Self Driving Car using Deep Learning

Sairam Chaganti
Masters in Computer Science
UMKC, Kansas-city
scmdt@umsystem.edu

Shravani Nalla
Masters in Computer Science
UMKC, Kansas-city
sny9m@umsystem.edu

Nikhila Chirumamilla
Masters in Computer Science
UMKC, Kansas-city
nccdx@umsystem.edu

Sravani Niharika Garapati
Masters in Computer Science
UMKC, Kansas-city
sgrgz@umsystem.edu

***Abstract*—An intriguing debate has been taking place in recent years about self-driving automobiles. In the race to develop self-driving cars, many well-known automakers, like Tesla, GM, and Nissan, are competing. Autonomous automobile technology is being developed by major technology firms including Google, Waymo, Baidu, and Aptiv. Radar, lidar, sonar, GPS, and odometry are only few of the technologies used in autonomous vehicles to recognize their surroundings. It's possible to automate navigation using sensor data fed into a control system. The CNN deep learning algorithm will be discussed in this article in order to recognize the local ecosystem in order to create the automated navigation needed by self-driving vehicles. In order to apply the learning results, an open simulation system will be created, and the data set will be learned. This simulation demonstrates a great degree of accuracy in the process of teaching an autonomous vehicle to navigate by examining its immediate surroundings.**

keywords: Self -driving, deep learning, CNN, Simulation, Accuracy.

I. INTRODUCTION

Many scientific fields, including as genetics, pharmacology, image segmentation, voice recognition, and natural language processing have turned to machine learning (ML) as a means of doing research. ML is also being used to make stock market forecasts.

One of the most significant advances in computer vision has been the development of convolutional neural networks and other deep architectures. For tasks such as image classification, image captioning, object recognition, or semantic segmentation, they easily surpassed earlier hand-crafted feature extraction-based systems and established a new standard of excellence. Feature extraction by hand, followed by a classifier, was the traditional approach to pattern recognition issues. The ability to automatically extract high-level features from training data is a significant benefit of CNNs for image recognition applications.

Additionally, by scanning a whole picture using convolution kernels, just a few parameters must be learnt in comparison with the total number of operations. During the last twenty years, CNNs with learnt features have shown their worth. However, in the last few of years, they've been integrated into genuine systems. It's basically because of two major advancements. In the first place, the success of parallel graphics processing units (GPUs) as a result of the reliability of CNNs learning processes. Mnist [1] and ILVRC [2] datasets, which may be utilized for both training and testing, are readily available.

An advanced neural network is described in this study. It learns the whole set of algorithms required to operate a car's steering wheel. In a simulated driving situation, this CNN was trained and evaluated using data. A front-facing center camera footage may be used to generate control orders for the car's steering once it has been trained.

II. RELATED WORKS

A mix of video cameras, radar, ultrasonic sensors, and lidar has been the usual concept for self-driving vehicles until now. Aptiv, Waymo, and other major developers of autonomous technology use this strategy. It has become more clear that the traditional approach may not be sufficient as many corporations perform trials in areas like California.

The main problem with autonomous vehicles is that they are currently unable to see well enough to safely operate in heavy traffic or see far enough ahead to manage highway conditions in any form of weather, regardless of the weather.

The brightness from the sun may fool video cameras. Radar can tell you the relative speed of things, but its perception is like that of Mr. Magoo. Because of the low acuity of ultrasonic sensors' sound waves, only adjacent objects can be detected. Despite its ability to produce 3-D pictures of people and street signs, Lidar (officially known as light detection and ranging) has a limited range and may be hindered by heavy weather. Because without perceptual data, even the most advanced artificial intelligence program cannot assist.

To train a self-driving vehicle system to the level of precision needed in a real-world driving experience, an end-to-end system would require an enormous quantity of data, which would be impossible to acquire in the time it would take to train the modular system. Sub-modules with diverse functions, such as pedestrian recognition or route planning, are divided down in a modular fashion. Depending on the results of the experiments, either machine learning or human engineering is used to train each module. Real-world self-driving vehicles and this paper's model have a number of key differences in terms of input.

Waymo and Tesla, for example, use a variety of sensors and cameras to construct 3D representation of the car's surroundings . A 360-degree representation of the environment requires expensive technology like as LIDAR [3], radar and ultrasonic sensors. As an example, the Tesla Model S makes use of a single back camera as well as broad and narrow front cameras that view within. Twelve ultrasonic sensors in the back, plus a RADAR in the front, allow the automobile to measure the speed of things in front of it. Additionally, these autos use SLAM algorithms that combine the data from all sensors into a single location.

III. METHODOLOGY

A. Data collection and Manipulation

The goal is to forecast the steering wheel movements of a self-driving automobile based on data from a camera in front of the vehicle. Image to angle mapping is done by our deep convolutional neural network. Figure below shows an example picture.

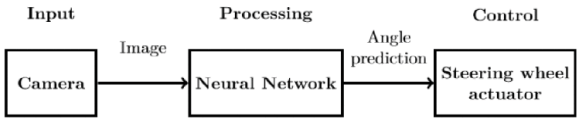

Fig. 1. System Driving Mode

Data collection mode and self-driving mode are two of the system's modes of operation. When the automobile is being driven manually, the data collection mode begins. To train the neural network, the camera records its pictures and sensors measurements. The camera inputs pictures straight into the neural network while our system is driving the automobile. The control signals are then predicted by it.

To begin gathering data, we must first alter the surrounding environment to our favor. On various courses with sharp and smooth curves, bumps, trees and traffic signals, the simulated cameras deliver views in front of the car. The background script gathers information on the driver's control of the steering wheel, throttle, and brake pedals.

The matching steering, throttle, and brake values are shown on each simulation picture. The photos captured by the camera are processed before they are sent to the network.

There are various steps before images from the camera are sent to the network:

- Selection procedure: data with desirable qualities is picked and separated into a training and a validation set, the training set consisting of 80% and the validation set consisting of 20% of the data.
- Changing the color scheme of the input photos may be accomplished via the use of the color scheme modification technique.
- The photos are then cropped to eliminate the front of the automobile, re-sized to 66x200, and converted from RGB to YUV since YUV takes into consideration human perception, enabling decreased bandwidth for chrominance components.
- Afterwards, the pictures are supplemented by adding simulated shifting and rotations to teach the network how to recover from a bad location or orientation. Using a normal distribution with a mean of 0, the magnitude of each of these disturbances is generated at random.
- The last step is to combine the pictures along with their related steering angle, throttle, and brake values and feed to neural networks as arrays.

B. Approach: Neural Networks

Neural Networks employ numerical vectors or matrices as input and conduct operations like additions and matrix multiplication, such as matrix multiplication to control the parameters. Each layer's activation function, the optimizer, as well as the loss function, have an impact on a neural network's performance.

In order to provide output for the next layer, activation functions require weights, biases, and a specified input. The findings are shown on the output layer, which is the last stage of the pipeline. For each input, the difference between the calculated and intended output is specified by a loss function. The optimizer in a neural network is used to minimize the discrepancy between predicted and actual results. For training, an optimization strategy starts with a random set of weights. Optimizer, activation functions, and loss functions are all available. Additional layer types and connections between layers are needed to create an array of neural and deep network architectures [6].

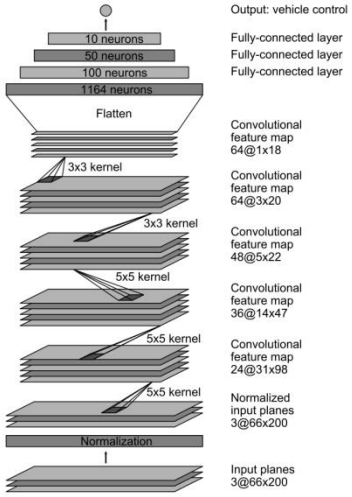

Fig. 2. CNN architecture

C. Architecture

Feed forward neural networks is NN architectural type we employed in our tests. We utilize networks with convolutional layers as the initial layers since we are dealing with images. First, images with dimensions of 66x200x3 are used as input, followed by five convolution layers with 24, 36, 48, 64, 64 neurons in each, each with a filter size of 5x5 and a 2x2 pixel jump at each filter pass (strides of sizes 2x2), and two final layers with filter sizes of 3x3 but no strides. There are several advantages to increasing the number of neurons in each layer of the neural network. Due to its quicker convergence to zero and more accurate output, all convolution layers utilize an activation function called elu [4]. Once the convolutional layers have produced their multi-dimensional output, they are followed by a flattening layer. There are 5

layers in total, each with 100 neurons and all elu activation except the output layer, which is linear since this is a regression issue. All of these levels are stored as models of type Sequential, and the output layer is saved as a linear model because this is a regression problem.

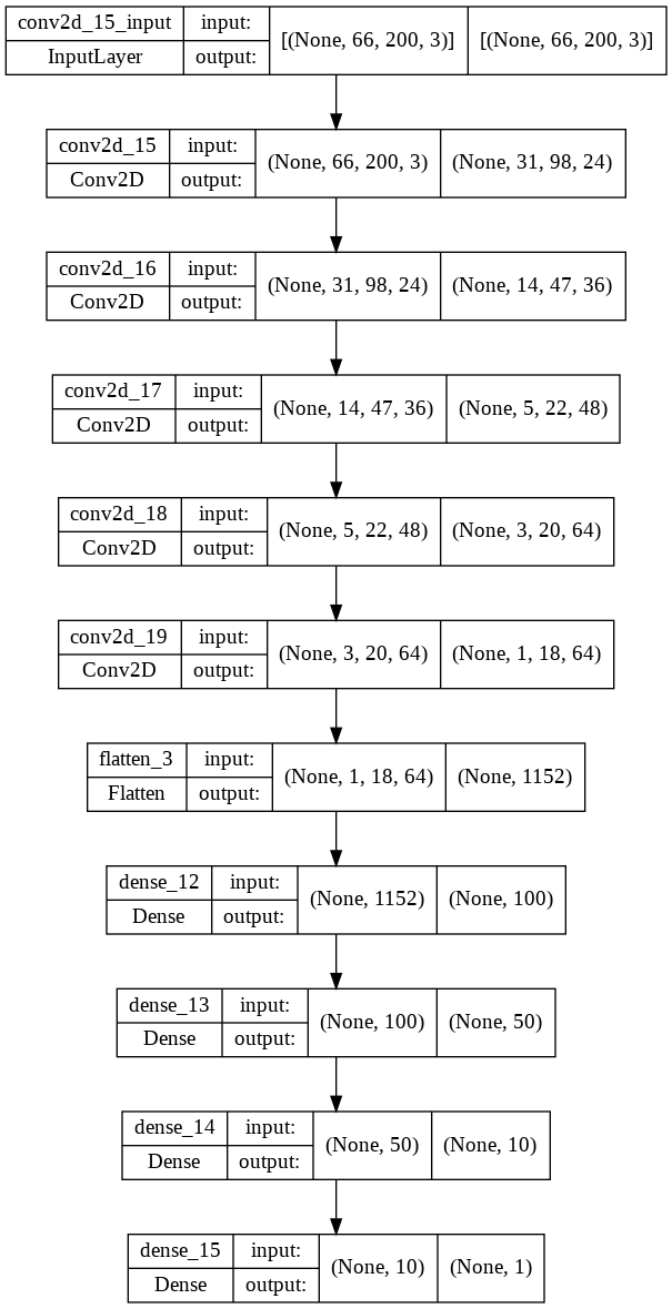

Fig. 3. Nvidia Model Architecture

The images are given to the model with steering angle, throttle, brake amount as labels in the form of arrays as input data. Hyper parameters (number of neurons, number of epochs, steps per epoch, learning rate etc) are selected experimentally and after a number of trials, the ideal values are : Validation Steps = 200, Number of time periods = 10, Steps per epoch=300.

D. Software

This section discusses the following software requirements that this project makes use of.

1) Simulator: This project makes use of the Udacity self-driving vehicle simulator. In order to educate students how to train automobiles to navigate road courses using deep learning, this was designed for Udacity's Self-Driving Car Nanodegree.

2) TensorFlow: The open source framework TensorFlow was developed to perform mathematic operations on multidimensional data arrays (Tensors). Different hardware configurations may utilize the same API because of the framework's flexible nature.

3) Keras: For the creation and evaluation of deep learning models, the open source Python package Keras provides a powerful and simple-to-use tool. Using only a few lines of code, you can create and train neural network models using Theano and TensorFlow, two powerful numerical computing frameworks.

4) Socketio: High-level framework Socket.IO incorporated into the Udacity project's current communications (SIO). It's likely that the decision to adopt SIO in the Udacity project was influenced by the ease with which it can be integrated into high-level programs like Python ones.

5) Flask: A lightweight Python web framework known as Flask makes it simpler to build web apps in Python by providing a variety of helpful tools and capabilities. It allows developers greater freedom and is easier to use for newcomers to the field.

6) OpenCV: It is an open source computer vision and machine learning software library known as OpenCV (Open Source Computer Vision Library). OpenCV was created to facilitate the use of machine perception in commercial goods by providing standard foundation for computer vision applications. Because OpenCV is a BSD-licensed software, companies may easily use and change the code.

7) Pillow: In order to handle images, you need the Pillow library, which offers all the necessary tools. You can resize, rotate, and modify your images. For statistical analysis and contrast enhancement, you may utilize Pillow's histogram technique of extracting statistics from images, which can then be utilised.

IV. Model Training

Once the information has been gathered and analyzed, There are around 3700 random samples chosen from CSV, we train the network by randomly drawing batches until the complete dataset is being utilized. This trained network is tested with validation set in regular intervals. We have used Google-colab environment for this which is a cloud that allows us to train models in a GPU-powered environment. We have loaded all the required libraries like tensor flow, keras, sklearn, numpy for the training process. We upload the data to a Github account and cloned the data to environment before we begin training. After that,Google Colab is utilized to get the training started.

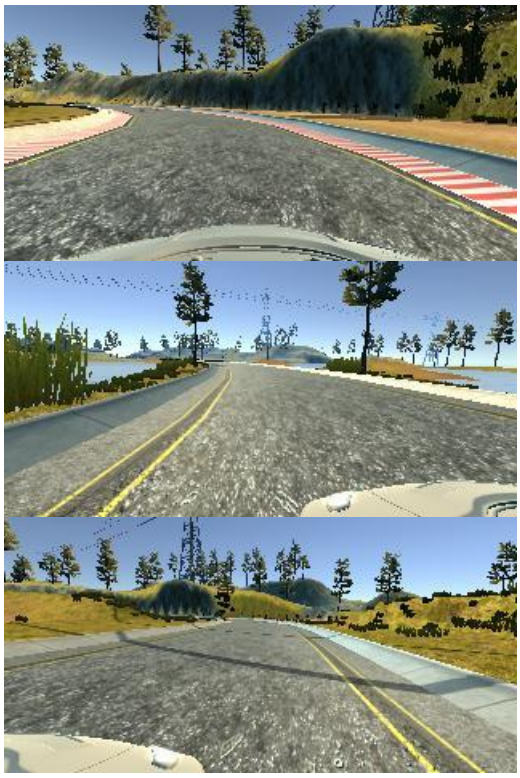

Fig. 4. Sample Images by centre camera, right camera and left camera

Using a batch generator and a model with random weights, input batches are fed into the system before being sent through the layers and compared to the provided labels. We are using adam optimizer [5] with learning rate of 0.001 adjusts the weights. The validation loss is calculated and records are maintained for every run through the inputs in the initial epoch and if the next epoch gave better validation loss than the inital, a copy of the model and its associated weights are saved as model.h5 file. model.h5 file is created to save the weights for future testing and operation of the self-driving cars when all the epochs have been completed.

V. Testing

We are using anaconda tool for running the python code and installed all the required libraries and dependencies. We have placed the .py file and the model.h5 file in a folder. First we create an environment and then activate it where we can run the code and execute it smoothly.

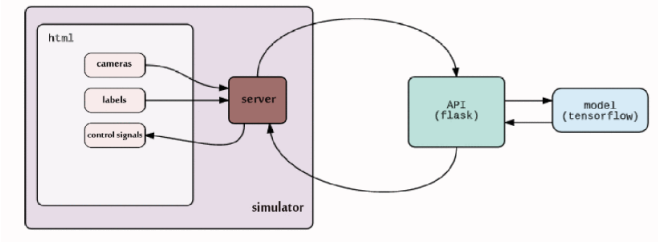

Fig. 5. Model and Simulator Flow

Figure above shows how the model and simulator flow. Initialize Socektio webserver which is used to perform realtime communication between client and server.When a client creates a single connection to websocket server, it keeps listening for new events from server, allows us to continuously update the client with server. It establishes bidirectional communication with the server.As soon as connection is established, we are setting the initial values and simulator is going to send the image details, based on the image, our model extracts features from the images and predicts steering angle. The figure below shows how the driving process during testing looks.

Fig. 6. Screenshot of Driving process after testing

VI. Results

The training and testing validation loss graph in figure shows that there is not over fitting in the model and the model was able to predict the turns effectively by using the center, left, right, steering angle and throttle amount retrieved from csv file. The loss achieved is 3% which is very less. If large amount of training data is used we can reduce the loss percentage even more. For a model to perform efficiently, it should train on huge amount of data to handle all kinds of situations.

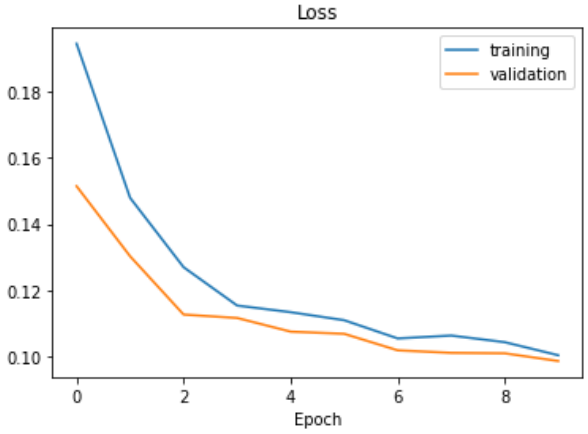

Fig. 7. Training and Validation Loss vs Epoch

Conclusion

This research reveals that convolutional neural networks can be taught using simulation data to run autonomous automobiles. Data from the udacity simulator is used to create and train a neural network. In order to verify the neural network's performance, data from the simulator was used to drive the automobile independently. We found that CNNs can learn the whole lane and road follow job without the need for human breakdown into lane recognition, semantic abstraction, route planning and control. This research empirically confirmed that CNNs can do this.A limited training set is all that is needed for the CNN to learn relevant road characteristics. Deep Neural Networks and classic computer vision methods were used in this study to see whether they might lessen risky driving behavior in self-driving cars. The process of setting up the investigation's environment and assembling the tools required a significant amount of time throughout development. Future autonomous driving initiatives will benefit tremendously from using a system that blends classic computer vision methods with the processing power of DNNs, according to the study.

FUTURE WORK

It's recommended that we collect additional training data in a variety of lighting settings to improve the CNN's capabilities. Definitely, this will increase the precision of the car's navigation system. It is also possible to enhance the model's functionality by adding features such as the ability to avoid obstacles encountered on the course.An accurate model might benefit greatly by the CNN being implemented on a real automobile. Simulator training isn't adequate to get the model ready for real-world use, even when the simulator is meant to be as realistic as possible. In the future, further study may be done to provide a framework for better measurements that indicate how a system is influencing its own actions. This simulator provides the opportunity to do so without the difficulties that frequently arise in the actual world.

REFERENCES

[1] G. Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6), 141-142

[2] Large scale visual recognition challenge (ILSVRC). URL: http://www.image-net.org/ challenges/LSVRC/..

[3] Himmelsbach,M.;Mueller,A.;etal.LIDAR-based3Dobjectperception. In Proceedings of 1st international workshop on cognition for technical systems, volume 1, 2008..

[4] Clevert, Djork-Arn, Thomas Unterthiner, and Sepp Hochreiter. ”Fast and accurate deep network learning by exponential linear units (elus).” arXiv preprint arXiv:1511.07289 (2015)..

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.

[6] Giusti A, Cireşan D C, Masci J, Gambardella L M and Schmidhuber J 2013 Fast image scanning with deep max-pooling convolutional neural networks 2013 IEEE Int. Conf. Image Process. ICIP 2013 - Proc. pp 4034–8.