REPORT VOLPATO PIETRO

Snake Project

1 - Environment

The board i decided to use for the project is a 6 * 6 board walls included, so the space in which the snake moves is 4 * 4. The original rewards provided were slightly modified to encourage the snake to make the least steps possible. The STEP_REWARD value has been modified from 0 to -0.005. The value is kept low otherwise the snake might prefer to hit the wall rather than undertake a long path (6 steps max.) to eat a fruit.

2- Algorithm

The algorithm implemented to interface with the snake environment is the DQN. This algorithm allows to combine the principles of deep neural-networks with Q-learning and uses the neural network's weights to learn the action-value function.

3 - Architecture

The architecture is structured as follows:

- **Convolutional layer**: 32 filters, (4 * 4)
- **2 Dense layers**: 2 64 neurons layers to process the output of the convolutional layer
- **Output later**: Another dense layer with dimension 4, the same of the action space

All layers but the last one use a ReLU activation function. The filters in the convolutional layer allow to learn the spatial features of the boards, while the dense layer process the input and maps it to the space of possible actions to take given a state.

3.1 Masking of illegal actions

Given a state, I've implemented a function that finds the position of the head in the board and scans the 4 blocks around the head. To each block (each possible action) assigns the relative reward and use it to modify the output of the neural network, to make a more informed selection of the actions to undertake and compute the targets.

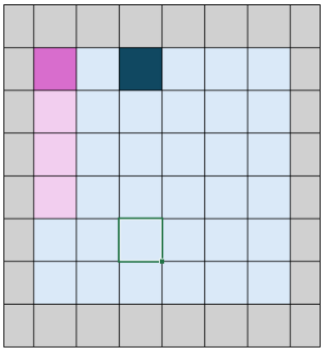


Figure 1

-0,153	0,124	-0,054	-0,076	Original output of the Neural Network
-0,2	0,124	-0,1	-0,1	Output modified by subsituting the outputs with values related to action leading to terminal states
-0,153	=-0,005 + 0,261	-0,054	-0,076	Output modified with the target value in place of the selected action
-0,2	=-0,005 + 0,261	-0,1	-0,1	Output modified with the target value in place of the selected action and rewards of terminal states

Figure 2

In the example above we have the head in dark pink, the body in pink and the fruit in dark blue. In the first row of the second picture we can see the output of the neural network. In the second the actual values with which the action is selected. In this example in both cases action selected is 1. The rewards and the value of the next state are computed and substituted as the target in the position of the selected action. The last row shows the target used by applying the described function, which also substitutes the rewards of terminal states. This allows to speed-up convergence with a more comprehensive loss, and to make more informed decisions, especially in the early stages of learning.

3.2 Enriched state representation

The state in this case study is a (*n_boards* * 6 * 6 * 4) tensor, giving a one-hot representation of the board. To enrich the state, I've added a 6-dimensional vector to the output of the first dense layer:

- **Manhattan distance:** Occupies one cell of the 6-D vector and represents the Manhattan distance between the head of the snake and the fruit.
- **Position of the fruit:** A 4-D binary vector that gives indications on how to reach the fruit. By using the example in *Figure 1,* only the cell related to action "right" would have the value 1, with all other cells being 0. → [0, 1, 0, 0].
- **Length of the body:** A 1-D vector indicating the length of the body

Those information help the agent to improve its decision making.

4 - Baseline

The strategy I implemented to compare my RL agent with is a simple baseline that guarantees to solve the snake game at any iteration, no matter the board size. The only constraint of the baseline is that the board size must be even. It consists in moving the head of the snake in the closest corner at the start of the game. Then, depending on the corner chosen, move up or down (depending on where the wall is) and continue straight until the last block before the wall. At that point, turn in the direction opposite to the lateral wall and turn again to move towards where the snake was coming from.

Ex: *To make an example, let's suppose the board is 8 * 8 walls included, so a 6 * 6 "walkable" space. If we start in the top left corner, the snake will have to move down 5 times, then turn right, and move upward 4 times.*

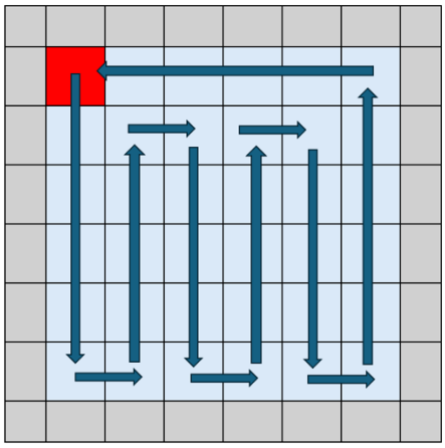
Then continue straight again until the second to last block before the wall, turn in the same direction as before, (leaving a one-block-gap between the head of the snake and the wall) and move towards the opposite direction where it was coming from.

Ex: *Continuing the example made before, the snake now turns right again and moves down 4 times, since this time there is a one-block-gap.*

Repeat the same process as before (continue straight until the last block before the wall, turn in the same direction as before, and move in the opposite direction until the second to last

block before the wall) until the snake arrives in the corner in the same diagonal as the one where it started, but in the opposite direction. This time move in the y-direction opposite to the wall until the snake reaches the corner In the opposite diagonal and same direction. This time move in the direction opposite to the wall in the x-direction, until the snake reaches the corner where it started the loop. Continue until convergence.

Ex: *By continuing the loop from the previous position, the snake will reach the bottom-right corner. From this point the snake moves upwards until it reaches the top-right corner, and continues left until it reaches again the top-left corner.*



Presented here on the lift a picture illustrating the strategy starting from the top-left corner. Walls are painted in gray while the arrows represent the actions the snake takes to complete a loop.

Figure 3

The problem with this baseline is its inefficiency in the early stages of the game, since the snake might have to make a full loop just to eat one fruit. In later stages of the game, where the majority of the board is occupied by the snake's body, the n° of fruits-per-loop will increase, making the policy more efficient.

5 – Training

The agent was trained for5000 iterations, over 1000 parallel boards. Epsilon was decreased exponentially, using a decay rate of 0.99935 and a minim value of epsilon of 0.1.

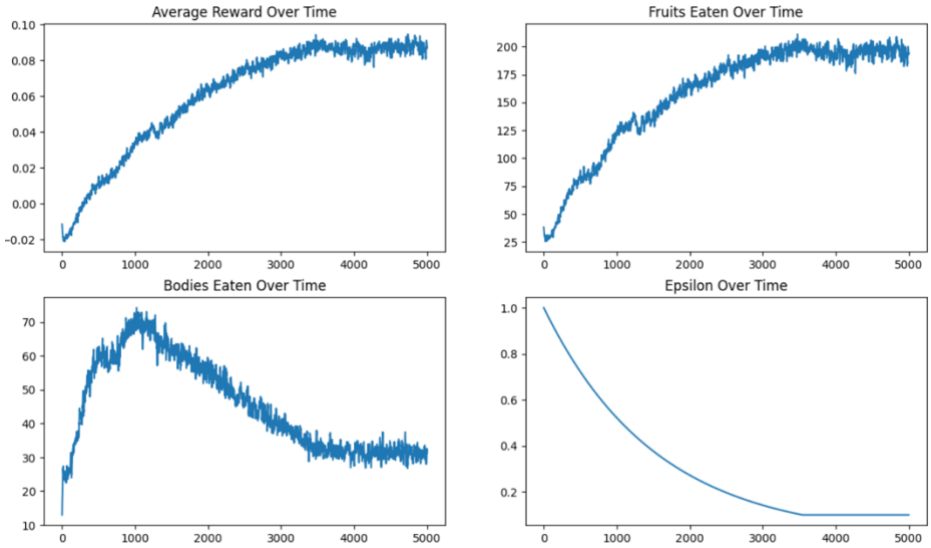
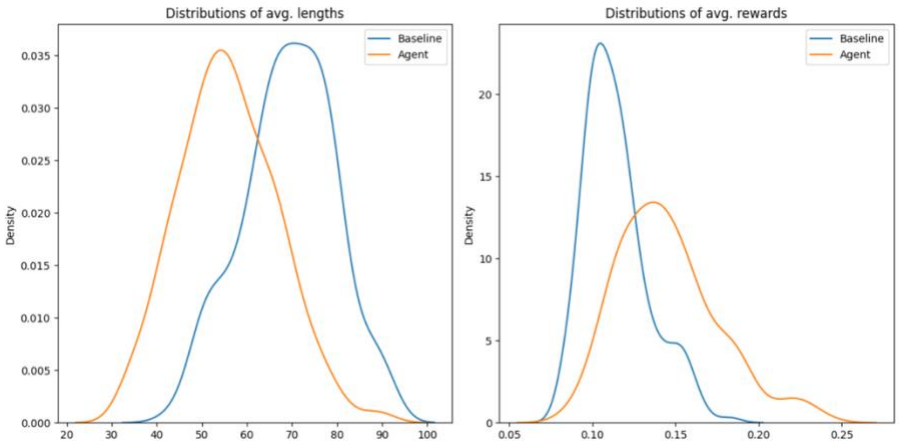


Figure 5

The results of the baseline were not plotted above, since the strategy is designed to always eat the maximum number of fruits possible and never eat its body. The real comparison with our algorithm can be made on the average reward, given that the flaw of the baseline is it's inefficiency.




```
Mean length of baseline : 69.28
Mean length of agent : 55.87
Mean rewards agent : 0.1452  -  SD : 0.03027
Mean rewards baseline : 0.1143 -  SD : 0.01897
```

Figure 6

From *Fig. 6* we can see how, when the agent completes the board, it has higher average rewards with respect to the baseline. The magnitude of the result is not enormous, but it is a good improvement in percentage. Bigger boards will return higher discrepancies between the agent and the baseline, while for boards of this size results are almost comparable.