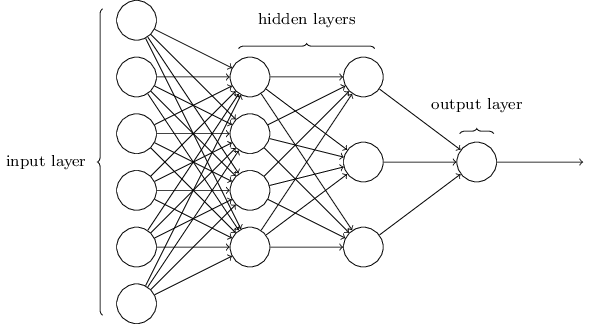
Literature Review

The topic I’m researching is whether we should use convolutional neural networks to clone human behaviour to power self driving cars, or use a mixture of hard code and computer visions. The introduction video *Self-Driving Car Nanodegree Program Overview*(Udacity, n.d.) provided me with general description of how each method works. However, this source did not privide any details on how to build any of the systems, neither the advantages of them all.

## Background Knowledge

As this project revolves around behavioral cloning, there are some background knowledge to understand before one could begin to understand this program. The normal way computer science works is you give a computer an algorithm an input, and it spits out an output. But as one could imagine, writing a program to mimic human behaviour given an image is not easy, so instead another process is used here called supervised machine learning. You give a computer a large number of inputs (called features) and a large of outputs (called labels) and the computer uses regression to find a path from the input to the output so that when given an unlabeled input, it could find the output.

A normal neural network works by connecting breaking down the input parameters into nodes, and connecting them to several hidden nodes organised into layers, then connecting the hidden layers to the output nodes, which gives the output.

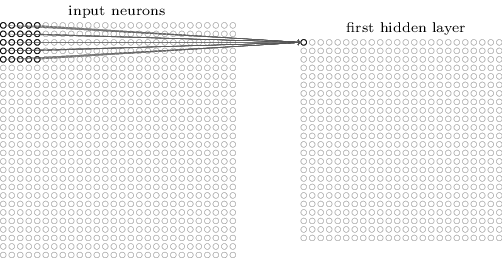


*(diagram from Neural Network and Deep Learning* )

As shown by the diagram above, every node is connected to every node in the next layer, and each connection is called a neuron. Each neuron could be expressed in the form of f(x)=mx+c, m is called the weight of the neuron, and c is called the bias. By adjusting the weight and bias of all the neurons in a neural network through a process called gradient descent, we could approach the labeled output. Then when the network encounters an input that it didn’t see before, it could calculate the output with desired reasonable accuracy.

To mimic a drivers behaviour when driving, we need to design a network that takes an input from cameras mounted on the car, and predict an output of steering angle. We could connect each pixel as an input node, however as an image has 3 dimensions (x,y,rgb), doing this will destroy some spacial information about the image as the input nodes is essentially a list of the pixels.

A better way of extracting output from an image is called a convolutional neural network, it uses similar concepts to a normal neural network, however it preseves the spacial dimention of an image.



*(diagram from Neural Network and Deep Learning* )

As shown in the diagram above, a convolutional neural network uses a local receltive field of specified square dimensions (e.g. 5\*5) and to scan across an image and produces a new hidden layer. Every complete scan is called a filter. This process could be repeated for a specified amount of time and the values are all stored in a new 2 dimensional array in the next layer. So the dimension of one layer of nodes would be x\*y\*number of filters.

The same process is repeated from the 1st hidden layer to the second and so on. Each time loosing 4 pixels from the x and 4 pixels from the y direction. To reduce the amount of computation needed and extract bolder features from the image, sometimes the layers are downsampled by combining several pixels to one.

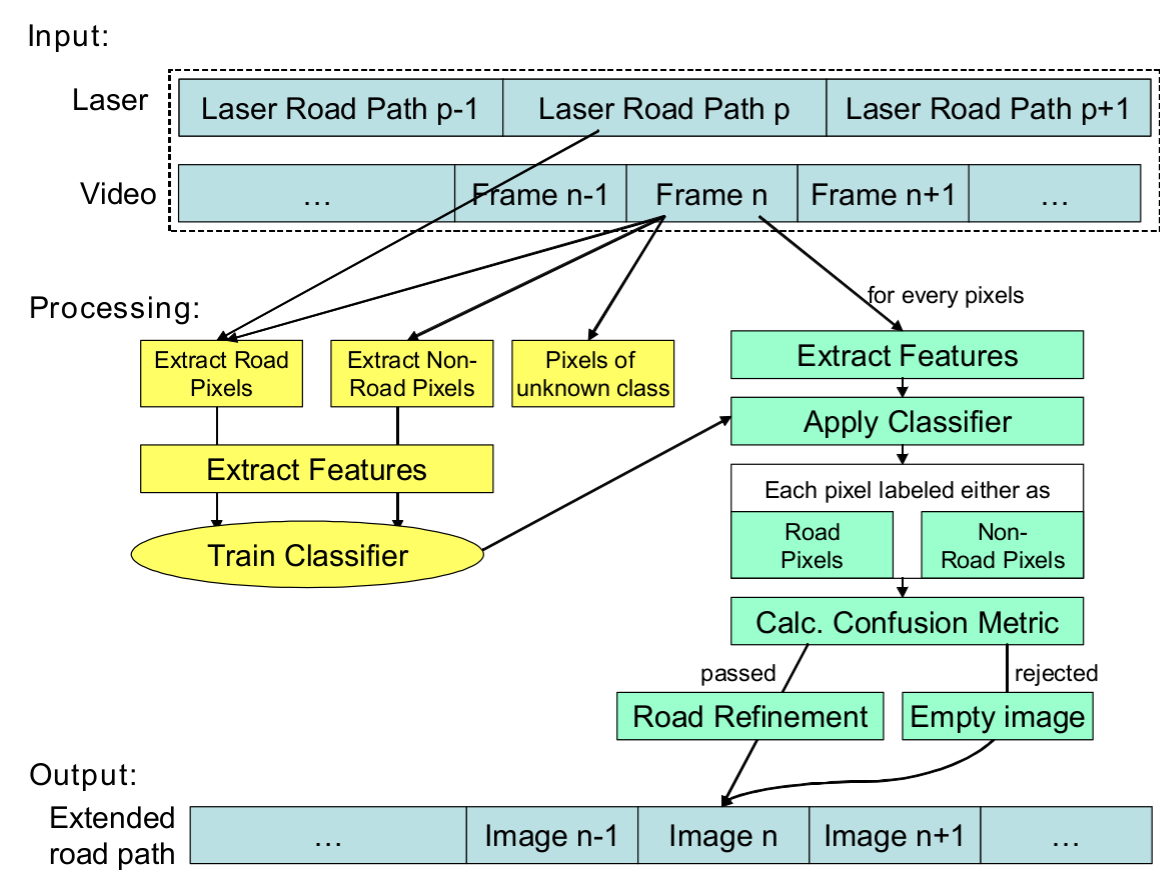
The whole process is then repeated until the x and y direction are both 1, so the dimension of this layer would be a 1 dimentional array of filters which is then connected to a normal neural network to give the output.

As one could see from the description above, the whole process has more parameters to tweek than comventional programs including but not limited: the dimension of the network itself, the learning rate of the network, and choosing the way the network evaluates success (the cost function).

This EPQ would be about how to choose these parameters as to make the steering angle output match what a human would decide to do, and analysing whether a similar system using end to end (straight from input image to output steering angle with no hard code algorithm in between) machine learning could realistically be used in autonomous driving and the benefits and drawbacks of this system vs one that uses hard code and sensors to decide the action of the vehicle.

### History

Many see the second DARPA grand challenge as the breaking point for autonomous vehicles as it was the first time a vehicle successfullly navigated through a desert without human intervention. The winning team(The Stanford Racing team) used computer vision to map terrain and use path planning algorithms to guide the car in the desired direction according to the official Stanford team website(“Stanford Racing :: Home,” n.d.) and *Using CART to segment road images*(Davies and Lienhart, 2006).



­

*Diagram from paper mentioned above.*

The paper provided an unbiased review of the technology used including both advantages and disadvantages, and even included links to the actual source code used. However, the method used is way too complicated for this project.

The winner of the DARPA urban challenge a few years later which focused on autonomous vehicles in an urban setting was from Carnegie Mellon University, the team used a preprogramed multi-level decision tree to respond to the terrain mapped by various sensors with little machine learning involved according to their website (“Tartan Racing @ Carnegie Mellon,” n.d.) and *A multi-modal approach to the DARPA Urban Challenge* (Urmson et al., 2007).

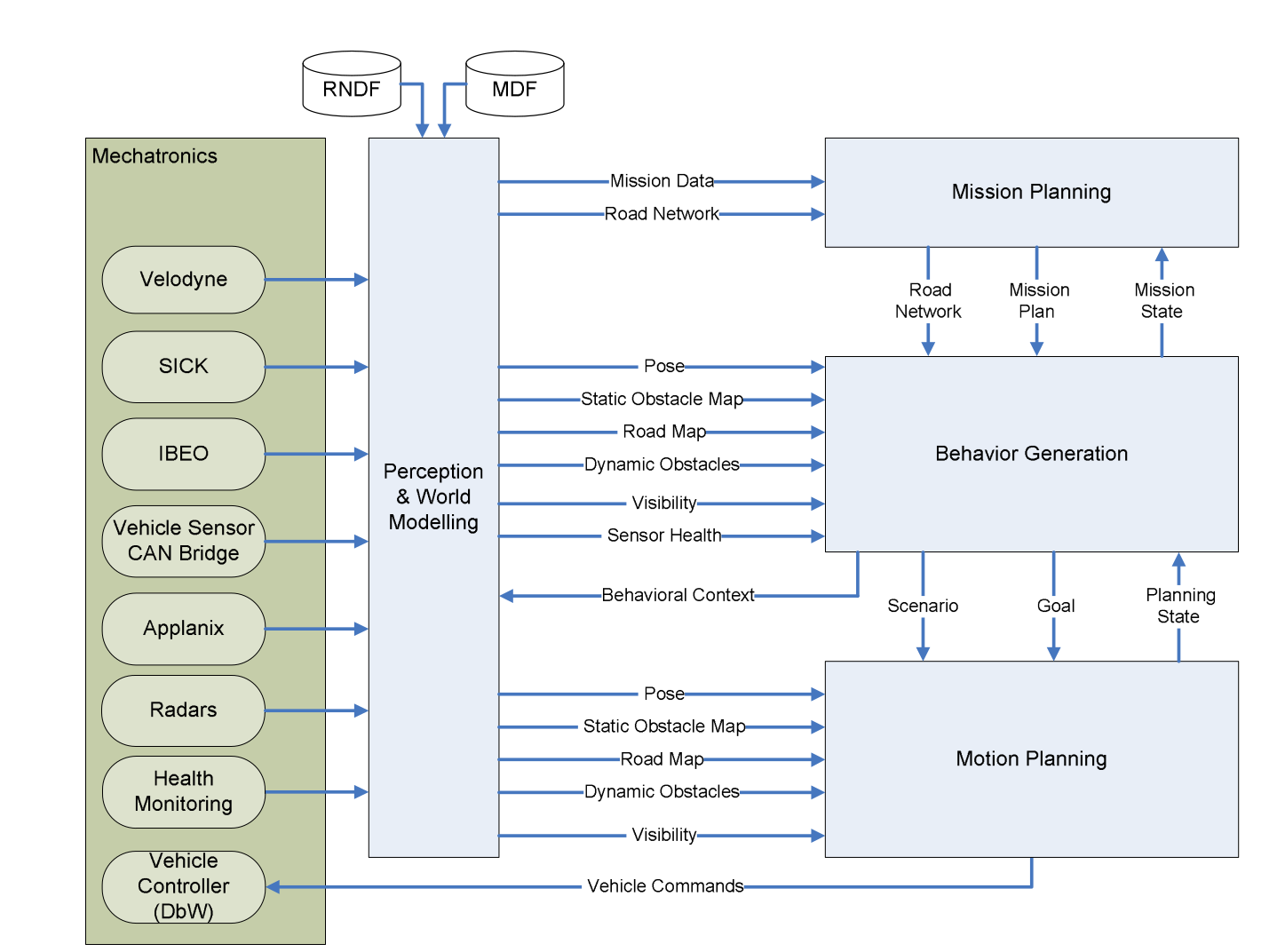
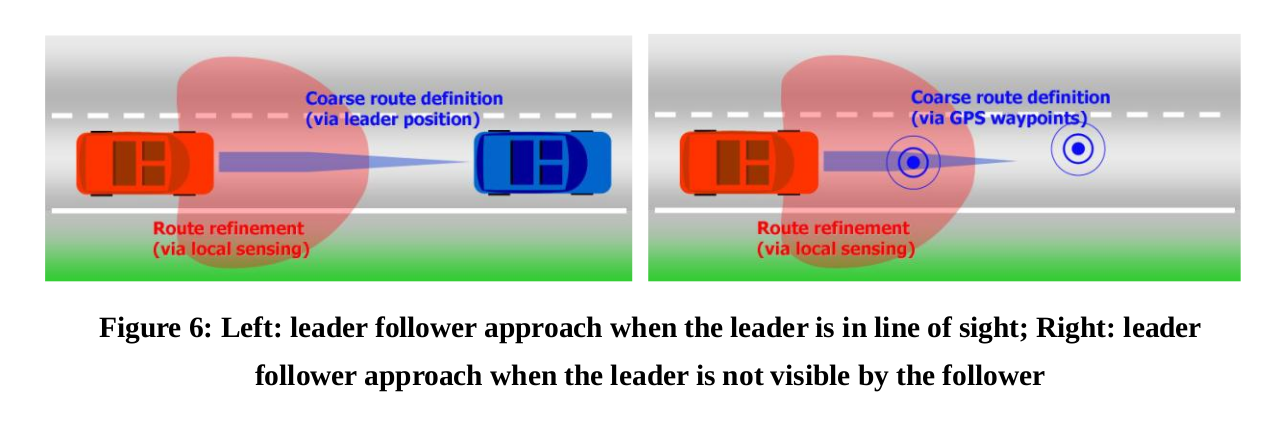


Diagram from paper mentioned above.

The source again was too advanced to be used in this project.

In 2010, Vislab from the University of Parma attempted a 13000 km drive from italy to China . According to this article at spectrum.ieee.org(Guizzo, 2010), the team used one van to navigate and map the landscape with human intervention, and another van that followed the first. The paper *The VisLab Intercontinental Autonomous Challenge: an extensive test for a platoon of intelligent vehicles* (Bertozzi et al., n.d.) and *The VisLab Intercontinental Autonomous Challenge: 13,000 km, 3 months,… no driver* *(Broggi et al., 2011)* confirmed the idea.

Diagram from paper mentioned above.

Although method used is indeed suitable for unmapped roads, it did not fit with the goal of this project.

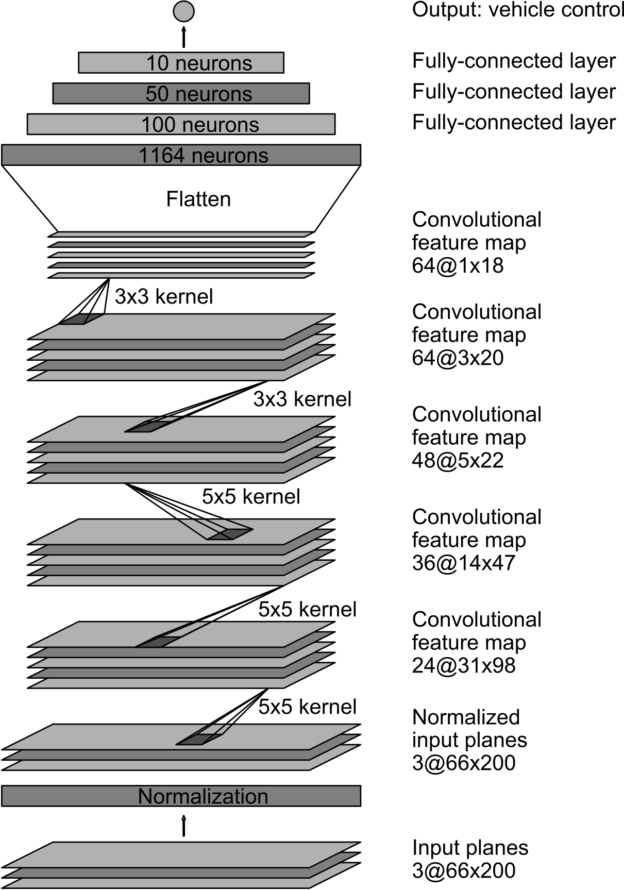
Stanford’s dynamic design labs has been participating in the development of autonomous vihecles, and in 2012, they have developed a car is designed to run at speeds of 100+ miles per hour. The specifics of the technology was not able to be found with reasonable effort, however, judging by the research done at stanford including *(*“Safe driving envelopes for path tracking in autonomous vehicles | Dynamic Design Lab,” n.d.)and (“Motion Planning for Autonomous Vehicles | Dynamic Design Lab,” n.d.), the team uses sensor fusion path tracking to plan the path of the car algorithmically.

Carnecie Mellon is also an active participant in the field with a fleet of testing vehicles running software that does object recognition and trajectory planning modified from the source used in the DARPA challenges. This system does actually try to mimic human behaviour when driving according to *Towards Fully Autonomous Driving: Systems and Algorithms* *(Levinson et al., 2011)* , however it differs from the method investigated in this project in which the machine learning process happens end to end.

## Current Technology

It is also worth noting that companies such as Tesla and Uber are actively trying to develop fully autonomous vehicles, with the most advanced by far being Google’s Waymo(formerly the Google Self-drivng Car Project). In May 2018, Waymo began an a public trial of it’s driverless car sharing program which is fully autonomous(“Early Rider Program,” n.d.) having driven 13,000,000 km already. It is hard to find information on the exact source code used by companies such as Waymo, however according to videos released by waymo(Waymo, n.d.), a self an self-driving car engineer on Quora(“What are some of the technologies used in the Google self-driving car? - Quora,” n.d.) and a TED talk by one of the Google Self-driving car members(Urmson et al., 2015), they use similar techniques such as object recognition and path planning mentioned above, the difference is that Google has a more servers optimized for machine learning, therefore can process data faster and produce a better result.

The computer hardware manufacturer Nvidia is also interested in autonomous driving technology and is a supplier for the uber team. They have investigated the possibility of using a convolutional neural network to read images from a camera and learn from a human the reaction to take with reasonable success, a research blog by Nvidia details the technology used and their development process (“End-to-End Deep Learning for Self-Driving Cars,” 2016), as the article goes in depth into the how the network comes together and the logic in each step, and that the method described is exactly the method this project aims to evaluate, this source is very valuable. The exact network structure used is show below:



### Practical Information

The open source code provided by Udacity’s nanodegree(*CarND-Behavioral-Cloning-P3*, 2018) is very valuable to the development of this project as it provided an easy way to get training data and test algorithms without resorting to using unrealistic games like Grand Theft Auto, or building a physical car and environment before knowing the viability of it all; however I did not enroll in the actual program(“Self Driving Car Engineer Nanodegree | Udacity,” 2018) as it was considered costly both in terms of time, and finance.

Stanford’s online course cs231n (“CS231n,” 2018) is another vital resource for this project as it helped familiarise me with basic python commands, and explained in detail how a convolutional neural network works at a level that is easily understood, and pointed me in the right direction for about similar topics.

Google’s machine learning crash course(“Machine Learning Crash Course | Google Developers,” 2018) was the resource I settled on after bouncing between courses on Udacity and several other sites. I find this source particularlly useful as it is very beginner friendly, and explains machine learning concepts that are hard to understand with careful descriptions and interactive diagrams. It also assumes zero prior knowledge of the topic so iteractive sessions actually help me familiarise with python’s datastructure, which is necessary knowledge for building a machine learning project as to maximise the efficiency of the machine.

The guide from Tensorflow on building an iris classification program(“Premade Estimators,” n.d.) was helpful as it provided a clear run down of the steps used to build a machine learning model using Tensorflow. It also cleared my confusion of several terminology such as “label” and “feature” which is necessary for understanding some of the sources listed above. The list at the Machine Learning Glossary(“Machine Learning Glossary | Google Developers,” 2018) is helpful in the same way as it cleared my confusion on terms such as “supervised” and “unsupervised”.

The book *Neural Networks and Deep Learning**(Nielsen, 2015)* provided a great source of knowledge for understanding the underlying mechanisms of machine learning, as well as explaining what the parameters in the code means. It is also more theoretical than the other sources listed therefore more general and versatile. This gave me the ability to understand the working of the wole system better, to judge what settings to use in my own implementation of the program, and gave me the ability to tweak theses same parameters later if my code does not work the first time it compiles. For example, it explained the concept of backpropagation better than the courses above could.

The blog post *An augmentation based deep neural network approach to learn human driving behavior* *(Yadav, 2016)* provides a detailed description of how to work with the specific task of teaching a car to lane-keep. It provides diagrams and suggestions on the tecniques used to generate training images including shifting, reflecting, and randomising the images as well as changing the brightness of the image, and gave a model of what the author’s best attempt at such an neural network. It also suggest future investigations to be done after this task.

The github repository of user subodh-malgonde(Malgonde, 2018) provides an example of developers attempting to do a similar project as this one. The author documented some of the techniques used, many of which is similar to the ones mentioned in the previous blog post, and an example of finished code which may come in useful when this project comes to the testing phase for evaluation. However, the code itself is less well commented and documented therefore harder to understand.

The github repository of user naokishibuya(Shibuya, 2018) also shows another developer attempting to use a convolutional neural network to train a car to drive around a track. This source is better than the previous ones in my opinion as it not only provides detailed description of the techniques used to generate additional training images, but a detailed explanation of the model used, as well as how each parameter is tweaked and why he tweaked them. These information may come in useful when doing the investigation later as it is a great source of knowledge for how to debug my own code. This source also provides an example of a trained model so it could potentially be used to evalute my own code later in the process. The exact network used by Naokishibuya is shown below:

| **Layer (type)** | **Output Shape** | **Params** | **Connected to** |
| --- | --- | --- | --- |
| lambda\_1 (Lambda) | (None, 66, 200, 3) | 0 | lambda\_input\_1 |
| convolution2d\_1 (Convolution2D) | (None, 31, 98, 24) | 1824 | lambda\_1 |
| convolution2d\_2 (Convolution2D) | (None, 14, 47, 36) | 21636 | convolution2d\_1 |
| convolution2d\_3 (Convolution2D) | (None, 5, 22, 48) | 43248 | convolution2d\_2 |
| convolution2d\_4 (Convolution2D) | (None, 3, 20, 64) | 27712 | convolution2d\_3 |
| convolution2d\_5 (Convolution2D) | (None, 1, 18, 64) | 36928 | convolution2d\_4 |
| dropout\_1 (Dropout) | (None, 1, 18, 64) | 0 | convolution2d\_5 |
| flatten\_1 (Flatten) | (None, 1152) | 0 | dropout\_1 |
| dense\_1 (Dense) | (None, 100) | 115300 | flatten\_1 |
| dense\_2 (Dense) | (None, 50) | 5050 | dense\_1 |
| dense\_3 (Dense) | (None, 10) | 510 | dense\_2 |
| dense\_4 (Dense) | (None, 1) | 11 | dense\_3 |
|  | **Total params** | 252219 |  |

*Table from source mentioned above.*

I chose to use Tensorflow as the library to provide the codebase necessary for machine learning as it is well documented, open source, and has a community of developers working on similar projects. *Build a Convolutional Neural Network using Estimators* (“Build a Convolutional Neural Network using Estimators | TensorFlow,” 2018) is a great source of well explained and well documented code on how to use Tensorflow for Convolutional neural networks such as this one. The code in the examples are also adaptable so may come in handy when I build my own network. However, this example is a classification problem so it may be different from when this project needs to solve which is a regression problem.

**Bibliography**:

Bertozzi, M., Bombini, L., Broggi, A., Buzzoni, M., Cardarelli, E., Cattani, S., Cerri, P., Debattisti, S., Fedriga, R.I., Felisa, M., Gatti, L., Giacomazzo, A., Grisleri, P., Laghi, M.C., Mazzei, L., Medici, P., Panciroli, M., Porta, P.P., Zani, P., n.d. The VisLab Intercontinental Autonomous Challenge: 13,000 km, 3 months,… no driver 11.

Broggi, A., Cerri, P., Felisa, M., Laghi, M.C., Mazzei, L., Porta, P.P., 2011. The VisLab Intercontinental Autonomous Challenge: an Extensive Test for a Platoon of Intelligent Vehicles,” Intl. Journal of Vehicle Autonomous Systems, special issue for 10 th Anniversary.

Build a Convolutional Neural Network using Estimators  |  TensorFlow [WWW Document], 2018. URL https://www.tensorflow.org/tutorials/estimators/cnn (accessed 8.20.18).

CarND-Behavioral-Cloning-P3: Starting files for the Udacity CarND Behavioral Cloning Project, 2018. . Udacity.

CS231n Convolutional Neural Networks for Visual Recognition [WWW Document], 2018. URL http://cs231n.github.io/convolutional-networks/ (accessed 8.19.18).

Davies, B., Lienhart, R., 2006. Using CART to segment road images, in: Chang, E.Y., Hanjalic, A., Sebe, N. (Eds.), . Presented at the Electronic Imaging 2006, San Jose, CA, p. 60730U. https://doi.org/10.1117/12.650164

Early Rider Program [WWW Document], n.d. . Waymo. URL https://waymo.com/apply/ (accessed 8.27.18).

End-to-End Deep Learning for Self-Driving Cars [WWW Document], 2016. . NVIDIA Dev. Blog. URL https://devblogs.nvidia.com/deep-learning-self-driving-cars/ (accessed 8.19.18).

Guizzo, E., 2010. Autonomous Vehicle Driving from Italy to China [WWW Document]. IEEE Spectr. Technol. Eng. Sci. News. URL https://spectrum.ieee.org/automaton/robotics/robotics-software/autonomous-vehicle-driving-from-italy-to-china (accessed 8.26.18).

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., Thrun, S., 2011. Towards fully autonomous driving: Systems and algorithms, in: 2011 IEEE Intelligent Vehicles Symposium (IV). Presented at the 2011 IEEE Intelligent Vehicles Symposium (IV), IEEE, Baden-Baden, Germany, pp. 163–168. https://doi.org/10.1109/IVS.2011.5940562

Machine Learning Crash Course  |  Google Developers [WWW Document], 2018. URL https://developers.google.com/machine-learning/crash-course/ (accessed 8.20.18).

Malgonde, S., 2018. A car trained by a deep neural network to drive itself in a video game: subodh-malgonde/behavioral-cloning.

Motion Planning for Autonomous Vehicles | Dynamic Design Lab [WWW Document], n.d. URL https://ddl.stanford.edu/publication-research-theme/motion-planning-autonomous-vehicles (accessed 8.27.18).

Nielsen, M.A., 2015. Neural Networks and Deep Learning.

Premade Estimators [WWW Document], n.d. . TensorFlow. URL https://www.tensorflow.org/guide/premade\_estimators (accessed 8.27.18).

Safe driving envelopes for path tracking in autonomous vehicles | Dynamic Design Lab [WWW Document], n.d. URL https://ddl.stanford.edu/publications/safe-driving-envelopes-path-tracking-autonomous-vehicles (accessed 8.27.18).

Shibuya, N., 2018. Built and trained a convolutional network for end-to-end driving in a simulator using Tensorflow and Keras: naokishibuya/car-behavioral-cloning.

Stanford Racing :: Home [WWW Document], n.d. URL https://cs.stanford.edu/group/roadrunner//old/technology.html (accessed 8.26.18).

Tartan Racing @ Carnegie Mellon [WWW Document], n.d. URL http://www.tartanracing.org/tech.html (accessed 8.26.18).

Udacity, n.d. Self-Driving Car Nanodegree Program Overview.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Dolan, J., Duggins, D., Ferguson, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T., Kelly, A., Kohanbash, D., Likhachev, M., Miller, N., Peterson, K., Taylor, M., 2007. Tartan Racing: A multi-modal approach to the DARPA Urban Challenge.

Waymo, n.d. Waymo 360° Experience: A Fully Self-Driving Journey.

What are some of the technologies used in the Google self-driving car? - Quora [WWW Document], n.d. URL https://www.quora.com/What-are-some-of-the-technologies-used-in-the-Google-self-driving-car# (accessed 8.27.18).

Yadav, V., 2016. An augmentation based deep neural network approach to learn human driving behavior [WWW Document]. Chatbots Life. URL https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9#.d779iwp28 (accessed 8.27.18).s