Autonomous Driving in Snowy Conditions

Mark-Robin Giolando
Oregon State University
Corvallis, Oregon
giolandm@oregonstate.com

Jonathan Turcic
Oregon State University
Corvallis, Oregon
turcicj@oregonstate.com

ABSTRACT

Driving through freshly fallen snow can be perilous due to the snow creating a slippery surface that causes vehicles to lose control and crash. Humans adapt to this challenge by following the tracks of other vehicles in order to stick to safer, cleared areas, and autonomous vehicles can draw motivation from this behavior to operate in wintry conditions. Our approach to this problem was to train a neural network via evolutionary algorithm to drive an autonomous vehicle capable of following the safe areas in the road that have been cleared out by previous vehicles driving on it while avoiding dangerous areas of the road that have not been cleared. Our preliminary results demonstrated an ability to solve navigation challenges on each map, and to generalize the solution to other maps, with the top performing solutions able to generalize their behavior to slightly more than half (7/12 trials) of the other maps after being trained on a different map. In the future we plan to continue modifying the parameters of our system so that it is able to find a solution that is able to generalize to all of the testing scenarios.

KEYWORDS

Autonomous Driving, Neural Networks, Evolutionary Algorithms, Snow

ACM Reference Format:
Mark-Robin Giolando and Jonathan Turcic. 2021. Autonomous Driving in Snowy Conditions. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3–7, 2021*, IFAAMAS, 8 pages.

1 INTRODUCTION

In recent years there has been a large push for research into developing fully autonomous cars that drive safely without input from the driver/passengers. Fully autonomous vehicles possess the potential to provide many benefits including: increased driver safety, reduction of traffic congestion, and increased mobility for people with disabilities who are unable to drive on their own [12]. Furthermore, a study done in 2012 showed that more than 90% of motor vehicle related accidents were caused by driver error, something fully autonomous vehicles may reduce by eliminating the driver from the equation[4].

Winter conditions exacerbate these issues, make driving even more perilous with snow and ice covering the road, and falling snow obscuring sensors, preventing the vehicle from accurately sensing it's surroundings while also making the road more slippery. Winter roads consist of several "grades" of quality dependant on how packed the snow is, how much traffic has cleared lanes in the snow and the existence of ice on the road. These values will impact the traction the vehicle will have, thereby potentially degrading the vehicle's control leading to more crashes. The snow can also cover the lane dividers increasing the confusion of where to drive. These factors greatly increase the danger of driving with approximately 17% of all car accidents occurring in winter conditions with over 115,000 people being injured and 1,300 people being killed annually in car accidents on snowy or icy roads [3].

In snowy conditions, repeated driving in a certain area tends to create new lanes in the road. These new lanes consist of a narrow safe zone where the snow has been crushed into either packed snow or melted, surrounded by semi-safe zones. The width of these zones is formed by, and defined by how many times a vehicle has driven over the area. Outside of these zones are areas vehicles have not driven on, which are very dangerous due to the fresh, unpacked snow. Driving in these areas results in the vehicle losing control, being unable to turn or stop. This can result in the vehicle colliding with other vehicles or ending up in ditches.

The same difficulties of driver operated vehicles in winter conditions are also present in fully autonomous vehicles since there is such a small margin or error in the position of the vehicle on the road. Even a small exposure to the unsafe road outside of the safe zone of packed/melted snow can lead to the vehicle losing control and crashing leading to injury of the passengers. Time is also an important factor when driving in winter conditions. If there is snow falling, as time progresses the road conditions will get worse and worse so driving quickly should be a priority. But the faster that a vehicle is driving, the more prone it will be to making small errors that could lead to a crash. Due to this, there is a tradeoff between the speed that it is safe to operate the vehicle and attempting to minimize the time that is spent driving in the bad conditions. This tradeoff along with the narrow margins of error lead to this being a difficult task to solve.

We intend to solve the problem of dangerous driving in winter conditions by training an algorithm to be able to autonomously drive a vehicle within the safe areas that have been previously cleared out by other vehicles. The system will look at the road to determine which areas of the road are safe for the vehicle to drive on and the vehicle will attempt to stay within this area while driving. The algorithm will attempt to drive as quickly as possible, while maintaining safety by not traveling over slippery areas where other vehicles have not traveled. The algorithm uses a reward fitness function that prioritizes cleared areas by giving the vehicle bonus points for staying within them. Smaller rewards are given for areas that are covered with slush and are less safe to drive on. The vehicle is also further incentivized to stay near the center of the cleared area. Any actions that lead to the vehicle encountering the areas marked as being covered in fresh snow result in the network failing. The algorithm will return speed up/slow down values as

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, London, UK.

well as right/left/straight directions based on the values of the map surrounding the vehicle.

2 BACKGROUND

2.1 Challenges in Driving in the Snow

Autonomous driving in the snow is challenging for a variety of reasons. Snow creates driving difficulties by reducing the friction between the vehicle's wheels and the pavement resulting in an inability to control the direction or speed of the vehicle. A further issue is that snow creates numerous perception issues ranging from obscuring obstacles and signs to creating false positives. [18]. To the best of our knowledge, there has not been any previous work that has combined solutions to all of these problems with an autonomous vehicle that is able to reliably drive in snowy conditions. However, there has been extensive work in these areas individually. The next sections will discuss the previous work that has been done to attempt to solve these individual problems.

2.2 Previous Work

2.2.1 Autonomous Driving. Previous work by Ozturk et al [13] used curriculum reinforcement learning to address different driving scenarios such as driving in inclement weather. Their work included simulating the effects of different types of weather on the simulated vehicle. One important consideration is that they treat the road as having uniform values per section (e.g., the rainy parts are equally wet across the road), whereas our work assigns different values to the road. Other work has developed algorithms to use a human to train an algorithm to navigate through bad conditions [15].

More work in this area involves using a reinforcement learning algorithm to train a vehicle to drive on a road with varying conditions including snow, coast, and cliffs as well as different types of turns [9]. This work is similar to ours in that it involves autonomous driving in adverse conditions but differs from our work since it assumes that the car will be able to use road markers as guidelines to follow while in our work, the car will not be able to see the road and will instead be following paths created by other vehicles. Their work also differs from ours in that they plan on driving on the slippery portions that are covered in snow, where our work aims to avoid that danger.

2.2.2 Sensing. Due to the difficulty of autonomous vehicles sensing their surroundings in snowy and rainy conditions, there has been significant amounts of research done in the area of cleaning up sensor readings. Previous work in sensing involved detecting objects, and cleaning images for increased clarity [11]. Other work focuses on how to use landmarks in snowy landscapes to perform localization on the autonomous vehicle [1]. More work involved evaluating the quality of the road conditions [14] or improving the response from LIDARs degraded by snow [2]. Rawashdeh et al, fused sensor data from LIDARs, radars, and camera data to detect drivable paths in snowy conditions [16]. However, for the sake of our work, we will be dealing primarily with the driving challenge, assuming the sensing problems are already solved and processed images can be provided to our algorithm.

2.2.3 Lane/Path Following. As our algorithm will be looking at the path in the snow that was left behind by previous vehicles and determining the portion of this area that is safe for the vehicle to drive in, our work essentially boils down to a path following algorithm. Extensive research has been done in this area including lane following algorithms that use the road markers to determine where the vehicle can drive [10] and other methods that involve driving on roads that do not have distinct markers for the car to follow [5] [8]

One of these methods that was created by Caraffi et al uses computer vision systems to allow autonomous vehicles to follow gravel/dirt roads that do not have distinct road markers [5]. This involved classifying which parts of the road are drivable and which are off-road and unable to be recognized by the sensors. This is similar to our work since we will also be looking to classify the path in the snow into parts that are safe to be driven on and unsafe to be driven on. However, this work differs from ours since they are still following a clear road with no snow, even though there are no road markers, while our work is going to be following a path that was created by another vehicle driving through the snow.

Another method created by Fassbender et al involves following another car on gravel/dirt roads while matching the speed of the car [8]. In this method, the follower car is looking at the road for the tracks that are left behind by the lead vehicle and uses these to traverse the road. The speed of the follower car is also determined by the speed of the car that is leading. This work is very similar to what we are trying to do as it involves following tracks that are left behind on the road by another car. However, it differs from ours in that it requires a lead car for the following car to determine what speed it should use.

2.3 NeuroEvolution of Augmenting Topologies

NEAT (NeuroEvolution of Augmenting Topologies) is a type of evolutionary algorithm for artificial neural networks. The NEAT functionality is enabled by the NEAT-Python library[6], and makes modifications to both weights and nodes. During the mutation step, nodes can be removed or added in addition to connections. This provides a signature ability to automatically address how many hidden nodes a solution requires, though a minimally sized network is prioritized. NEAT algorithms will often begin with 0 hidden nodes. This, combined with the slow, incremental rate of growth of structural mutations results in the sequential networks being as small as possible.

The algorithm progresses through a set number of generations, each produced by reproduction and mutation from the fittest genomes of the previous generation as evaluated by a fitness function evaluating the quality of individual genomes. Genes are made up of connection genes and node genes make up these genomes and can be added, removed or modified. Genome candidates are grouped into species based on how close they are to one another by genomic distance. NEAT contains more competition within these species than between the different species.

The use of speciation allows the system to tell which ancestors the genes came from by tracking a global innovation number that marks the order in which genes were created, enabling the crossover mutation. This also enables genomes to inherent genes from the better performing parent. Solution diversity is also protected by

having solutions compete within their own species, preventing one solution set from dominating all others and stifling innovation. [17]

2.4 NeuralNine Car Simulator

For our work, we took advantage of an existing car simulator developed by NeuralNine [7]. The simulator developed a solution to navigating a race track as quickly as possible with a priority placed on a lightweight solution. To this end, the NeuralNine AI Car opts not to use hidden layers, and uses 5 input measurements of the track around the vehicle to choose one of four actions: increase speed by 2, decrease speed by 2, turn right by 12 or turn left by 12. On more complicated tracks, they opted to only use two output nodes for turning right and left by 12 degrees. Their solution was able to solve multiple tracks, but struggled at times to deal with narrow paths, such as those that will arise in our problem space. Their solution also prioritized finding a solution that would simply navigate the track without crashing while our solution was focused on how safely the car was able to drive around the track. Pygame was used to demonstrate the learning and performance of the cars.

We further expanded upon the simulation by dividing the track into areas of differing values, and expanded with a more complex fitness function. We also increase the measurements that the car is making 5 to 7 sensor readings adding more inputs to the neural networks. The outputs are also changed in our solution, including a fifth option for the vehicle to take, that of maintaining direction and speed. We also modified the feed forward neural network architecture and mutation settings.

3 METHODS

3.1 Algorithm Description

Our solution to the danger of driving in the snow will involve training a neural network to be able to follow cleared paths that were left behind in the snow by other vehicles. We will be simplifying the road to be three areas that are as follows: the green areas that the vehicle is safe to drive in, the yellow areas that it is unsafe for the vehicle to drive in, and the white areas which will cause the vehicle to crash. An example of the road used can be seen in Figure 1.

We used an Evolutionary Algorithm to modify the initially random neural networks to search for a network that is able to complete the testing tracks safely and without crashing. Each generation of the Evolutionary Algorithm there will be 30 cars that attempt to drive safely on this road and each of these cars will have their own neural network that they will use to choose their actions. The inputs to the neural network will be seven sensor readings that will be computed at each step of the simulation. Each of these sensors will return the distance from the center of the car to the edge of the green safe zone at various angles and these distances will be the inputs to each car's respective neural network. More information about the sensors is available in the sensing section below.

Every car in each generation will attempt to drive within the safe zone as best as it can. The fitness of each of the neural networks will be computed based on the following criteria: the time that the car was in the safe zone, how close the car was to the center of the safe zone, and the total distance that the car traveled in the whole generation. More information on the fitness function is available in the fitness function section below. Using the fitness values for each

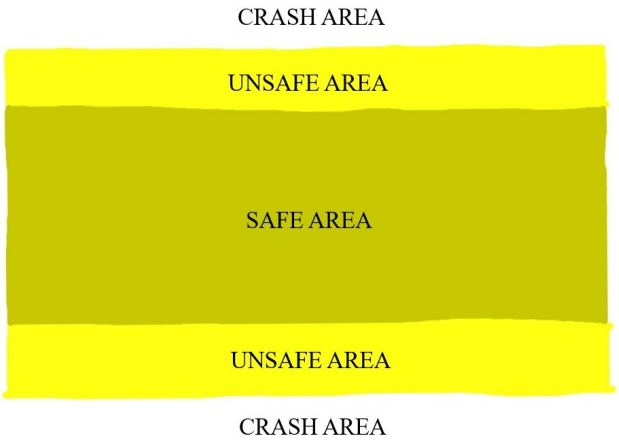


Figure 1: Safe, Unsafe, and Crash Areas of Simplified Road Model

of the neural networks, the evolutionary algorithm will be used to determine which cars will be carried on to the next generation and which will be modified/mutated with the goal of increasing its fitness. More information about the parameters of the evolutionary algorithm is available in the evolutionary algorithm section below.

Once all generations of the evolutionary algorithm are completed or the desired fitness value is reached, the neural network that returned the highest fitness value will be tested on different tracks of varying difficulty that contain features that were not present on the training track.

3.2 Assumptions

Due to the scope of this project, there were multiple simplifications that were made to both the model of the road as well as the model of the cars. As was mentioned above, we simplified the model of the road to simply the safe area (green), the unsafe area (yellow), and the area in which the car crashes (white). In real life, these areas of the road would not be this clearly defined but this simplification was made because accurately modeling the paths that are created by vehicles driving through snow is outside the scope of this class.

Another simplification that was made was the kinematic model of the cars. We simplified the model of the car to be essentially a point that can be rotated in any direction. Realistically, only the front wheels of the car would be able to be turned and the motion of the front and back wheels would be different and would both be dependent on the steering angle of the front wheels. But this simplification of the kinematic model of the car was made because generating an accurate model of the kinematics of a four wheeled car is outside the scope of this class.

3.3 Sensing

The inputs to the neural network are determined by the distances that are returned by the seven sensors present on the vehicle. There is a sensor on each side of the car that points outward and one on the front of the car that points forward. There are also two more sensors between these on each side of the car that point forward at

various angles. The names of these sensor values in order starting from the driver side of the car and going clockwise are D1, D2, D3, D4, D5, D6, and D7. The vehicle with labeled sensor values can be seen in Figure 2. These sensors move outward from the center of the car and will stop when they reach the edge of the green safe zone. If the car is not near the edge of the safe zone, the sensor will stop at a certain defined maximum distance. Each of these sensors will return the distance from the center of the car to the edge of the green safe area and these seven distances will be fed into the neural network as inputs.

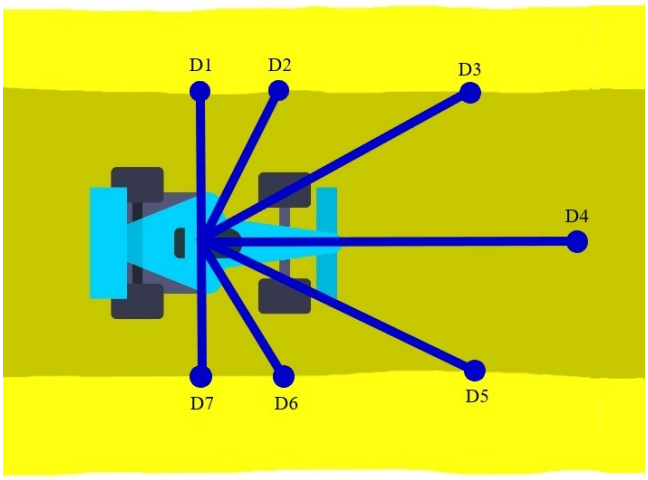


Figure 2: Labeled Sensors that Feed Distance Returned into Neural Network

3.4 Feed Forward Neural Network

We chose to use a feed forward neural network to choose the actions that the cars will make based on the inputs that are returned from the sensors. For our work, we found a neural network with 7 inputs, one hidden layer with 3 hidden units, and 5 outputs as seen in Figure 3 worked best for the cars to be able to generalize their actions to most tracks. If the number of inputs or outputs of the system was increased, a neural network that was able to complete the training track without crashing could be found but the system became too complex for the neural network to be able to adequately generalize to the testing tracks. If the number of hidden units was increased, the neural networks would over fit to the training track and be unable to complete the testing tracks without crashing. The activation function that was used for the hidden and output nodes was chosen to be a sigmoid function. The system was also able to function when a hyperbolic tangent activation function was used for the hidden and output nodes but the results were not significantly different from when the sigmoid activation function was used so we decided to use the sigmoid function for simplicity.

As was mentioned previously, the inputs to the neural network are the seven distances that are returned by the sensors. At each time step, these values are fed through the neural network and the output node with the highest value is chosen as the action for the current time step. The possible actions are as follows: turn left which changes the angle of the car by 4 degrees, turn right which changes the angle of the car by -4 degrees, speed up which increases the speed of the car by 2, slow down which decreases the speed of the car by 2, and maintain speed and heading which keeps the speed and angle of the car constant. The minimum speed was set to be 10.

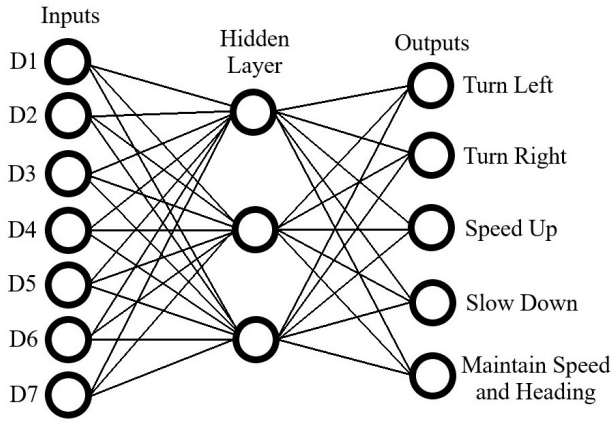


Figure 3: Structure of Best Neural Network Found using Evolutionary Algorithm

3.5 Evolutionary Algorithm

We decided to use an evolutionary algorithm to modify parameters of the neural networks to find a network that is able to complete the training track successfully. At the start of the algorithm, a population of 30 neural networks is created with all of their parameters being initially random. This first generation of random networks is then used to control the actions of the population of the 30 cars based on their sensor inputs. The generation is completed after it runs for 1200 time steps or after all of the cars in the population have crashed. At this point, the three neural networks that have the highest fitness values are directly passed into the next generation without any modifications. The rest of the population is then filled with mutated versions of the best networks in the population. This process is then repeated for a set number of generations or until a network with a suitably high fitness value is found.

The parameters of the neural networks that can be mutated in each generation of the evolutionary are the values of node biases of the hidden and output layers and the values of connection weights both in the input-hidden layer as well as the hidden-output layer. There is also a chance that the structure of the neural network can be mutated between generations with a chance of hidden layer nodes and node connections in the input-hidden and hidden-output layers being and added or deleted. Initially the system starts with zero hidden units in the hidden layer all nodes in the network are assumed to be fully connected. Table 1 outlines information on how the previously mentioned values are mutated in each generation of the evolutionary algorithm.

Parameters Mutated	Variables Needed to Mutate Parameters
Node Bias	**Initial Node Bias Mean = 0.0** **Initial Node Bias SD =1.0** **Node Bias Max = 30** **Node Bias Min = -30** **Node Bias Mutate Power = 0.5** **Node Bias Mutate Rate = 0.7** **Node Bias Replace Rate = 0.1**
Connection Weights	**Initial Node Bias Mean = 0.0** **Initial Connection Weight SD =1.0** **Connection Weight Max = 30** **Connection Weight Min = -30** **Connection Weight Mutate Power = 0.5** **Connection Weight Mutate Rate = 0.7** **Connection Weight Replace Rate = 0.1**
Node Connection	**Connection Add Probability = 0.5** **Connection Delete Probability = 0.5**
Nodes	**Node Add Probability = 0.2** **Node Delete Probability = 0.2**

Table 1: Mutation Parameters for Evolutionary Algorithm

The node bias and connection weight mutations both share similar parameters. The mean and standard deviation (SD) represent the distribution of newly created node biases or connection weights with the mean being the average and the standard deviation being the range in which these values can initialize. The max and min values for these parameters represents the range in which node biases and connection weights can exist. If a mutation to one of these parameters occurs that increases or decreases it above or below its respective max or min value, the value will be set to its max or min value. The mutation power for both of these values represents the standard deviation of mutations that are made to the respective node biases and connection weights. The mutate rate represents the probability that these parameters will be mutated after each generation and the replace rate parameter represents the probability that the node biases and connection weights will be completely replaced with a newly initialized value.

The mutations to the structure of the neural networks happen slightly differently. In each generation of the algorithm, there is a probability that node connections (denoted in Table 1 as Connection Add/Delete Probability) and nodes (denoted in Table 1 as Node Add/Delete Probability) will be either added or deleted. As was mentioned previously, when initialized, the network nodes will be fully connected and the number of hidden nodes will be initialized as zero. As the algorithm progresses, node biases, connection weights, number of hidden nodes, and node connections will be mutated in order to attempt to find a network with a higher fitness value while prioritizing networks with simpler structures.

3.6 Fitness Function

The fitness of each neural network at the end of each generation is the sum of the rewards that the car accumulated throughout the entirety of the generation. The reward at each step is calculated using the following values: a running count of the time that the vehicle spent in the safe zone called safe time, a ratio that determines how close the car is to the center line of the safe zone called lane ratio, and the total distance that the car travels in the generation called distance. The calculations of the reward at each time step and the overall fitness can be seen below in Equations 1 and 2.

Reward at Each Time Step

$$R = (SafeTime) * (LaneRatio)/20 + (Distance/50) \quad (1)$$

Fitness at Each Time Step

$$Fitness_{new} = Fitness_{old} + R \quad (2)$$

There are two main parts of the reward that the car receives at each time step. The safety value that is calculated using safe time and lane ratio, and the distance value that is calculated using the speed of the car and the size of the time step. Each of these values are reduced by different amounts. The safety value is divided by 20 and the distance value is divided by 50. These reductions differ due to the safety of the vehicle is prioritized over the distance that the vehicle traveled. Both are important to the reward of the vehicle but since the safety of the vehicle is more important than the distance, this value is reduced by less. Also to note, these values are being reduced since the fitness value of successful cars becomes very large over the time of each generation and these reductions in the reward values are used to simply reduce the magnitude of the fitness value without changing the comparison of fitness values between cars. More in depth descriptions of how each of the parameters in the reward and fitness equations are calculated is available below.

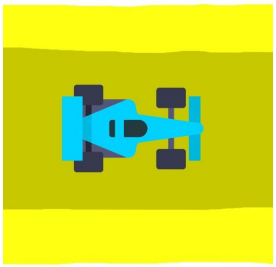
3.6.1 Safe Time. At each time step of the generation, the car will check to see if it is within the green safe zone. This is done by checking the RGB value of the pixels at each of the four corners of the vehicle model. If the RGB value of all four of these pixels matches the predefined color of the green safe zone, the car will be considered safe at that time step. Figure 4 shows examples of cars that are considered safe and not safe. Based on whether the vehicle is safe or not, the safe time value for the current time step will be calculated. If the vehicle is safe, 5 points will be added to its safe time and if the vehicle is not safe, its safe time will be multiplied by 0.8. Equations 3 and 4 show how safe time is calculated at each time step.

If Vehicle is Safe

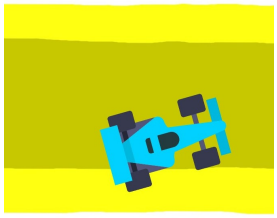
$$SafeTime = SafeTime + 5 \quad (3)$$

If Vehicle is not Safe

$$SafeTime = 0.8(SafeTime) \quad (4)$$



(a) Safe Car with All Four Wheels in Green Area



(b) Unsafe Car with Three Wheels in Green Area and One in Yellow Area

Figure 4: Examples of Cars that are Safe and Unsafe

Safe time is a running summation of these values for the entire generation so cars that manage to stay within the safe zone for the whole generation will receive exponentially large returns and vehicles that are not often in the safe zone receive little to no rewards.

3.6.2 Lane Ratio. At each time step of the generation, the car will check to see how close it is to the center of the safe zone. For this value to be calculated, the vehicle needs to be inside of the safe zone and this is determined the same way that it is for the calculation of the safe time variable. If the vehicle is not in the safe zone, the lane ratio will be zero. If the vehicle is in the safe zone, the width of the safe zone will be calculated based on the sum of the distances returned from the sensors on each side of the vehicle which are D1 and D7 (see Figure 2 for information on sensors). The sum of these distances will be the width of the safe zone and half this width will be the location of the center line of the safe zone. The car will then use the sensor values of D1 and D7 to determine how close the vehicle is to the center line. This calculation can be seen in Equations 5- 12. Figure 5 shows examples of how lane ratio changes depending on the vehicle's position in the safe zone.

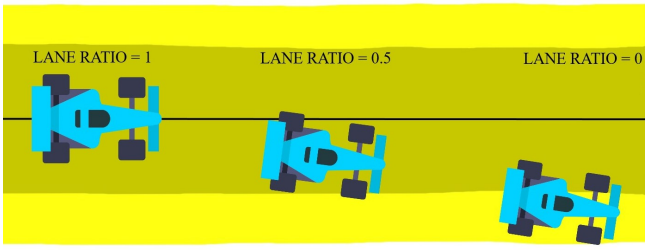


Figure 5: Examples of the Car on Different Parts of the Road with Different Lane Ratios. Note: Black Line Represents center line of Safe Zone

Calculation of Safe Zone Width and Center Line Location

$$SafeZoneWidth = D1 + D7 \tag{5}$$

$$CenterLine = SafeZoneWidth/2 \tag{6}$$

If D1 = D7 (Car is on Center Line)

$$LaneRatio = 1 \tag{7}$$

If D1 > D7 (Car is Right of Center Line)

$$\delta = D1 - CenterLine \tag{8}$$

$$LaneRatio = (CenterLine - \delta)/CenterLine \tag{9}$$

If D7 > D1 (Car is Left of Center Line)

$$\delta = D7 - CenterLine \tag{10}$$

$$LaneRatio = (CenterLine - \delta)/CenterLine \tag{11}$$

If Car is Not in Safe Zone

$$LaneRatio = 0 \tag{12}$$

When the reward of the current time step is calculated, lane ratio will be multiplied by safe time to discount this value depending on the vehicle's position in the safe zone. A lane ratio value of zero would mean that the vehicle is on the very edge of the safe zone or in the unsafe zone and this would reduce the safe time value at the current time step to zero since this position would not be safe. A lane ratio of one would mean that the vehicle is directly on the center line of the safe zone and this would reward the car with the full safe time value since the car is well inside of the safe zone. The purpose of the lane ratio parameter is to encourage the vehicles to stay near the center of the safe zone since the vehicles will receive higher rewards depending on how close they are to the center line.

3.6.3 Distance. For each of the cars in the population, the total distance the car traveled throughout the entire generation will be calculated using the speed that the car is traveling at. Since the time steps are uniform and equal to one, the total distance that the car traveled will be the sum of the speed that the car was going at each time step. Equation 13 shows how this is calculated.

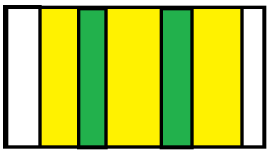
Distance Calculation

$$Distance = Distance + Speed * dt \tag{13}$$

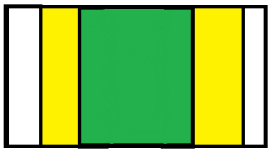
As was mentioned previously, the distance that the car travels is prioritized less than how safe the vehicle is over the time spent in the generation since vehicles that drive safely but move slower should be considered better than vehicles that move fast but drive unsafely.

4 EXPERIMENTAL SETUP

4.1 Environmental Map



(a) Two lane realistic depiction of lane.



(b) Simplified depiction of lane.

Figure 6: Simplifying assumption of lane values

For the experiment, several simplifying assumptions were made. As shown in Figure 6a, the real world realistic scenario involves

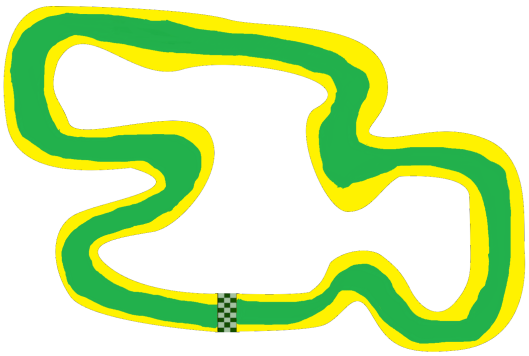


Figure 7: Sample map

two lanes of cleared snow. To simplify our experimental setup, we simplified this into one lane, as seen in Figure 6b. In these figures, green and yellow represent driveable areas, with green being the preference.

To train the system, we created a series of path maps to train and test the resultant networks. The maps consist of a series of closed loop tracks of varying complexity, such as the one shown in Figure 7. These maps are composed of three values, a good area (green), a worse area (yellow) and an undrivable area (white). If the vehicle encounters the white areas, it fails. The best genome upon completion of the evolution algorithm is saved off.

This saved genome will be loaded into evaluation software that runs it through each map. Several metrics are collected to assess performance: if the car is able to survive on the circuit, the distance the vehicle is able to travel, the ratio of time spent in the safe green area and the calculated fitness value.

4.2 Training and Verification

Our evolutionary algorithm uses a population of 30 feed forward networks to search for a solution to the navigation training problem. Each solution is run through a training map to have their performances evaluated. Solutions that navigate into the white area are discarded, and solutions with high fitness scores are prioritized when new solutions are mutated. This is repeated through all of the generations to train optimal solutions.

Once solutions were trained, the highest performing solutions were saved and tested on other maps to compare the generalizability of the results. Data is collected on the performance of each solution in terms of survivability, time spent on safe green areas and distance covered.

These training and verification steps were repeated on 5 maps (1 train, 4 verify) for generation counts in the set [30, 40, 50, 60, 70, 80, 90, 100]. The evaluation maps were run for 1200 time steps or until the vehicle crashed, whichever came first. Results are displayed in Section 5

5 EXPERIMENTAL RESULTS

For our experiments, we used five maps of varying difficulty. We selected one to train on, and then evaluated the best performing vehicle's performance over all 5 maps. Below, we included a table of the results of training on map 1 for 40 generations before testing on the rest of the maps. Survival indicates if the best trained solution on the trained map was able to survive, or if it crashed. Distance Covered is how much distance the vehicle was able to drive in the time span tested. Safe Path Ratio is how much of the time was spent on the safe green path, compared to the less safe yellow path. Fitness Value is a reflection of the accumulated fitness over the testing period. Table 2 shows the results after being training on Map 1, and Table 3 and Table 4 shows the results after being trained on Map 4 and Map 5 respectively for 40 generations.

Map	Survived	Distance Covered	Safe Path Ratio	Fitness Value
1	Yes	12000	1	165202.74
2	Yes	12000	0.982	105838.66
3	No	1950	0.964	4323.41
4	No	1950	0.964	4341.66
5	Yes	12000	0.978	84893.94

Table 2: Initial Results when trained on Map 1 for 40 generations

Map	Survived	Distance Covered	Safe Path Ratio	Fitness Value
1	Yes	12000	1	502723.57
2	Yes	12000	0.934	390510.87
3	No	4080	0.944	49168.72
4	Yes	12000	0.913	385931.61
5	Yes	12000	0.973	410379.33

Table 3: Initial Results when trained on Map 4 for 40 generations

Map	Survived	Distance Covered	Safe Path Ratio	Fitness Value
1	Yes	12000	1	515096.15
2	Yes	12000	0.999	494674.02
3	No	1960	0.964	13270.51
4	No	1960	0.964	13244.85
5	Yes	12000	1	514640.70

Table 4: Initial Results when trained on Map 5 for 40 generations

Further experiments have yet to be conducted regarding the impact of training generations upon the results, as well as further experimentation with the weights of the fitness function. As it currently stands, the fitness reward is heavily dominated by the safety value, causing the vehicle to stay in the middle of the green area moving slowly rather than increasing it's speed in areas of the track that are safe. This results in the speed being minimized as any movement forward takes the vehicle away from the center line. It would also explain the uniform behavior we are seeing in

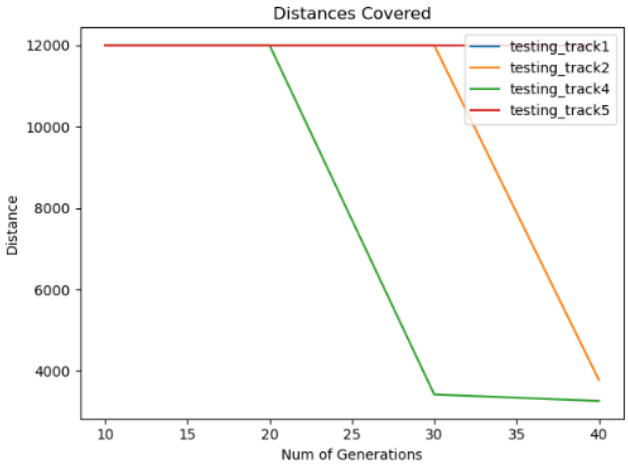


Figure 8: Distances covered on each map

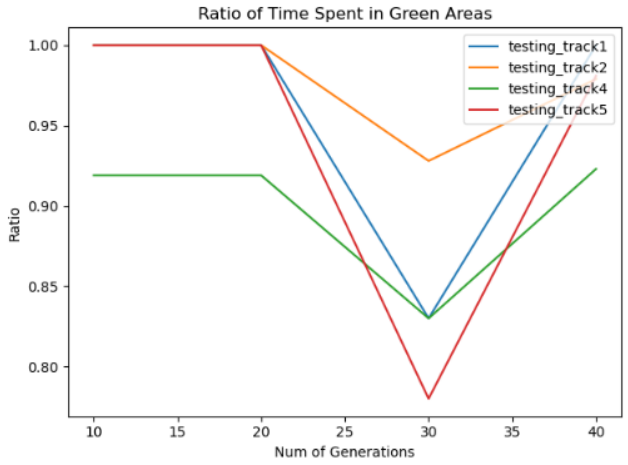


Figure 9: Ratio of time spent in a green area

the distances covered (every map having the vehicle move at the minimum speed of 10). We were able to do some testing with various generation sizes, as shown in Figure 8 and Figure 9, however the results do not seem to be very good.

Based on previous experiments, we expect that a more balanced fitness function would result in a variety of speeds being used. Overall, we expect this will result in the solution covering more distance, potentially as a minor cost to the Safe Path Ratio. This will be due to the system not rewarding the vehicle as strongly to stay towards the center.

6 WORK DISTRIBUTION

Organization. Mark: 60 Jon: 40

Technical. Mark: 40 Jon: 60

Coding. Mark: 50 Jon: 50

Writing. Mark: 50 Jon: 50

REFERENCES

[1] Mohammad Aldibaja, Akiuse Kuramoto, Reo Yanase, Tae Hyon Kim, Keisku Yonada, and Noaki Suganuma. 2018. Lateral Road-mark Reconstruction Using Neural Network for Safe Autonomous Driving in Snow-wet Environments. In *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, 486–493.

[2] Mohammad Aldibaja, Naoki Suganuma, and Keisuke Yoneda. 2017. Robust intensity-based localization method for autonomous driving on snow–wet road surface. *IEEE Transactions on industrial Informatics* 13, 5 (2017), 2369–2378.

[3] Tony Arevalo. 2021 [Online]. Winter Driving Statistics for Driver Safety – 2020. https://carsurance.net/blog/winter-driving-statistics/

[4] Michele Bertoncello and Dominik Wee. 2021 [Online]. Ten ways autonomous driving could redefine the automotive world. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world#

[5] Paolo Grisleri Claudio Caraffi, Stefano Cattani. 2007. Off-Road Path and Obstacle Detection Using Decision Networks and Stereo Vision. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 8, NO. 4, DECEMBER 2007* (2007).

[6] LLC CodeReclaimers. 2021 [Online]. NEAT Overview. NEAT Python. https://neat-python.readthedocs.io/en/latest/index.html

[7] Florian Dedov. 2021. ai-car-simulation. https://github.com/NeuralNine/ai-car-simulation.

[8] Thorsten Luettel Hans=Joachim Wuensche Dennis Fassbender, Benjamin C. Heinrich. 2017. An Optimization Approach to Trajectory Generation for Autonomous Vehicle Following. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017).

[9] Marin Toromanoff Raoul De Charette Etienne Perot, Maximilian Jaritz. 2017. End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning. *a2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2017).

[10] S. Ernst L. Kelch J. Goldbeck, B. Huertgen. 1999. Lane following combining vision and DGPS. *Institute of Flight Guidance and Control, TU Braunschweig, Hans-Sommer-Str. 66, 38106 Braunschweig, Germany* (1999).

[11] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. 2019. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484* (2019).

[12] United States Department of Transportation. 2021 [Online]. Automated Vehicles for Safety. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#:~:text=What%20are%20the%20safety%20benefits,to%20hu

[13] Anil Ozturk, Mustafa Burak Gunel, Resul Dagdanov, Mirac Ekim Vural, Ferhat Yurdakul, Melih Dal, and Nazim Kemal Ure. 2021. Investigating Value of Curriculum Reinforcement Learning in Autonomous Driving Under Diverse Road and Weather Conditions. *arXiv preprint arXiv:2103.07903* (2021).

[14] Guangyuan Pan, Liping Fu, Ruifan Yu, and Matthew Muresan. 2018. Winter road surface condition recognition using a pretrained deep convolutional network. *arXiv preprint arXiv:1812.06858* (2018).

[15] Mingxing Peng, Zhihao Gong, Chen Sun, Long Chen, and Dongpu Cao. 2020. Imitative Reinforcement Learning Fusing Vision and Pure Pursuit for Self-driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3298–3304.

[16] Nathir A Rawashdeh, Jeremy P Bos, and Nader J Abu-Alrub. 2021. Drivable path detection using CNN sensor fusion for autonomous driving in the snow. In *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2021*, Vol. 11748. International Society for Optics and Photonics, 1174806.

[17] Kenneth O Stanley and Risto Miikkulainen. 2002. Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, Vol. 2. IEEE, 1757–1762.

[18] Kyle Wiggers. 2020 [Online]. New data set helps train cars to drive autonomously in winter weather. https://venturebeat.com/2020/02/03/new-data-set-helps-train-cars-to-drive-autonomously-in-winter-weather