Comparative Analysis of Digital Self-Driving Vehicles: Genetic Algorithm vs Reinforcement Learning

Blake Oakley , **Brandon Willett** and **Eric Watson**
University of Central Florida
{BRobersonO, BWillett, EWatson2}@knights.ucf.edu

Abstract

This paper analyzes the results of a genetic algorithm (GA) evolving a self-driving car as represented in a digital 3-dimensional (3D) environment. The results of the GA-evolved car are compared to a car trained by a widely available deep learning method of machine reinforcement learning (RL) [Juliani *et al.*, 2020] as well as a human-controlled car. From the experiments conducted as a part of this study it was discovered that a neuro-evolved GA outperforms the ML-Agents RL cars.

1 Introduction

In many video games, when parameters are kept equal, artificial intelligence (AI) greatly under-performs human capabilities. What possibilities do GA's [Holland, 1975] hold in comparison to other machine learning (ML) techniques in furthering AI competitiveness against humans? GA's have previously been found to successfully navigate the vast search space that limits other methods of training video game AI's. [Kim and Ahn, 2018] Games in general, and video games more specifically, have been heavily utilized in researching the effectiveness of various algorithms and their combinations in the field of AI. [Logofatu *et al.*, 2019] The study of evolutionary algorithms have also been successfully applied to evolving neural networks in self-driving cars. [AbuZekry *et al.*, 2019] These works are extended here by combining the worlds of neuro-evolution of autonomous vehicles and 3D video game AI, and analyzing the observable differences between traditional RL, neuro-evolution via a GA, and human capabilities, in racing digital cars around a track.

This paper details a set of experiments conducted on both RL and GA agents, chart their respective evolution, and concludes with a comparative analysis. Ultimately, it is shown that while a genetically evolved car doesn't reach the same performance level as a human, it does outperform cars trained by reinforcement learning. These experiments were conducted within a 3D digital environment created using the popular game engine Unity, a tool available for free when used non-commercially (and even commercially within limits). Each agent, both for RL and GA, is controlled by a neural network, and is trained on a traditional stadium-shaped track with 18 checkpoints used for fitness scoring [Schuchmann, 2020] (see Figure 1). RL and GA agents were trained on an arcade style car controller. [SpawnCampGames, 2020]

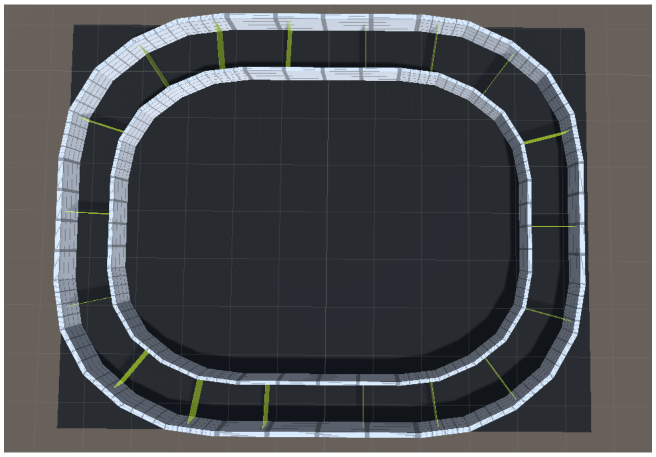
Figure 1: stadium-shaped track in Unity with checkpoints

2 Reinforcement Learning

ML-Agents is a package/plugin for Unity, available openly through the software's asset manager that requires the installation of pytorch. [Juliani *et al.*, 2020] This allows developers to create AI with ML techniques. Of the many ML approaches available through this package, the RL method was used to train agents over a series of experiments, or runs. In Unity, scripts, or code that dictates how an agent navigates and interacts with the digital environment, control how agents are trained. After setting up the ML-Agents package the training environment for the RL agents was established. [Monkey, 2020] Five experiments were ran with 500,000 steps (decisions made by the RL training) each, on a population of 20 cars.

Run 1: Each of the 20 agents attempted to complete one lap around the track. When an agent ran into either the inner or outer wall bordering the track, it received a negative reward. A positive reward was granted for completing the lap. This experiment was limited in that agents were not encouraged to complete the lap with a high degree of speed, leading to a need to add additional time for agents to run on the track in order to increase performance. Run 2: In this iteration of

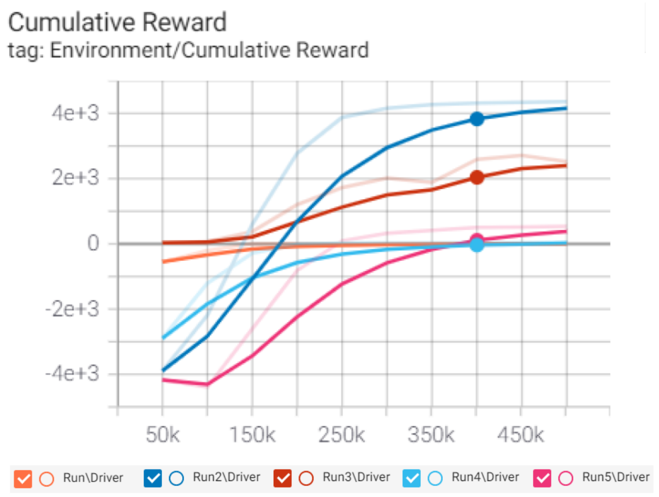

Figure 2: Results for the Reinforcement Learning

Run	Best	Average
Run 1	18.55	19.08
Run 2	16.43	16.88
Run 3	17.68	18.17
Run 4	16.72	17.23
Run 5	14.41	14.92

Table 1: Lap Time in Seconds for RL Runs

the experiment, agents were not terminated after completing one lap. Allowing the agents to run multiple laps translated into an increase in both speed and navigation around track. Run 3: In this run, instead of only giving a bad reward for an agent's collision with a wall, the agent's entire episode was also terminated and restarted. The hypothesis that this would improve the rate at which agents were changed positively proved unjustified, however, as the agents were reinforced very strongly to avoid walls, thereby both reducing speed and risk taken to take the sharp curves closer to walls needed for optimal finish times. Run 4: This run re-tested agents where walls did not terminate the agent's episode and the agent was able to attempt multiple laps, allowing for greater diversity in the experience needed for training to take root. Run 5: In this final run, rewards were implemented that strongly reinforced agents who travelled both fast and far. These rewards were weighted such that distance received more importance than speed, and produced superior results relative to all other runs, as displayed in Table 1.

3 Genetic Algorithm

In the GA created to evolved cars, each individual in the population is an agent containing a neural network that serves as the brain of the car. The jagged array-based implementation of the neural network used in this study is taken from an open source and modified. [Jet, 2017] The neural network contains 3 input nodes which receive values from ray-casts spreading outward from the central point of the car and have the ability to sense walls and checkpoints on the track. Next is a number

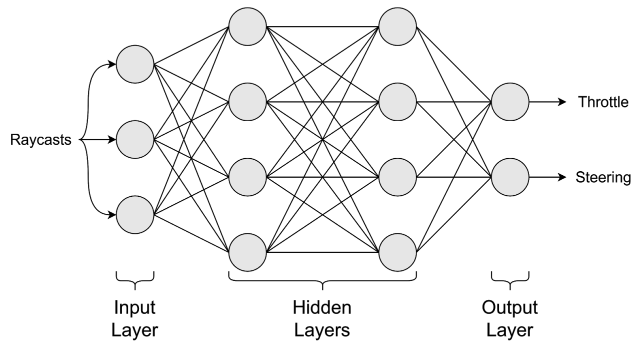

Figure 3: Neural Network Structure

of hidden layers (from 1 to 3) with each hidden layer having a number of neurons (nodes - noted on each run). Finally, the neural network contains 2 output nodes, each producing values for acceleration and turning radius. Every node from one layer to the next contains an edge which holds a weight.

From generation to generation, elitism is enforced for the best individual of each generation. This represents a 1% elitism, as population size is held at 100 for each run. The remaining 99% of the population for the next generation is created via crossover with parents being chosen in a tournament-style selection process where two groups of 4 from the population are chosen, with the best of each serving as parents. Crossover produces a child which is created by taking the weights of every other layer from the 2 parents. Products of crossover then undergo mutation, where each of their weights have a 0.01% chance of being replaced by a randomly generated weight. Each run consisted of 50 generations with a maximum iteration of 10 and each generation had a maximum time limit of 120 seconds.

The GA also runs in tandem with a hidden layer algorithm that generates hidden layers randomly between a minimum and maximum layer count and neuron count. This provided our GA at the beginning of every iteration with a brand new hidden layer combination that had not been used previously. The hidden layer algorithm kept track of all hidden layer combinations and, by receiving the average fitness of the iteration from the GA, tracked the best performing combination. This really improved our GAs ability to discover the best hidden layer combinations (see Figure 3).

4 Results and Analysis

The runs tested the effects of variance in hidden layer size and number of neurons. The results of these runs are seen in Table 2.

Table 2 demonstrates a general trend of more layers producing better lap times for the best agent of the run in both its best lap and average over all laps. However, as can be seen with an anomaly that might bring this conclusion into question, Run 3 had only 2 hidden layers, yet produced an agent which outperformed Run 4's 3 hidden layers. This finding suggests that the number of neurons in each hidden layer also plays an important role in the performance of the agent. This is one reason why the decision was made to vary the num-

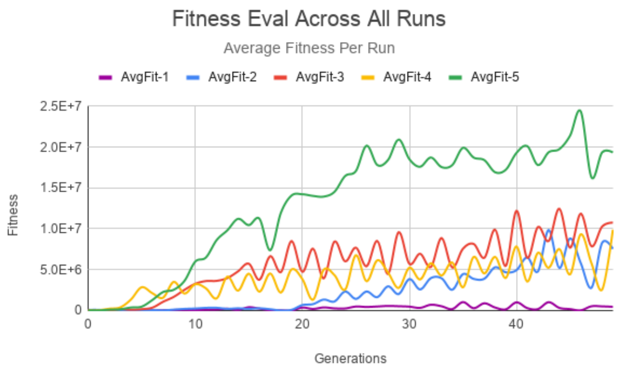

Figure 4: Average Fitness per Generation

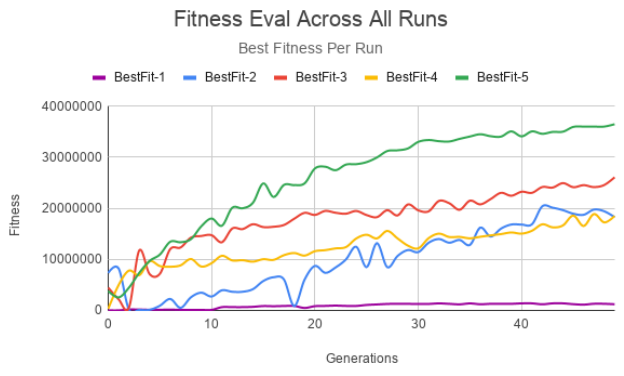

Figure 5: Best Fitness per Generation

ber neurons layer by layer for Runs 4 and 5, while keeping the numbers stable for Runs 1-3, and observe any meaningful differences. The numbers chosen generally reflect what was observed to perform well from preliminary runs. Future experimentation with a more systematized exploration of hidden layer count and number of neurons might bring more clarity to the nature of their relationship. Nevertheless, the higher neuron count of 10 in Runs 3 and 5 give some insight into what such an experiment would reveal, as it is these runs, despite having a different number of hidden layers, which outperformed the others. Even with this being the case, a direct comparison between these two, as seen in Figures 4 and 5, shows how substantially better (higher fitness score) Run 5 performs relative to Run 3, suggesting that layer count cannot easily be dismissed as the more important factor in determining performance.

Interestingly, as shown by the similarities between Figures 6 and 7, the average lap time of the best agent and the best lap time of the best agent follow almost the exact trajectories. This indicates that, when it comes to an individual neural network within an agent, there is very little variation lap to lap. This demonstrates that the learning is occurring gradually lap to lap rather than in big changes.

When comparing the best laps achieved by different agents, the results are very similar between a human driver, an RL agent, and a GA agent, with times of 11.84 seconds, 14.41 seconds, and 13.11 seconds, respectively (see Table 3). The GA agent which scored the best lap time had a neural network of 3 hidden layers with 5 neurons in the first hidden layer, and 10 in the remaining two. The results demonstrated by the GA agents prove promising, as they outperform the times achieved by the commercial RL method, with many parameters and optimization techniques associated with GA's available for future experimentation. As this study is limited mostly to neural network structure, there exists more room for study into questions such as: Do different implementations of neural networks have an effect of the evolution of an agent? How do differing recombination/operational techniques (such as NEAT) impact the results found in this study? How adaptable is a trained neural network for an agent on tracks of other shapes and distances?

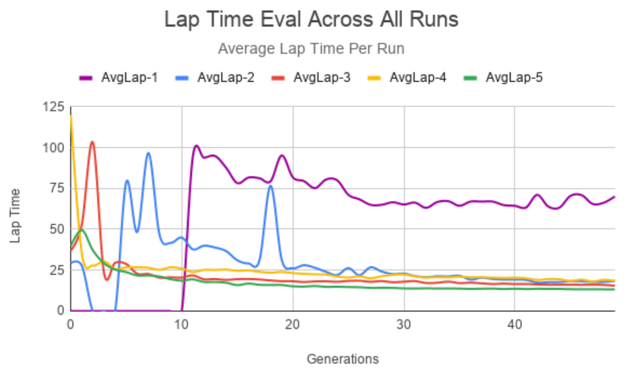

Figure 6: Average Lap Time per Generation

Run	Layers	Neurons	Gen	Best	Average
Run 1	1	2	11	63.25	63.25
Run 2	2	8, 8	2	17.04	17.44
Run 3	2	10, 10	1	14.58	15.56
Run 4	3	8, 8, 5	3	17.42	18.13
Run 5	3	5, 10, 10	1	13.11	13.24

Table 2: Results of GA Runs (Best and Average are of the Best Car in Population, Gen is the Generation of the first completed lap)

5 Conclusions

Evolutionary algorithms have already proven to outperform popular ML algorithms at training agents to self-drive in two-dimension pixel digital environments. [AbuZekry *et al.*, 2019] This work has been extended here, as a GA is used on agents in 3D environments, demonstrating their already-

Agent	Best	Average
Human	11.84	12.32
RL	14.41	14.92
GA	13.11	13.24

Table 3: Best and Average Lap Time

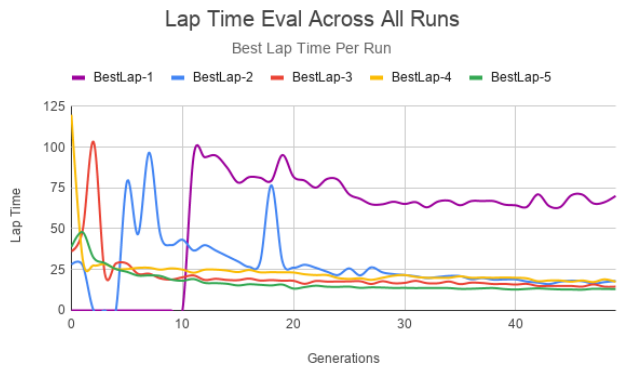

Figure 7: Best Lap Time per Generation

established ability to handle greater complexity in video game settings. [Logofatu *et al.*, 2019] [Perez-Liebana *et al.*, 2019] Without heavy experimentation on parameters, nor exploring different methods of crossover, a GA, working on a simple implementation of a neural network, was able to outperform a commercially available method of RL in training a car to self-drive around a track. It accomplished this simply by testing diverse structures of the neural network controlling the car. These results build anticipation for even more complex actions and environments for GA's to tackle in the realm of video game AI.

References

[AbuZekry *et al.*, 2019] Ahmed AbuZekry, Ibrahim Sobh, Mayada Hadhoud, and Magda Fayek. Comparative study of neuroevolution algorithms in reinforcement learning for self-driving cars. *European Journal of Engineering Science and Technology*, 2(4):60–71, 2019.

[Holland, 1975] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[Jet, 2017] Underpower Jet. Tutorial on programming an evolving neural network in c# w/ unity3d. https://www.youtube.com/watch?v=Yq0SfuiOVYE, 2017.

[Juliani *et al.*, 2020] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *Computing Research Repository (CoRR)*, abs/1809.02627:1–28, 2020.

[Kim and Ahn, 2018] Man-Je Kim and Chang Wook Ahn. Hybrid fighting game ai using a genetic algorithm and monte carlo tree search. In *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 129–130. Association for Computing Machinery, 2018.

[Logofatu *et al.*, 2019] Doina Logofatu, Florin Leon, and Fitore Muharemi. General video game ai controller-integrating three algorithms to bring a new solution. In *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2019.

[Monkey, 2020] Code Monkey. How to use machine learning ai in unity! (ml-agents). https://www.youtube.com/watch?v=zPFU30tbyKs, 2020.

[Perez-Liebana *et al.*, 2019] Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General video game ai: A multi-track framework for evaluating agents, games, and content generation algorithms. *IEEE Transactions on Games*, 11(3):195–214, 2019.

[Schuchmann, 2020] Sebastian Schuchmann. Ml-agents 1.0+ creating a mario kart like ai. https://www.youtube.com/watch?v=n5rY9ffqryU, 2020.

[SpawnCampGames, 2020] SpawnCampGames. Arcade style car controller —— unity tutorial. https://www.youtube.com/watch?v=TBIYSksI10k, 2020.