AI playing Snake Game with Deep Q-Learning

Selected Topics in Artificial Intelligence 2021. Professor: Velarde, Gissel

Ayllón Durán, Mauricio
UPB
La Paz, Bolivia
mauricio.ad1707@gmail.com

López Rojas, Jorge Ignacio
UPB
La Paz, Bolivia
georgeilr99@gmail.com

Vejarano Torres, Mauricio
UPB
Santa Cruz, Bolivia
maveto13@gmail.com

Pomier Salas, Kevin Mauricio
UPB
La Paz, Bolivia.
kevinpomier.kp@gmail.com

Abstract — **In this project, we explore the application of Reinforcement Learning to the game Snake. We use Deep Q Learning implementing a Neural Network to help the agent (the snake) to learn what action should take given a state of the game. The proposed model takes a state of the game, represented by an array with 11 values, as an input for a feed forward neural network and it gives as an output 3 values, each one representing one of the three possible actions the agent can take (turn right, turn left, keep straight). At last, we measure the results by comparing the top scores and the mean score of the Artificial Intelligence (AI) with our own scores to determine if it can do better than the average human.**

Key Words—Deep Learning, Deep Q-Learning, Agent, Environment.0

I. Model

Our implementation follows the model presented in [1]. The model uses Reinforcement Learning to make the agent (the snake) choose the best action to take depending on the state of the game to maximize the rewards. To understand how the model works we need to understand the 4 main concepts and how they are implemented (the state, the actions, the rewards, and the agent).

The state is the representation of the game that the agent can perceive, it contains the essential information the agent needs to process to know which action to take. Knowing how to implement the state is important because if you put in a lot of information, the agent will perform better but it will be slower, on the other hand, if we give less information, the agent will process faster but its performance will be worse. The implementation of the state in this model is given by an array with 11 values that gives the agent the information it needs to process and predict the best action. Each value can be 0 or 1 and represents different things: danger straight, danger right, danger left, direction left, direction right, direction up, direction down, food left, food right, food up and food down.

The actions are moves that the agent can make given a certain state. In this model the agent can make only 3 moves: turn right, turn left, and keep straight. These actions can be represented by an array of 3 values from 0 to 1. Each value represents the probability of each action to be the best option.

The reward is a value that tells the agent how well it is doing. You can sum some value to the reward if the agent does something good and sum a negative value if the agent does something bad. For this model the rewards are simple: it gains 10 points if it eats the food, loses 10 if it dies or if it takes more than 5 seconds for the agent to reach the food, the last punishment is in order to avoid bucles.

The agent is the AI that processes the state and gives an action, then it claims the rewards and learns how to improve for the next game. The way it learns is by implementing Deep Q Learning. For this model we use a Feed Forward Neural Network that takes as input the state and outputs the Q values of each action. The structure of the neural network consists of 3 layers: the input layer that takes the 11 values of the state, 1 hidden layer with 256 neurons and Relu activation function, and the output layer of 3 values, each one corresponding to the Q-value of each action. The algorithm of Deep Q Learning starts processing a state, which will return the Q values of all actions, then the agent will select the action with max Q value (or a random action depending on the epsilon-greedy policy), perform the action and receive the reward, calculate the loss function, and last use back propagation to minimize the loss [2]:

$$Loss = (r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2$$

II. Experiments

We performed separate experiments. First, each participant played the snake game in the best way they could. It became an essential part of the project since its results will later be very useful, becoming a baseline for the project and to make a comparison of Human vs AI.

Likewise, the second part of the experiments was carried out only by the AI where it learns to play. It should be noted that the AI at the beginning has no idea how to play and through random movements it performs an exploration. As the game progresses, it exploits the known, thus generating a game strategy. Specifically, we let the AI carry out a training (Exploration-exploitation) of 200 games. The same model was trained with different optimizers (Adam, SGD and Adamax) to compare their performance and choose the best model to compare with the human results.

III. Results

To assess the results of the models and compare them, we used 3 parameters, the high score to compare the best game from each model, the mean score to compare the general performance of each model, and the standard deviation to see the distribution of the scores each model achieved. The mean score is computed as:

$$MeanScore = \frac{TotalScore}{NumberOfGames}$$

For the second part of the experiments, we only compare the model with Adam Optimizer, which was the best optimizer, with the human results, only using 2 parameters, the high score and the mean score as seen in Table 1.

Model	Highest Score	Mean Score	Standard deviation	Number of Games
Optimizer Adam	56	16.14	12.804	200
Optimizer SGD	7	0.63	0.518	200
Optimizer Adamax	17	0.62	0.58	200

Table 1 - Model results for different optimizers

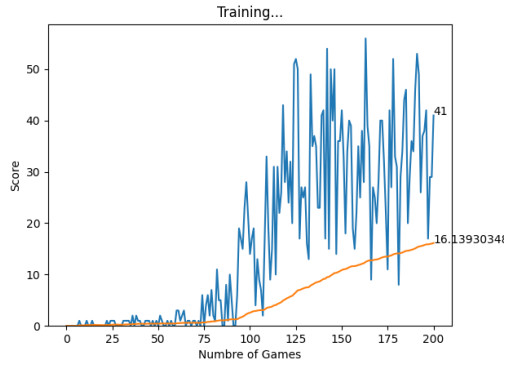

Figure 1 – Results of Model with Optimizer Adam in 200 games

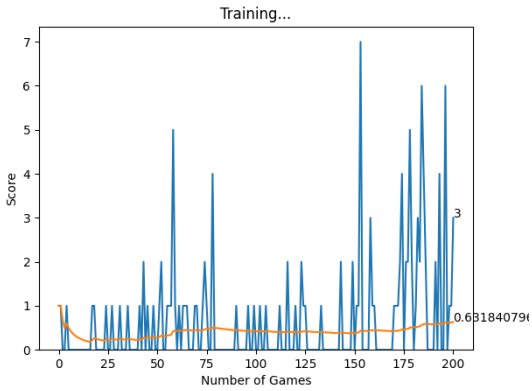

Figure 2 – Results of model with Optimizer SGD in 200 games

Player	Highest Score	Mean Score	Standard deviation	Number of Games
Model – Adam optimizer	56	16.14	12.804	200
Jorge López	35	5.05	3.76	20
Mauricio Ayllón	50	9.7	7.68	20
Mauricio Vejarano	61	12.95	9.89	20
Kevin Pomier	45	6	5.50	20

Table 2 - Results that each player got when playing vs Best Model

IV. CONCLUSIONS

In this project, we have shown an implementation of Q-learning together with the change of different parameters. Anticipating that the difference between the performances of the model could become more stable after a long period of training, and the result did not verify our expectation. Furthermore, we compare this method with human players. While none of the models could achieve the perfect game, it was found that after a certain number of games the best model could learn how to play like a human would do, developing its own strategy. We observed that agent seems not to be aware of its body, only its head. Therefore, the algorithm could be improved to achieve a higher score by improving its state representation. This is worth exploring for future work.

V. REFERENCES

[1] Teach AI To Play Snake - Reinforcement Learning Tutorial With PyTorch And Pygame (Part 1). (2020, December 20). [Video]. YouTube. https://www.youtube.com/watch?v=PJl4iabBEz0&list=PLqnslRFeH2UrDh7vUmJ60YrmWd64mTTKV&index=1&t=670s

[2] Deep Q-learning: An introduction to deep reinforcement learning. Analytics Vidhya. (2020, April 27). Retrieved October 25, 2021, from https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.

IX. CONTRIBUTION

Jorge Lopez contributed to the construction of neural networks
Mauricio Ayllon: contributed to the data collection and construction of graphics when analyzing the game
Mauricio Vejarano: contributed to the training of the machine, executing it in the game snake.
Kevin Pomier: contributed to the data collection to be able to train the machine