An Exploration of Machine Learning’s Use for Self-Driving Cars

To what extent does the amount of data given to a machine learning program affect its ability to self-drive a car?

Subject: Computer Science

Word Count: 3539

Contents

[**Introduction** 3](#_Toc89979650)

[Background Information 4](#_Toc89979651)

[Investigation 5](#_Toc89979652)

[Conclusion 14](#_Toc89979653)

[Bibliography 16](#_Toc89979654)

[Appendix A 19](#_Toc89979655)

## Introduction:

Computers, coding, and technology are fundamental parts of our twenty-first century society. For those in Generation Z, technology has been integrated into our lives from the day we were born, through the toys we played with and the smartphones our family used. As someone engrossed in the world of networking and computer science, I carefully considered the ways that these topics will influence our future. Wanting to explore the specific technologies that will majorly affect our future lives, I decided to tackle the idea of machine learning.

Throughout this essay, I will be attempting to answer the question, “To what extent does the amount of data given to a machine learning program affect its ability to self-drive a car?” But what is machine learning? According to IBM, it is “a branch of Artificial Intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do so” (Kavlakoglu). In essence, machine learning uses previously known and inputted data to accurately predict future data. A machine learning program improves its accuracy the more data it is given, but is there a limit to how much it can improve? I hope to understand if there is a saturation point where additional data does not change the accuracy of the algorithm, if a saturation point does not exist so the algorithm always improves, or if there is a different outcome entirely. This is very useful for deducing if there can be “too much” training data given to a machine learning program and if there is a point where one should stop adding data. More specifically, I want to uncover how this applies to the field of self-driving cars. I have long been interested in the topic of fully autonomous vehicles because of how outlandish the concept always seemed. The idea that we could use computers to replicate activities that could only be accomplished by humans for decades is incredible, and I am excited to research the topic for this essay.

The technology for Artificial Intelligence is relatively new, but it is already implemented in many applications and is constantly improved, such as through the development of self-driving cars. Now that fully autonomous vehicles seem to be more within reach, given Tesla’s partially self-driving cars available for sale to the public and Waymo’s work on creating completely autonomous taxis for public use, I thought this served as a great opportunity to research the technology and understand its development better.

# **Background Information:**

As our desires as a society increases, with us needing new conveniences of life, there needs to be new advances in technology. One such advancement that is becoming increasingly more important is that of Artificial Intelligence. Artificial Intelligence is the concept of using computers to replicate complex tasks that normally could only be completed by a human. Artificial Intelligence is already very prevalent in our lives despite being continuously developed through the use of machine learning. For example, Phone assistants like Siri use machine learning to improve their voice recognition and understand our sentences better, while email spam filtration is improved through machine learning by better recognizing suspicious emails.

Machine learning is a subset of Artificial Intelligence that improves over time through experience and learning. Machine learning programs use artificial neural networks that imitate the information-transfer and decision-making process of the human brain through algorithms. Through the collection of large sets of data, a machine learning program can create an algorithm that allows it to “learn.” According to Sanderson, a Stanford graduate and former Khan Academy educator, in order for an Artificial Intelligence program to learn, it must make use of neural networks, which uses multiple layers of nodes that contain data to imitate the process that the human brain goes through (Sanderson). The nodes are analogous to synapses in the brain, as they transfer data forward for different uses in the same way that synapses transfer signals. Using neural networks is considered deep learning, which is a subset of machine learning. The neural network starts with an input layer of nodes based on the initial data given. The data goes through multiple layers that feed-forward into the next layer, going through a specific change each time until it eventually reaches the desired final layer, the output layer.

The process used to accomplish this is large and complex, but it is often simplified through the use of the Python programming language. Python is the primary coding language used for the development of machine learning programs for two reasons. It is very easy to understand compared to other languages, and it contains a large list of comprehensive libraries used for machine learning such as scikit-learn, which provides efficient tools for predictive data analysis, and TensorFlow, which makes it easier to build and deploy machine learning models.

# **Investigation:**

The overall goal of my investigation is to find whether or not constantly adding data into a machine learning program will continuously increase its effectiveness through both online research and through my own experiment. Before conducting my research, I created a hypothesis; I believe that adding more data into a machine learning program will consistently improve its effectiveness in completing its assigned tasks. For example, with the development of Tesla’s autonomous driving feature for its vehicles, there are still many issues that arise while using them, such as cars being unable to identify certain obstacles or acting carelessly during dangerous situations. If my hypothesis is correct, then continuously adding data to the machine learning program will allow the vehicles to continuously improve their accuracy while driving, eventually becoming more reliable than a human driver.

Alongside the source-based research I conduct, I am also using a program known as Udacity’s Self-Driving Car Simulator to help determine the veracity of my hypothesis. Udacity’s simulator was created for a college-level self-driving car engineering program to assist in teaching how to train cars to automatically navigate road courses using deep learning. The program is open sourced on GitHub which provided easy access to this tool for my research.

The first step I took in my research was studying how to actually use and manipulate the simulator for my specific purpose. Guides to manage the simulator were scarce, so I needed to learn mostly on my own. I used Anaconda, which is an application that allows simple package management for Python, to manage the different libraries I needed to install and to run the necessary python files I required in order to train the data I collected. I spent over a month slowly researching Anaconda and Udacity’s simulator to finally understand how to collect data from the simulator, train the data using a pre-made machine learning algorithm, and then run the trained model in the simulator to see how well it drove through two different simulated roads.

While I did not create the simulator myself, I have taken significant time to also try and understand the code behind what is actually happening within the application. The simulator itself is rather simple, as it is essentially a driving game made in the Unity game engine. Its complexity comes with its ability to easily collect data from the movements of the vehicle, which then can be used with a Python machine learning training program to create a model that can run within the simulation. While using the program, the code itself does not know how to identify if it is on the road or not, or if it is improving or not. The code is simply trying to replicate the data that I provide it to the best of its ability through converting as many variables as possible into numbers and making equations out of it. The simulation allows for collection of 7 independent variables: three images from the car of its centered point of view, leftmost point of view, and rightmost point of view; the angle of steering; the state of the throttle from 0 to 1; the state of the brakes from 0 to 1; and the speed of the vehicle. The program puts all these variables into a machine learning algorithm; it uses the images taken from the car and the numerical values of the steering, throttle, brakes, and speed to create a gradient function that most consistently lines up with the data it has received. Using the mean squared error function, the program attempts to find the smallest error between the values it calculates and the data it has received. The closer the error value is to zero, the more accurate its results should be to the data. Since there is so much variable data, the actual model will always end up different, even slightly, from the data it is given as it can be compared to finding the ‘average’ of all the inputted data. This still, however, allows it to create a model that enables the vehicle to drive accurately to the given data on the simulated tracks. This can be seen in the example code in Appendix A.

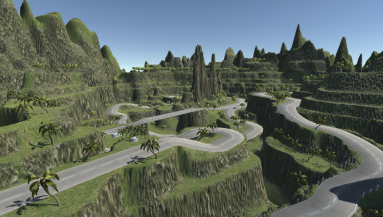
Once the model is created, a different file is used to run it through the simulation and allow the car to drive without any input. For every image it receives in real time from the simulated car, the program uses the model to try and calculate the next course of action, whether it is to speed up to go uphill, or slow down before a turn or going downhill.

With the Udacity simulator, I drove a sample car across two different tracks using the keyboard to control the throttle, steering, and the brakes. The data that the training model would eventually receive was taken from my driving of the car in the simulator. While driving the vehicle, the simulator recorded all the necessary variables that would be needed for it to create a prediction. With the new trained model I created, I ran it through the simulation to have the car drive autonomously and to see how well it managed to drive on the road. Ideally, the trained model would drive with a similar accuracy to my own and would consistently circle around the track without making a mistake. In order to compare the data I collected with my hypothesis, I planned to extensively train the model by driving the car many times through the same track to build up a large spreadsheet of data. The simulator contains two tracks: the first being a simple flat loop that is easy to drive around without mistakes for data collection and the second being a more complex track with lots of turns and pathways going uphill and downhill which made it more difficult to collect large amounts of data on it.



*Figure 1 A photo of what the simple driving track looks like within the simulation*

The first of the two tracks in the simulator was a simple loop on flat ground, on which it was rather simple for me to collect data; hopefully, it would be simple for the model to replicate. With the Udacity simulator, I spent significant time testing different scenarios of driving to see their effects on the outcome for its autonomous functionality. My data collection ranged from driving very small sections of the road to driving it through loops many times over. The results from my test seemed to mostly fit my hypothesis. During tests where the car only drove one lap or less, it tended to crash and veer off the track much more often. When the car drove multiple laps around the course, it ended up consistently driving loops around the track without external support. Adding more iterations of the car driving around the loop seemed to consistently improve its accuracy and lowered its chance of veering off track. This can be attributed to the track being a simple loop on flat ground, so it was not difficult for the algorithm to find its way with no issues. However, significant issues began to occur when using the second, more complex road offered by the simulation.



*Figure 2 A photo of what the complex driving track looks like in the simulation*

Through my different tests on this track, I found the results were drastically different from the first track. This track has many tight turns and slopes, and this makes it difficult for the program to navigate itself around the course the same way, though adding more data by taking extra samples of the car driving carefully around the tight turns increased its effectiveness at doing so. The addition of the element of slopes into this track made the machine learning model have a more difficult time navigating, because it had to process more data whenever it was forced to go uphill or downhill.

At this point is where the limitations with this simulator began to appear and cause issues with my experiment. The simulator uses pictures taken of the road in order to identify where the vehicle is in relation to the road and to stay within its boundaries. However, the simulator’s process seems to consistently cause errors when the onboard “camera” is trying to decide between two different roads that happen to be next to each other in its sight. In these two examples, the program consistently fails and ends up crashing into the fencepost dividers, because it is trying to stay in the middle between the two adjacent roads that it sees in front of it.





Figure 3 Photos of the overhead view of the simulated vehicle and of the view the code sees

*Figure Photos of the overhead view of the simulated vehicle and of the view the code sees*

In this problem, the model used the images it collected to identify whether it is too close to running off the road on the right side, so it attempts to drive to the left to stay away from the edge. In the process, the car would crash into the dividers. In my attempt to fix this, I added more data as my hypothesis predicted this would solve the issue. I trained the model further by giving it a large additional amount of data of driving through this track. Even so, it seemed that it still failed every time in one or both of the locations; it never was able to complete a full lap. Based on this result from my experiment with the simulator, it seems that constantly adding data to a machine learning program does not always consistently improve its efficiency as I thought in my hypothesis.

While this was my result when adding more data, it is important to note that the simulator lacks in other areas that would be incredibly important for real-life cars, such as pedestrian obstacles, stop signs and traffic lights, diverging roads, and more. These are all elements that must be accounted for when driving on a real road which would require the simulator to be much more complex. Since this simulator does not contain all these elements, it makes the results obtained from the simulator harder to compare to the real self-driving cars being developed by Tesla and Waymo.

Though the simulator is not completely accurate to real-life, it does not mean that the conclusion I found was wrong, as this issue seems to occur in the current development of autonomous cars. The current overarching goal for self-driving cars is to reach level 5 autonomy. The Society of Automotive Engineers created an official chart of how to identify the level of driving automation, with level 0 being extremely minor assistance like a blind spot warning and level 5 being that the vehicle can drive itself under all conditions (SAE International). Additionally, according to the U.S. National Highway Traffic Safety Administration, level 5 self-driving cars would have to be vehicles that “can do all the driving in all circumstances, [and] the human occupants are just passengers and need never be involved in driving” (Dickson). In short, this means that the vehicle does not need driver intervention, nor would it allow it. Level 5 vehicles would ideally have no steering wheels and would drive near perfectly under all possible conditions. Tesla CEO Elon Musk previously said in 2020 that he was confident that they would have completed the basic functionality for level 5 cars by the end of the year, though that ended up not being true. Currently, Tesla vehicles, even with their “Full Self Driving” beta, only reach about level 2 or level 3 autonomy, which is partial assistance where the driver still must maintain their focus on the road and be ready to take control at a moment’s notice.

While Elon Musk believes that Tesla can achieve true self-driving capability simply through software updates due to the vehicles containing 8 cameras that analyze the surrounding environment, software engineer Ben Dickson believes otherwise. While there is some sound logic to Tesla’s approach, given humans primarily use our limited vision to drive effectively on the road, the system has flaws because “current neural networks can at best replicate a rough imitation of the human vision system... [Neural Networks] don’t have the flexibility of humans when facing a novel situation not included in their training data” (Dickson). Neural networks and deep learning algorithms currently are not advanced enough to improvise in unknown situations as humans can. In fact, many deep learning and machine learning algorithms today need to be pre-trained for every possible scenario in order to work well in any given scenario. However, training for every possible scenario with driving is practically impossible because there are near limitless things that can occur on the road due to road obstacles and peoples’ actions. To adjust to the chaotic nature of humans, a self-driving car needs to process information in the same way that humans can, but that is unfortunately impossible with what we currently know about Artificial Intelligence.

This leads into a larger problem of interpolation solutions versus extrapolation solutions when considering the development of more complex Artificial Intelligence. Direct-fit interpolation is the idea that a program can learn by constantly learning new things and then filling in the spaces between data points based on what it has already seen, which would eventually cover every possibility leading to true human-level performance. On the other hand, ideal-fit extrapolation is the idea that deep learning is fundamentally flawed because it can only interpolate. Human beings naturally extrapolate data from their surroundings and their memories in order to decide what to do next, so giving that extrapolation ability to an artificially intelligent program would also lead it to human-level performance.

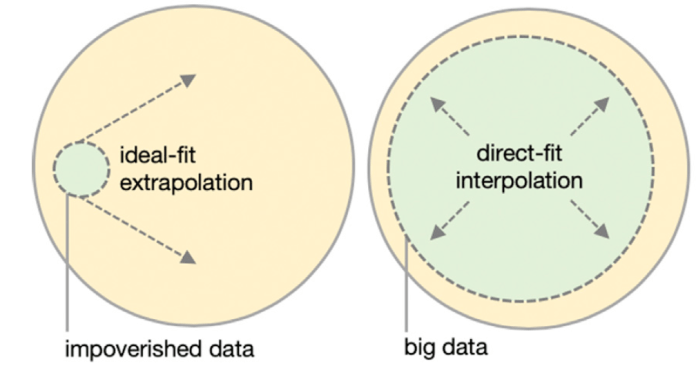


Figure 4 Extrapolation (left) tries to extract rules from big data and apply them to the entire problem space. Interpolation (right) relies on rich sampling of the problem space to calculate the spaces between samples.

Dickson, B. (2020). *Ideal-fit Extrapolation versus Direct-fit Interpolation* [Diagram]. TechTalks. https://bdtechtalks.com/2020/07/29/self-driving-tesla-car-deep-learning/

*Figure SEQ Figure \\* ARABIC 5 Extrapolation (left) tries to extract rules from big data and apply them to the entire problem space. Interpolation (right) relies on rich sampling of the problem space to calculate the spaces between samples.*

In order to create a device that truly acts like a human brain, I believe that it requires elements of both ideas to work to the fullest extent, and this is especially true for self-driving cars. Human drivers need a lot of data or experience from driving in order to drive safely, which is why experienced drivers are generally safer and more reliable than new drivers, but they also need to improvise when something goes wrong. An autonomous car needs the same to function properly: plenty of data about road safety and avoiding obstacles, but also the ability to extrapolate data quickly like humans. Situations requiring extrapolation include, when we see or hear a car that isn’t slowing down and become aware of the potential obstacle, if we know we are driving in an area where drivers are reckless so we have to act accordingly, or any other unusual scenario that is a result of human error. If self-driving cars both know enough about road laws to drive safely on the road and account for and predict situations that could occur due to the error of other drivers or the environment, only then can we finally accomplish that incredible idea of getting in a car and never having to touch a steering wheel.

# **Conclusion:**

As I have learned through the investigation of this essay, the topic of Artificial Intelligence and machine learning in self-driving cars is not something that is easy to research and experiment on because it is a very complex topic still being researched and developed. Given this, finding an answer to the question of how the amount of data can affect a machine learning program for self-driving cars was difficult. Through my research, I have discovered that constantly adding data to machine learning algorithms for autonomous cars will not be the only solution to creating level 5 vehicles. With the Udacity simulator, despite its simplicity, I discovered that constantly increasing my pool of data will not solve every problem that arises from replicating human driving. The simulator had its limitations through missing key aspects of real-world driving such as lane switching, pedestrian obstacles, and road signs, but it had significant value because it still showed the abilities and limitations of real self-driving algorithms. In fact, autonomous cars currently on the market also demonstrate this as they seem to be limited to level 2 and level 3 autonomy due to fundamental issues with deep learning. Current technology is limiting the development of autonomous vehicles from having the same improvisation skills that humans naturally have, causing constant struggle in developing cars that can even reach level 4 autonomy without issue. Attempting to add data for every single edge case while developing autonomous cars is a largely impossible task because of how many scenarios exist given the vast number of variables in the natural environment, from natural human error to changes in the weather. Overall, it is clear that the amount of data needed to self-drive a car is very important to helping it work properly, but it only helps to a certain extent. Reaching true autonomy will require more creativity in the field of Artificial Intelligence.

# Bibliography

Bojarski, Mariusz, et al. “End to End Learning for Self-Driving Cars - Nvidia.” *Nvidia*, 25 Apr. 2016, https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf/n.

Brown, Aaron. “Udacity/Self-Driving-Car-Sim: A Self-Driving Car Simulator Built with Unity.” *GitHub*, 3 May 2017, <https://github.com/udacity/self-driving-car-sim>.

“CICV, Marmara University and Bigtri Join the International Alliance for Mobility Testing & Standardization - SAE INTERNATIONAL RELEASES UPDATED Visual Chart for Its ‘Levels of Driving Automation’ Standard for Self-Driving Vehicles.” *SAE International*, SAE International, 12 Dec. 2018, https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles.

Dickson, Ben. “System 2 Deep Learning: The next Step toward Artificial General Intelligence.” *TechTalks*, 2 Feb. 2020, https://bdtechtalks.com/2019/12/23/yoshua-bengio-neurips-2019-deep-learning/.

Dickson, Ben. “The Limits and Challenges of Deep Learning.” *TechTalks*, 11 Aug. 2019, https://bdtechtalks.com/2018/02/27/limits-challenges-deep-learning-gary-marcus/.

Dickson, Ben. “Understanding the Limits of CNNS, One of Ai's Greatest Achievements.” *TechTalks*, 2 Mar. 2020, https://bdtechtalks.com/2020/03/02/geoffrey-hinton-convnets-cnn-limits/.

Dickson, Ben. “Why Deep Learning Won't Give Us Level 5 Self-Driving Cars.” *TechTalks*, 3 Aug. 2020, https://bdtechtalks.com/2020/07/29/self-driving-tesla-car-deep-learning/.

Johnson, Daniel. “Difference between Deep Learning and Machine Learning vs Ai.” *Guru99*, 6 Oct. 2021, https://www.guru99.com/machine-learning-vs-deep-learning.html.

Kavlakoglu, Eda. “AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?” *IBM*, 27 May 2020, https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

Rahav, Siraj. “How to Simulate a Self-Driving Car.” *YouTube*, commentary by Siraj Rahav, 17 May 2017, https://www.youtube.com/watch?v=EaY5QiZwSP4

Raval, Siraj, et al. “LlSourcell/How\_to\_simulate\_a\_self\_driving\_car.” *GitHub*, 28 May 2017, https://github.com/llSourcell/How\_to\_simulate\_a\_self\_driving\_car.

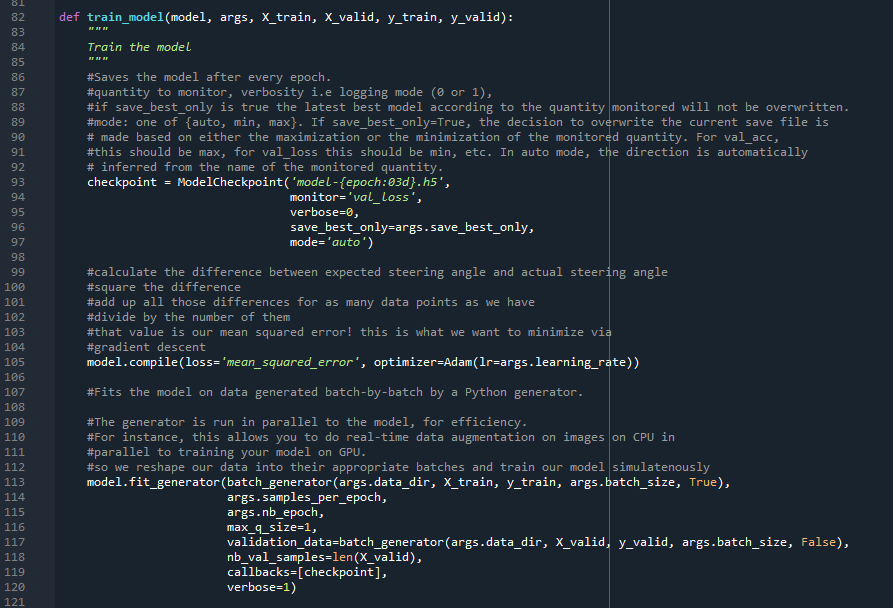
Sagar, Abhinav. “How to Train Your Self Driving Car Using Deep Learning.” *Medium*, Towards Data Science, 26 Nov. 2019, https://towardsdatascience.com/how-to-train-your-self-driving-car-using-deep-learning-ce8ff76119cb.

Sanderson, Grant. “Gradient descent, how neural networks learn.” *YouTube*, commentary by Grant Sanderson, 16 Oct. 2017, https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1\_67000Dx\_ZCJB-3pi

Silva-Zapata, Omar. “Self Driving Car with Udacity Simulator.” *Instructables*, Instructables, 14 Jan. 2019, https://www.instructables.com/Self-Driving-Car-With-Udacity-Simulator/.

# Appendix A

Code Sample



A sample of the code in the model.py file used to train a model in the simulation from my data; the code was created by Siraj Rahav.