Autonomous Vehicle Navigation Control Using Neuroevolution of Augmenting Topology (NEAT) Inspired Algorithm

Abstract. Autonomous vehicles (AVs) currently travel the roadways of the United States and many other countries. The current debate about AVs does not center on whether they should or should not be deployed; they are currently in use. Instead, the focus is on how new transportation networks, our social environment, and the people who live in it will be affected by such technologies, as well as whether such systems should be fully automated or remain under direct human supervision. What's more, how will mobility change if these self-driving cars start sharing and eventually dominating our roads? This paper presents a NEAT (Neuroevolution of Augmenting Topologies) inspired algorithm for autonomous vehicle navigation. The study of neuroevolution is regaining popularity. Several renowned artificial intelligence institutes and researchers are testing it (NEAT), and new opportunities for influence in deep learning are developing as a result of numerous recent developments. And as more people become aware of its potential, it is beginning to draw more attention.

Keywords: Neuroevolution, Topologies, Artificial neural networks, Deep Learning, Autonomous vehicles.

1 Introduction

One of the most anticipated technological advancements of our day, autonomous vehicles have captured the public's imagination maybe more than any other kind of transportation in the previous 50 years. A regular stream of news stories confirms the fierce competition among companies striving to manufacture autonomous vehicles, including new partnerships between technology firms and conventional manufacturers. The fact that there is such widespread interest indicates how important cars are everywhere. Over the course of the 20th century, they became firmly ingrained in our urban societies, bringing about "new socialities of commuting, family life, community, lei-sure, and the delights of movement." Our subjectivities, everyday routines, and the development of our cities have all changed as a result of automobiles. [1].

Closed systems that have already adopted autonomous systems include production lines that are highly automated, air traffic, vast logistic systems in warehouses, and metros and equivalent rail systems. In open systems like road traffic, only preliminary testing in public transportation is being done, primarily on divided or marked lanes, and prototypes from various manufacturers are being tested in small numbers [2].

As was previously noted, autonomous vehicles are already being used in a number of nations. They are being implemented merely because, in the opinion of many researchers, autonomous vehicles will have a substantial impact on the world's transportation industry. The introduction of autonomous vehicles will improve both road safety

and our quality of life. The amount of traffic collisions will significantly decline. Autonomous vehicles can also help with parking challenges, such as finding a spot far from a city center [3]. These automobiles will use less fuel and cost less to insure.

Despite all the advantages that driverless vehicles bring to our society, we shouldn't overlook their drawbacks. Computers are now able to make decisions that resemble those of humans thanks in large part to artificial intelligence, yet there are some situations where machines fall short. It is difficult to teach autonomous machines to make judgments similar to those of humans. An autonomous vehicle must carefully consider a number of variables before choosing the best course of action when navigating along an unregulated route. A vehicle must also engage with other human drivers in a variety of environments.

If all vehicles are replaced by autonomous vehicles, and assuming that all vehicles will communicate before making a decision, the problem of autonomous vehicles navigating via our current road network can be readily solved. Realistically, autonomous vehicles cannot in the near future completely replace the automobile sector. It is still crucial to permit autonomous vehicles to operate in situations where there are both human and robotic drivers.

No business has yet been able to market a completely autonomous car that can operate on any road, in any weather, and without assistance from a human [4]. Numerous strategies are continually being developed by research organizations and businesses to address the issue of navigation in autonomous cars.

2 Literature Review

This paper will illustrate a way to simulate autonomous vehicle navigation that is inspired by NEAT (neuroevolutionary augmented topologies). The study of neuroevolution is making a comeback. Several renowned artificial intelligence institutes and researchers are testing it (NEAT), and new opportunities for influence in deep learning are developing due to numerous recent developments. Additionally, it is beginning to draw more attention as more people become aware of its potential [5]. A branch of artificial intelligence (AI) and machine learning (ML) called neuroevolution aims to simulate in a computer the evolutionary process that resulted in the creation of our brains. In other words, the study of using evolutionary algorithms to evolve neural networks is known as neuroevolution. Neuroevolutionary algorithms come in various forms, such as NEAT, HyperNEAT, and novelty search [5].

2.1 Types of Neuro Evolutionary Augmented Topologies

2.1.1 HyperNeat

The term HyperNEAT, which stands for Hypercube-based Neuro-Evolution of Augmenting Topologies, was initially proposed in 2007. It is a method for creating indirect encoding artificial neural networks (ANNs). By using a cutting-edge indirect encoding method called Com-positional Pattern Producing Networks (CPPNs), which does not

require a conventional maturation stage, HyperNEAT made a contribution to the field of neuroevolution (i.e. evolving artificial neural networks). The size and resolution of evolving ANNs can scale up or down even after training is complete, and neural structures can be created to take advantage of issue geometry, to name a few examples. Large ANNs can also be compactly stored by small genomes.

The intersection of two study fields is where HyperNEAT is located. For instance, neuroevolution proposes that artificial neural networks can be developed via evolutionary computing (ANNs). The second area, generative and developmental systems (GDS), explores how small encodings may be used to explain large-scale, intricate structures in order to promote evolution. Since each gene in the encoding does not correspond to a single matching unit of design in the phenotype, these encodings are sometimes referred to as indirect encodings. The objective in both situations is for created artifacts to one day match the complexity and power of products of natural evolution. The concept that indi-rect encoding can help ANNs to evolve by utilizing development features lies at the intersection of these two fields [6].

2.1.2 Novelty Search

In contrast to conventional evolutionary techniques, novelty search is a relatively new artificial evolution tool. This approach rewards ideas more for their novelty than for their ability to accomplish a certain objective. The absence of a fixed target prevents premature convergence due to a deceptive fitness function.

An evolutionary technique known as novelty search rewards solutions just for being novel in their behavioral characteristics. Fitness-based evolution often has a static objective, and individual assessments are typically made independently of one another. A dynamic metric that rates candidate solutions according to how dissimilar they are from previously evaluated solutions in terms of their behavior is used to evaluate people in novelty search. Premature convergence has little effect on novelty search since there is no fixed target [7].

2.2 Underlying Technology

AI and many neuroevolutionary algorithms are built on artificial neural networks. Artificial neural networks, or ANNs, are high-performance computer systems built on the idea of biological brain networks. ANNs are referred to as "artificial neural systems," "parallel distributed processing systems," and "connectionist systems." A large number of units in an artificial neural network are linked together in some way to enable communication between them. These basic processors, also known as nodes or neurons, cooperate to do tasks.

Artificial neurons, also referred to as nodes or neurons, are the central processing units in neural networks. In a condensed mathematical model of the neuron, connection weights that change the impact of associated input signals serve as a representation for synapses, and a transfer function serves as a representation for the nonlinear behavior that neurons exhibit. The weighted sum of the input signals is converted by the transfer function, and the neuron impulse is then computed [8].

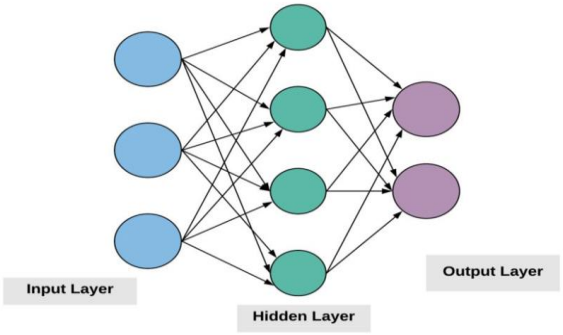

Figure 1 Basic Artificial Neural Network with a Single Hidden Layer[8]

Deep learning is currently attracting the attention of numerous scholars and industries. Deep learning refers to machine learning architectures that combine many multilayer perceptrons into a single hidden layer as opposed to a single hidden layer. The more "deep" or "hidden" layers a deep neural network has, the more complex patterns it may be able to learn. Fully connected layers or entirely interconnected networks are terms used to describe deep layer networks composed of neurons. This refers to the reality that every neuron in an environment is connected to every other neuron therein. Different deep learning architectures can be created by combining fully linked networks with additional machine learning capabilities [9]. The amount of hidden layers contained in each network is what distinguishes a deep neural network from a simple artificial neural network (ANN). More than one hidden layer can be found in a deep neural network. Convolutional neural networks, recurrent neural networks, auto encoders, generational adversarial networks, and many more are just a few of the many diverse types of deep learning approaches.

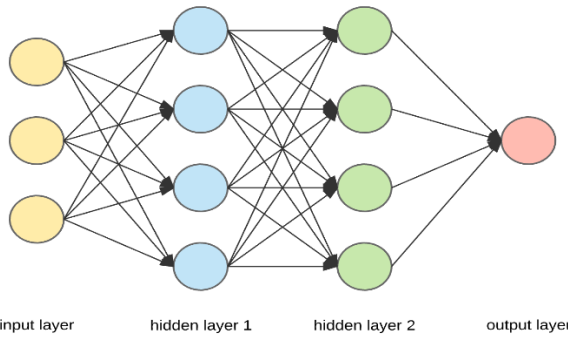

Figure 2 Deep Neural Network with Multiple Hidden Layers [9]

Deep Q learning is now one of the most well-known methods for developing autonomous vehicles. A novel avenue for addressing navigation and autonomous vehicle control issues is deep reinforcement learning. By analyzing its actions and interactions

with the environment, the controlling agent of the vehicle can learn from mistakes using Deep Q-network (DQN), a widely used deep reinforcement learning method [10]..

2.3 How does Deep Q learning work in Autonomous vehicles?

The development of autonomous self-driving vehicles has profited recently from the rapid expansion of machine learning and artificial intelligence, notably in the domain of camera perception. Deep Neural Network (DNN) technology has gained more attention and interest due to the advent of deep learning and improvements in computer power, notably in the Graphic Processing Unit (GPU). Sentence categorization and speech recognition are only two examples of the various uses for DNN, or the stacking of several layers of Artificial Neural Networks (ANN). The time and effort needed to manually engineer feature extractions has been reduced thanks to the DNN architecture's success in learning feature representation. By adding more hidden layers to a DNN, learning capacity and task performance may be enhanced. One such technique that has seen significant advancement since the introduction of DNN is Reinforcement Learning (DRL), which has been applied in a number of applications such as autonomous voltage control for power grid operations, battery management systems, network traffic signal control, and human-machine collaboration [10].

2.3.1 Neuro Evolutionary approach to Deep Q learning:

We previously covered how Deep Learning/Deep Reinforcement Learning excels in areas like autonomous car control jobs. However, three significant obstacles still prevent the widespread implementation of these strategies to issues in the real world: Long-term temporal credit assignment with little incentives, little diversification in exploration, and brittle convergence features.

First of all, it's challenging to correlate actions with results when the incentive is insufficient (ap-pearing only after a sequence of acts). The temporal credit assignment problem, as it is commonly known, affects the majority of real-world fields. Bootstrapping is utilized in real-world settings to get around this problem, but it frequently fails when the time horizons are long and the rewards are limited. Multi-step returns help to solve this issue, albeit they perform best when used in on-policy scenarios. Off-policy multi-step learning is robust, according to recent studies, but it necessitates additional corrective procedures like significance sampling, Retrace, and V-trace, which can be computationally expensive and constrained.

Second, RL relies on exploration to set up superior rules and avoid hasty convergence to local optima. DRL continues to struggle with high-dimensional action and state space exploration. Numerous approaches, such as count-based exploration, intrinsic motivation, curiosity, and variational information maximization, have been suggested to address this problem. One group of methods concentrates on exploring the parameter space of the agents by directly adding noise to it. However, each of these approaches either makes use of sophisticated auxiliary structures or incorporates sensitive elements that are task-specific. Creating a generic exploration technique that can be used to different domains and learning algorithms is a current research area [11].

Finally, DRL methods usually show brittle convergence characteristics and are renowned for being sensitive to hyperparameter selection. This is particularly true for DRLs that violate policy and use replay buffers to save and reuse prior experiences. Although the replay buffer is necessary for sample-efficient learning, the convergence properties become incredibly fragile when paired with a deep non-linear function approximator [11].

ERL also carries over EA's population-based methodology, which generates redundancies that support convergence characteristics and enhance the learning process. ERL also integrate parameter and action space investigation using the population, leading to a variety of policies that effectively investigate the domain.

2.3.2 Evolutionary Reinforcement Learning:

In this hybrid technique, different experiences are created using a population-based methodology to train an RL agent, and then gradient information is periodically injected into the EA population. The key conclusion is that the main challenges of DRL can be addressed by an EA without compromising the usage of gradients for improved sampling efficiency. ERL inherits EA's ability to resolve the issue of temporal credit assignment by employing a fitness metric that consolidates the return of an entire episode. Based on this fitness, the ERL selection operator provides selection pressure to portions of the policy space that produce a higher episode-wide return.

His method causes the state distribution to be biased in favour of regions with higher long-term returns. This implicit prioritization performs best in fields with lengthy time horizons and few incentives.

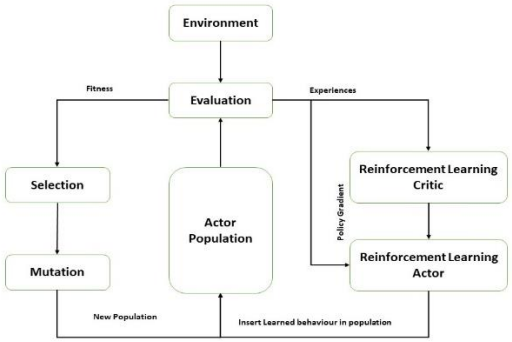

Figure 3 Evolutionary Model for Deep Reinforcement Learning [11]

ERL also carries over EA's population-based methodology, which generates redundancies that support convergence characteristics and enhance the learning process. ERL also integrate parameter and action space investigation using the population, leading to a variety of policies that effectively investigate the domain.

3 System Details

3.1 Simulation Environment

The system allows the user to run a default simulation where the environment for autonomous vehicle navigation is pre-set or create their own environment using a mini drawing application built into the system. Once the environment for the simulation is determined, the system allows the user to configure the various parameters of the NEAT algorithm. Configuration of parameters and environment leads to the start of a simulation, where the user can view the vehicle's performance. The simulation environment operates at 30 fps (frames per second), meaning every single frame lasts for 1/30 second.

The vehicle's fitness is calculated by counting the number of seconds it lived in the simulation environment Sl. We then take an arbitrary threshold T and subtract the number of seconds from it. This entails that the faster the vehicle navigates through the environment, the fitter the vehicle is. The fitness function is demonstrated by the equation below.

$$f(x) = T - S_L$$

Where f(x) represents the fitness function. Variable T represents the arbitrary threshold, and S_L represents the seconds lived by the vehicle. Once the desired fitness of a vehicle is reached, the simulation ends, and the final structure of the best-performing genome (solution) is presented to the user along with many other data visualizations that graphically present the performance of the NEAT Algorithm.

3.1.1 Vehicle Actions:

In our implementation, a vehicle can increase or decrease its speed, the vehicle can perform movements like steering left, right or going in a straight direction. Such movements and speed are determined by the output of the evolving neural network. The vehicle can also recalculate its coordinates to a starting position once the finish line on the simulation environment is reached.

3.1.2 Vehicle Sensors:

Figure 4 below shows various sensors that the vehicle has. These sensors represent distance values from the vehicle's current coordinates to the road's edges or other objects such as obstacles. The distance value is calculated using the Euclidean distance formula, as shown in the equation below. The distances that are obtained from the vehicle's current location later serve as inputs to our neural network.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

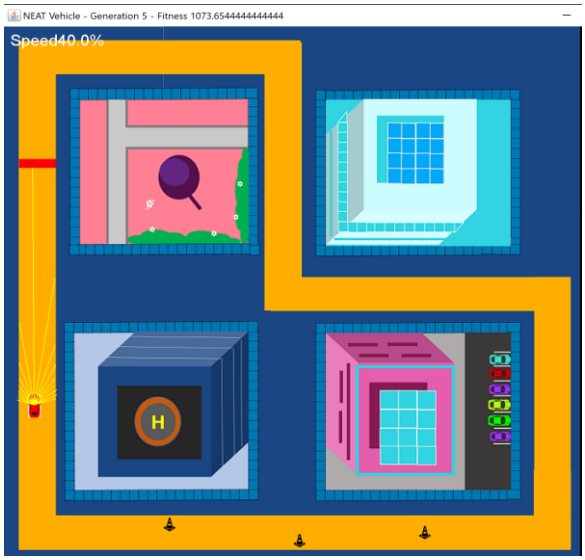

Figure 4 Default simulation environment

4 Our Approach

In this paper, we model autonomous vehicle navigation as an optimization problem. The approach uses NEAT (Neuroevolution of Augmenting Topologies) inspired algorithm to guide autonomous car simulation towards optimal results, as mentioned in Section 2. Our approach falls under the category of genetic algorithms for the generation of evolving artificial neural networks (a neuroevolution technique). It modifies network architecture and weighting factors to strike a balance between the variety and fitness of developed solutions [12].

The proposed solution is divided into three main steps, which are as follows:

1. Input or data collection
2. Prediction using the NEAT Algorithm
3. Simulating the outputs of the neural network in a simulation environment

The main architecture of the NEAT algorithm involves Artificial Neural Network and Genetic Algorithm as architecture optimizers. Predictions from the neural network are passed to the simulation environment to determine the best movement by the vehicle in the environment. The prediction outcome differs from other AIs since NEAT will produce a collection of probability for all potential actions rather than the precise projected action. The simulator will then process this collection of possibilities.

4.1 Collection of Sensor Input

To predict the vehicle's movement, we must analyze the various distances between the vehicle and the road's edges. Getting such distances requires us to gather sensor data in

real time. The data will become the attribute for the prediction in the next step. Currently, the data consists of 18 distances obtained from each car sensor, as shown in figure 4 above. Therefore, 18 different distance values will be collected during this stage at each coordinate of the vehicle itself. Sensor input collection is done using Algorithm I.

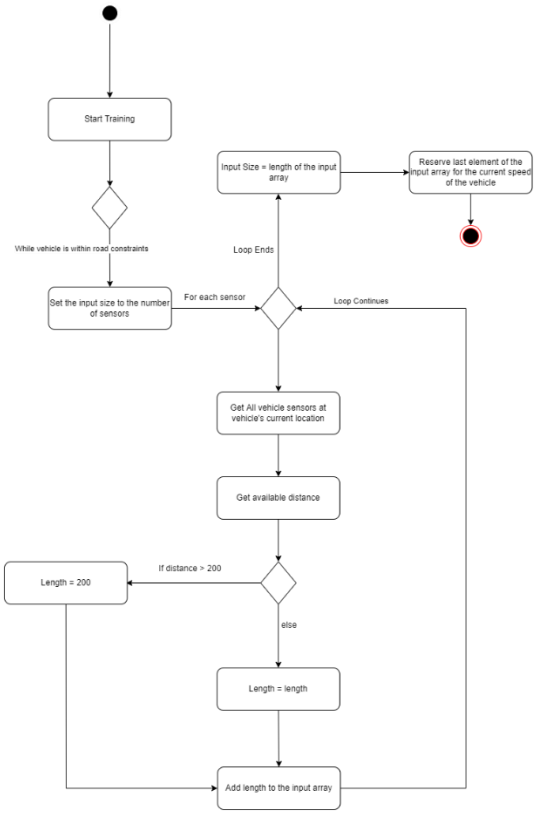

Figure 5 Algorithm I Activity diagram for sensor input collection

In Algorithm I, we also reserve the input array's last element for the vehicle's speed at its current location. This implies that our neural network will have two outputs and 18 inputs. The first output generates a double value used to determine the vehicle's direction, and the second output determines the vehicle's speed.

4.2 Prediction using NEAT Algorithm

Once the data has been collected to analyze the vehicle, we pass all the gathered data from the sensors to the NEAT algorithm. There are three steps that the NEAT algorithm follows, which include prediction, learning and evolution.

The NEAT algorithm starts with a simple ANN structure comprising 18 input and two output nodes. In NEAT's terminology, the ANN's initial architecture is considered a genome with 18 input node genes and two output node genes. Initially, the genome does not have any hidden layers. The genome represents the vehicle in our simulation. This architecture will become the initial point of the NEAT algorithm. The output of such architecture is the best performing genome out of all the species and their respective genomes (genome with the highest fitness). It serves as the input to our simulation environment, which displays the vehicle's performance.

As the simulation progresses, at every generation, a group of elite genomes are preserved that are to be carried to the next generation based on their respective finesses. Genomes within the same specie groups undergo a process of mutation where slight changes are made to the topology and the weights of the genomes. The genomes then go through the crossover process to generate a new offspring or a child's genome. At each step, underperforming genomes undergo a process of stagnation where they are removed from the specie groups. Once again, based on fitness, the best genome is selected from the entire population simulated in the simulation environment.

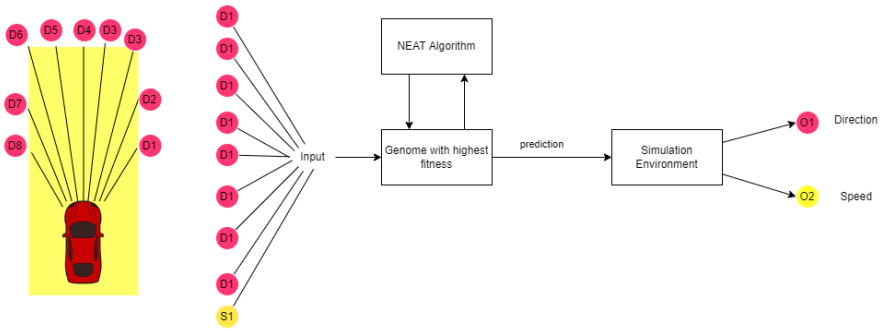

Figure 6 Simulation and Prediction steps

4.3 Simulation

After the prediction step, the prediction will be processed in the simulator component of the system. The simulator will simulate all the combinations of the vehicle's movement and speed at every coordinate on the simulation grid. During the simulation, the genome with the highest fitness is simulated in the environment as shown in the figure 7 below.

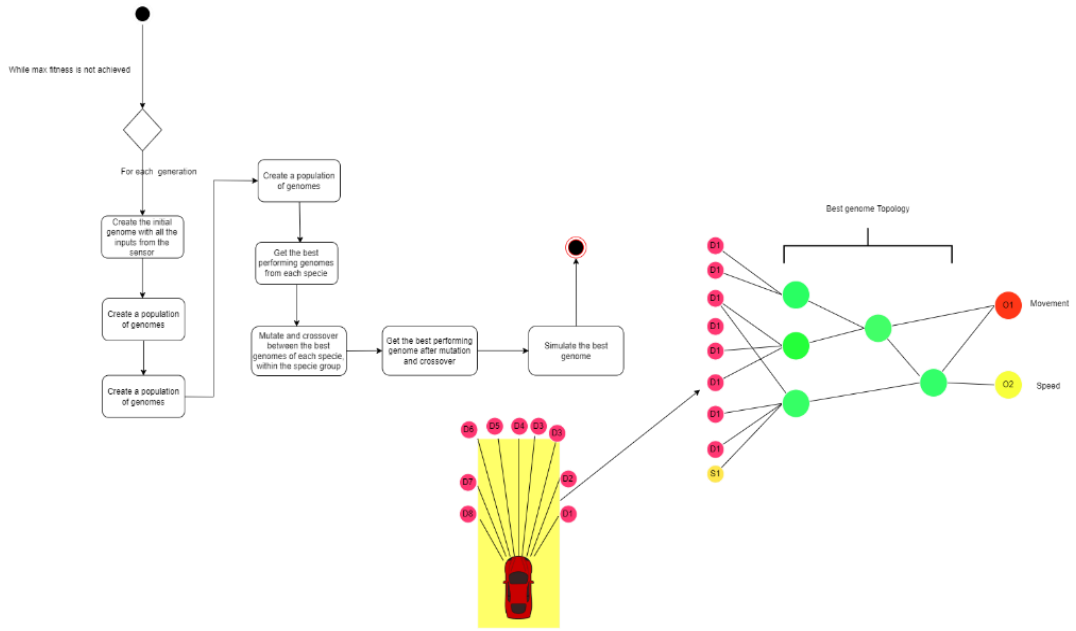

Figure 7 Overall simulation structure

Results

During the course of study, several experiments were carried out to view the overall performance of the NEAT Algorithm. Figure 8 below shows a generational fitness graph describing how average generational fitness fairs against the fittest genome's fitness in that generation.

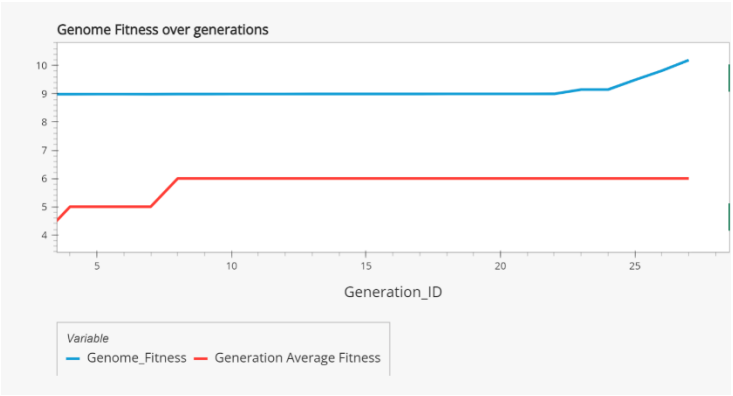

Figure 8 Generational fitness per generation

The graph above shows how the highest fitness, represented by a blue line, remains constant for several generations till we have a steep increase in fitness. This alludes to the evolutionary process where in real life, the evolutionary process does not stop but goes into hiatus until the need to change a characteristic emerges. We observe that the average fitness of the rest of the genomes in the generation is always lower than that of the fittest genome.

Figure 9 below shows the number of active species per generation. During the evolutionary process, the fittest parents in each specie group tend to produce one or more offspring. Once an offspring is created, there is a slight chance that the newly generated offspring will undergo mutation. As new features and topologies emerge through evolution, we see new specie groups forming with distinct characteristics. The figure below shows an increase in active species per generation. This conforms to the fact that the genomes in our population are going through the steps of crossover and mutation to produce more offspring leading to new species groups.

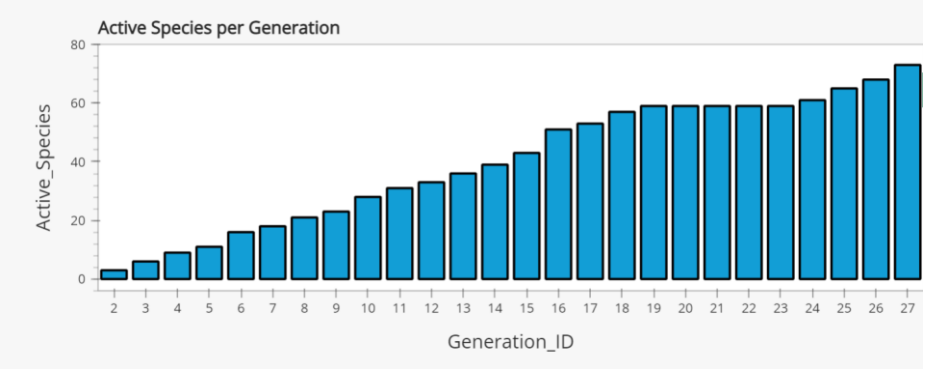

Figure 9 Active species per generation

Conclusion

The proposed paper successfully implemented autonomous vehicle navigation control in a simulation environment using the Neuroevolution of Augmenting Topologies. Our methodology modelled autonomous vehicle navigation as a learning problem where the input to our algorithm was the various distance measures from the edges of the road and the vehicle's speed. During our simulation, every generation tends to produce a vehicle more suitable to the simulation environment. Topology of the genome also evolved from a minimal structure as the evolutionary process progressed through discrete steps conforming to the theory behind the Neuroevolution of Augmenting Topologies.

Since its inception, NEAT has seen enormous breakthroughs in research, from the XOR function's evolution to the complicated present-day models. Future NEAT research may include further analysis and experiments on how to incorporate learning better to meet the higher computational demands of autonomous vehicles. Our survey findings lead us to the conclusion that the NEAT approach will exceed the most recent

state-of-the-art models in the future and play a dominant role. Future advancements in computing power will make it possible for the feature-designing portion of neural networks to transition to automation, replacing the existing human interaction techniques.

References

1. Bissell, D., Birtchnell, T., Elliott, A., Hsu, E.L.: Autonomous automobilities: The social impacts of driverless vehicles. Current Sociology. 68, 116–134 (2018).
2. Schoitsch, E., Schmittner, C.: Ongoing cybersecurity and safety standardization activities related to highly automated/autonomous vehicles. Lecture Notes in Mobility. 72–86 (2020).
3. Wiseman, Y.: Intelligent transportation systems along with the COVID-19 pandemic will significantly change the transportation market. The Open Transportation Journal. 15, 11–15 (2021).
4. Hancock, P.A., Nourbakhsh, I., Stewart, J.: On the future of transportation in an ERA of automated and Autonomous Vehicles. Proceedings of the National Academy of Sciences. 116, 7684–7691 (2019).
5. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation. 10, 99–127 (2002).
6. Kowaliw, T., Bredeche, N., Doursat René: Growing adaptive machines combining development and learning in Artificial Neural Networks. Springer Berlin, Berlin (2016).
7. Gomes, J., Urbano, P., Christensen, A.L.: Evolution of swarm robotics systems with novelty search. Swarm Intelligence. 7, 115–144 (2013).
8. Abraham, A.: Artificial Neural Networks. Handbook of Measuring System Design. (2005).
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. 521, 436–444 (2015).
10. Quek, Y.T., Koh, L.L., Koh, N.T., Tso, W.A., Woo, W.L.: Deep q-network implementation for simulated autonomous vehicle control. IET Intelligent Transport Systems. 15, 875–885 (2021).
11. Khadka, S., Tumer, K.: Evolutionary Reinforcement Learning. V1, (2018).
12. Kristo, T., Maulidevi, N.U.: Deduction of fighting game countermeasures using neuroevolution of augmenting topologies. 2016 International Conference on Data and Software Engineering (ICoDSE). (2016).