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 is to train a neural network via an evolutionary algorithm to drive an autonomous vehicle capable of following the safe areas in the road that have been cleared out by previous vehicles, while avoiding dangerous areas of the road that have not been cleared. This problem is difficult to solve using learning as the vehicle needs to be able to generalize its actions to any environment and road conditions can change rapidly in snowy weather. We demonstrated that our algorithm is capable of training neural networks to be able to navigate in simulated conditions. After training on one map, the top performing solutions were able to successfully generalize to multiple other testing maps by using the safe area of the road as a guideline to follow for safe driving.

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, 10 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 [14]. 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 making it more difficult for drivers too see where the road is. 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 factors will impact the traction the vehicle will have, thereby potentially degrading the vehicle's control leading to more crashes. This potential loss of vehicle control greatly increases 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].

There are many factors that increase the danger of driving in winter conditions and further necessitate the need for the increased driving safety that fully autonomous vehicles can offer. But autonomous vehicles struggle to drive in snowy conditions due to the fact that the lane dividers and other distinctive road features that they would normally use to guide their actions are obscured by snow or ice [12], [7]. However, 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 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 a ditch.

We are proposing to use the new lanes in the snow that have been created by previous vehicles traveling as a guideline for the autonomous vehicle to follow. We used a neural network to control the autonomous vehicle and an evolutionary algorithm to train the neural network to use the new lanes in the snow as a guideline for safe driving.

The contributions of this paper are to:

- Create a new framework for autonomous driving in snow using cleared lanes in snow as guide for autonomous vehicle
- Train a neural network for control of an autonomous vehicle using neuro-evolution for safe driving in snowy conditions

Sensor readings from the car will be taken to determine the vehicles position on the road and will then be fed into the neural network to output the safest action for the car to take. The network will be trained to drive the vehicle as quickly as possible while maintaining safety by staying within the areas of the road that have been cleared out by other vehicles. The algorithm uses a reward fitness function that prioritizes cleared safe areas by giving the vehicle large rewards 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 by receiving additional rewards that increase in value the closer the vehicle is to the center of the safe area.

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. © 2021 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

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. [21]. 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 [15] 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 [17].

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 [10]. 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 [13]. 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 [16] 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 [18]. However, for the sake of our work, we will be dealing primarily with the driving challenge, assuming the sensing problems are already solved and accurate sensor readings can be made by the autonomous vehicle.

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 using this to determine what actions would be the safest, 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 [11] and other methods that involve driving on roads that do not have distinct markers for the car to follow [5] [9].

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 classifying the path in the snow into parts that are safe and unsafe to be driven on. However, it differs from our work since the primary focus of their work is classifying the areas of the road from images taken from the car, while in our work, we are assuming that the car already has the knowledge of which areas of the road are safe and unsafe. Their work also 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 [9]. 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 travel at.

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. [19]

2.4 NeuralNine Car Simulator

For our work, we took advantage of an existing car simulator developed by NeuralNine [8]. 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 SIMPLIFYING ASSUMPTIONS

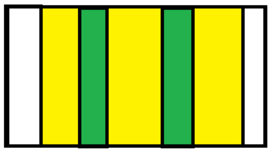
In order to successfully complete this work, we had to make a series of assumptions.

3.1 Road Model

We decided to simplify the model of the road from a realistic two track path in the snow down to a one track path. Figure 1a shows the realistic two tracks model while Figure 1b shows the simplified one track model that we used for this project. In our road model, the green area in the center represents the region where the snow is melted and it is safe to drive, the yellow areas on the sides represent the regions with slushy snow where it is unsafe to drive, and the white area represents the region with deep untouched snow where driving will likely result in a crash. Figure 2 shows the simplified road model with the different areas of the road labeled. Realistically, 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.

3.2 Car Model

We also decided to simplify the kinematic model of the vehicle 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.


(a) Two lane realistic depiction of lane.

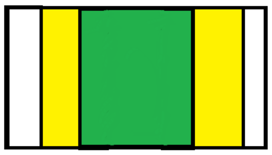

(b) Simplified depiction of lane.

Figure 1: Simplifying assumption of lane values

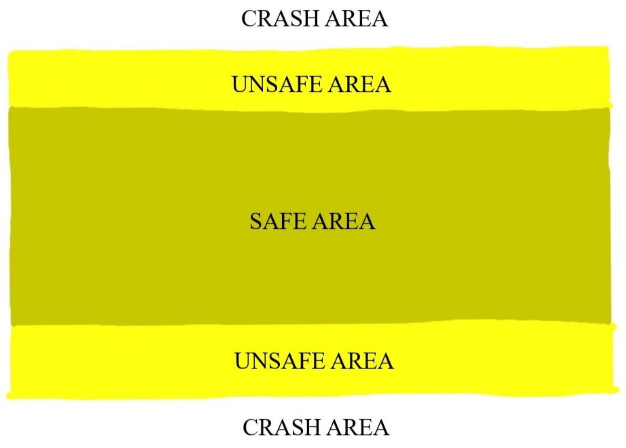


Figure 2: Safe, Unsafe, and Crash Areas of Simplified Road Model

3.3 Sensing

As was mentioned earlier, in this project we are assuming that the sensing issues that are present with autonomous vehicles driving in snow have been solved and our autonomous vehicles are able to get accurate sensor readings. However, we are also making some simplifications to how the sensors on the vehicle function. Realistically, autonomous vehicles would likely be equipped LIDAR sensors that are used to detect distances to objects in the vehicles environment [20]. We have simplified the LIDAR sensor to 7 distance measurements that are taken at different angles around the vehicle as can be seen in Figure 3. These measurements return the distance from the center of the vehicle to the border of the safe and unsafe areas of the road.

4 METHODS

We used a multi-layer feed forward neural network to control the autonomous vehicle and an evolutionary algorithm to train the networks to drive the vehicles successfully. The objective of our algorithm is to train vehicles to drive safely in snowy conditions by using the area in the road that has been cleared out by other vehicles as a guideline for the autonomous vehicles to follow.

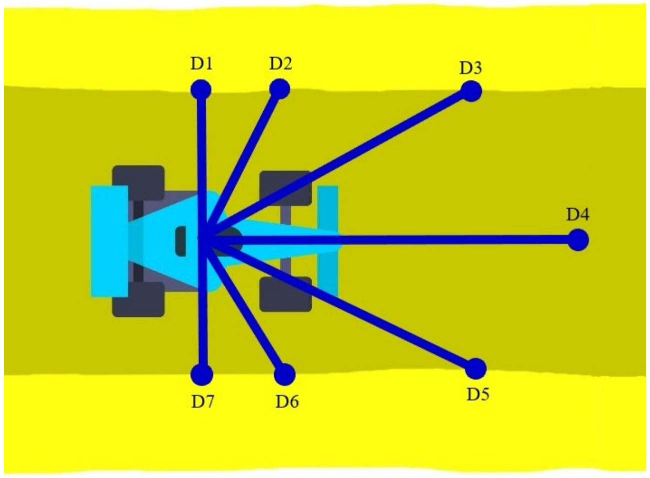


Figure 3: Labeled Sensors that Feed Distance Returned into Neural Network

4.1 Feed Forward Neural Network

In our solution, multi layered feed forward neural networks were used to control the actions of the vehicles. The neural network structure that worked best for this application had seven inputs, one hidden layer three hidden units, and five outputs. At each time step of the simulation, the car takes 7 sensor readings of its environment. These sensor readings are the distances from the center of the vehicles to the edge of the safe area of the road at seven different angles around the vehicle. Figure 3 shows these sensor reading around the car as well as labels them respectively from left to right as D1-D7. These sensors readings are fed through the neural networks and the output node with the highest value was chosen as the action for that time step. The possible actions are as follows: turn left which changes the angle of the car by -7 degrees, turn right which changes the angle of the car by 7 degrees, speed up which increases the speed of the car by 0.1, 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.

4.2 Evolutionary Algorithm

In order to train the neural networks to drive the cars safely, we decided to use NEAT evolutionary algorithm. The NEAT algorithm is made for optimizing the parameters of neural networks by mutating both the internal parameters of the neural networks as well as the actual structure of the neural networks with the goal of increasing the overall fitness. The internal parameters that are mutated are the nodes biases in the hidden and output nodes as well as the connection weights in both the input-hidden and hidden-output layers. The structural components of the networks that can be mutated are the node connections and number of hidden units in the hidden layer with probabilities that both of these can be added or removed in the mutation step of the algorithm. Figure 5 shows a simplified

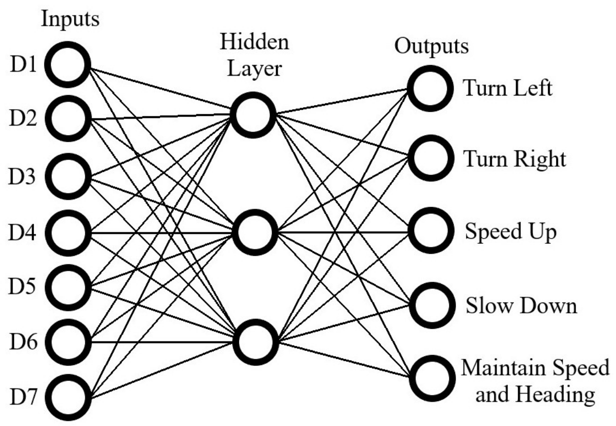


Figure 4: Structure of Best Neural Network Found using NEAT Algorithm

block diagram of how the NEAT algorithm works. In each mutation step of the algorithm, NEAT will mutate the highest fitness networks of the previous generation with the goal of creating new networks with higher fitness scores. The NEAT algorithm's ability to mutate both the internal parameters as well as the structure of the networks creates interesting mutation behavior since NEAT will prioritize networks with simpler structure and will slowly increase the complexity of the networks as needed in order to increase their fitness.

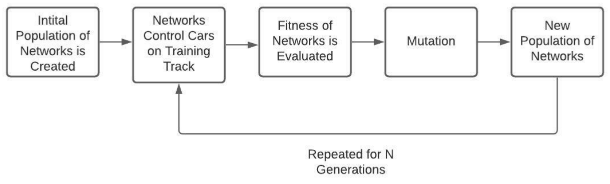
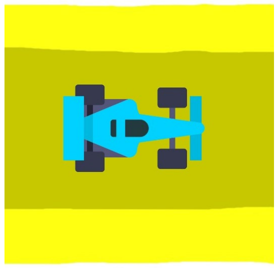

Intital Population of Networks is Created
Networks Control Cars on Training Track
Fitness of Networks is Evaluated
Mutation
New Population of Networks
Repeated for N Generations

Figure 5: NEAT Algorithm Block Diagram

4.3 Fitness Function

There are two main areas of the neural networks performance that are evaluated at the end of each generation. The first is how safely the networks were able to control the vehicle which takes into account how much time the vehicle spent in the safe zone of the road as well as the vehicles actual position in the safe zone. The second is how fast the network was able to drive the vehicle while maintaining safety.

4.3.1 Safe Value. At each time step of the generation, the car will receive safe value (SV) depending on what area of the road that it is in. The area of the road that the vehicle is in is found 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. If one or more of the RGB values at the corners of the vehicle do not match the color of the safe zone the vehicle will be considered unsafe. Figure 6 shows examples of cars that are considered safe and unsafe. If the vehicle is safe, its safe value for the current time step will be 10 and if the vehicle is unsafe, its safe value for the current time step will be 1. The vehicles safe value will be used in the calculation of its reward at each time step so vehicles that manage to drive within the safe area will in turn receive higher rewards.


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

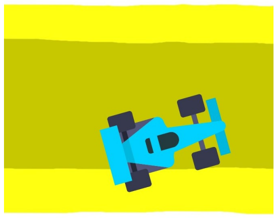

(b) Unsafe Car with Three Wheels in Green Area and One in Yellow Area

Figure 6: Examples of Cars that are Safe and Unsafe

4.3.2 Lane Ratio Bonus. At each time step of the generation, if the car is inside of the safe area of the road, it will receive an bonus value (BV) depending on its position in the safe area. This bonus value will then be discounted by the lane ratio (LR) depending on how close the car is too the center line of the safe zone. Lane ratio is a discount factor with values between 0 and 1 that is calculated using the sensors on the left and right side of the car (D1 and D7) to first determine the width of the safe zone and location of the center line, then determine how close the car is too the center line. Examples of how lane ratio changes based on the vehicles position in the road can be seen in Figure 7 and the calculation of the lane ratio can be seen in equations 1 - 7. In the reward calculation, the bonus value will be multiplied by the lane ratio to decrease the bonus value as lane ratio decreases. The maximum bonus value that a car can receive for being inside of the safe zone is 10 when the vehicles is directly on the center line with a lane ratio of 1. As the vehicle moves away from the center line and towards the unsafe yellow area, the value of lane ratio will decrease from respectively from 1 to 0 depending on how far from the center line the vehicle is which will in turn reduce the bonus value that the car receives for being inside of the safe zone. The purpose of the lane ratio bonus is to encourage the cars to drive as close as possible to the center line of the safe zone since this is the area of the road that is the safest.

Calculation of Safe Zone Width and Center Line Location

$$SafeZoneWidth = D1 + D7 \tag{1}$$

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

If D1 = D7 (Car is on Center Line)

$$LR_t = 1 \tag{3}$$

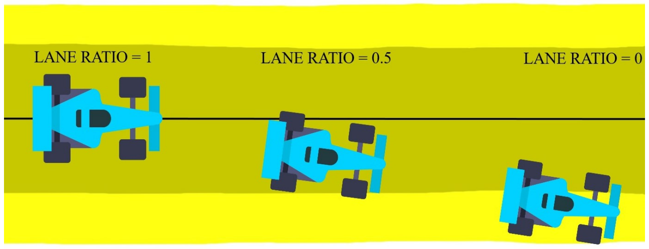


Figure 7: Examples of the Car on Different Parts of the Road with Different Lane Ratios. Note: Black Line Represents center line of Safe Zone

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

$$\delta = D1 - CenterLine \tag{4}$$

$$LR_t = (CenterLine - \delta)/CenterLine \tag{5}$$

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

$$\delta = D7 - CenterLine \tag{6}$$

$$LR_t = (CenterLine - \delta)/CenterLine \tag{7}$$

4.3.3 Speed. At each time step, the vehicles current speed (V) speed will be used to directly reward the vehicle for driving quicker. Since the vehicles speed is a known property, there is no calculation necessary. We are trying to minimize the time that the cars have to spend driving in the snowy conditions so vehicles that move at higher speeds are rewarded more. However, increasing the vehicles speed will make it more prone to small errors that could put the car into unsafe areas of the road.

4.3.4 Overall Reward and Fitness. Using the vehicles safe value (SV), Lane Ratio Bonus (BV*LR), and speed (V), the reward at each time step and overall fitness over the time span of the generation can be calculated as seen in equations 8 and 9. Since it is more important that vehicles drive safely rather than quickly, the rewards that the vehicles can receive at each time step for driving safely are larger than what the rewards received for driving quickly. This is why the vehicles speed (V) is reduced by a factor of 10 in the reward equation. The goal of the reward equation is to heavily encourage vehicles to drive safely by giving them large rewards for safe actions and lightly encourage vehicles to drive quickly by giving them small additional rewards for safe actions completed at higher speeds.

Reward at Each Time Step

$$R_t = SV_t + BV_t * LR_t + V_t/10 \tag{8}$$

Fitness over Full Generation

$$Fitness = \sum_{t_0}^{t_{end}} R_t \tag{9}$$

5 EXPERIMENTAL SETUP

5.1 Training Setup

The training process begins with the creation of a population of 30 neural networks that are initialized with random parameters. These networks are then used to control the actions of 30 cars on a training track. It should be noted that though the population of 30 cars attempt to drive on the training track simultaneously, from the perspective each car, they are driving alone. This means that they are not able to collide with the other cars and do not have to consider avoiding the other cars on the track while they are driving. The other cars will also not be modifying the track (e.g., they will not turn yellow areas green by driving on them). As mentioned previously, the simplified road model of the training tracks will consist of a green safe area in which it is safe for the cars to drive, a yellow unsafe area in which the cars are still able to drive but less safely, and a white dangerous area which will lead to the cars crashing. The training maps are closed loop tracks consisting of this road model with many different road features including straight roads, soft curves, sharp curves, etc. as can be seen in Figure 8. The control networks will attempt to drive the cars around the training track as safely as they can without crashing and each generation will be completed when the maximum number of time steps in reached (1200), or all of the cars in the population have crashed into the dangerous white area of the road. The fitness of the networks will be evaluated based on the criteria in Section 4.3 with the highest fitness networks passed into the next generation, and the rest of the population being filled with mutated versions of these networks. This process will then repeat for a set number of generations and the network that returned the highest fitness over all of the generations will be used for the testing process.

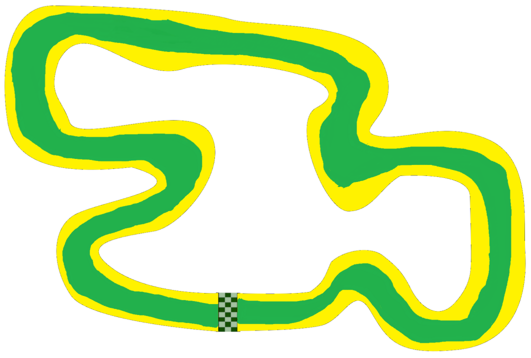


Figure 8: Example of Track that Networks were Trained On

5.2 Testing Setup

The testing process will take the network with highest fitness from the training and test its performance on a set of testing maps. The testing maps are also closed loop tracks consisting of the simplified road model except they have different road features than the map that the network was trained on. In total there are 5 maps that are used for training and testing all with varying complexity of road features. To show the generalization of the best network from training, the network will be tested on the maps that it was not trained on. The testing process will take the best trained solution from the training map and test its generalization to the other 4 maps (e.g., train on map 1, test on maps 2-5, etc.). Several metrics will be collected in order to evaluate the performance of the networks including if the network was able to complete the testing map without crashing, the ratio of time spent in the safe green area of the map, as well as the distance that vehicle was able to travel.

6 EXPERIMENTAL RESULTS AND DISCUSSION

For our experimental analysis, we used a three step process. First, we tested if the maps are solvable, and how many generations are required to find a solution. We also look at the number of survivors that exist per generation, representing the number of potential viable solutions. Using the estimate for the number of generations required to reliably reach solutions, we then investigated the generalizability of the trained networks via cross validation. In this second step, we compared the results of training on a fixed number of generations, and testing on all maps. Finally, we conducted a deep dive into the one training map and one test map. Here we evaluate the changes in speed over time, distance covered and amount of time spent in the safe green area.

6.1 Map Solvability Analysis

For our initial experiments, we looked into the capabilities of our solution to learn and solve map tracks. To this end, we looked into how long it took to evolve the first solution that would survive for the full 1200 time steps. We further investigated the number of solutions per generation, as well as how much distance the best solution per generation was able to cover. These values would also provide us with a more objective approximation of how complex each map is, as more complex maps would, on average, require more generations to produce a viable solution (aka one that survives the entire time span).

To this end we ran 10 training rounds on each of the maps, and averaged the number of survivors per generation as well as the distances covered by the best performing member of the population. The results are displayed in Figures 9 and 10. Figure 9 shows the average number of survivors present in each generation for each map, while Figure 10 shows the distance covered by the network solution with the highest fitness value.

These results demonstrate that each of the maps can be trained on, and none of the maps are unsolvable. Furthermore, they suggest that Map 4 is the most complex map as more generations are required to find a solvable solution, and even once solutions are found, there are fewer of them (there are less survivors), as demonstrated in Figure 9. One drawback to these figures is that adding errorbars made them unreadable.

We found that some of the maps (1,2 and 3) were able to find a potential solution within just a few a few generations, however this appeared to largely be the result of luck, as the number of survivors would stagnate at that point, requiring a further 10 generations to find more solutions. At that point, enough viable candidates were in the population pool for innovation to occur, and more survivors covering most distance could be found. In order to reliably find multiple solution options, 30-40 generations were required to be

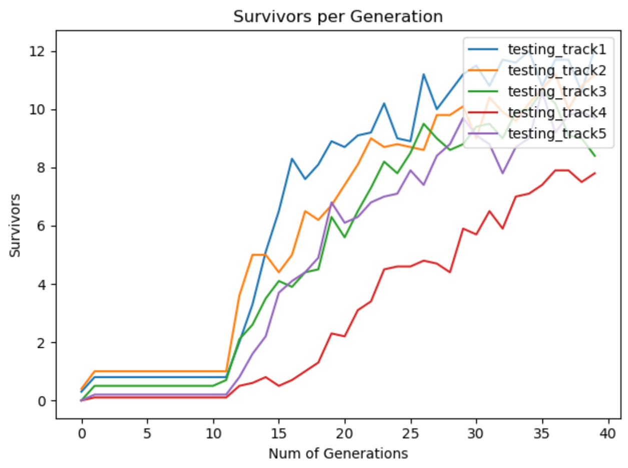


Figure 9: Number of survivors per generation in each map

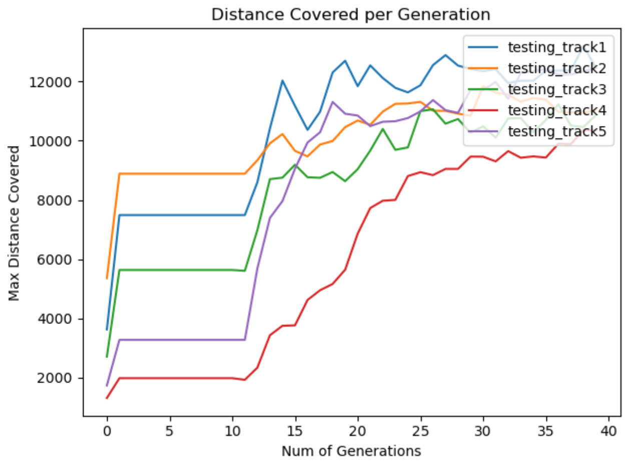


Figure 10: Distance covered by best performing agent in each generation

run. We found that the more generations used to train a solution, the better the results created.

6.2 Cross Validation

In the previous results section, we were able to determine approximates for how many generations were needed for each map in order to train a solution. In this section, we next look into testing the generalizability of the trained networks. This will also demonstrate that our networks are not just memorizing the maps, and that the resultant solution networks are able to function in new situations.

To this end, we trained for 40 generations on one map, and then tested the best performing solution on all five maps. This was repeated for 3 iterations (due to time constraints we could not do more than 3) and the results averaged. This process was repeated for each of the maps (train on 1, test on all). The results are displayed in Table 1. In the table, the rows are the map trained on and the columns represent that testing map (e.g., the 3nd row from the top represents the results of training on Map 2).

Train\Test	1	2	3	4	5
1	1	1	0.67	0.67	1
2	1	1	1	1	1
3	1	1	1	1	0.67
4	1	1	1	1	1
5	1	1	1	0.33	1

Table 1: Success rate for training on one map and testing on all

We found that the more complex training maps identified in the previous subsection had superior generalized performance compared to the simpler maps such as Maps 1, 2, and 5. Map 4 was the most complex map consisting of many sharp curves and narrow road sections, and when the networks were trained on this map, their actions were able to be generalized successfully onto all of the testing maps with a 100% survival rate. However, when the networks were trained on simpler maps, in particular Map 1, which was the simplest map consisting of just a loop with minimal turning, the best network found in training was only able to reliably generalize to Maps 2 and 5, which were comparable in simplicity to Map 1.

Table 2 demonstrates the ratio of time, on average, each solution spends in the safe green area, compared to the less safe yellow area. It should be noted that these are the ratios of time that was spent in the safe green area while the car was alive, and the values are not reduced for the cars that ended up crashing during testing. Since some of cars that ended up crashing during testing may have managed to stay inside of the safe green area of the track until moments before they crashed, their ratio will still be close to 100%. Even though results trained on Map 1 ended up crashing some of the time, while the cars were alive on the track, they was able to stay in the safe area of the road for around 99% of the time. This demonstrates that even though the cars may end up crashing, they were able to partially learn the behavior of staying in the safe area .

Train\Test	1	2	3	4	5
1	1	0.990	0.992	0.990	1
2	0.998	0.999	1	0.995	0.999
3	1	0.999	1	0.996	0.980
4	1	0.999	1	1	1
5	1	0.999	1	0.986	1

Table 2: Ratio of time on green for training on one map and testing on all

However, when the networks were trained on Maps 2 and 4, the best network was able to complete all of the testing tracks without crashing so the ratio of time spent in the safe area is accurate. The trained network from these tracks was able to keep the car inside of the safe area of the road between 99 and 100 percent of the time. This shows that the networks that were able to generalize well were also able to complete the tracks safely since they spent close to the entire time of the test in the safe area of the road.

These results suggest that using more complex training sets will improve the accuracy of testing the solution on other maps as Maps 3 and 4 were able to generalize fairly well. The reason for these differences in performance may be due to simpler maps with fewer turns not providing as many differing scenarios for the vehicle to learn from. The networks are able to learn to complete the simple training track, but when they encounter complex road features in the testing maps that were not present in their training, they do not know what to do which results in them crashing. However, when trained on the more complex maps such as Maps 3 and 4, they have already encountered the most complex road features in our set of testing maps, and are able to complete the simpler tracks with ease since they have seen all the road features in those tracks before. We found that more complex maps with many different types of road features are essentially larger and more diverse training sets that are able to successfully teach the networks how to react in many different situations while the less complex maps only teach the networks how to react to a small set of road features.

6.3 In-Depth Map Analysis

In the previous two sections, we demonstrated that each of the maps were trainable, and the resulting trained networks were generalizable. In this section, we performed a more detailed investigation and analysis of a single map's performance. In this case, we chose to use a new Training Map to further validate our results. We trained a solution for 40 generations on the new training map and then tested the solution on our previous training and testing maps and reviewed the results.

Our first step is to confirm the performance of the solution on other maps. Table 3 displays the distances covered on each map, as well as how much of the time was spend on the safe green area. This represents how quickly the vehicle was able to travel, as well as how much of it was done safely. Across the 5 tested maps, the average test run spent 99.5% of the time on green, safe areas. However, when the new trained solution was tested on map 4, it was only able to drive safely on the track for a short time until crashing which is why the distance covered on this map is lower than the others.

	1	2	3	4	5
Distance Covered	11426.8	10398.5	10320.9	2836.6	11024.4
Percent On Green	1	0.998	1	0.978	1

Table 3: Distance and Percent of Time on Green on Each Map Trained on New Training Map

Next, we took a closer look at a single solution. A solution was generated on the Training Map and tested on Map 3. In Figures 11 and 12, we look at how the speed of the vehicle changes over time, as well as how far the vehicle travels from the center line of the green area. These are important considerations towards the end objective, as maximizing speed and safety were our two, main concerns. Maximizing speed to get off the dangerous road as soon as possible, and maximizing safety so the vehicle does not crash. Having the vehicle stay as near as possible to the center of the safe area also keeps it as far as possible away from the dangerous areas where crashes are likely.

When trained on the new map, the network was able to successfully increase and decrease the speed of the vehicle as necessary depending on what area of the track that the vehicle was in as seen in Figure 11. When the speed is increased, the vehicle is in a section of the track that only contains relatively simple road features like straight road or softly curved road and when the speed is decreased, the vehicle is in a section of the track containing more complex road features like sharp curves or consecutive left to right curves. This behavior of increasing and decreasing the speed of the vehicle depending on the complexity of the current section of the track demonstrates that the network was able to successfully learn that it is safe to move quicker in some areas of the rather than others.

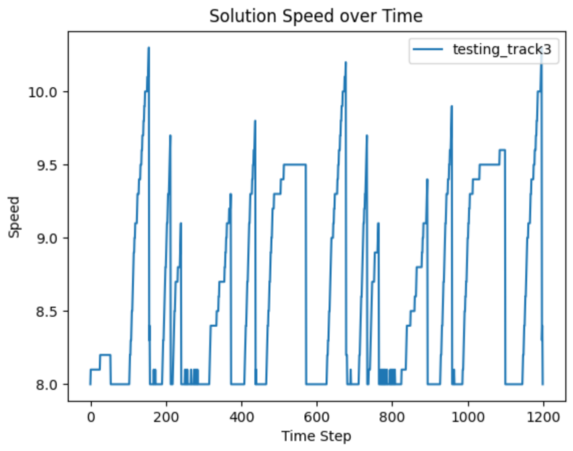


Figure 11: Speed per time step for a network trained on the new Training Map, tested on Map 3

Even with increasing the speed of the vehicle in some areas of the track, the network was able to keep the vehicle inside of the safe area of the track for all time steps. This is shown in Figure 12 which shows the lane ratio of the vehicle at all time steps. A value of 1 in this figure means that the vehicle is directly on the center line of the safe area of the road and a value of 0 would mean that the vehicle outside of the safe area. As the shown in the figure, the vehicle maintains a lane ratio between around 0.6 and 1 for all of the time steps meaning the vehicle is inside of the safe area at every time step. These results combined with the vehicles ability to change it speeds depending on the section of the track that it is in, shows that our algorithm was able to successfully train vehicles to both drive safely as well as maximize their speed without jeopardizing the safety of the vehicle.

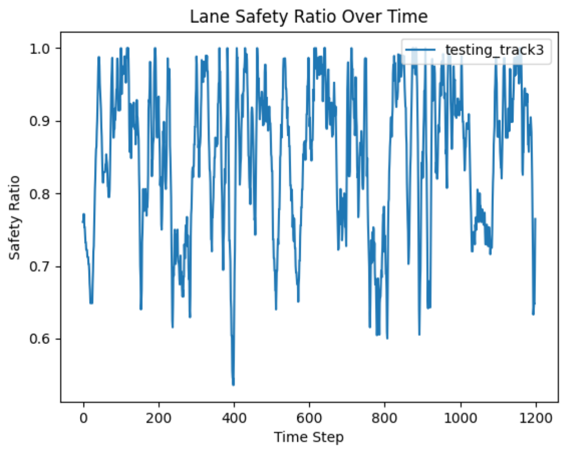


Figure 12: Lane Safety Value per time step for a network trained on Training Map, tested on Map 3

7 CONCLUSION

In this paper, we presented our solution for teaching an autonomous vehicle to navigate a simulated map of snowy conditions by following lanes created in the snow by other vehicles, as a guide for the vehicle to follow. This was done by creating a map containing representations of a series of drivable conditions ranging from drivable to likely to cause a crash. To solve this problem we developed a Feed Forward Neural Network for control, and searched for a solution using an Evolutionary Algorithm with NEAT topology.

Overall, our experiments confirmed the viability of our work. The evolutionary algorithm, and fitness function we used were able to find feed forward neural networks the could control a vehicle through each of the maps. The majority of these, 91.7% per the results in Table 1, were further generalizable to other maps. We also found that more complex training maps would result in increased generalization when tested on the other maps. In particular, we found that Map 4 had the best performance, being generalizable to each of the other maps.

We also found that the networks had difficulty in modifying the speed of the cars. Many of the networks simply drove the cars at the minimum speed while changing the heading of the car in order to safely navigate road features as necessary. This is likely due to the fact that the slower the cars moved, the more closely they would stay to the center line of the green safe zone. This behavior would be rewarded by the fitness function we used that heavily prioritized positioning near the center line of the green area, resulting in large changes in speed that would take the vehicle away from the center to be suppressed.

However, the algorithm was able to find several networks that were capable of both increasing and decreasing the speed of the car while also managing to keep the car in the safe area of the road as seen in Section 6.3. But networks such as this were not able to be consistently found using the algorithm and the majority of the networks would simply drive the cars at the minimum speed. This behavior of driving the car at the minimum speed is not what we initially hoped for, but does coincide with the behavior of human drivers when driving in snowy conditions who opt to drive slowly in order to maintain as much control over the car as possible.

7.1 Future Work

In the future, we hope to make a series of improvements to our code. In particular, we would like to further increase the number of output action options the vehicle has such as being able to choose combination actions (e.g., speed up and turn right). We would also like to expand the work to provide more sensitive control. As it currently stands, the turning actions and the speed up/slow down actions are all fixed values. Enabling the system to control the degree to which a turn is more or speed is increased would likely help to further improve the performance of the system. We do predict that this would require longer training times due to the increased complexity of the model.

Finally we would also like to look further into adjusting the number of initial hidden nodes our system starts with. It would be interesting to figure out exactly how adjusting the number of starting hidden nodes would impact the training time (as higher numbers of hidden nodes require more time to train), as well as the resultant vehicle behavior. In particular, we would want the vehicle to demonstrate a greater adaptability in how it changes its speed.

8 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] C.R. Kelber C.R. Jung. 2004. A robust linear-parabolic model for lane following. In *Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing*. IEEE.
[8] Florian Dedov. 2021. ai-car-simulation. https://github.com/NeuralNine/ai-car-simulation.
[9] 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).
[10] 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).

[11] 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).

[12] M.H. Han K.B. Lee. 2008. Lane-following method for high speed autonomous vehicles. In *International Journal of Automotive Technology*. Springer Link.

[13] 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).

[14] 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

[15] 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).

[16] 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).

[17] 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.

[18] 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.

[19] 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.

[20] Udacity Team. 2021. How Self-Driving Cars Work: Sensor. https://www.udacity.com/blog/2021/03/how-self-driving-cars-work-sensor-systems.html.

[21] 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