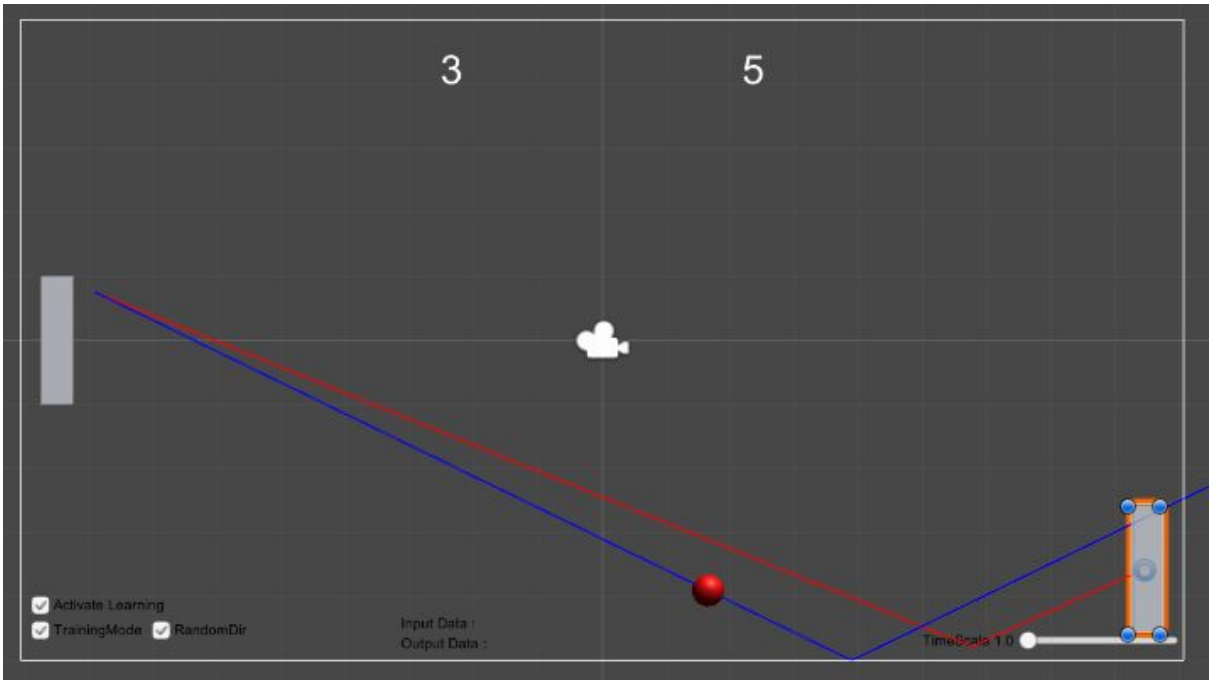
Neural Network AI Pong Project

1. Objectives

- Implement a neural network AI in the game
- Find and implement an algorithm that allows the AI to anticipate the ball's path.
- Save the training data in a JSON file for reusability.

2. Procedure

In this project, the ball will constantly try to calculate the ball's path every time it is coming to it.



The above picture shows :

- A blue line: it represents the ball's actual path. You can see the ball following it.
- A red line: it represents the path the AI has calculated through his neural network.

● Input Layer

The calculation process starts when the player sends the ball to the AI's side. The AI will then receive 4 input values:

- The X and Y values of the starting position. This can be either the ball's position when the player sent it or when it bounced off a wall (see Output Layer)
- The X and Y values of the ball's direction.

● Hidden Layer

This neural network's complexity has one hidden layer containing 15 neurons.

● Output Layer

After calculation, the AI will return 3 output values:

- The X and Y values of the estimated ball position before it bounces off.
- A value called "finishedForesight" which can be 0 or 1.
    - (0) means the AI thought the ball would bounce on the wall, in which case he will initiate another path calculation.
    - (1) means the AI thought the ball would arrive at his side, in which case he will move the paddle to the Y coordinate he calculated.

● Calculation process

Following the example from the picture above, here is the entire calculation process:

1. When the ball hits the player's paddle or the player just launched it, the ball's position and travelling direction will be sent to the Neural Network.
2. The AI notices the ball will collide with the bottom wall. As a result, it will send the collision's position as an output value. The finishedForesight output value is 0.
3. If finishedForesight = 0, then the collision position it sent as an output earlier will be used as an input for the next path calculation. The direction will be reflected to represent the bounce, and be also sent as an input value. The AI will repeat the process as long as it thinks the path is not finished. Such procedure could be risky, because the AI can indefinitely think that is the case, resulting to an infinite loop. To avoid this, we set a limit to the number of calculations it can do.
4. If finishedForesight = 1, then the AI will move its paddle to the output coordinates.
5. Backpropagation will be processed at the end of these calculations.

3. Why use this method?

The objective here is to limit the variations of the neuron weight. Before using this method, I ran some tests and noticed the AI could not find the ideal weight, because the ball's direction varies a lot. When the AI tried to estimate the final positions, the outputs were fluctuating a lot.

By using this approach, I made the AI proceed step by step, by splitting the path according to the number of bounces the ball is going to do on the wall. This allowed me to limit the randomization.

That being said, the more bounces there is, the less precise the outputs will be. Since we use the AI's estimations as inputs to the next step's calculation, if the estimation was wrong, so are the following outputs. This makes the backpropagation process less efficient.