Training an Autonomous Racecar using a Neuroevolutionary Algorithm

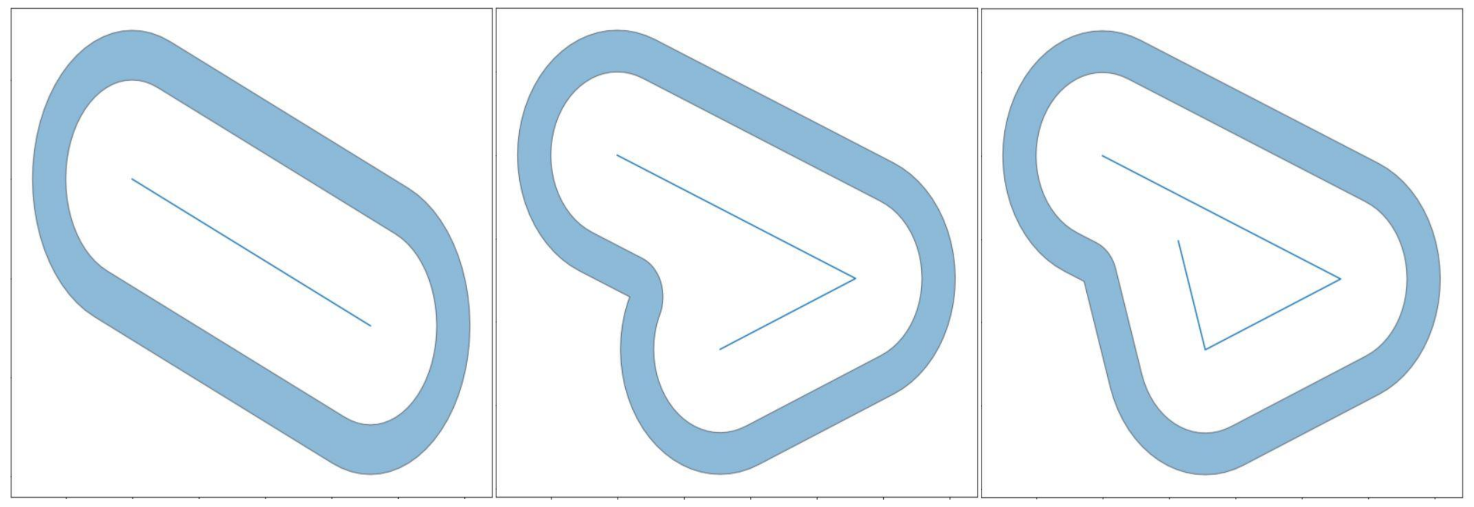
Pouya Tobias Strand Nikoui s165946 | Marius Cristian Mic s193282 | Francisc Vlad Loghin s193204
Technical University of Denmark

Introduction

Genetic algorithms are a strong contender to backpropagation-based reinforcement learning, because they are able to better avoid local minima issues due to the diversity generated by crossover and mutations. Also they have a better mean learning time due to not needing to compute all the derivatives necessary for backpropagation. Ziyi Xiang has shown that despite their higher variance genetic algorithms are well-suited for navigation tasks [1].

Curriculum Learning

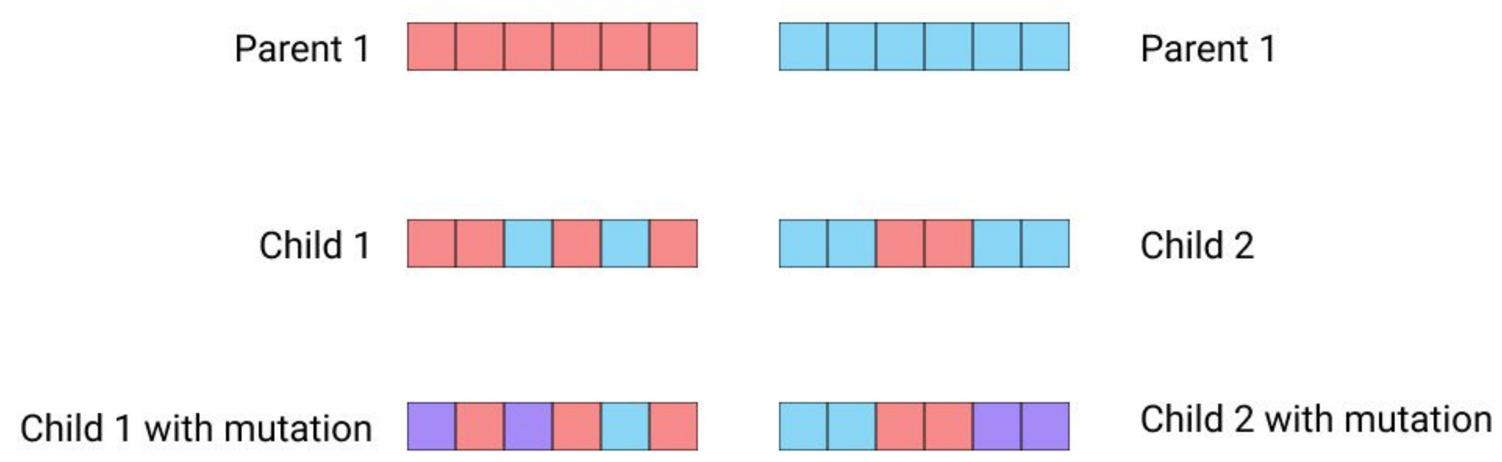
The basic idea is to start small, learn easier subtasks and slowly increase the difficulty. Researchers have found that neural networks can learn faster with a curriculum in place [2]. For this project the curriculum is to start off with a zero-turn map, save the final weights and load them to the model as a starting point for a one-turn map, then a two-turn map and so on.



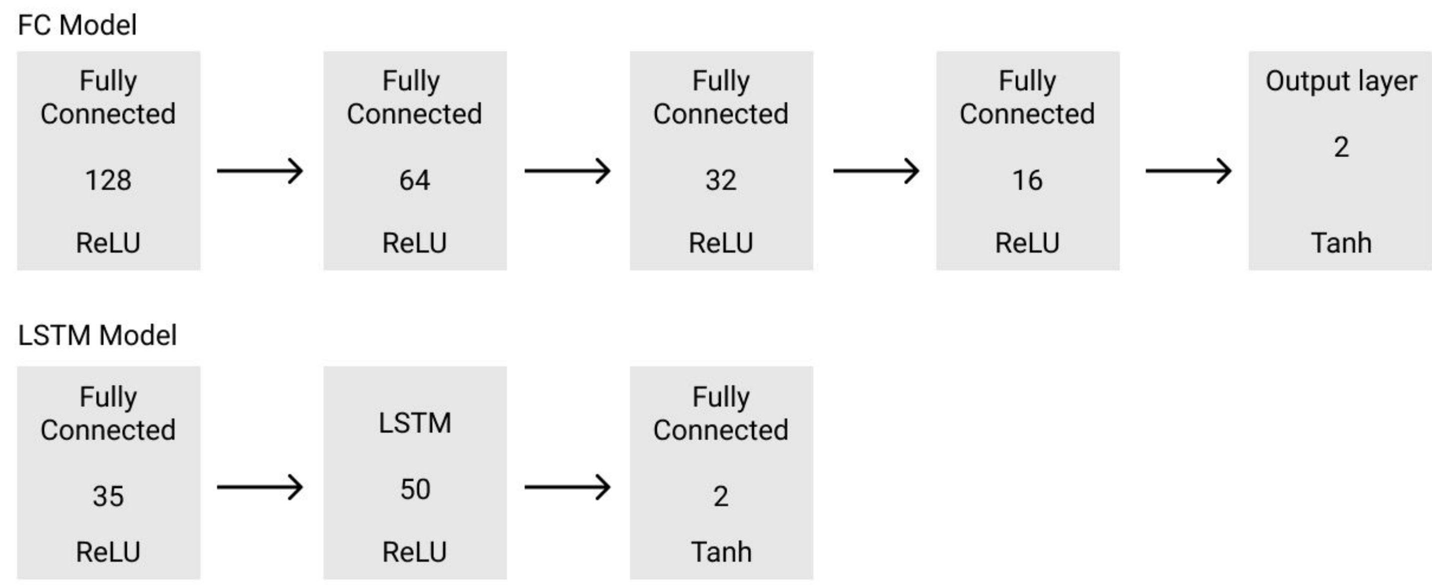
Genetic Algorithms

GAs are methods for solving optimization problems that draw inspiration from Darwinian Evolution. They start from a set of randomly generated agents and then, by observing each agent's performance the algorithm selects the most successful ones to 'pass on their genes' and be the basis for future agents. This happens repeatedly until, after evaluating multiple iterations, the models converge towards an optimal solution.

Each solution (model) represents a chromosome and each iteration represents a generation where the most successful agents in each generation are called parents (i.e. they pass their information on). Each agent inherits its solution from its parents, which is mutated randomly. In each generation the algorithm will select two parents that perform the best and use their chromosomes to create children. Children are then multiplied and mutations are applied randomly to all children, at random times. This adds stochasticity and thus more variety in the 'gene pool'. As results get better the probability of mutation gets lower and lower [1].



Models



Fitness Determination

In this report distance based penalties have been utilized to determine whether the subsequent state following a given action is desirable. Agents are originally penalized for hitting a wall or neither hitting a wall or the goal (A), however an additional penalty has been implemented for the distance to the goal of the new state being larger than the previous one (B). The fitness is thus the total penalty and the child generation is then selected, to some top limit, by ranking the results from least to greatest. The penalty functions are:

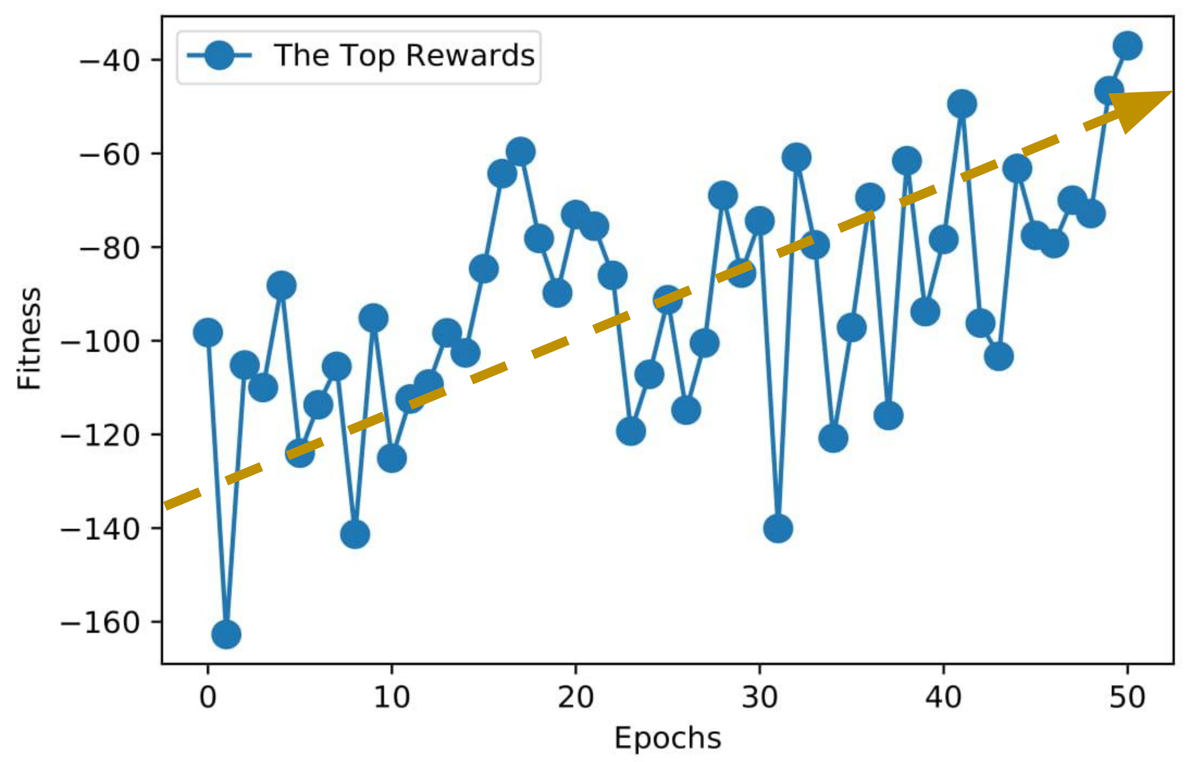
A) $$R_{Update} = R_{Update} + \sqrt{(Goal_x - Car_x)^2 + (Goal_y - Car_y)^2} \cdot 10 \cdot 3$$

B) $$R_{Update} = R_{Initial} + |(D_{Previous} - D_{Update})| \cdot 10 \cdot 3$$

Results

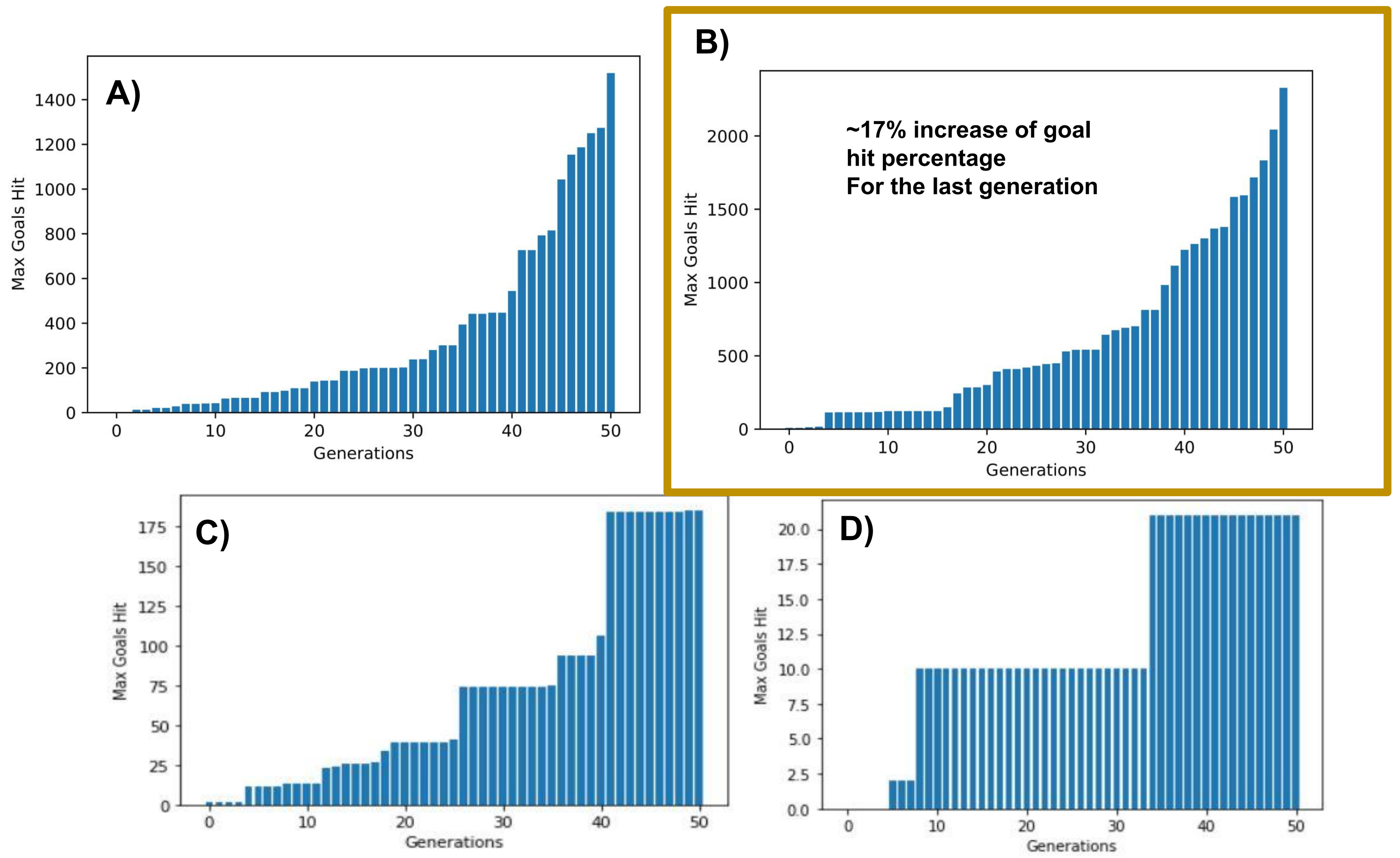
Number of Agents	Number of Generations	Crossovers	Mutation Power	Top Limit
100	51	20	0.1	10

The highest fitness among all the agents, which is negative in this case, was plotted for each generation. Below it is seen that, for a map with zero turns, the fitness fluctuates over time however the general trend is increasing and should converge toward zero given enough time.



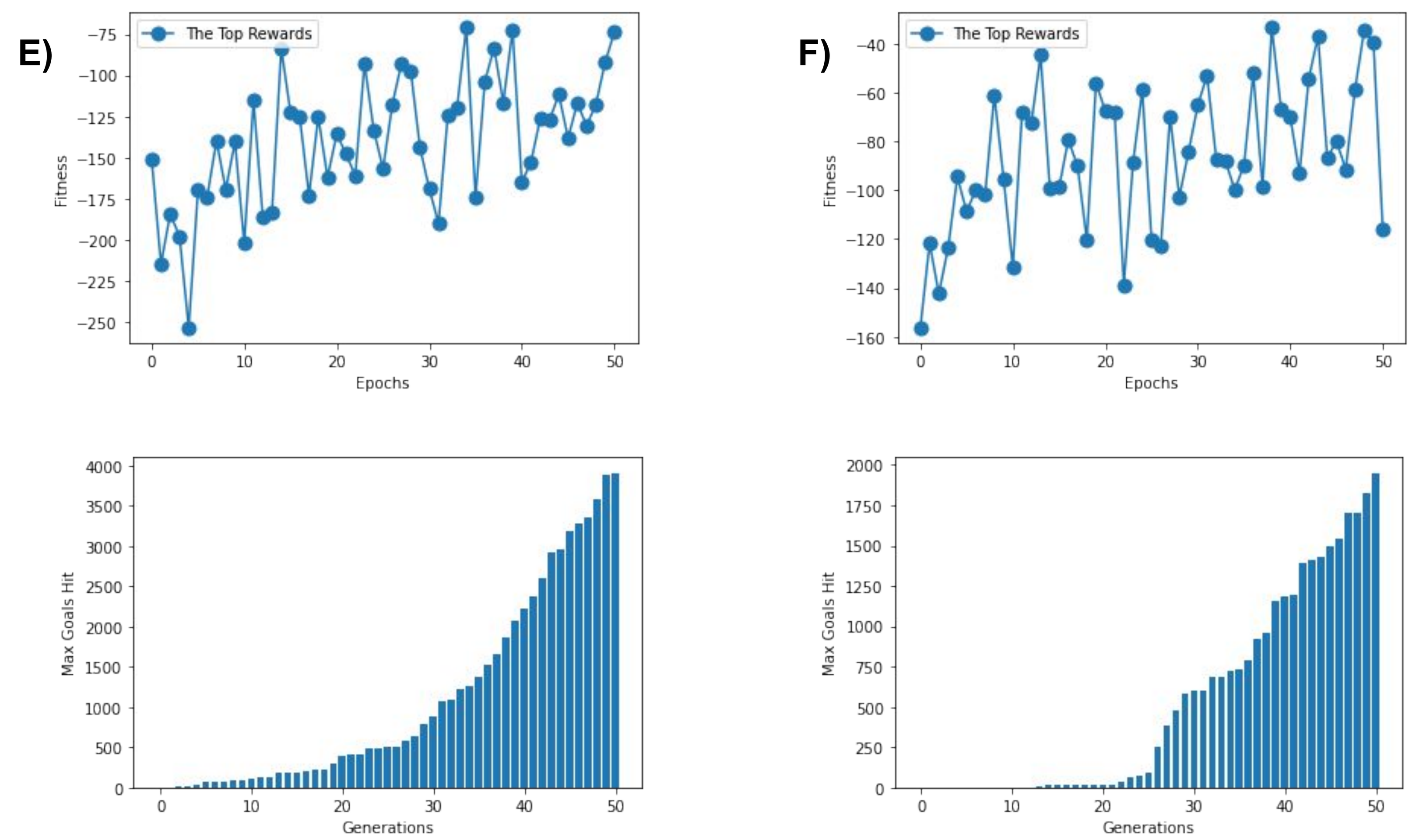
Results (FFNN)

Figure A and B show the cumulative goals hit without and with the additional penalty implementation, respectively. The additional penalty drastically outperforms the original algorithm. From the three stages of curriculum learning (0,1,2 turns) in figure B, C, D it is seen that the network learns over time and can apply what it learned to a more difficult task effectively.



Results (LSTM)

In order to investigate the performance of a different network, an RNN was tested. In figures E and F, the fitness and cumulative goals hits are plotted for maps of zero and one turn, respectively. A reduction of the plateaus are evident in the figure F, indicating that the training speed increased.



[1] Project, D., Technology, I. N., & Cycle, F. (2019). A comparison of genetic algorithm and reinforcement learning for autonomous driving. [2] Cirik, V., Hovy, E., & Morency, L.-P. (2016). Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks. 41–48. Retrieved from http://arxiv.org/abs/1611.06204
[3] National Highway Traffic Safety Administration (12.2020). National Motor Vehicle Crash Causation Survey - published July 2008. Retrieved from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059

Training an Autonomous Racecar using a Neuroevolutionary Algorithm

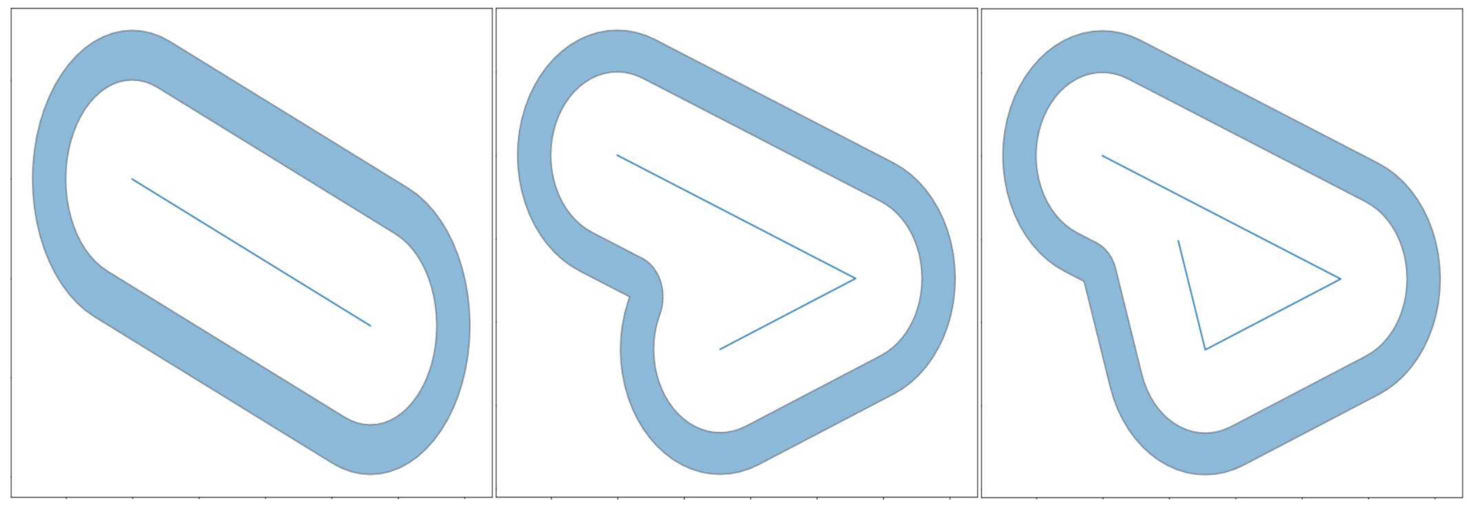
Pouya Tobias Strand Nikoui s165946 | Marius Cristian Mic s193282 | Francisc Vlad Loghin s193204
Technical University of Denmark

Introduction

Genetic algorithms are a strong contender to backpropagation-based reinforcement learning, because they are able to better avoid local minima issues due to the diversity generated by crossover and mutations. Also they have a better mean learning time due to not needing to compute all the derivatives necessary for backpropagation. Ziyi Xiang has shown that despite their higher variance genetic algorithms are well-suited for navigation tasks [1].

Curriculum Learning

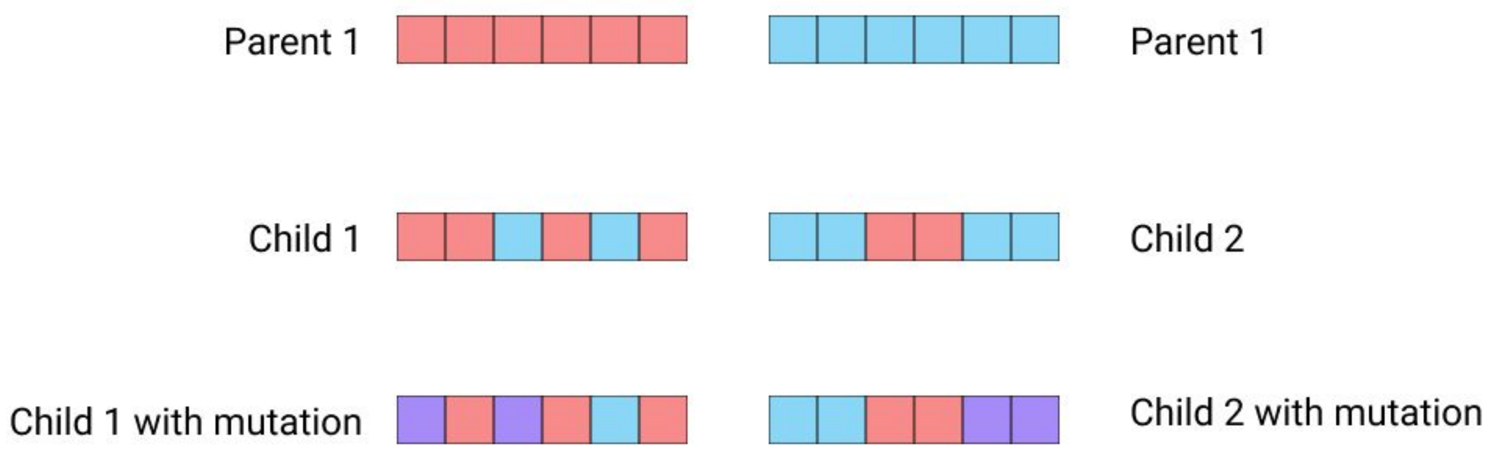
The basic idea is to start small, learn easier subtasks and slowly increase the difficulty. Researchers have found that neural networks can learn faster with a curriculum in place [2]. For this project the curriculum is to start off with a zero-turn map, save the final weights and load them to the model as a starting point for a one-turn map, then a two-turn map and so on.



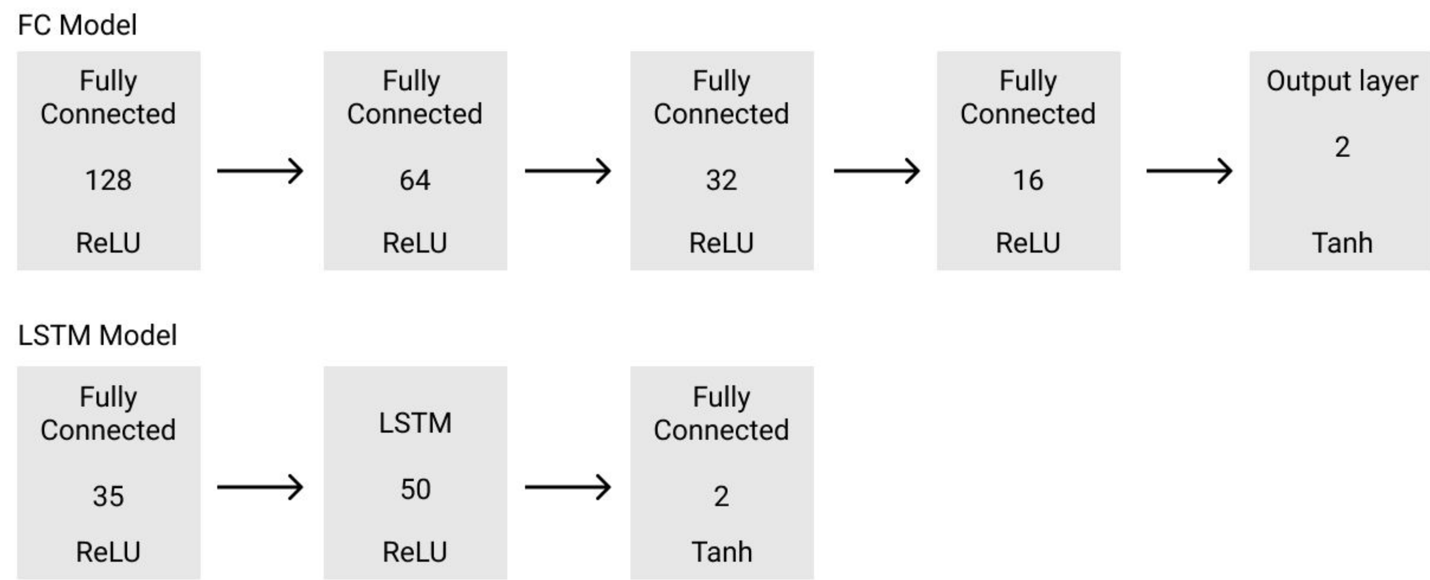
Genetic Algorithms

GAs are methods for solving optimization problems that draw inspiration from Darwinian Evolution. They start from a set of randomly generated agents and then, by observing each agent's performance the algorithm selects the most successful ones to 'pass on their genes' and be the basis for future agents. This happens repeatedly until, after evaluating multiple iterations, the models converge towards an optimal solution.

Each solution (model) represents a chromosome and each iteration represents a generation where the most successful agents in each generation are called parents (i.e. they pass their information on). Each agent inherits its solution from its parents, which is mutated randomly. In each generation the algorithm will select two parents that perform the best and use their chromosomes to create children. Children are then multiplied and mutations are applied randomly to all children, at random times. This adds stochasticity and thus more variety in the 'gene pool'. As results get better the probability of mutation gets lower and lower [1].



Models



Fitness Determination

In this report distance based penalties have been utilized to determine whether the subsequent state following a given action is desirable. Agents are originally penalized for hitting a wall or neither hitting a wall or the goal (A), however an additional penalty has been implemented for the distance to the goal of the new state being larger than the previous one (B). The fitness is thus the total penalty and the child generation is then selected, to some top limit, by ranking the results from least to greatest. The penalty functions are:

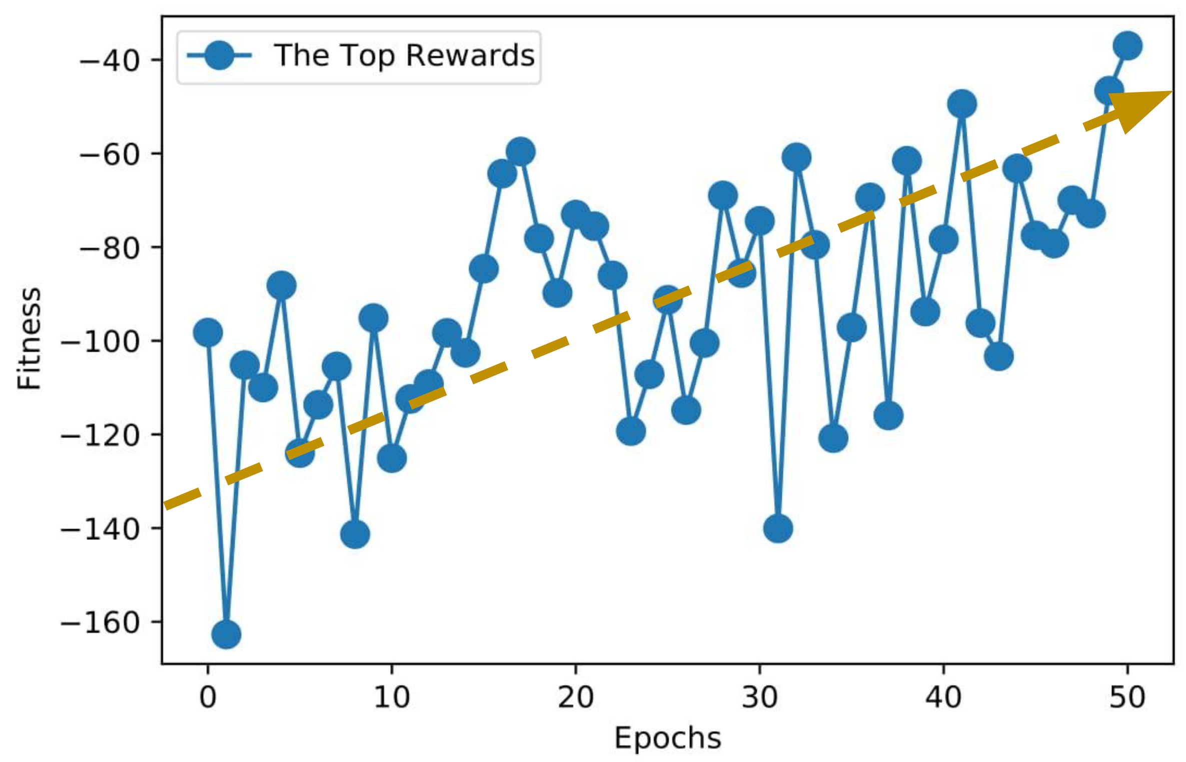
A) $$R_{Update} = R_{Update} + \sqrt{(Goal_x - Car_x)^2 + (Goal_y - Car_y)^2} \cdot 10 \cdot 3$$

B) $$R_{Update} = R_{Initial} + |(D_{Previous} - D_{Update})| \cdot 10 \cdot 3$$

Results

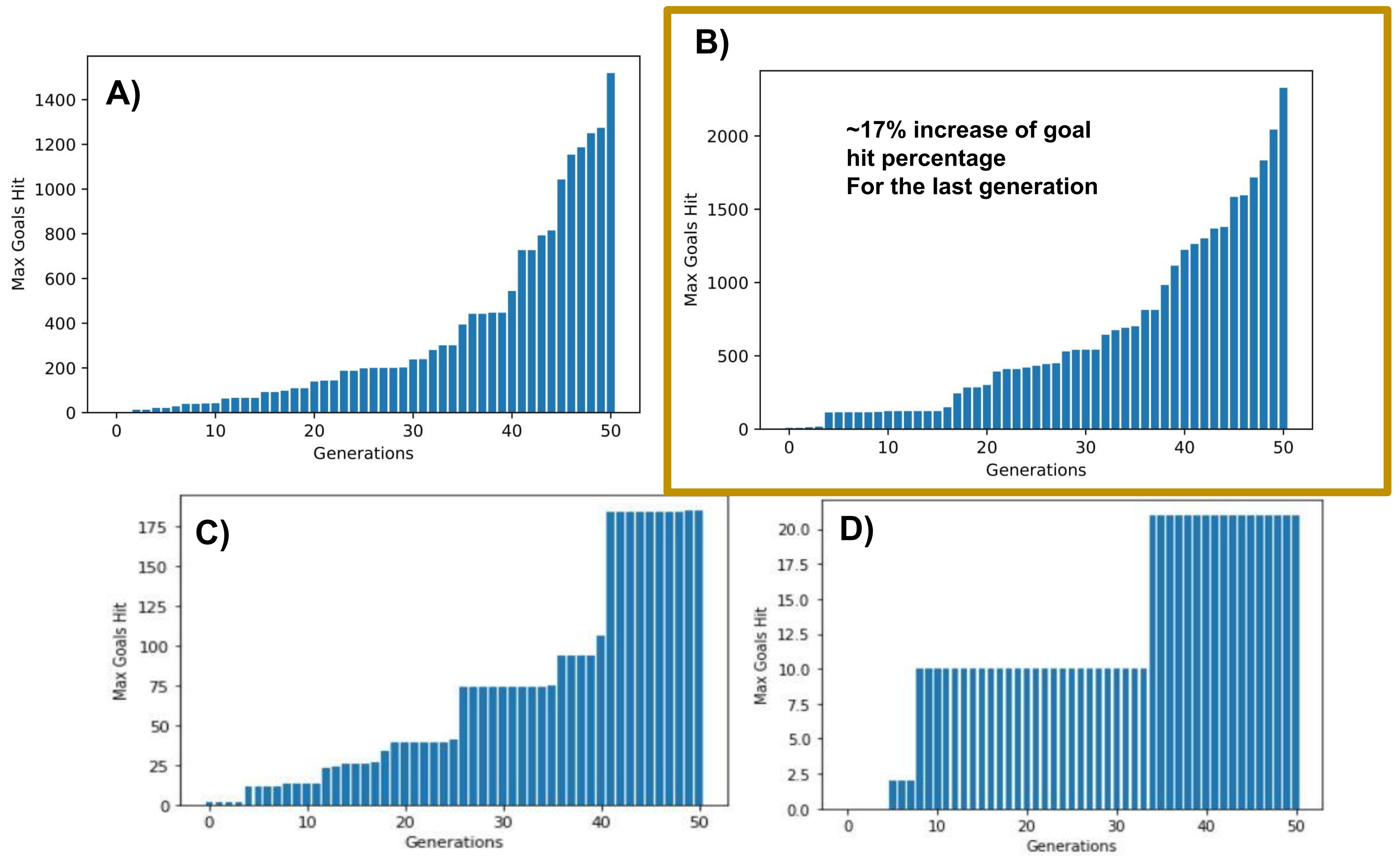
Number of Agents	Number of Generations	Crossovers	Mutation Power	Top Limit
100	51	20	0.1	10

The highest fitness among all the agents, which is negative in this case, was plotted for each generation. Below it is seen that, for a map with zero turns, the fitness fluctuates over time however the general trend is increasing and should converge toward zero given enough time.



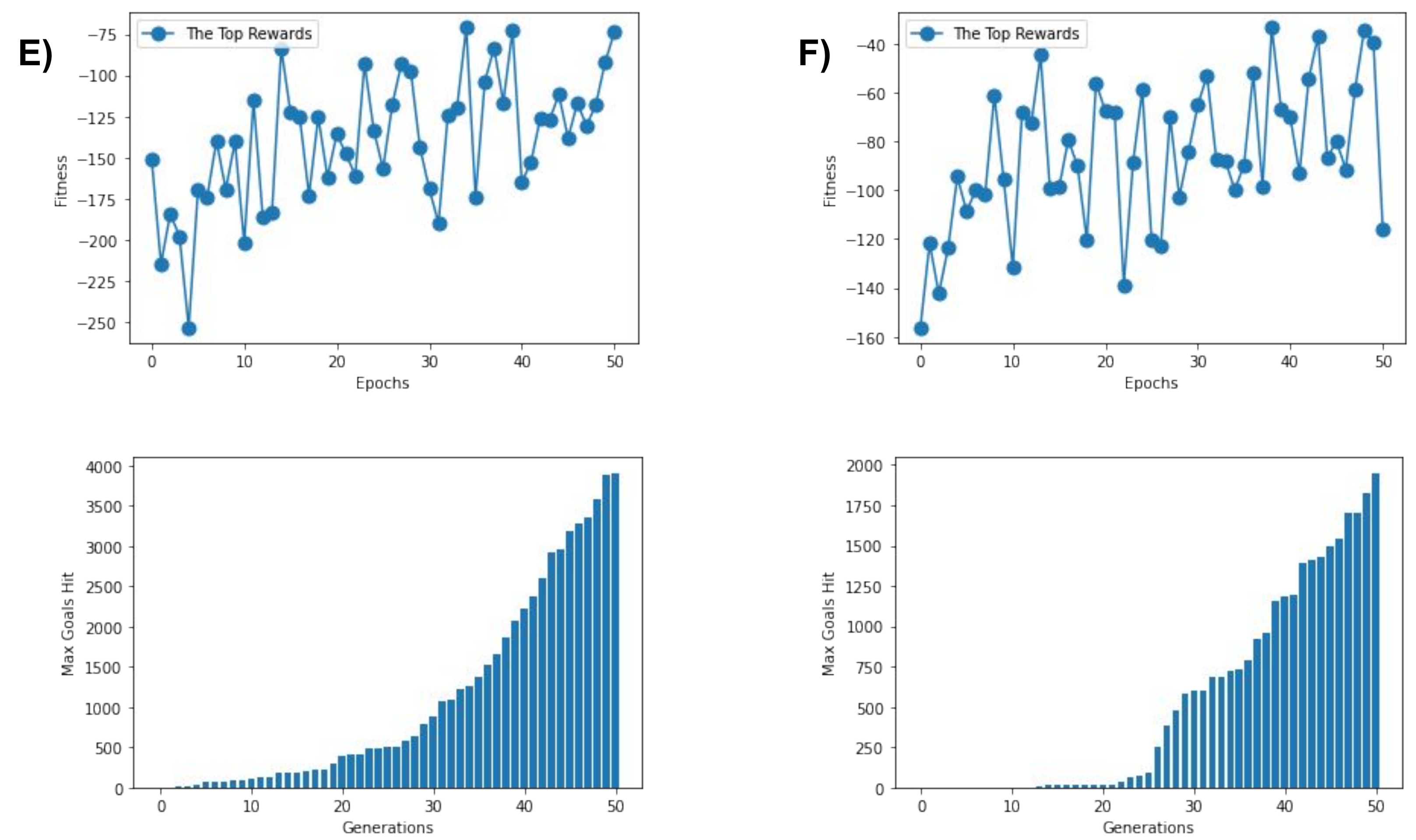
Results (FFNN)

Figure A and B show the cumulative goals hit without and with the additional penalty implementation, respectively. The additional penalty drastically outperforms the original algorithm. From the three stages of curriculum learning (0,1,2 turns) in figure B, C, D it is seen that the network learns over time and can apply what it learned to a more difficult task effectively.



Results (LSTM)

In order to investigate the performance of a different network, an RNN was tested. In figures E and F, the fitness and cumulative goals hits are plotted for maps of zero and one turn, respectively. A reduction of the plateaus are evident in the figure F, indicating that the training speed increased.



[1] Project, D., Technology, I. N., & Cycle, F. (2019). A comparison of genetic algorithm and reinforcement learning for autonomous driving. [2] Cirik, V., Hovy, E., & Morency, L.-P. (2016). Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks. 41–48. Retrieved from http://arxiv.org/abs/1611.06204
[3] National Highway Traffic Safety Administration (12.2020). National Motor Vehicle Crash Causation Survey - published July 2008. Retrieved from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059

Training an Autonomous Racecar using a Neuroevolutionary Algorithm

Pouya Tobias Strand Nikoui s165946 | Marius Cristian Mic s193282 | Francisc Vlad Loghin s193204

Technical University of Denmark

Introduction

Self-driving cars are a major area of research from both universities and the private sector nowadays. They will revolutionize transportation in a similar way that the first Ford cars did in the 19th century. Some of the important areas where they will have an impact are in road safety (where driver behaviour or error are a factor in 94 percent of crashes [3], higher independence for disabled people, reduced congestion and environmental benefits.

The two main methods of achieving this are traditional backpropagation reinforcement learning algorithms (RL) and genetic algorithms (GA).

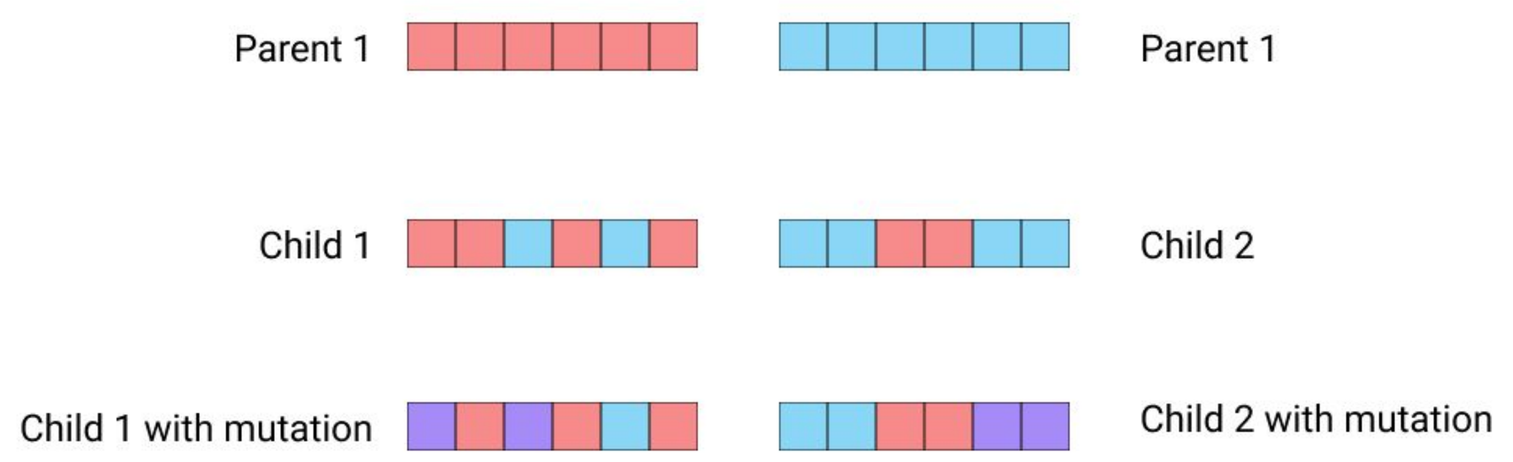
Genetic algorithms are a strong contender to backpropagation-based reinforcement learning, because they are able to better avoid local minima issues due to the diversity generated by crossover and mutations. Also they have a better mean learning time due to not needing to compute all the derivatives necessary for backpropagation. Ziyi Xiang has shown that despite their higher variance genetic algorithms are well-suited for navigation tasks [1].

In this report, GAs will be used in combination with curriculum learning. The basis for this project was provided by the people at Deloitte with whom we have collaborated throughout the project.

Genetic Algorithms

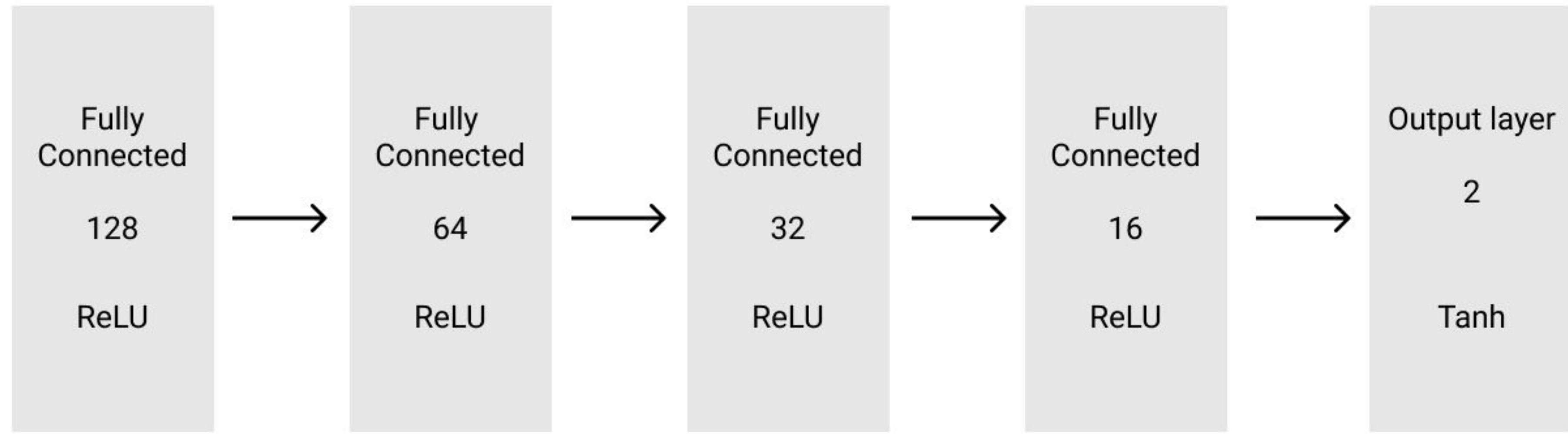
GAs are methods for solving optimization problems that draw inspiration from Darwinian Evolution. They start from a set of randomly generated agents and then, by observing each agent's performance the algorithm selects the most successful ones to 'pass on their genes' and be the basis for future agents. This happens repeatedly until, after evaluating multiple iterations, the models converge towards an optimal solution.

Each solution (model) represents a chromosome and each iteration represents a generation where the most successful agents in each generation are called parents (i.e. they pass their information on). Each agent inherits its solution from its parents, which is mutated randomly. In each generation the algorithm will select two parents that perform the best and use their chromosomes to create children. Children are then multiplied and mutations are applied randomly to all children, at random times. This adds stochasticity and thus more variety in the 'gene pool'. As results get better the probability of mutation gets lower and lower [1].



Models

Network architecture in this report:



Fitness Determination

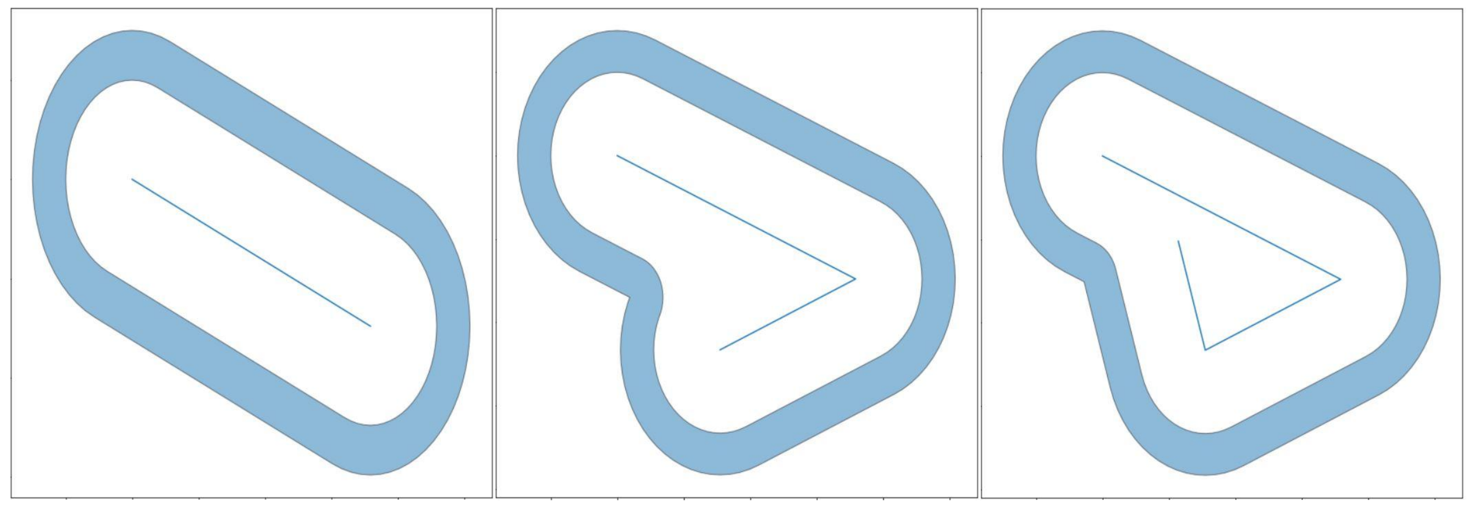
In this report distance based penalties have been utilized to determine whether the subsequent state following a given action is desirable. Agents are originally penalized for hitting a wall or neither hitting a wall or the goal (A), however an additional penalty has been implemented for the distance to the goal of the new state being larger than the previous one (B). The fitness is thus the total penalty and the child generation is then selected, to some top limit, by ranking the results from least to greatest. The penalty functions are:

A) $$R_{Update} = R_{Update} + \sqrt{(Goal_x - Car_x)^2 + (Goal_y - Car_y)^2} \cdot 10 \cdot 3$$

B) $$R_{Update} = R_{Initial} + |(D_{Previous} - D_{Update})| \cdot 10 \cdot 3$$

Curriculum Learning

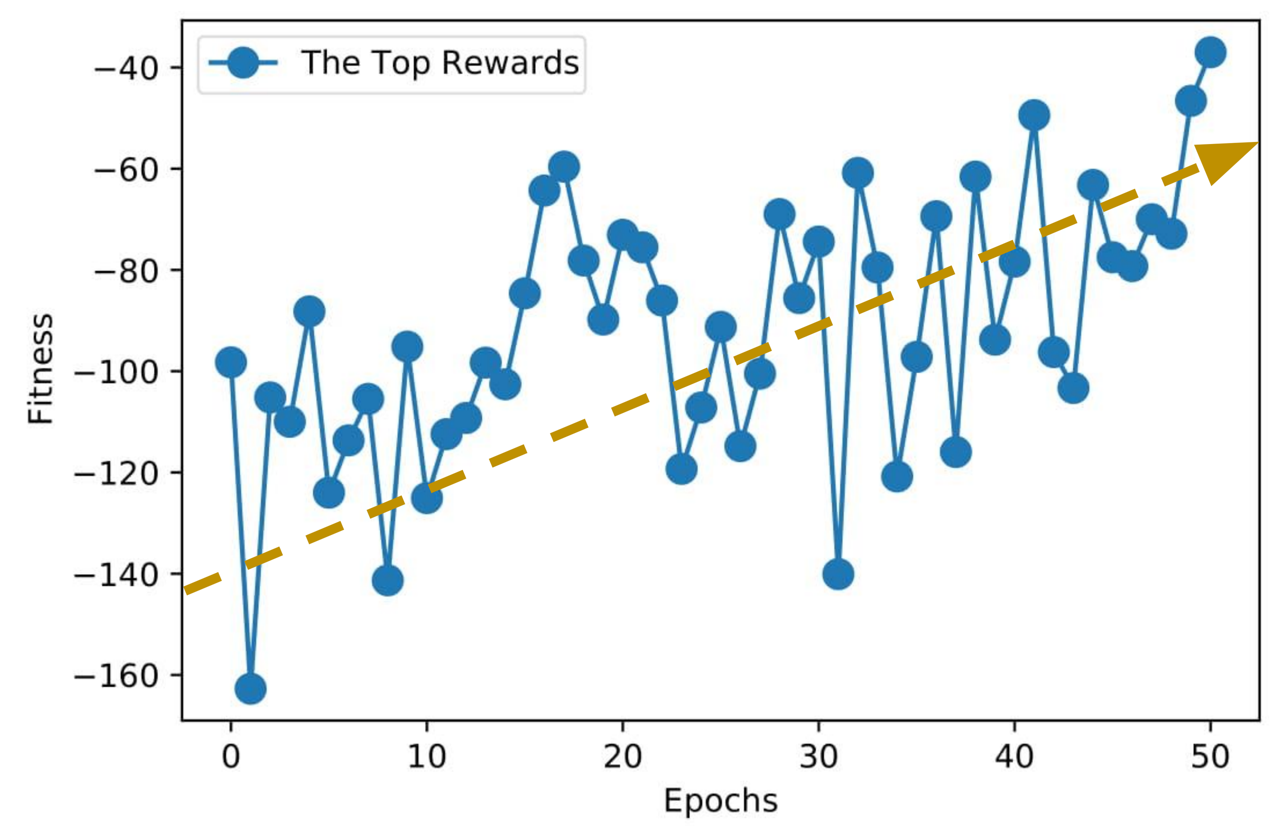
In schools human children learn following a highly organized curriculum that is meant to take them through lessons step by step, building on previously existing knowledge. Curriculum learning in the context of machine learning does exactly the same thing. The basic idea is to start small, learn easier subtasks and slowly increase the difficulty. Researchers have found that neural networks can learn faster with a curriculum in place [2]. For this project the curriculum is to start off with a zero-turn map, save the final weights and load them to the model as a starting point for a one-turn map, then a two-turn map and so on. This way the simpler features can be learned in the easier maps and transferred to the more complex ones.



Results

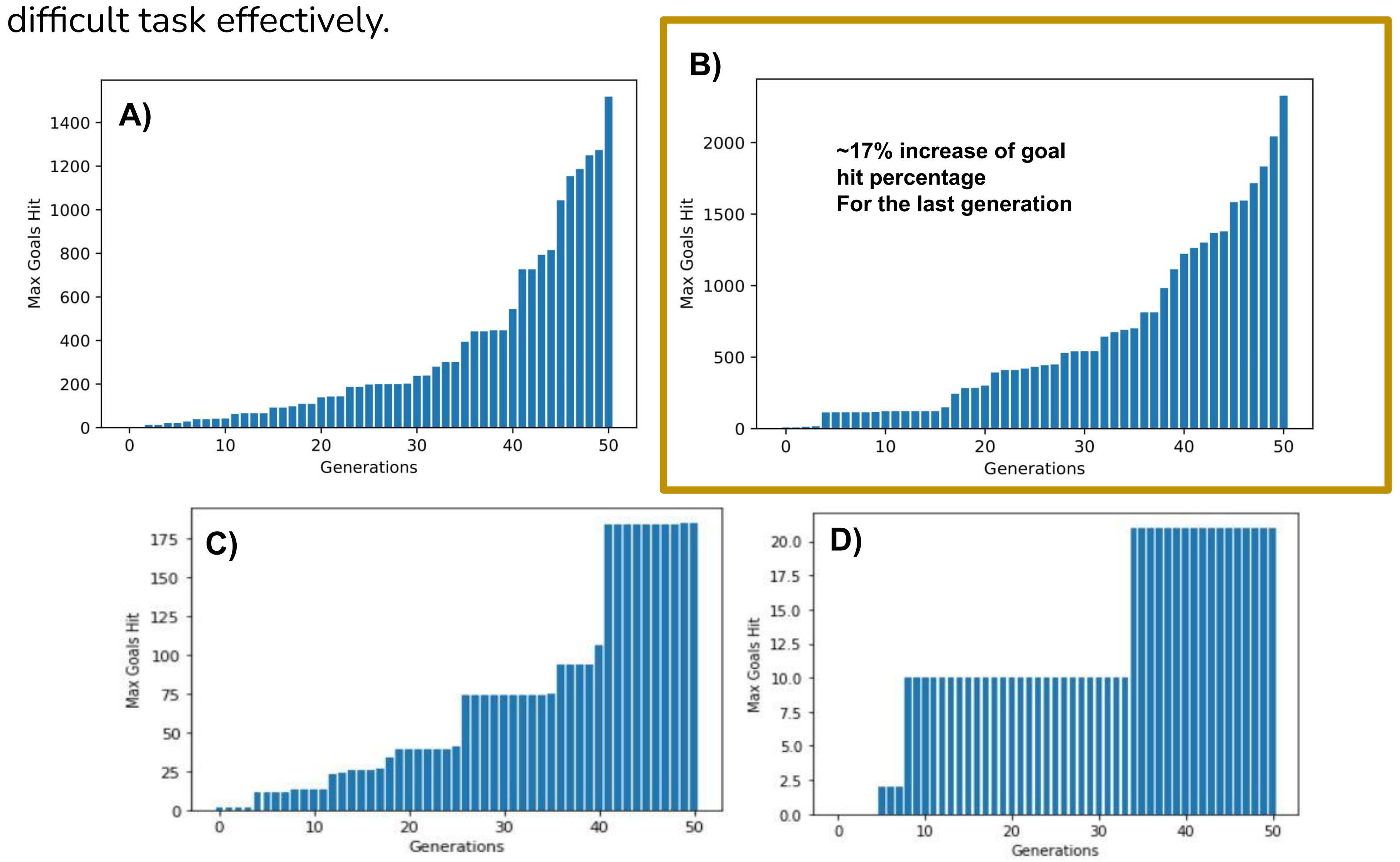
The highest fitness among all the agents, which is negative in this case, was plotted for each generation. Below it is seen that, for a map with zero turns, the fitness fluctuates over time however the general trend is increasing and should converge toward zero given enough time. The table shows the arguments passed to the training function.

Number of Agents	Number of Generations	Crossovers	Mutation Power	Top Limit
100	51	20	0.1	10



Results (Quantitative)

Figure A and B show the goals hit without and with the additional penalty implementation, respectively. The additional penalty drastically outperforms the original algorithm. From the three stages of curriculum learning (0,1,2 turns) in figure B, C, D it is seen that the network learns over time and can apply what it learned to a more difficult task effectively.



[1] Project, D., Technology, I. N., & Cycle, F. (2019). A comparison of genetic algorithm and reinforcement learning for autonomous driving.
[2] Cirik, V., Hovy, E., & Morency, L.-P. (2016). Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks. 41–48. Retrieved from http://arxiv.org/abs/1611.06204
[3] National Highway Traffic Safety Administration (12.2020). National Motor Vehicle Crash Causation Survey - published July 2008. Retrieved from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059

Training an Autonomous Racecar using a Neuroevolutionary Algorithm

Pouya Tobias Strand Nikoui s165946 | Marius Cristian Mic s193282 | Francisc Vlad Loghin s193204

Technical University of Denmark

Introduction

Self-driving cars are a major area of research from both universities and the private sector nowadays. They will revolutionize transportation in a similar way that the first Ford cars did in the 19th century. Some of the important areas where they will have an impact are in road safety (where driver behaviour or error are a factor in 94 percent of crashes [3], higher independence for disabled people, reduced congestion and environmental benefits.

The two main methods of achieving this are traditional backpropagation reinforcement learning algorithms (RL) and genetic algorithms (GA).

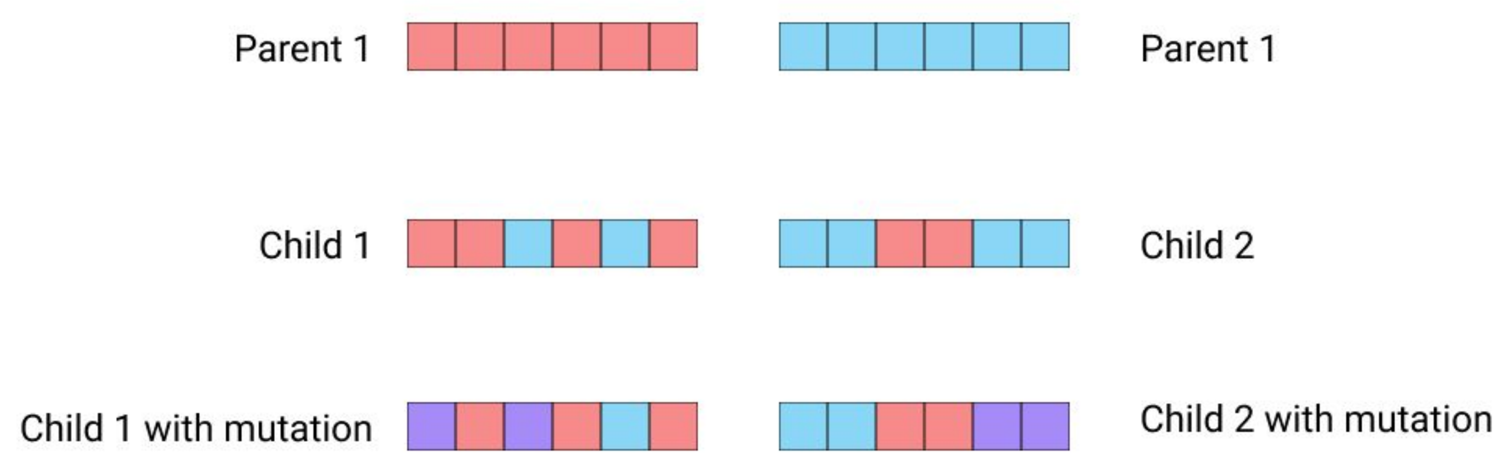
Genetic algorithms are a strong contender to backpropagation-based reinforcement learning, because they are able to better avoid local minima issues due to the diversity generated by crossover and mutations. Also they have a better mean learning time due to not needing to compute all the derivatives necessary for backpropagation. Ziyi Xiang has shown that despite their higher variance genetic algorithms are well-suited for navigation tasks [1].

In this report, GAs will be used in combination with curriculum learning. The basis for this project was provided by the people at Deloitte with whom we have collaborated throughout the project.

Genetic Algorithms

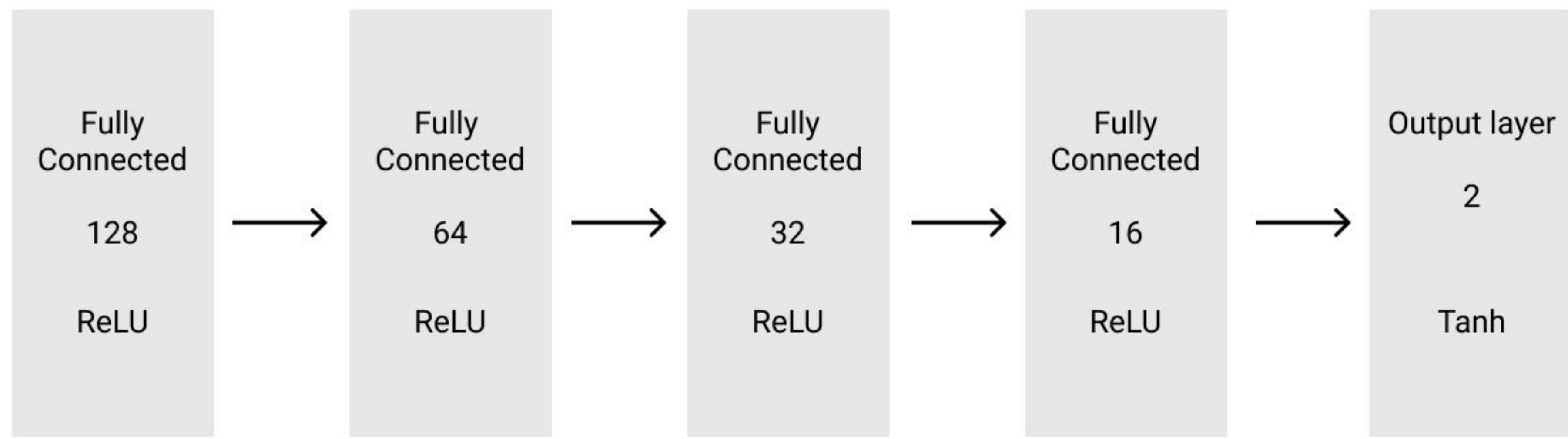
GAs are methods for solving optimization problems that draw inspiration from Darwinian Evolution. They start from a set of randomly generated agents and then, by observing each agent's performance the algorithm selects the most successful ones to 'pass on their genes' and be the basis for future agents. This happens repeatedly until, after evaluating multiple iterations, the models converge towards an optimal solution.

Each solution (model) represents a chromosome and each iteration represents a generation where the most successful agents in each generation are called parents (i.e. they pass their information on). Each agent inherits its solution from its parents, which is mutated randomly. In each generation the algorithm will select two parents that perform the best and use their chromosomes to create children. Children are then multiplied and mutations are applied randomly to all children, at random times. This adds stochasticity and thus more variety in the 'gene pool'. As results get better the probability of mutation gets lower and lower [1].



Models

Network architecture in this report:



Fitness Determination

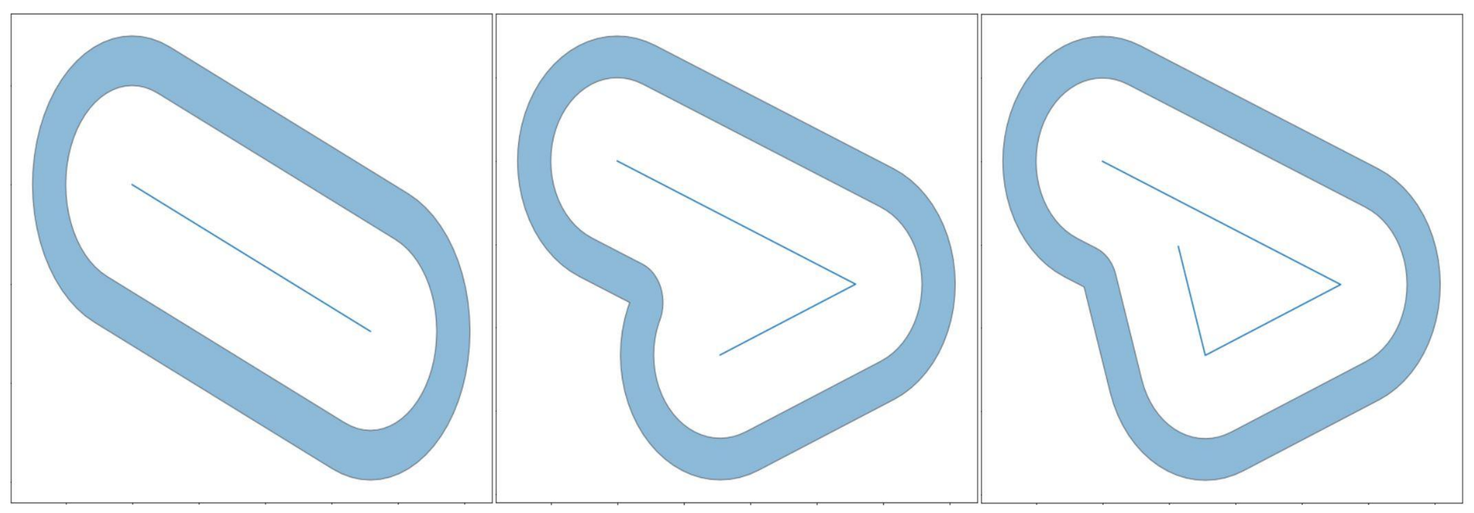
In this report distance based penalties have been utilized to determine whether the subsequent state following a given action is desirable. Agents are originally penalized for hitting a wall or neither hitting a wall or the goal (A), however an additional penalty has been implemented for the distance to the goal of the new state being larger than the previous one (B). The fitness is thus the total penalty and the child generation is then selected, to some top limit, by ranking the results from least to greatest. The penalty functions are:

A) $$R_{Update} = R_{Update} + \sqrt{(Goal_x - Car_x)^2 + (Goal_y - Car_y)^2} \cdot 10 \cdot 3$$

B) $$R_{Update} = R_{Initial} + |(D_{Previous} - D_{Update})| \cdot 10 \cdot 3$$

Curriculum Learning

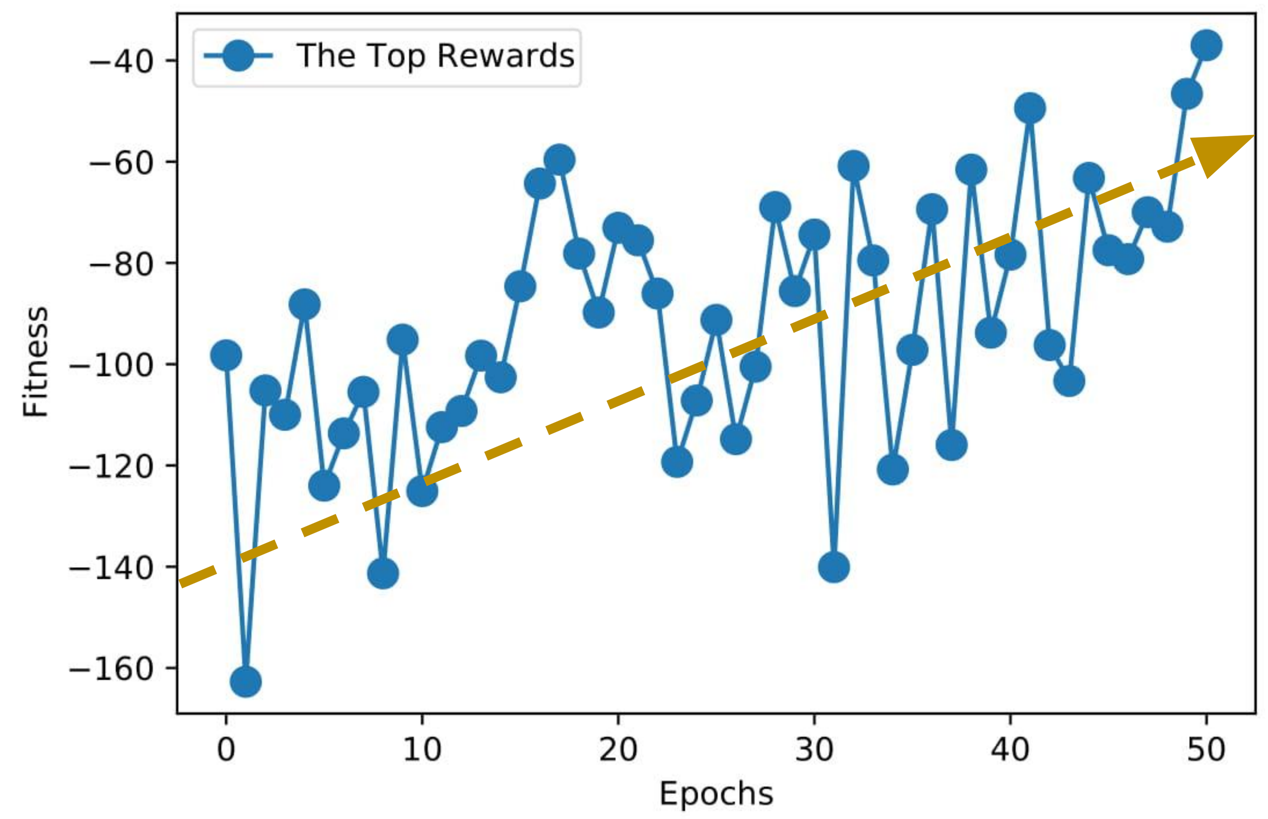
In schools human children learn following a highly organized curriculum that is meant to take them through lessons step by step, building on previously existing knowledge. Curriculum learning in the context of machine learning does exactly the same thing. The basic idea is to start small, learn easier subtasks and slowly increase the difficulty. Researchers have found that neural networks can learn faster with a curriculum in place [2]. For this project the curriculum is to start off with a zero-turn map, save the final weights and load them to the model as a starting point for a one-turn map, then a two-turn map and so on. This way the simpler features can be learned in the easier maps and transferred to the more complex ones.



Results

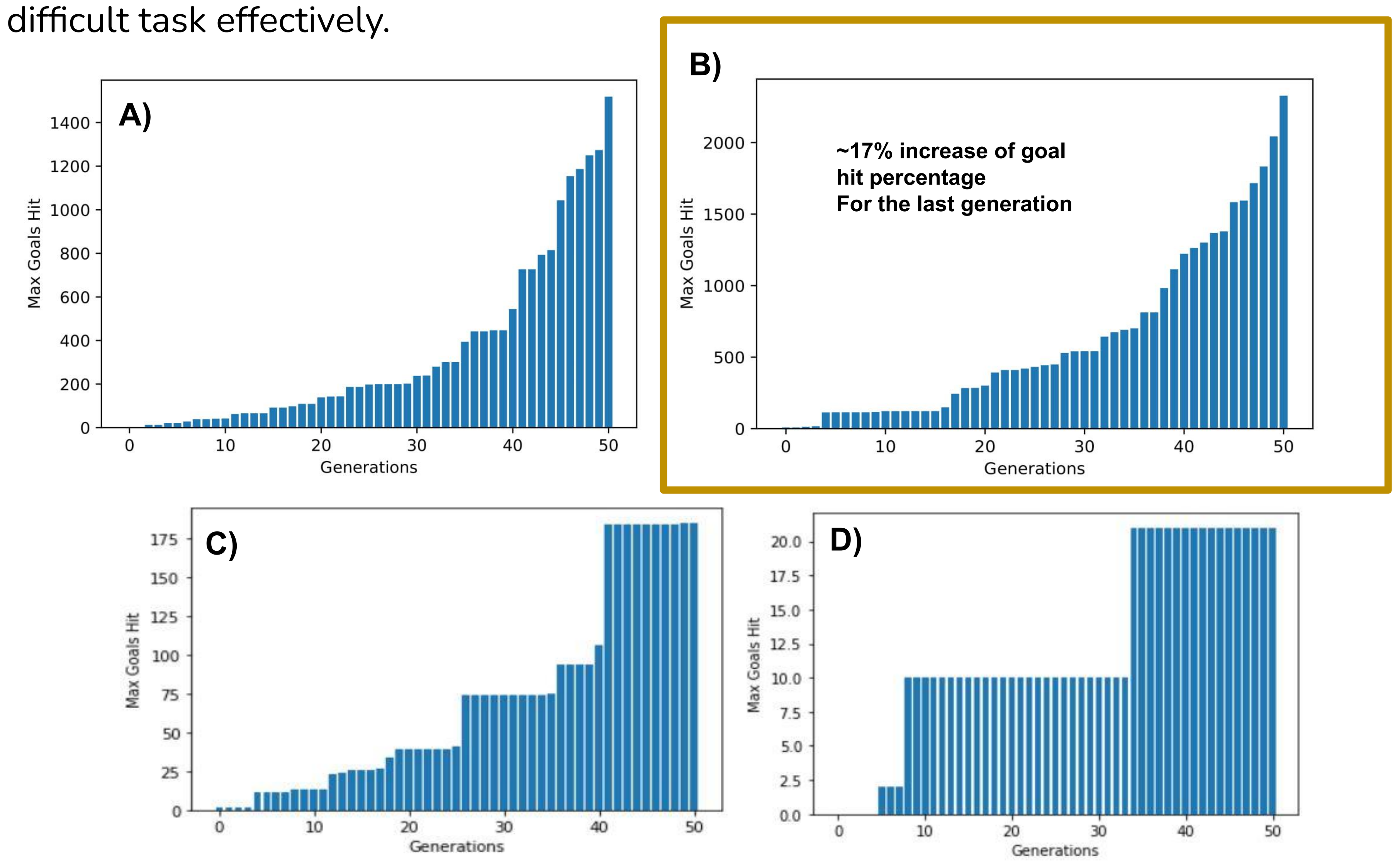
The highest fitness among all the agents, which is negative in this case, was plotted for each generation. Below it is seen that, for a map with zero turns, the fitness fluctuates over time however the general trend is increasing and should converge toward zero given enough time. The table shows the arguments passed to the training function.

Number of Agents	Number of Generations	Crossovers	Mutation Power	Top Limit
100	51	20	0.1	10



Results (Quantitative)

Figure A and B show the goals hit without and with the additional penalty implementation, respectively. The additional penalty drastically outperforms the original algorithm. From the three stages of curriculum learning (0,1,2 turns) in figure B, C, D it is seen that the network learns over time and can apply what it learned to a more difficult task effectively.



[1] Project, D., Technology, I. N., & Cycle, F. (2019). A comparison of genetic algorithm and reinforcement learning for autonomous driving.
[2] Cirik, V., Hovy, E., & Morency, L.-P. (2016). Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks. 41–48. Retrieved from http://arxiv.org/abs/1611.06204
[3] National Highway Traffic Safety Administration (12.2020). National Motor Vehicle Crash Causation Survey - published July 2008. Retrieved from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059