# Comparison of Genetic Algorithm and Monte-Carlo implementation on Ms Pacman

## Abstract

This paper compares the viability of using a Genetic Algorithm and Monte-Carlo policy within a game space to play the game Ms Pacman. The method implemented are a Standard Genetic Algorithm using Roulette Wheel Selection for parent selection and a Monte-Carlo state-action value algorithm with a soft policy [1] to ensure all states have a non-zero probability of being used. The methods are used within the OpenAI Gymnasium on the Ms Pacman environment using a deterministic implementation of the game.

The Genetic Algorithm made a final score of \_\_\_\_\_ and the Monte-Carlo Policy got a final score \_\_\_\_\_

[results]

[conclusion]

## Introduction

*This drone will fly within a defined space to hit a number of predefined targets. The drone is controlled using two motors which define the thrust independently allowing for navigation, and is subject to gravity and drag forces on rotation and planar motion. The task is to write a flight controller which uses reinforcement learning to have the drone traverse space to hit as many targets as it can within a specified amount of time.*

In the field of machine learning there have been many different algorithms developed to encompass a range of different situations from image processing to pathfinding. This project is aimed around implementing a machine learning algorithm into the game space Ms Pacman provided by OpenAI Gymnasium. While there are many ways to go about solving Ms Pacman, this project focuses on Genetic Algorithm and Monte-Carlo implementation within a deterministic Ms Pacman simulation.

## Methodology

### Environment

For the environment of this project, I used the OpenAI Gymnasium with the MsPacmanDeterministic-v4 environment. This environment creates the entire Ms Pacman game.

The Pacman and Ms Pacman environments at each time step returns the following variables: observation, reward, terminated, truncated, info.

I utilise only the first 5 actions for the algorithms, noop, up, right, left, and down as this represents all of the directions that I believe to be necessary and avoiding unnecessary additional actions where the difference is only seemingly needed for when using a joystick.

I used the deterministic version of the game so to make the ghosts do the same actions each time. This is because when trying to train a Genetic Algorithm on an environment that has enemies which display randomness, a high score on that attempt may not be an accurate representation as when ran another time could show any value between 0 and even higher than the previous score it got without the instructions changing.

In order to do the following algorithms I needed to get the positioning of the objects on the board. To do this I had to utilise both the RGB observations and the RAM observation settings provided by OpenAI Gymnasium.

I used the RAM observation setting to gather the coordinates for each of the ghosts, Ms Pacman, and the cherry by manually printing out the observations into a CSV file for roughly the first 200 frames of the game and setting the action of Ms Pacman to be moving left then upwards. From there I colour coded the CSV file so that if the current cell is larger than the previous cell it would be coloured green, and if it is smaller than the previous cell it would be coloured green. This helped me to decern whether the object is moving in a positive or negative direction from its previous state.

By doing this and watching what Ms Pacman does given the action of go left and upwards I could figure out which frame it changes direction and match that up with the CSV to find an entry which is moving negatively (indicating left movement) until that frame and then doesn’t change whilst also seeing where another entry changes from not moving to moving negatively (indicating upwards movement). I then repeated this process of watching the ghosts for their movement of the red moving in a certain path and the other ghosts moving in the starting square in a predictable pattern I figured out which output from the RAM observation the ghosts represented by. This showed that the RAM observation produces all the X coordinate values of Ms Pacman and the ghosts first and then does the Y values. I noticed there was a gap between the all the X and Y values which had no change from the start that I guessed to be the cherry which after testing found out to be true.

After getting the

### Genetic Algorithm

For the Genetic Algorithm, I followed a standard methodology for the genetic algorithm described by the following pseudo code:

To represent this within the program I created 3 classes, the brain being the holder of all the instructions that the agents would follow, the agent class where the information about the agent is stored such as its fitness and the ability to create a child, and the population which does the main functions such as natural selection, genome crossover, etc.

When training I created a population with a size of 50 and ran the algorithm for 1000 generations taking the best agent from each population to move to the next generation, a mutated version of the best agent, and then used the roulette wheel algorithm to choose the remainder of the population. Then applied one point genome crossover on the population during natural selection.

### Monte Carlo Soft policy

I have set up the algorithm to have three behaviours to choose from, run away from the ghosts, collect pills, and chase the ghosts.

## Observations

The Genetic Algorithm works particularly well when it comes to eating the ghosts when they go blue

## Experiments

Number of pills eaten

Highest scoring

## Results

Genetic Algorithm performed better with X

Monte-Carlo Policy performed better with Y

## Future Work

Adapt the Genetic Algorithm into using genetic programming as to act on behaviours rather than a list of instructions

NEAT

## Conclusion

## References