoid eating itself. This idea isn't implemented in our base model. This causes us to take a different approach.





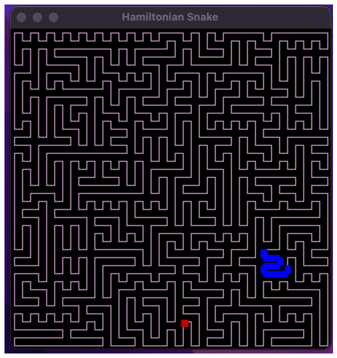
```
Game 98 Score 33 Record: 52
Figure(1280x960)
Game 99 Score 22 Record: 52
Figure(1280x960)
Game 100 Score 34 Record: 52
Figure(1280x960)
Game 101 Score 24 Record: 52
Figure(1280x960)
Game 102 Score 18 Record: 52
Figure(1280x960)
Game 103 Score 13 Record: 52
Figure(1280x960)
Game 104 Score 35 Record: 52
Figure(1280x960)
```

Result after 30 minutes(100+ games) of Training.

Hence, to solve the issue of dying by self consumption and winning the entire game we move to a mathematical model, the Hamiltonian Cycle.

2.2 Hamiltonian Cycle.

The problems faced in the Base Model were mainly the time taken to train and the over exploitation of the greedy algorithm causing a large snake to eat itself in order to reach to the food.



Hamiltonian GUI.

The Hamiltonian Cycle or famously known as Hamiltonian Path is a mathematical model presented by William Rowan Hamilton (Olliver 2022). This model is an integral path of Graph theory, where this model creates a random path in an enclosed space that connects the entire region in a closed path which doesn't overlap itself at any path.

Hence this model helps us create an amazing trick to solve two of our major issues:

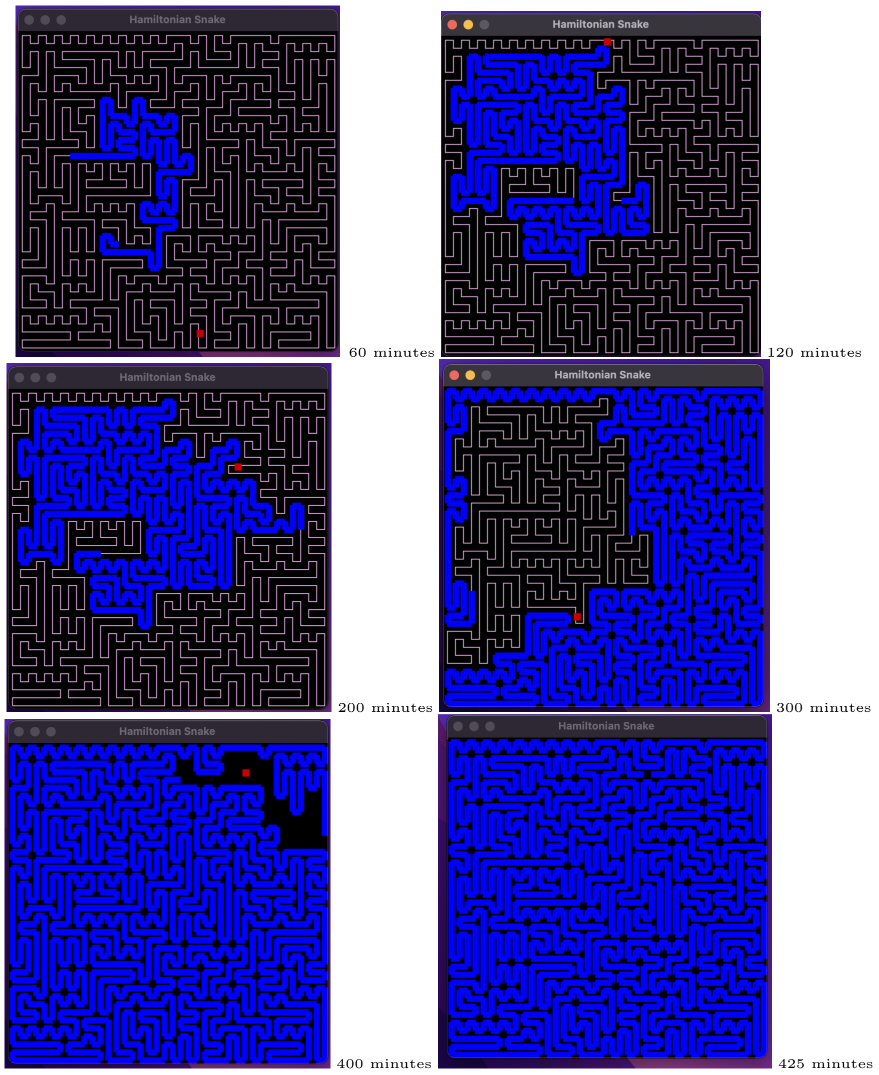
1. Solution for Infinite Loop and Self Consumption: This model eradicates the possibilities of any overlapping paths, hence the possibility of any self eating scenarios and infinite loops are out of the picture.

2. Solution for Extra training time to learn the environment. The only training to consider is the training for the Hamiltonian Cycle to create the path, once that has completed its a cake walk for the snake to just follow it and win the entire game easily.

However, there is a major drawback to this algorithm too. This is the failure to cut into the path and eat the food using the shortest path, therefore it does complete the game perfectly but after a really long period of time.

As the snake follows only the path and has no learning towards the shortest path the snake relies only on the Hamiltonian path and does not learn any shortest path moves to progress faster, hence if it takes 100 moves to eat the 1st food at position n, even though the position of the food at the same place after a few moves it will take 100 moves at least to eat it again. In short, if the food is just a few moves away from the snake it will ignore the food and continue following the Hamiltonian Path.

The Time dependency can be understood by observing the time the snake takes to reach the final food. This is shown below.



Output for Hamiltonian Cycle.

Although the output is promising and the result is perfect the time taken to reach this output is extremely high (7 hours). Introducing an agent specific method of cutting the Hamiltonian cycle when it is safe to activate the shortest path algorithm can optimise our snake.

This is covered in the next section.

2.3 Advanced Model.

So, until now the Hamiltonian cycle has worked perfectly yet the only point of improvisation is the time taken to complete the cycle.

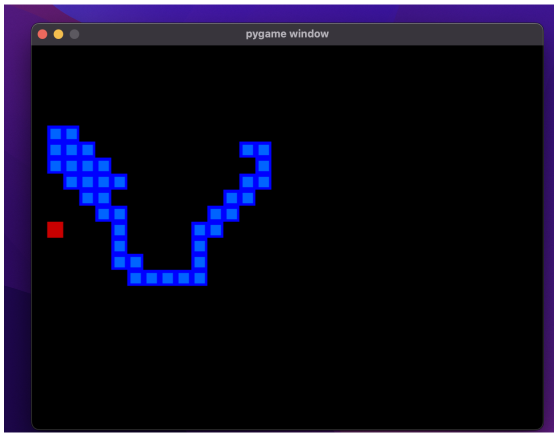
Hence, the only improvement to be carried out now is to add in the shortest path algorithm. The steps undertaken for this method are:

1. Applying the A* search algorithm to check when it is safe to use the closest path algorithm.
2. Applying the Hamiltonian Cycle as the start point for the snake.

The A* search Algorithm is a path finder algorithm that converts the space complexity of the initial position(head of snake) to the final position(food) into weights in ascending order. Hence by finding the path with the least weights we can get a straight(short-cut) towards the food.

Hence, we wont be just moving in 4 directions instead diagonally now, by combining two immediate adjacent directions together.

As you can see in the figure above, the snake doesn't move horizontally or vertically anymore, nor does it move in a fixed cycle as the Hamiltonian path, but it follows the path until it is close to the food and breaks out to eat it diagonally.



Snake Moving Diagonally.

3 Conclusion.

To conclude this experiment, I would like to share my observations as an insight to future development in this field.

- Using a basic Agent, State and Action System while Inducing a Reward I created a Snake Game AI that took around 3 hours and over 85 games to train a snake that gets a high score of 52 in a game of 360.
- So to improve this I introduced the Hamiltonian Cycle Path creator, a complete Mathematical Model which performed perfectly but, took 7 hours to do so!
- Finally, using A* search algorithm with the semi-perfect Hamiltonian Cycle i created a close to perfect snake game that takes around 120 minutes to complete the entire game, however using an extremely high computation power.

To improvise this code one can explore multiple avenues.

1. Using a Mathematical Model to complete the game is a good approach however, one can develop a complete Reinforcement Learning Algorithm to add in multiple state and action parameters to fine tune the agent to achieve the same goal as the Hamiltonian Cycle.
2. Creating a function that can exploit the A* closest path Algorithm in the initial phase and as the snake increases its size to a certain limit switch over to the Hamiltonian Cycle. In this case the initial time spent on following the Hamiltonian cycle will be saved and the in the second segment when the snake grows large it will take the Hamiltonian cycle to win the game in complete Totality.

4 Artifact Content List:

- Zip Folder Named "Snake_Game_AI" with Three Directories- Advanced-SnakeAi, Base_Model, Hamiltonian_AI
- Presentation named "Snake_Game_AI"

References

Bruderer, Herbert (2016). “The birth of artificial intelligence: first conference on artificial intelligence in paris in 1951?” In: *IFIP International Conference on the History of Computing*. Springer, pp. 181–185.

Goswami, Vedant (2022). *Ai driven snake game using Deep Q Learning*. `https://www.geeksforgeeks.org/ai-driven-snake-game-using-deep-q-learning/`. Accessed:2022-07-28.

Kaelbling, Leslie Pack, Michael L Littman, and Andrew W Moore (1996). “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4, pp. 237–285.

Olliver (2022). *Hamiltonian path*. `https://en.wikipedia.org/wiki/Hamiltonian_path`. Accessed:2022-07-04.

Szita, István (2012). “Reinforcement learning in games”. In: *Reinforcement learning*. Springer, pp. 539–577.