**Parvulescu Valeriu**

**Stefan Andrei**

**1231A**

**NN&GA**

**Ai Avalanche**

# Main idea:

The purpose of the project is to make the computer learn to play a specific game created in python.

We used the game created at HCI (AVALANCHE v1.0)

# Input data:

-player\_piece(the CPU players)

-current\_piece(current falling object)

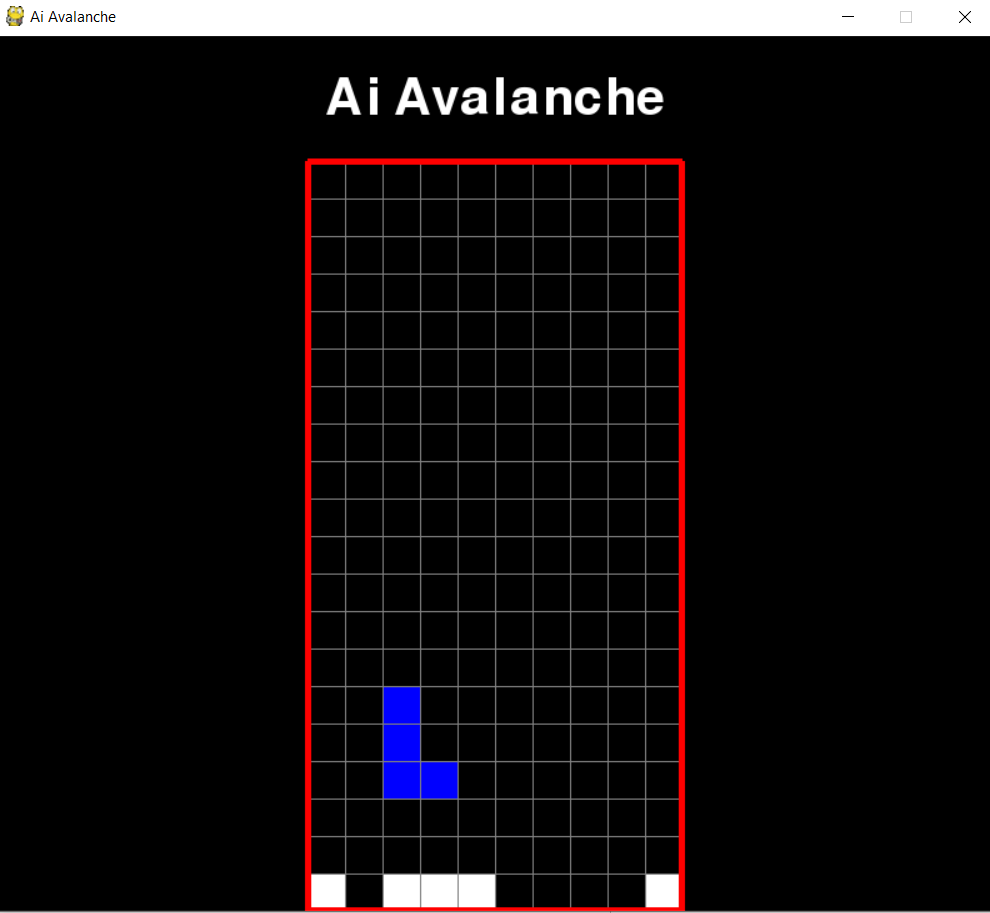
-next\_piece(next falling object)

-red-bordered grid of 20 rows and 10 columns(the area where the action happens)

# AI analysis:

We’ll start from the data known from the HCI report.

For this project,the in-game window will look like this:



The players are the CPU and are called ‘genomePieces’.Each player has as inputs the shape of the object falling,the distance between the player and the object,the rotation of the object,the distance of the player from the walls and the current position of the player.The corresponding weights are random at first.

The fitness will represent the score made by each player before he dies.

While the game runs,the players process the inputs and the weights using the ‘tanh’ function and make decisions(meaning going left or right) based on the value obtained.

As players die,we observe that there is one that the last remaining are a lot smarter than the previous ones.This means that some of the random data for the weights works for the learning process.

# Selection:

After all players died we do the following:

-in the list ‘genomes’ we sort the players decreasingly with respect to the fitness score and keep only 20% of the players.

We do this because we want to keep only the best players and our goal is to create more players

with similar learning rate as these 20%,meaning we make ‘children’ for these ‘parents’.

# Crossover:

Here we create the children by making various computations using the weights obtained from the parents.

2 parents are selected randomly from the list of intelligent players and are used to make the children.

We make 2 new children for each pair of 2 parents,and append them to the list so we can obtain the initial population.(in our case population is 130 and after a game we keep the best 26 players and make children of them to recover the remaining 104)

# Mutation:

Here we perform the mutation for the renewed list of genomes.

We use some random data and based on that we either add or substract the weights with their values multiplied with the random data and some external finite data.

After this the whole process starts again after a new game starts.

# External file:

To make the learning faster after we close the game completely we put the last update of genomes in an external file,so that when we run the game again it will take the last genomes from the file and continue from that point.

# Conclusions:

We can say that it is a sort of over-fitting method,as we do not expose a clear rule for the players to follow.But the specific computations help in the learning,and after many games(or generations in our case) it will become impossible to kill the players,and this will result in an infinite loop.(Meaning our goal was reached)

# Bibliography:

<https://en.wikipedia.org/wiki/Artificial_neural_network>

<https://www.youtube.com/watch?v=aircAruvnKk>