[redacted] End to End Self-Driving Car Simulation with Convolutional Neural Network

[redacted] [redacted] [redacted]
[redacted] [redacted] [redacted]

Abstract

In this project, we trained a CNN model to implement a self-driving car in a simulator. The self-driving car in the simulator can drive itself with three input cameras: left camera, center camera, and right camera. These images from these three cameras are inputs of our CNN model and the outputs are the steering angles. The steering angles are the main factor to direct the car and keep it on the right course of the road. The simulator we used is Udacity self-driving car simulator. We can drive the car in two modes: manually and autonomous. We use both the online dataset and data generated locally in the training mode. Then we perform a transformation on the images captured while driving to achieve better performance. The result of our project is satisfying and the self-driving car is able to drive on the correct course.

1. Introduction

The self-driving car is the research focus of both academic researchers and industries. There are huge benefits of self-driving technology. By Urmson, Over 1.2 million people are killed by traffic accidents every year in the world [6]. Among all these accidents, 90% of the accidents are caused by the wrong actions made by the drivers. For example, 30% of the accidents are caused by drunk driving and 10% are caused by distraction for the drivers. Self-driving vehicles have the potential of reducing accidents and they can provide safety for traffic. This technology also makes driving for disabled people possible.

Self-driving cars are vehicles that have the ability to drive under different road conditions and follow different road markings. Self-driving cars should be able to drive without or with less human interventions. Limit to equipment restrictions, we decided to use a simulator to simulate a self-driving car. This simulated self-driving car has cameras as same as a real self-driving car. We developed a Self-Driving Car (SDC) in the simulator Udacity self-driving car simulator with a CNN model. The SDC is able to drive along the main course of the road. It can successfully drive on different sections of a simulated road with different backgrounds, road markings, light conditions, and road conditions.

The input image is gathered by three built-in cameras: left, center, and right camera. The images collected are our raw dataset. The dimension of the raw data is $3 \times 160 \times 320$. The image augmentation is performed to achieve better model performance. We first cut the top and bottom part of the image and flip the new image along the vertical axis. The final dimension of the input data is $3 \times 70 \times 320$. To create the training dataset, we manually drive the car in the simulator and collected the driving images. These images are used to train our CNN model.

CNN models perform a key role in the SDC. The reason is that SDC only has images as its inputs. CNN models have better performance on this kind of input because they can reduce the number of connections compared to fully connected layers and it loses as little information as possible. Especially in the driving image, the difference between images with a short time elapsed is relatively small, we do not need every pixel to determine the next direction. As the results prove, CNN is a well-performed model for our task. We used a CNN model proposed by Mariusz Bojarski, which has 1 normalization layer, 5 convolution layer, 3 fully connected layers, and 1 drop-out layer.

After we trained the model, we implement this model to our self-driving car and observe its performance manually, and execute human intervention while necessary. To evaluate the general performance of the SDC, we set up a road section to have our SDC self-driven. We count the time that the SDC need human intervention (THD), compared to the time that SDC self-driven (TSD), we use this ratio to evaluate the general performance of the SDC:

$$\text{Human drive rate} = \frac{\text{THD}}{\text{TSD}}$$

We have a human drive rate 0.709% as our final result.

2. Related Work

Self-driving cars have drawn great attention since the day they are proposed. The first contest for self-driving cars started at the year of 2004 where self-driving vehicles need to drive 150 miles in the desert. However, none of the vehicles made it in the game. In the next year, 5 out of 23 vehicles successfully finished the contest [6]. In the year of 2007, the DARPA contest for self-driving cars was held. In this contest, the vehicles need to drive in a simulated urban environment and 6 cars made it in the contest. Besides all these competitions, many companies joined the wave of self-driving cars. The company Waymo, whose parent company is Google Inc, experimented on the road for more than 1 million miles and achieved zero accident. More companies like Tesla, Baidu, TuSimple and traditional car companies are also researching self-driving cars.

The level of self-driving cars can be decided by how they depend on human manipulation. In SAE rating, the level of self-driving cars ranges from 0 to 5. In level 0, the vehicles do not have any abilities to drive by themselves. In level 1, the cars have a basic driving assistant in either the steering wheel or brake assistant, the rest actions are done by humans. In level 2, the cars need to have more multiple assistant functions than level 1. In level 3, the vehicle is able to perform almost all the actions and the human drivers only need to watch the road in case emergencies happen. In level 4, the vehicle can perform all the driving actions and human drivers do not need to focus on the road, but this is restricted for some certain road and environmental conditions. In level 5, the vehicles have identical functions, but they are able to work on all the roads and environments. The table1 lists different key characteristic of different levels of self-driving cars.

By M. Daily, Self-driving technology has made great progress in the past years [2]. The development of Self-driving technology benefits from the development of deep learning and sensor technology. New advanced sensors are created and updated over the days. The fast development of sensor technology provides more and more data. NVIDIA proposed an advanced data collection system: DAVE 2 System which can collect time-stamped video and also collect the steering angle data [1]. Image-based self-driving technology is widely used. Recently, the rise of convolutional neural networks (Convolutional Neural Network, CNN) has greatly improved the ability to detect objects. However, there are also many challenges for autonomous vehicles. They are facing uncertainty in the rural areas, as well as the complicated traffic condition and road markings[6].

At first, CNN simply integrated into the sliding window approach. However, this does not solve the precise positioning of the object in the picture. Therefore, Girshick et al. (2014) proposed R CNN (Region-based CNN) to solve the problem of positioning in object recognition [3]. They use selective search to generate many candidate regions, then use CNN to extract feature vectors from each region, and use linear support vector machines to classify candidate regions that contain entities. While RCNN solves the positioning problem, the time complexity of the model itself is too high, resulting in a series of efforts (ResNet, Fast R-CNN, Faster R-CNN) to optimize the model, resulting in an end-to-end framework. The original CNN could only work with fixed-size images, not candidate bounding boxes of different sizes.

However, the effect of the above methods in practice is not significant, mainly due to the variable physical scale in the data set, as well as the extent of being blocked or truncated. And the above method is only improved on the detection network, and the selective search used to filter the candidate boxes does not apply to this scenario, so Ren et al. (2015) proposed RPN (region proposal networks) instead of selective search [5]. RPN is a distortion of CNN that predicts the type and probability of entities contained in each candidate box in the form of a sliding window on the convolutional feature of the image. The RPN, combined with Fast RCNN, enables end-to-end learning.

There are also other perceptions of self-driving vehicles. From a public perspective, autonomous vehicles can solve the problems of traffic problems and change the way of transportation. People are attracted to the benefits of the safety and the convenience of self-driving vehicles, and the most adopted form of self-driving cars are taxis [4].

3. Proposed Method

3.1. Deep Learning Appoach

Machine learning is widely used in autonomous driving, focusing on self-driving vehicle awareness of the environment and decision-making. The application of machine learning in environment awareness falls under the category of supervised learning, such as object detection of images in a camera, which requires a large number of images labeled as solids as training data for deep learning methods to identify objects from new images. The use of machine learning in behavioral decision-making generally belongs to the category of intensive learning, intelligent body needs to interact with the environment, every step of the intelligent body behavior will affect the environment, and environmental changes will also affect the behavior of the intelligent body. Intensive learning is learning the mapping relationship between environment and behavior from a large amount of sample data that interacts with the environment so that the intelligent body can 'intelligently' behave every time it perceives the environment.

The collection and processing of environmental information is the basis and premise of no one driving independently. The perception of the environment of unmanned vehicles is mainly based on various sensor technologies, the sensors used generally have cameras, millimeter-wave radar, lidar and so on. For the safe and accurate perception

SAE	Definition	Who drivers	Take actions	Environments
L0	Human driving	human	human	none
L1	Assistant driving	human and vehicle	human	certain roads
L2	Partial self driving	vehicle	human	certain roads
L3	Conditional self driving	vehicle	human	certain roads
L5	Fully self driving	vehicle	vehicle	all conditions

Table 1. Comparison from L0 to L5

of the environment, self-driving vehicles work together with multiple sensors, with millimeter-wave radar and lidar primarily responsible for medium- and long-range ranging and environmental awareness, while cameras are primarily used for the identification of traffic signals and other objects.

The main research direction in the field of traditional computer vision is the visual problems of visible-light cameras. In self-driving, real-time and accurate detection of surrounding objects is critical. At the same time, object detection is also one of the research focus of computer vision.

3.2. Convolutional Neural Network

A Convolutional Neural Network (CNN) model that we trained leads the execution of our self-driving car. In CNN, we use kernel as feature detectors to create multiple feature map. In our CNN model, the kernel size of 5 and 3 are used in our project. The kernel is used to compute the output channel:

$$\sum_{j=1}^{25} w_j^{(@k)} x_j \text{(For kernel size 5)}$$

And the output channel size is:

$$O = \frac{W - K + 2 * P}{S} + 1$$

The convolution of the CNN model is actually a kind a 2-D convolution. The kernel can move from x and y axis - it is moving vertically and horizontally. But it cannot move across the different channel. The CNN models consists of convolution, activation, and pooling layer. Although pooling layer is not used in the CNN model that we developed in this project, we found convolution layers and fully connection layers are enough for the purpose of our project. The computation of the convolution is:

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v] A[i-u, j-v]$$

Our input image is RGB image, which consists of 3 color channels: red, green and blue. These 3 channels are the map volume of our input.

3.3. Model Architecture

We implement the model similar to the model proposed by Nvidia as shown in figure 1. The first layer of the network normalizes the input image (the structure is a 66*200 matrix data structure). According to the introduction of the paper, the normalization layer is hard-coded and does not need to be changed during the learning process. The hard-coded implementation can ensure sufficient computing efficiency. Normalizing the entry data of the network model can make the normalization process adjust according to the network structure, and can facilitate the GPU to accelerate the processing process.

Following this is the convolutional layer. The convolutional layer is used to extract the features of the input data (graphic files/frames at a certain speed provided by the camera). It is empirically selected based on the results of a series of parameter experiments. It can be seen that there are a total of five layers of convolutional layers with convolution function. Although the parameters of each layer are different (mainly different convolution kernels), the ultimate goal is only one, that is, to find out from the complex visual image The most sensitive part of the computer, and greatly reduce the amount of image data.

The feature image processed by the first two layers of convolution clearly outlines the boundary of the road. The first layer of convolution calculation output on the lower left and the second layer of convolution calculation output on the right. The most important road boundary information is not lost, but The information of the overall picture has been simplified a lot

This shows that the convolutional neural network has automatically learned to recognize useful road features mainly boundary features. But it should be noted that the paper repeatedly emphasized that the designer of this end-to-end implementation has never trained this end-to-end system to recognize road shapes, that is, no road sample set is provided for special CNN training for image recognition. , Otherwise the system does not belong to the end-to-end autonomous driving system, and a break is formed in the first link of image recognition.

The true end-to-end system training method is only related to the final output of the manual driver's operation. If we simplify the problem to a typical "lane keeping function", the training sample is the steering angle of the human driver, which is directly linked to the image signal captured by the camera for targeted training.

Therefore, after the five convolutional layers, the system docks three fully connected layers (relatively black box), and finally outputs a control number, that is, the turning angle (mathematically processed as the inverse of the turning radius). The design intent of the fully connected layer here is to exist as a mathematical realization of the direction controller.

The data source of the entire end-to-end system is only the camera (the left, middle and right position is to capture the complete forward road information) and the direction operation of the human driver. After the CNN deep neural network processes the visual information, it will give a steering wheel angle based on the previous learning experience through the fully connected layer, and at this moment, the real human operation will also be introduced and compared with the output of CNN. The result of the comparison is sent back to the CNN network to correct the fully connected layer parameters in the black box state. The specific number of parameters is not clear, but this so-called back propagation weight re-adjustment strategy is used in many CNN implementations. Proven effective.

In this framework, as long as sufficient training data is provided, that is, human drivers drive a vehicle with a camera to drive a large amount of mileage, plus the limit road state of the artificially created system-various working conditions that deviate from the road line, CNN Will be fully trained and become strong enough.

4. Experiments

4.1. Description of Dataset

The dataset used to train the model consists two parts:

- images captured by the left, center and right camera
- state of the car, including steering angle, throttle, reverse and speed

The dimension of the images is 160 * 320 in RGB. Only the steering angle is used in the training process as the output of the network.

4.2. Transformation of Data

Before feeding images into the model, the top and bottom of the image are removed, so the the dimension is reduced to 70 * 320. Half of the images and corresponding steering angles are flipped to increase robustness of the model.

Due to the nature of the simulator, the steering angle, unlike in reality, is not continuous, so a smoothing function is applied to the collected steering angle before training.

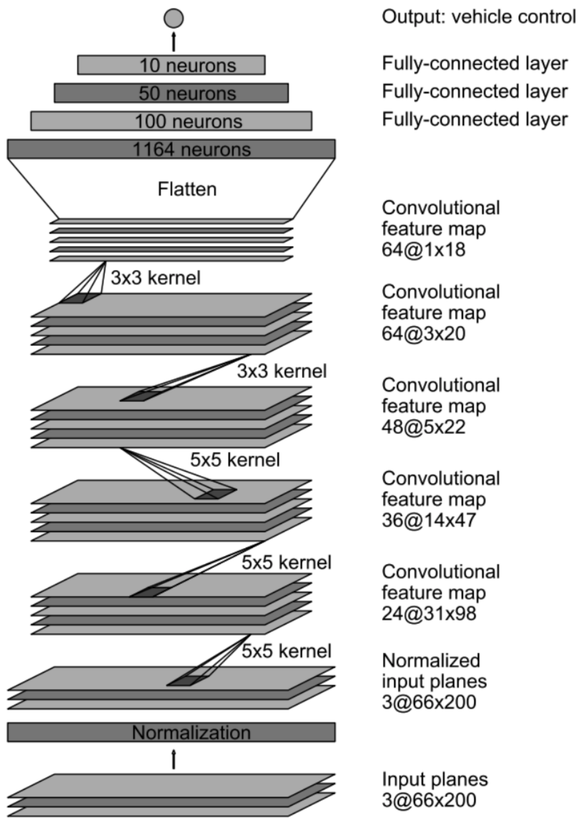

Figure 1. the architecture of CNN model proposed by Nvidia

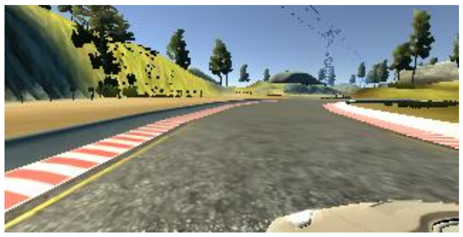

Figure 2. View of Left Camera

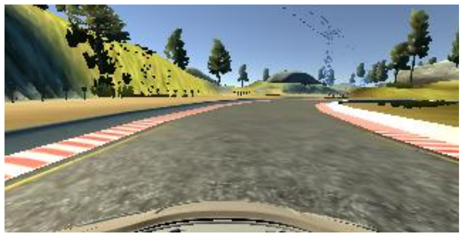

Figure 3. View of Center Camera

Images captured by the left and right cameras are trained in the same model as the central camera, so at the same position, images from different cameras will yield a different steering angle. To correct this effect, an offset is added to the steering angle before training the model.

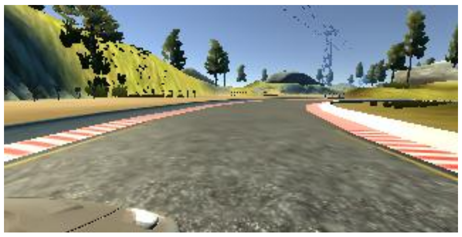
Figure 4. View of Right Camera

4.3. Software

Apart from regular editors, software used are:

- Python 3.9
- PyTorch 1.8.1
- Udacity Self-driving Car Simulator

4.4. Hardware

Training and simulation are both run by CPU on a local machine. Due to the large number of image files, Google Colaboratory has an IO speed significantly lower than a local machine with SSD when reading files from Google Drive.

The specifications of the PC for training are as follows:

- CPU: Intel Core i7-8809G
- RAM: 32GB
- OS: Windows 10 20H2

5. Results and Discussion

5.1. Simulation Results

Since the models can only be evaluated manually in the simulator, models trained with different hyper-parameters, datasets and epochs are mounted to the simulator to drive the car autonomously.

The test road is ring-shaped, so we only complete the whole ring once. All the models trained are not able to complete the entire route without any manual correction. In the final models proposed, most models require less than three times of short manual course correction.

Particularly, in the demo video, three interventions were conducted, which has a final drive rate of

$$\text{Human drive rate} = \frac{\text{THD}}{\text{TSD}} = \frac{2s}{282s} = 0.709\%$$

These actions are taken including one to move the car to the center of the road and the other one preventing the car from running off the road.

In general, the SDC model works well and drives the car in a normal lane of the road when the road is straight or only has a moderate turn. It can also identify bridges and intersections. The performance degrades when the turn is sharp or when an exit occurs. In these cases, the model will drive the car too close to the curbs.

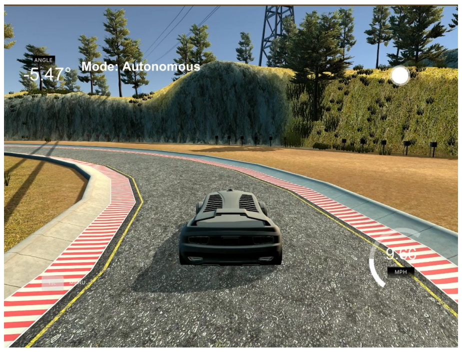

Figure 5. Normal Turn

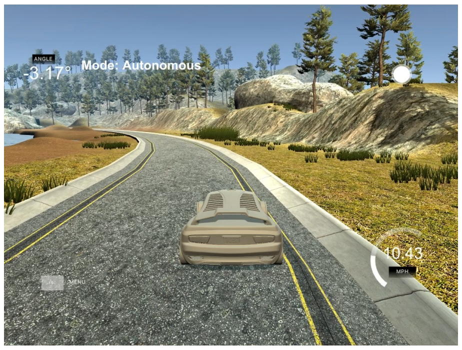

Figure 6. Close to Curb

5.2. Training Process

The training process is more interesting than the actual results, several unexpected phenomena are observed and finally solved with different methods.

5.2.1 Over-fitting

Initially, the number of epochs set to train the model is 50. However, in real practice, the model is quite efficient in capturing the features and converges at a tremendous speed. On average, after 4 epochs the model is proficient to run without much intervention.

On the contrary, since the loss function we used is MSE loss, which is not maximizing the time the vehicle drives on the road, the more epochs the model is trained, the more noise or weird driving behavior it learns. Even though the MSE loss is decreasing with each minibatch, the car may finally learn to run out of the road at the very beginning of the simulation.

To prevent this problem, based on the model proposed in the paper, we added a dropout layer to overcome the over-fitting problem. In addition, at most 20 epochs will be trained to both eliminate over-fitting and reduce training time, and each model trained after the 4th epoch will be accessed manually in the simulation to find the one with the best performance.

5.2.2 Steering Angle Smoothing

The design of the simulator training mode has some deficits. It has three input methods: keyboard, mouse and game joystick, in which the keyboard does not yield a continuous steering angle. Each time the user hits the key on the keyboard, the steering angle goes directly to a large value like 14 degrees and when the key is released, it drops back to 0 promptly.

This has caused a serious problem in the training process when using a dataset generated with a keyboard. The frames do not vary too much in a long curved road, but the steering angle is not consistent, which has mostly zeros and several large values. This makes it tough for the model to learn from a dataset with such huge variance in the single output. Also, when the model is fully trained, the car in the simulation drives with the front wheels turning and shaking on a large scale constantly, which is not the driving experience that everyone is expecting.

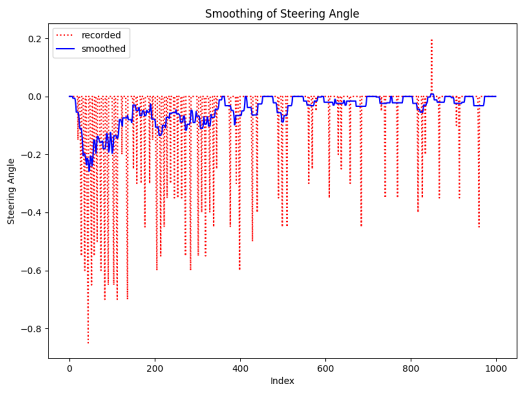

Figure 7. Smoothing of Steering Angle

Inspired by financial data processing techniques, this problem is solved by applying a smoothing function to the recorded steering angle before they are fed into the model. In this case, the smoothing window is quite large, approximately 15 frames. After smoothing, in a turn, the angles are not zeros but a small angle towards the right direction.

A second approach is to use a different dataset. After the first few models were trained, we created our own training dataset with an Xbox controller, which yielded a consistent and smooth steering angle right from the beginning. In this scenario, the smoothing window is narrowed down to just 5 frames.

5.2.3 Data Shuffling

In many cases shuffling training data is optional, but it has a huge impact in this particular case.

When the data is not shuffled, the model learns to drive on the beginning of the entire route pretty well in the first few minibatches. As is mentioned before, the model converges pretty fast, and it tends to only drive perfectly at the beginning, but when it comes to the rest of the road, it performs below expectation.

An example here is that at the beginning the curb is illuminated by the sun, and soon the model learns to drive along the white shiny curb. However, the entire route is a ring, on the other side of the ring, the curbs are actually in the shadows, and the model is no longer able to drive along. It also takes more time for the model to learn to recognize the new features of the road.

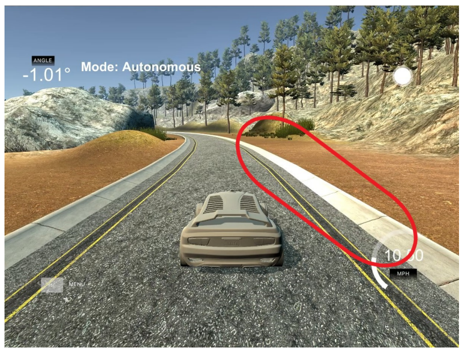

Figure 8. Illuminated Curbs

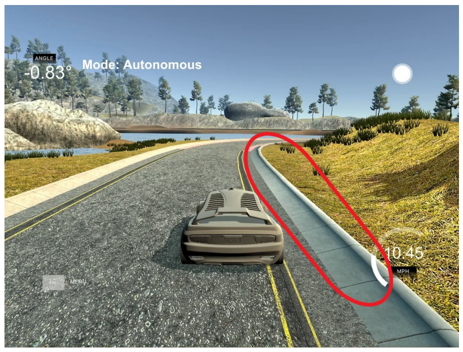

Figure 9. Curbs in Shadows

By shuffling the data, the model is not learning just from a certain part of the road. It is trained with the fragments of the entire route to prevent the problem.

5.2.4 Driving Behavior

Quite similar to reality, bad driving behavior in the data will teach the model to drive like a drunk person.

In some datasets used, the car drives on the lines all the time, which results in the model driving on the line and right next to the curb. In some other datasets, there is no indication on how to correct it self in a hazardous situation, but only perfect driving. As a result, the model never knows how to resume when hitting a curb.

The solution to this is to use a dataset with a focus on good driving behaviors and correction in dangerous situations.

6. Conclusions

In this project, we implemented a self-driving car in the Udacity self-driving simulator. We developed and tuned a CNN model that can process image input and output the steering angle. Different data transformation techniques were used to increase the robustness and accuracy of the model. We presented a possible solution to the self-driving technology. To deal with image input, we performed data augmentation to achieve better model performance and less computational cost.

Finally, our SDC can successfully deal with different road conditions, degree of shadow, and road marks and it can roll on the correct course. We put large amount of effort in the parameter tuning and problem solving as this is the most crucial part for our model to achieve good performance. We have proved that the possibility for CNN models to perform the tasks of self-driving. We have also proved that multi-angle cameras can increase the performance of the self-driving technology.

7. Acknowledgements

We really appreciate the help from Udacity for creating this open source driving simulator for everybody to use.

We would also like to express our gratitude to Kaggle and some participants for sharing their dataset and ideas in the Kaggle competition.

8. Contributions

[redacted]

- In charge of whole project architecture and coding software framework.
- Coded for the main files for model and training parts.
- Evaluated models trained with different hyper-parameters and epochs in the simulator.
- Collect different data resources and create dateset.
- Drafted *Experiments and Results* and *Discussion* of this report.

[redacted]

- Coded driving, data visualization and utility functions parts.
- Generated our own training dataset apart from the online dataset manually.
- Proposed and experimented new ways to transform the image dataset.
- Lead meetings and make presentation slides.
- Drafted *Abstract*, *Related Work* and *Proposed Model* of this report.

[redacted]

- Help model evaluation and result analysis.
- Debug the installment of the simulator.
- Proposed new approaches to solve the mentioned problems.
- Generated the videos for demonstration.
- Drafted *Introduction*, *Proposed Model* and other parts of this report.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] M. Daily, S. Medasani, R. Behringer, and M. Trivedi. Self-driving cars. *Computer*, 50(12):18–23, 2017.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[4] D. Howard and D. Dai. Public perceptions of self-driving cars: The case of berkeley, california. In *Transportation research board 93rd annual meeting*, volume 14, pages 1–16, 2014.

[5] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[6] C. Urmson and W. R. Whittaker. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2):66–68, 2008.

A. Appendix

A.1. Related Links and Resources

- Code of this Project: Github
- Udacity Self-Driving Simulator: GitHub
- Online Datasets: Download Link, Kaggle
- Self-Generated Datasets: Google Drive
- Demo Videos: Google Drive