Autonomous Car Simulation

Kieran Apps (B822620)

Loughborough University



Table of Contents

1. Introduction and Description
   * + 1. Brief Description
       2. Aims and Objectives
       3. Short Summary of the Project
2. Literature Review
   1. Machine Learning Methodologies; Supervised, Unsupervised and Evolutionary
   2. Neural Networks and Deep Learning
   3. Genetic Algorithms (and NEAT)
   4. Reinforcement Learning
   5. Methods for Automated Vehicles
3. Methodology Used
4. Implementation
5. Results
6. Conclusion and Further Work
7. References

Section 1: Introduction and Description

1.1 Brief Description

Choosing this project was relatively easy for me, cars are a big passion of mine, and the area of self-driving and autonomous cars has exploded in popularity in the recent years, with the likes of Tesla and their own Autopilot system in their vehicles.

Creating this project will give me great knowledge of this new emergent field of A.I. and allow me the learn and practically try out some of the key techniques that are used to create autonomous driving systems. The project itself will be a simulation of an autonomous car using Unity to host the simulated environment, track and car itself, then, Python will be used to program the Artificial Intelligence used to control the cars actions thanks to its useful libraries and lots of support around the language for A.I..

Using this combination plays both to my strengths of my knowledge of Python already and allows me to explore a new area of Python, that being the A.I and Machine Learning side of the language, as well as explore a new language C# with Unity. Thus, developing my skills of developing A.I and learning new language quickly and efficiently such that I can implement the simulation needed.

I believe this project will also allow me to explore some Machine Learning and A.I. techniques that truly interest me, allowing me to develop a deep understanding of such techniques so that I will be able to apply them to many other fields. On top of exploring the new aspects of A.I. that I have not previously been exposed to, I will also be able to collect some knowledge from previous University modules into this one task proving that I can apply what was previously taught to a new situation.

For the track that the car will have to navigate, I will be taking a real racetrack found in northern England called Cadwell Park, this way, I can ensure a wide variety of corner types, angle as well as some lengthy straights for the car to adapt to. Not only does it ensure a wide range of corners for the car to learn, but it will allow for a direct comparison to real cars on the same track such that we can see how similarly this A.I. performed to real experienced drivers on the same track. The car will likely not behave in precisely the same way, however this comparison could give some extremely useful insight into the next steps of the car, and the current successes that the car made within this project.

1.2 Aims and Objectives

My main task for this project is to create a successful autonomous vehicle that can navigate a predefined track successfully avoiding any collisions with the track itself (i.e., without hitting any walls). Building on from this base task, I want to try and create a car that is able to navigate the course at speed, meaning that the vehicle is able to learn to accelerate and decelerate (or brake) for corners and straights, rather than the car just maintaining one constant slow speed to travel around the track.

There have been previous papers and projects exploring the simulation of an autonomous vehicle, however, most of these projects aim for the base case that I want to achieve, this being a car that can navigate around a predefined track safely without colliding with any of the environment. Alternatively, these projects used a slightly different case which is navigating around in a traffic light environment meaning following road laws, signs and potentially even avoiding collisions with other motorists.

My project differs in the aspect that I want to achieve a more advanced car that as mentioned, is able to understand a path and use this knowledge to accelerate and brake accordingly to provide a faster lap, whilst also remaining safe and avoiding the environment.

I also plan to have the car ‘see’ in a similar way to which real autonomous vehicles ‘see’ using Lidar by implementing a similar method into the simulation to gather data about the environment that the car will use to navigate.

In a simplified bullet point list, the progress and objectives will be as follows, from the most basic and beginning of the project, to implementing the A.I. into the car:

* Create 3D model of Cadwell Park
* Create Unity project, load in the track and create a car model
* Create the car controller
* Create all of the event handling within Unity (for collisions, resetting the car and maintain knowledge of the current world state)
* Start the Python side of the project
* Add in a WebSocket to both Unity and Python
* Set up the Neural Network generation and forward pass
* Implement the Genetic Algorithm (this choice will be explained further in this report)
* Train the cars using a multitude of different network topologies
* Analyse the performance of these different topologies

1.3 Short Summary of the Project

All in all, I will be developing an autonomous car simulation using the latest methods in Artificial Intelligence such as Deep Neural Networks (Deep Learning) and Genetic Algorithms to help evolve the car into a safe, effective and fast autonomous vehicle which then will be comparable to a nearly identical real-world scenario to find the strengths, weaknesses and improvements needed for this A.I. system.

Section 2: Literature Review

As part of this project, a considerable amount of research into relevant and potential methods on how to create, control and train the A.I. of the car was undertaken. Outlined in this section is a report on all of my findings and how they can be applied to this area of Artificial Intelligence.

2.1 Machine Learning Methodologies; Supervised, Unsupervised and Evolutionary:

In machine learning, there are numerous ways for a computer to ‘learn’, these main methods being Supervised and Unsupervised learning. Each of these methods has its benefits and drawbacks as well as being more applicable to certain applications that to others.

Supervised Learning:

Supervised learning is the most common type of learning in the Machine Learning space currently and as a result has many algorithms associated with it. This method relies on having knowledge of a data set provided [1] and then uses this knowledge, after making its own predictions, to adjust the parameters within the network using a concept known as back propagation [2] so that the next predictions it makes will be more in line with what is actually labeled/categorized within the data set.

We can use this method to help classify areas within an image that the car/vehicle is using to see to help defined certain objects within the image. For example, the roadway can be given as training data [3] and used to teach the machine, thus the car, where it is allowed to navigate to, as well as where within this navigable space is free from obstructions. Not only is this method useful for classification of roadways, but also pedestrian detection can be performed using this method [4].

Convolution Neural Networks (CNNs) are most typically used to process and perform the object detection and classification of these types of images. This type of network is extremely similar to the usual Neural Network architecture in that there are many layers of neurons connected by weighted edges [5] however, the convolution layers usually behave differently in how they achieve their own output values and is also mainly used to process images. These layers take the image and pass a filter over each of the pixel values in the form of a matrix to produce the output of that convolution and convolution layer. A typical filter size is 3x3, hence, this will also shrink the image size on output.

Therefore, supervised learning with the aid of a CNN is a potential combination of techniques to allow a car to know where to drive and recognize objects of importance within its field of view, such as sign posts, pedestrians and other cars allowing it to then make decisions on where to navigate to.

Overall, this method of taking video from a vehicle and analysing it to perform a whole manner of operations like segmentation (finding the drivable area, as described) and object detection (such as for pedestrians) [6] is widely used in the pursuit of autonomous vehicles.

Unsupervised Learning:

Usually, unsupervised learning is used in a case where there is an extremely large volume of data provided to a system and is unlabeled or uncategorised.

Within this method there are again lots of different techniques and algorithms that can be applied to better fit the situation that it will be used in with some of the most widely known being Hierarchical Learning and Data Clustering [7].

Data Clustering [8] is perhaps one of the most popular methods of this learning type and is used to find patterns throughout a large set of unlabeled data and then use this to group (cluster) together similar points of data. One method for this type of clustering is k-means clustering. K-means clustering [9] uses a point called a centroid which upon initialization are randomly placed within the data space then used to calculate the Euclidian distance to all other points with these points then assigned to the cluster centroid closest to them. After this, the centroid is recalculated to be the centre of the cluster created and the process repeats. The process repeats k times, hence k-means, since this method can be prone to converging in an undesirable location, or local optima, so running multiple times can help relieve this issue to gain a result that will be successful enough in a practical scenario.

These types of methods however cannot easily be applied to an autonomous driving scenario as this is mostly a data mining technique which under the circumstances of an autonomous vehicle would not aid in creating a safe and efficient self-driving machine.

Evolutionary:

Evolutionary learning is a relatively new technique within the Artificial Intelligence space and at its core, takes inspiration from the concept of Darwinian evolution [10] and applies this to the Neural Networks or systems that need to be taught to perform a certain task. Because of this inspiration from nature and the natural path of evolution that life takes, a way to judge and discriminate against different individuals in a population must be conceived so that only the best can move forward to produce the next generation, thus improving the A.I.’s ability to perform the given task.

The function required to determine this performance is called a fitness function and is used to perform the fitness evaluation of each individual which then allows for the reproduction of these best individuals to take place.

Reproduction, as the name suggests, is creating a new set of individuals (reproducing them) to populate the next generation, however this itself does come with some challenges. Firstly, is the method to perform this reproduction, and the second is how to prevent the individuals from becoming too similar too quickly and thus not being able to explore a large enough search space to find acceptable results. To combat these issues, cross-over and mutation [11] can be implemented into an evolutionary algorithm to reproduce the individuals chosen during fitness evaluation.

Cross-over:

Crossover is the act of taking a gene/chromosome from one of the parents in the set that passed the fitness evaluation the best and using this to ‘split’ the two parents into two separate genome sequences. After this point is chosen the two parents swap portions to mix together their chromosomes creating this child individual in the new population.

Another way to perform cross-over is with uniform cross-over which is where single chromosomes of an individual are randomly taken from one of the parents and placed into the child individual at the same point this gene was taken from.

Figure: 1 (Showing uniform crossover)

Text, letter

Description automatically generated

Here (Figure 1, above) you can clearly see that with uniform cross-over for the first gene in the sequence of Child One is taken from Parent One and the second in Child One is from Parent Two.

Mutation:

Mutation is used to at random, change the value in any specified slot in a genome of an induvial during the cross-over phase of reproduction to any other valid value. The rate at which mutations occur can be variable and given by a value such as 0.1 to mean a 10% chance that a mutation will occur for this individual.

For a binary individual, such as Figure 1 and Figure 2, the mutation is as simple as flipping a bit from 1 to 0, or 0 to 1. In more complex examples such as a Neural Network the value can be a replacement of a weight or bias to a new random value within the specified range that these weights and biases can exist in, for example, between 0 & 1 or -1 & 1 are some commonly used values.

Figure: 2 (Showing mutation)

Graphical user interface, text, application

Description automatically generated

Both of the above techniques are extremely useful and relevant when using a Genetic Algorithm (GA) [12] as the main algorithm for this reproduction method.

Genetic Algorithms, as well as some other methods such as NeuroEvolution of Augmenting Topologies (NEAT) [13], will be discussed later in this paper’s literature review (Section 2.3).

2.2 Neural Networks and Deep Learning:

Neural Networks (NN) [14] are the core component to a lot of A.I. applications and take inspiration from the neurons within the human (or other mammalian) brains. This meaning that there are neurons that take input from other neurons and depending on the signal provided, produce some sort of output that is either strong enough to trigger an action or is too weak to do anything, and some other actions is taken.

How is this concept applied to computation?

To digitize this concept of a neuron, the first thing to be done is find a neat way to represent this neuron in a computer system. The representation of this neuron consists of the value of the node, and a set of weights and biases associated with each incoming node used to signify the strength of a certain node (or neuron) similarly to how connections in the human brain can be strong and weak.

Diagram

Description automatically generatedFigure: 3 (Simple Neural Network (Visualisation created using: <http://alexlenail.me/NN-SVG/index.html>))

In Figure 3 above, an extremely basic NN topology is provided to help aid in visualizing how these weights and biases link the inputs to outputs. Just two inputs, x1 and x2 are connected to an output node, x3 and both of these connections have a random weight (when initially assigned) w1 and w3 with the output node x3 also having a bias attached to it, b1. Together, using the input values, weights and bias the value of the output can be calculated. The formula for this is single node as follows:

This is repeated for all nodes in a given layer, Figure 3 only needs this computed once for the single output, but if there were more outputs, or this was a hidden layer with multiple nodes the calculation may need to be computed for 3, 4 or n many times for the size of the new layer.

Training a Neural Network can involve many different techniques, such as some explained in sections 2.1, with the main method being back propagation for a supervised learning technique.

Back propagation is performed after a forward pass of a given network has been completed and the predicted results are compared to the actual results in the training set of data. At the core of this method is the concept of ‘loss’ which is essentially the error rate of the network during its forward pass [15], and this is then used to ‘back propagate’ across the network updating the weights using the root mean squared error as it travels back through. This means that the next time the forward pass is performed all of the weights will produce a slightly more accurate prediction since this loss alters them slightly in the correct direction of the learning gradient.

Another method of training a Neural Network is using the Evolutionary Learning methods mentioned in section 2.1, with one common evolutionary algorithm being Genetic Algorithms, which will be explained in section 2.3 of this report.

Most NN’s are much more complicated than Figure 3 shows and tend to look more like Figure 4 (below), with many more layers creating a Deep Neural Network.

Figure: 4 (Complex Neural Network)

Diagram

Description automatically generated

Deep Neural Networks (DNN) and Deep Learning are still in essence extremely similar to usual Neural Networks and learning; however they operate on a much larger scale. DNNs are architecturally the same as a normal Neural Network however, they have many more hidden layers between the input and output. For example, a normal Neural Network can have 0 or 1 hidden layers, and if there are 2 or more hidden layers, this then constitutes as a Deep Neural Network.

With this larger topology of Neural Networks, the usual training method of back propagation does not always work as the loss can diminish as it travels further up the network, thus is not able to alter the weights correctly. These networks can also (if the correct training method is used) be overfit if there is not enough data to train the network, meaning that the size of a network can also be slightly dependent on the data provided to it.

To solve this loss training problem, a solution called a residue block can be implemented to ensure the strength of the signal/loss is not lost. In simple terms, this allows the signal to be passed back along a network separately from the signal/loss that is also being used to process the changes needed to the weights.

Figure 5: (Residue Block)

Diagram

Description automatically generated

Employing this method allows for these larger networks to be trained effectively and make proper use of their size and capabilities with more layers and nodes. An example of a network that operates this way is the ResNet [16] architecture.

2.3 Genetic Algorithms (and NEAT):

Genetic Algorithms (GA) [17] are a popular form of evolutionary learning that as mentioned in section 2.1 take direct inspiration from Darwinian evolution and apply this technique to how the network is modified and evolves to create a better performing Neural Network. This algorithmic technique takes all of the key concepts from the whole of evolutionary learning and applies them, such as the cross-over and mutation functions.

GAs use chromosomes to represent the attributes within a network take some genetic operators to help guide the population of individuals into the correct direction to reach a global optimum [18]. Parameters are initialized to be a random set, for a Neural Network this would mean initializing a random set of weights and biases to fit the size of the network that is desired, with each of these values being the genes or chromosomes to be used within the GA.

Once the population have been tested and run, each will have gained a fitness value calculated by their performance within the given task, some examples of parameters to factor in while creating the fitness function could be the accuracy of the output or time taken to perform the task at hand. Next the parents of the next generation are selected using these fitness values, with a higher value being the better performing individuals, therefore giving them a higher chance of being picked for the cross-over. Alternatively, the top 2 or 3 performing individuals can be picked, removing the chance element of the selection process, ensuring the best genes are carried into the next generation.

In a pseudo-code format this could look something like:

Figure 6: High level pseudo-code of a Genetic Algorithm

initialize individuals with random weights

while **min** **performance** **NOT** met:

for each **individual** in the **population:**

perform the task

evaluate performance

save fitness value to the individual

end for

select best **individuals** from **population**

for **newIndividual** in **newGeneration:**

select crossover point

swap genes and save as n**ewIndividual**

end for

end while

2.4 Reinforcement Learning:

asd

2.4 Methods for Automated Vehicles:

asd

Potential sources:

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6889488>

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4250190>

<https://reader.elsevier.com/reader/sd/pii/B9780128154809000153?token=85B4DB0FAF627AB0D2EBD20C68C0168A246BC22CC8535DC07A45B22037D1A3116004C39E5440C7C40F0410E3FE7ED5C9&originRegion=eu-west-1&originCreation=20220330153109>

<https://arxiv.org/pdf/2006.15175.pdf>

<https://arxiv.org/pdf/2109.06126.pdf>

Section 3: Methodology used

Why am I using what I am using….? Base this off of the literature review section and what I found in that

Section 4: Implementation

Section 5: Results

Section 6: Conclusion and Further Work

References:

[1] Schrider, Daniel R., and Andrew D. Kern. "Supervised machine learning for population genetics: a new paradigm." *Trends in Genetics* 34.4 (2018): 301-312.

[2] LeCun, Yann, et al. "A theoretical framework for back-propagation." *Proceedings of the 1988 connectionist models summer school*. Vol. 1. 1988.

[3] P. Sharma, D. Austin and H. Liu, "Attacks on Machine Learning: Adversarial Examples in Connected and Autonomous Vehicles," 2019 IEEE International Symposium on Technologies for Homeland Security (HST), 2019, pp. 1-7, doi: 10.1109/HST47167.2019.9032989.

[4] Fridman, Lex. "Human-centered autonomous vehicle systems: Principles of effective shared autonomy." *arXiv preprint arXiv:1810.01835* (2018).

[5] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." *arXiv preprint arXiv:1511.08458* (2015).

[6] Chiaroni, Florent, et al. "Self-supervised learning for autonomous vehicles perception: A conciliation between analytical and learning methods." *IEEE Signal Processing Magazine* 38.1 (2020): 31-41.

[7] M. Usama et al., "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," in IEEE Access, vol. 7, pp. 65579-65615, 2019, doi: 10.1109/ACCESS.2019.2916648.

[8] N. Grira, M. Crucianu, and N. Boujemaa, ‘‘Unsupervised and semisupervised clustering: A brief survey,’’ Rev. Mach. Learn. Techn. Process. Multimedia Content, vol. 1, pp. 9–16, Jul. 2004.

[9] Aristidis Likas, Nikos Vlassis, Jakob J. Verbeek, The global k-means clustering algorithm, Pattern Recognition, Volume 36, Issue 2, 2003

[10] J. Zhang et al., "Evolutionary Computation Meets Machine Learning: A Survey," in IEEE Computational Intelligence Magazine, vol. 6, no. 4, pp. 68-75, Nov. 2011, doi: 10.1109/MCI.2011.942584.

[11] Sacha Gobeyn, Ans M. Mouton, Anna F. Cord, Andrea Kaim, Martin Volk, Peter L.M. Goethals, Evolutionary algorithms for species distribution modelling: A review in the context of machine learning, Ecological Modelling, Volume 392, 2019

[12] F. H. F. Leung, H. K. Lam, S. H. Ling and P. K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," in IEEE Transactions on Neural Networks, vol. 14, no. 1, pp. 79-88, Jan. 2003, doi: 10.1109/TNN.2002.804317.

[13] L. Chen and D. Alahakoon, "NeuroEvolution of Augmenting Topologies with Learning for Data Classification," 2006 International Conference on Information and Automation, 2006, pp. 367-371, doi: 10.1109/ICINFA.2006.374100.

[14] Bishop, Chris M. "Neural networks and their applications." *Review of scientific instruments* 65.6 (1994): 1803-1832.

[15] Vogl, T.P., Mangis, J.K., Rigler, A.K. *et al.* Accelerating the convergence of the back-propagation method. *Biol. Cybern.* **59,**257–263 (1988). <https://doi.org/10.1007/BF00332914>

[16] Targ, Sasha, Diogo Almeida, and Kevin Lyman. "Resnet in resnet: Generalizing residual architectures." *arXiv preprint arXiv:1603.08029* (2016).

[17] Holland, John H. “Genetic Algorithms.” *Scientific American*, vol. 267, no. 1, 1992, pp. 66–73, [http://www.jstor.org/stable/24939139. Accessed 6 Apr. 2022](http://www.jstor.org/stable/24939139.%20Accessed%206%20Apr.%202022).

[18] Adaptive and Natural Computing Algorithms, 8th International Conference, ICANNGA 2007, Warsaw, Polamd, April 11-14, 2007, Proceesings, Part II, Bartlomiej Beliczynski, Andrzej Dzielinski, Marcin Iwanowski**,** Bernardete Ribeiro