# 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

*Drones have become a useful tool in the modern era, with many applications such as agriculture, photography and warfare. Defining movement of a drone is a nontrivial task with many variables to consider. While there are many ways to control a drone such as user controlled, this project focuses on implementing autonomous flight for a 2d simulated drone. 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.*

## Methodology

### Genetic Algorithm

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

Trained for 1000 generations with a population size of 50 per generation.

Standard Genetic Algorithm using roulette wheel algorithm for parent selection

### Monte Carlo Soft policy

### Environment

For the environment of this project, I used the OpenAI Gymnasium with the MsPacmanDeterministic-v4 environment.

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.

## Observations

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

## Experiments

## 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