Mulac Nino
Vermot-Desroches Matthias

2024-2025
ROBO5

Reinforcement Learning Project : Dinosaur game

Introduction

During this project, we wanted to use the knowledge learned during our Reinforcement Learning courses to make an IA that could play the Dinosaur Game optimally.

The Dinosaur game is originally a browser game developed by Google where the player controls a Dinosaur in a side-scrolling environment. The goal of this game is to avoid obstacles and get the highest score.
In order to avoid obstacles, the player can either :

- press the “up arrow” key to jump
- press the “down arrow” key to crouch or fall faster during a jump
- do nothing

In such an environment, we can code our agent to be able to do these actions. Its reward will be determined by its survival, so the total reward would be the score.

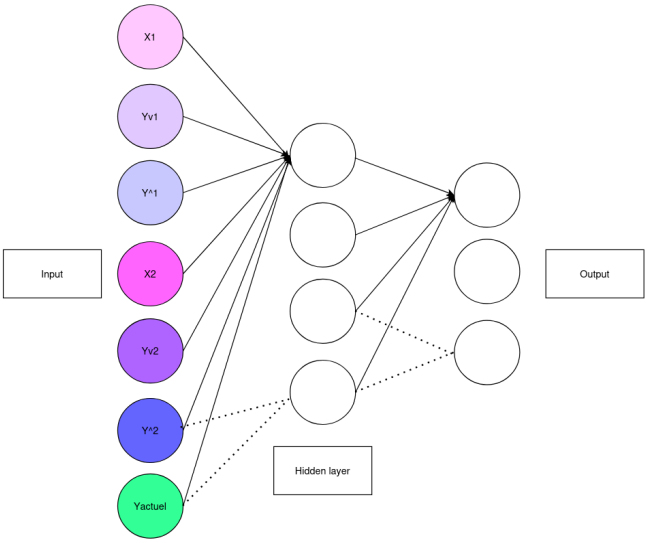
Re-creation of the game

First, we decided to recreate the game/environment using the Ursina library in python. It would allow us an easier time when training our agent.
The normal way of controlling the dinosaur would be pressing keys on the keyboard, but we can’t do that if we want to have multiple AIs training in parallel. So we decided to add another control method: we write the action/input to a file that our game will then read to make the action. This should theoretically not cause any delay, since the update of the file will be faster than the game frame update.
The AI inputs and rewards are also passed like this, in another file.

After all this, we started the training. In theory, the AI should be able to play indefinitely by making the correct actions.

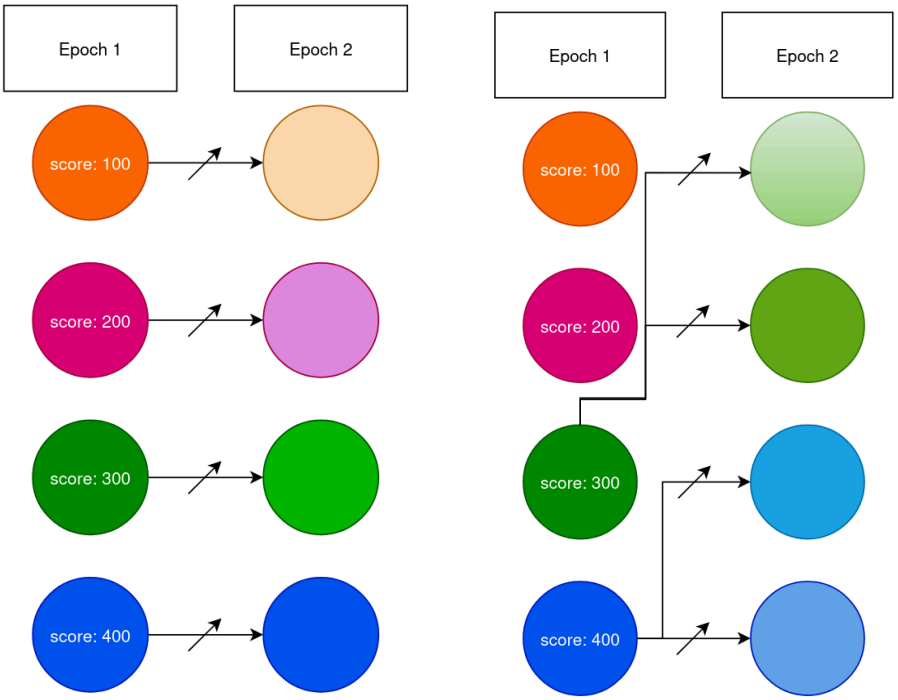
AI and Training

Our AI is based on a basic neural network. Its inputs are the position and sizes of the two next obstacles, as well as the height of the dinosaur. Its outputs are the possible actions: jumping, crouching, or doing nothing, so three neurons.



For the training, we tried several approaches, based on genetic reinforcement.
Genetic reinforcement is based on neural networks with weights serving as *genomes*, randomly selected at the start of the training. After every epoch, each genome can survive or not to the next generation, with more or less changes, depending on its performance during the epoch.

- Using 10 different genomes, and inflicting them a variation proportional to the inverse of their score. Everyone is selected to go to the next epoch.
- Using 10 different genomes, and selecting the five best of them, which will be able to “reproduce” themselves with slight variations.



Schematic of both approaches (for 4 genomes). The color variation depicts the weight's update and its intensity.

Results and conclusion

While the first approach wasn’t really successful, the second one performs well in most cases, except when there are too many obstacles at once.
Since our game can yield impossible situations, with several obstacles gathered, for example, it also limits the artificial intelligence effectiveness. But seeing it trying to overcome these limits is interesting, that’s why we decided not to patch the impossible cases of the game.
This project helped us learn more about reinforcement learning, especially genetic reinforcement.