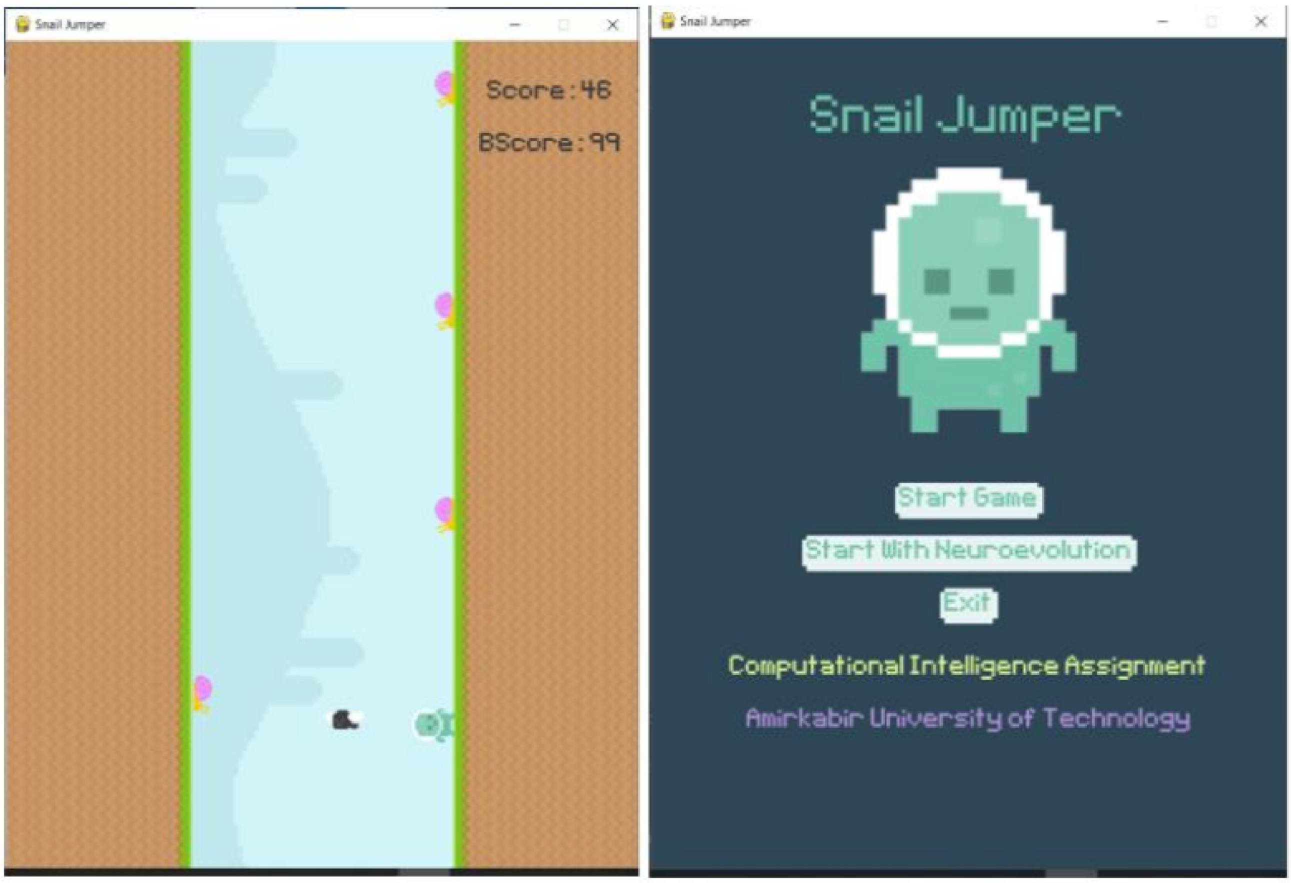
## Introduction

The aim of this project is to use an evolutionary algorithm to learn neural network in an environment where there is not enough data to learn. One of these environments is the game, where there is always something new happening, and therefore creating trainable data is almost impossible.

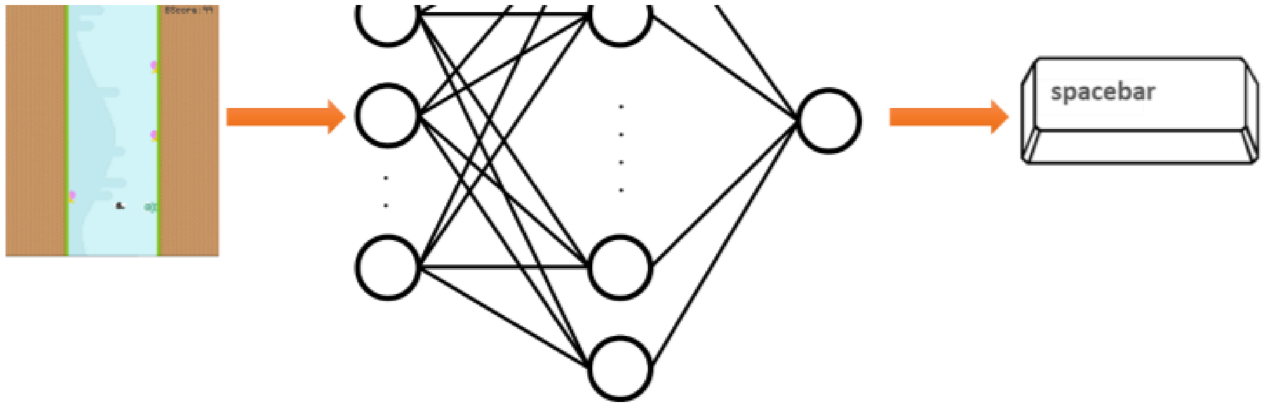
The game that we're trying to train computer for is called **Snail Jumper**.


Snail Jumper
Score:46
BScore:99
Snail Jumper
Snail Jumper
Start Game
Start With Neuroevolution
Exit
Computational Intelligence Assignment
Amirkabir University of Technology

This game can be played in two modes, manual and neuroevolution.

In the follow up, we will get acquainted with how neural evolution is implemented and we will see how evolutionary algorithms will help in learning neural network.

To advance the game using neural evolution, we must design a neural network that takes important decision-making parameters under input and then generates the corresponding output. In the end, the output produced is similar to pressing the *space* button defined in the game.



Therefore, after determining the important parameters in the decision and building the neural network architecture, the Feedforward operation is easily performed.

After the Feedforward operation we should define a cost function and in the following, update weights and biases using backpropagation till the cost turns to minimum.

But in our situation, there are no data to train. Therefore we can use evolutionary algorithms. We produce 300 players which each player has a neural network where their weights and biases are initialized to zero and a normal random value.

Then, according to the neural network architecture and the available initial values, each of them shows a different function by observing the obstacles. Some will hit obstacles and some will cross. The more the player continues on his path, the more fitness value it will acquire. Thus, according to the principle of evolution, players with better performance will always be passed on to the next generation, and by considering the crossover and mutation operators, after passing a few generations, it is expected to see better performance in player.

● ✕