Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

Self Driving Car using Deep Learning Technique

Chirag Sharma
Vellore Institute of Technology,
Chennai Tamil Nadu, India

S. Bharathiraja
Vellore Institute of Technology,
Chennai Tamil Nadu, India

G. Anusooya
Vellore Institute of Technology,
Chennai Tamil Nadu, India

***Abstract*— The biggest challenge of a self-driving car is autonomous lateral motion so the main aim of this paper is to clone drives for better performance of the autonomous car for which we are using multilayer neural networks and deep learning techniques. We will focus to achieve autonomous cars driving in stimulator conditions. Within the simulator, preprocessing the image obtained from the camera placed in the car imitate the driver's vision and then the reaction, which is the steering angle of the car. The neural network trains the deep learning technique on the basis of photos taken from a camera in manual mode which provides a condition for running the car in autonomous mode, utilizing the trained multilayered neural network. The driver imitation algorithm fabricated and characterized in the paper is all about the profound learning technique that is centered around the NVIDIA CNN model.**

Keywords— ***Deep learning, neural network, Convolutional neural network, stimulator, NVIDIA model.***

I. INTRODUCTION

Going by means of vehicle is at present one of the most perilous kinds of transportation with over a million passing every year around the world. As nearly all car crashes (particularly fatal ones) are caused by driver error, driverless vehicles would viably dispose of about all dangers related to driving just as driver fatalities, road safety, mobility for everyone, and injuries. The self-driving vehicle is the dormant beast that can make a huge difference. The self-driving vehicle is a vehicle outfitted with an autopilot framework and is equipped for driving without the help of human administrator. This innovation was fabricated initially using autonomy approach yet with the progress in the field of PC vision and ai we can utilize the deep learning approach. There are a couple of troubles that need to have been met before executing self-driving car.

One of the most important functionalities of a self-driving vehicle is autonomous lateral control. Autonomous lateral motion traditionally depends on image processing. The process is divided into detecting byways, locating by way centers, track planning, track following, and a control logic. The precision of such frameworks depends essentially on how well-tuned the picture processing filters are. These strategies are amazingly sensitive to assortments in lighting and are slanted to false identification. An algorithm tuned to detect lane markings in a place where it is bright and sunny may not do well in a place where it is dark and gloomy. This framework utilizes a convolutional neural system to yield controlling points dependent on street images basically the model is set up to copy human driving behavior. Since an end-to-end model doesn't take a shot at truly described standards, it is less affected by changes in lighting conditions.

At the end of this paper, you will try to go in detail of how to control autonomous car, we will develop a model of a car motion and will test our control design in our stimulator.

II. RELATED WORK

Convolutional neural networks (CNNs) have painted a whole new picture of pattern recognition; moving forward to large scale adoption of CNNs, many of the pattern recognition projects were finished with an initial stage of self-created components which were extracted and further followed by classifiers. The features we used, acquired knowledge by using training examples of growth of CNNs. This method is mostly used in image recognition as the convolution activity captures 2d nature of the specific image. Likewise, the convolution kernel peruses the entire picture by including it few parameters that peruses contrasted with the complete number of operations. In recent time and light put on the features and discoveries of CNN they are used for commercial purpose more often than they were used 20 years from now. There are a few reasons to it they are: 1) huge data sets such as ImageNet Large Scale Visual Recognition Challenge (ILSVRC) have become more accessible and can be validated.

2) CNN's learning algorithm have been put in parallel graphics processing units which promote fast-track learning

DARPA Autonomous Vehicle (DAVE) established the potential of end-to-end learning and was used to validate starting the DARPA Learning Applied to Ground Robots (LAGR) program. However, DAVE's success was not reliable enough to support a full alternative to more modular approaches to off-road driving: its average space between crashes was about 20 m. Recently, a new application has been started at NVIDIA, which aims to build on DAVE and create a strong system that is capable of learning the whole task of line and path monitoring, without the need for manual decomposition, marking, semantic abstraction, path planning, and control. The agenda for this project is to avoid the requirement to recognize certain features, such as lane markings, guardrails, or other cars.

Pomerleau did a fabulous work by creating DAVE-2 which was done when he bought together autonomous land vehicle in neural network dated in 1989. It featured how a trained neural network could run a vehicle on roads.

III. OBJECTIVE

In this paper, we will do the writing overview of self-sufficient self-driving vehicles. We will probably comprehend and get settled with the activity of a self-driving

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

vehicle utilizing deep learning calculations, for example, NN and CNN.

The self-driving car is one of the biggest and the most interesting topic these days, the industry will continue to need outstanding engineers to produce innovative solutions for driving. In this paper, our objective is to train and test the manual functioning of the car which will be done with the help of image processing and will make a model which we will run in our stimulator, after successfully training the model will make our car run on the different track and will try to decrease the error rate as much as possible and make our vehicle run productively in the various track.

IV. WORKING PRINCIPLE OF CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN every image is represented in the form of pixel values and it compares images piece by piece. It is commonly used to examine visual pictures by handling information with a grid-like topology.

CNN has the following layers:

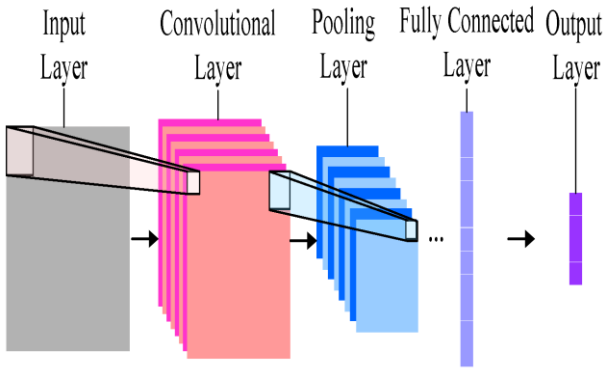

FIGURE 1: LAYERS OF CNN

If any of the layers fail to perform their task then the process will never be executed.

In the convolutional layer we first line up the feature along with picture and then multiply the pixel value with the corresponding value of filter and then adding them up and driving them with the absolute value of pixels.

Relu layer represents rectified layer unit crafted by this layer is to expel all the negative qualities from the filter picture and afterward change it with zero. The node is activated if the image is above a particular quality and it is linearly dependent on the variable considering the input is above threshold value. Image received from Relu is shrunk in the pooling layer.

The actual classification is done in a fully connected layer. We take the shrieked image and up that in a single list. And then we compare that image with our previously-stored list and judge the image. The last is the output layer which gives the output of the classified image.

V. WHY WE USED STIMULATOR?

Driving a vehicle in a stimulator is unquestionably not equivalent to driving a vehicle in reality, there are so many similarities. Given the present condition of game designs, pictures caught in a recreated situation (street, markers, scene) are a decent estimate of pictures that could be caught in reality.

The stimulator additionally gives security and convenience. Information assortment isn't troublesome we can gather information effectively, and if the model fails then there is no risk to life. To study and improve different model structures a stimulator is the best stage. We can implement our model afterward in a real car with real cameras, but for that, we should have to make it successful run efficiently in our stimulator. [1]

VI. BEHAVIORAL CLONING

[1] Behavioral cloning is a technique by which sub-psychological aptitudes like - perceiving objects, understanding while simultaneously performing an action can be captured and imitated in a computer program. The skills performed by human entertainers are recorded alongside the circumstance that offered rise to the activity. The learning program yields a lot of rules that reproduce skilled behavior. This method is used to create automated control structures for complex tasks where the classical control view is inadequate.

In simple words, the work of behavioral cloning is to gather the information and train the model to impersonate the behavior with the data that has been collected.

In summary, these include Deep Neural Network(DNN) we are essentially going to download a self-driving car simulator which is open source by Udacity, we will then use this stimulator to create our very own training data for our model for driving a car through the training track inside the stimulator, as we drive the car through the stimulator we are going to take images at each instance of the drive, these images are going to represent our training dataset and the label of each specific image is the angle on which the car will turn at that given point. After that we will show all the pictures on our CNN and let her learn how to drive independently by studying our behavior as a manual driver, this key difference that our model will learn to adjust is the car's steering ability at any time, it will learn to effectively turn the steering angle to the right level based on the stimulation it finds itself in.

After preparing the model we test it performance on a unique test track where the car will be driven independently.

This behavioral cloning technique is very helpful and plays a big role in a real-life self-driving car as well.

After learning this behavioral cloning technique, you will effectively be able to understand and even apply the science of self-driving cars.

VII. NETWORK ARCHITECTURE

The algorithm comprises of modules that start from collecting data which contain images, steering angles, and speed. Then came the balancing part followed by processing which is very important for the proper preparation of data entering the neural network.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

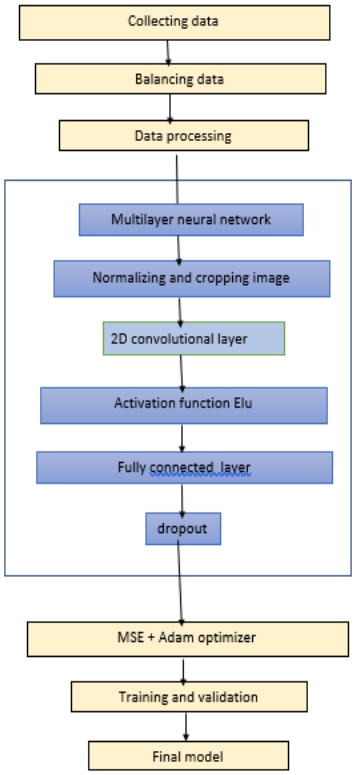

FIGURE 2: BLOCK DIAGRAM OF DRIVER IMITATE ALGORITHM

Three technique was used to preprocess the data

1. Normalization
2. Argumentation
3. Cropping the image so that unwanted data like the sky and hood of the car will remove.

The multilayer neural network model which is the key part of the driver cloning algorithm. It consists of 5 convolutional layers, one leveling layer, flatten layer, and 3 fully connected layers. The first 5 layers are monitored with the ELU function. Then the MSE and Adam optimizer adjust the hyperparameters. The MSE (mean square error) limits the distinction among the predicted steering angle and the actual steering angle. The last is the training and the validation in which the data is divided into two sets, 80% training, and 20% validation.

A. Collecting data

We are going to drive our car through the stimulator using our keyboard keys. The more we drive the car precisely the more accuracy will come in our model i.e. cloning the behavior and hence the term called behavioral cloning.

We downloaded the simulator provided by Udacity organization which is open-source. There is another open-source stimulator also which is available for this purpose for example air sim which is based on an unreal engine.

We have to train the model by driving the car with the help of our keyboard button as we use to do in our video games. This track is organized in a manner to provoke the neural network to conquer sharp turns. The main thing to remember is to attempt to drive the car in the center of the road to ensure a balanced dataset.

The car built with three cameras is situated on the left, right and one on the center. Images are collected, they collect the value of the steering wheel and speed throttle and brake at the current image as sown in figure 3.

FIGURE 3: EXAMPLE OF COLLECTED IMAGE

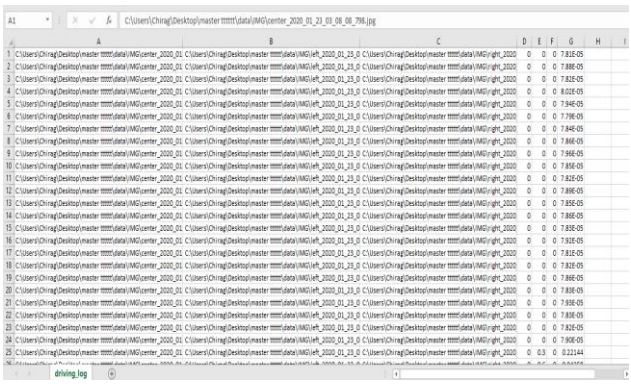

FIGURE 4: EXAMPLE OF CSV FILE

The steering angle is labeled from 1 to -1 where 1 is for the right turn, -1 is for the left turn, and 0 denotes that the car is moving in the straight path.

Here the vertical axis specifies the values of our histogram i.e. frequency of each steering angle during stimulation.

B. Balancing data

In the chart, we can see that we have so many 0 steering angles i.e. the recursion of 0 is higher. This will create a problem by making our model biased toward driving down straight. This would cause data imbalance.

To overcome this problem, we will improve this by eliminating all the samples above a given threshold so that the data will not be biased in driving straight all the time.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

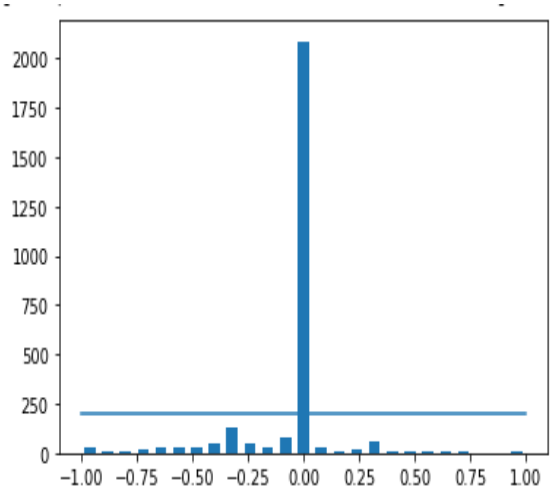

FIGURE 5*:* BAR GRAPH OF STEERING ANGLE

C. Reprocessing data

This is one of the important steps, as we can see that in our image there is a lot of areas which are not of our use or we can say that it isn't imperative to concentrate on, so we wiped out from the image. Features like sky mountain which is there in the picture and the hood on the vehicle in the base. These portions are not at all relevant for our car to determine steering angles. So, we cropped this by using simple NumPy array slicing.

As we can clearly see in our image that the image of axis 160x300 in which the height from 160 to 135 is the hood of the car and the range from 60 to 0 is the sky and landscape which is not of any use so we removed that portion from the photo and set its axis from 60 to 135. This will allow to concentrate more on significant features.

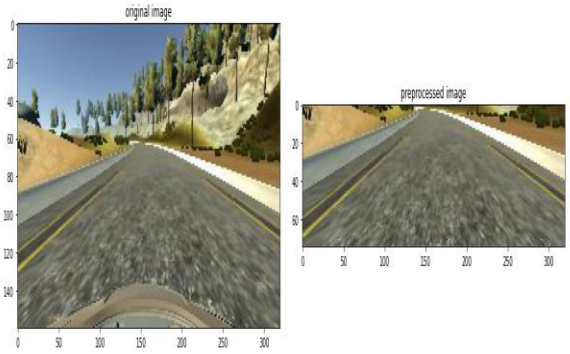

FIGURE 6: CROPPED IMAGE

The next step is to change the color space of the image, this is important for preprocessing because the model we use recommends that we use YUV color space where Y stands for luminosity or brightness of the image and UV represent chromium which adds color.

We show a processed image by adding gaussian blur technique which helps in smoothing and reducing noise of the image.

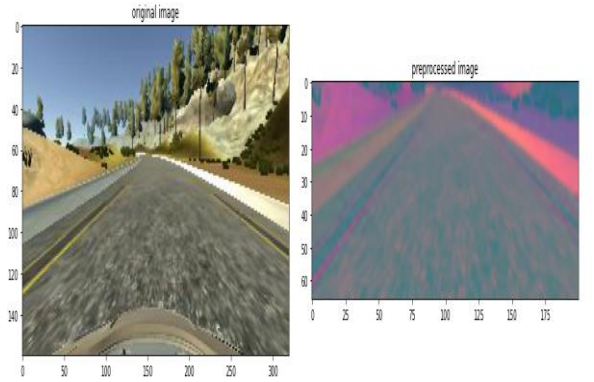

FIGURE *7*: PREPROCESSED IMAGE

D. Model

In behavioral cloning, our dataset is more complex than any other dataset because now we are dealing with images having dimensions 200x66. The popular model use foe behavioral cloning is the NVIDIA model and it also has to be implemented in real-time self-driving cars.

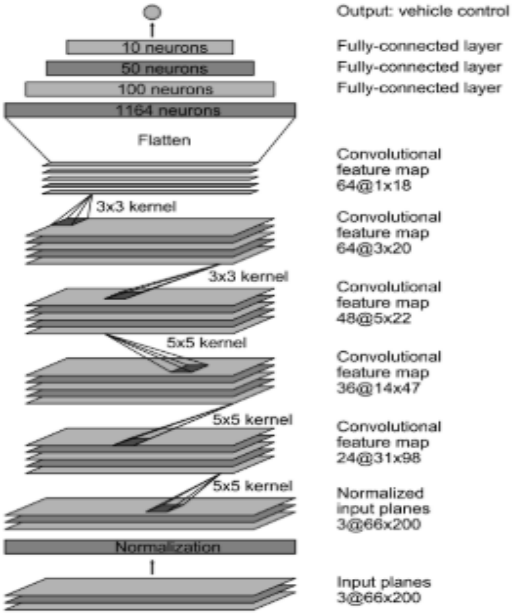

FIGURE 8*:* OUR MODEL FOR BEHAVIORAL CLONING

This model will work in the following steps.

Step 1: Normalization layer (hard-coded) divided by 127.5 and subtract by 1.

Step 2: Convolutional layers with 24,36,48 filters and 5x5 kernel and stride of 2.

Step 3: 2 Convolutional layers with 64 filters, 3x3 kernel, and stride 1.

Step 4: A flatten layer.

Step 5: 3 fully connected layer with an output size of 100,50,10.

Step 6: final output layer that gives the steering layer.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 18, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```

FIGURE 9*:* MODEL SEQUENCE

Next, we did the training process and validate by using the epochs. Here we perform cycle of 30 epochs and each batch contains 100 elements which helps in decreasing the loss which is shown in fig. 10.

```
Train on 662 samples, validate on 166 samples
Epoch 1/10
662/662 [==============================] - 7s 10ms/step - loss: 0.6617 - val_loss: 0.2204
Epoch 2/10
662/662 [==============================] - 0s 394us/step - loss: 0.1945 - val_loss: 0.1587
Epoch 3/10
662/662 [==============================] - 0s 397us/step - loss: 0.1453 - val_loss: 0.1444
Epoch 4/10
662/662 [==============================] - 0s 394us/step - loss: 0.1362 - val_loss: 0.1505
Epoch 5/10
662/662 [==============================] - 0s 406us/step - loss: 0.1309 - val_loss: 0.1412
Epoch 6/10
662/662 [==============================] - 0s 390us/step - loss: 0.1243 - val_loss: 0.1392
Epoch 7/10
662/662 [==============================] - 0s 395us/step - loss: 0.1166 - val_loss: 0.1333
Epoch 8/10
662/662 [==============================] - 0s 397us/step - loss: 0.1154 - val_loss: 0.1269
Epoch 9/10
662/662 [==============================] - 0s 396us/step - loss: 0.1091 - val_loss: 0.1233
Epoch 10/10
662/662 [==============================] - 0s 400us/step - loss: 0.1172 - val_loss: 0.1248
```

FIGURE 10: EPOCH WITH LOSS VALUE

RESULT

Behavioral cloning results in a simulated driving scenario behavioral cloning we stimulate driving scenarios which are implemented using CNN in which we use activation function which is Elu. Elu function uses mean square of error loss which is used for validating data. As we had expected, performance after the training of the data is better but the two values of trained and tested date are very close which lead us to a conclusion that the model portrays more of general qualities.

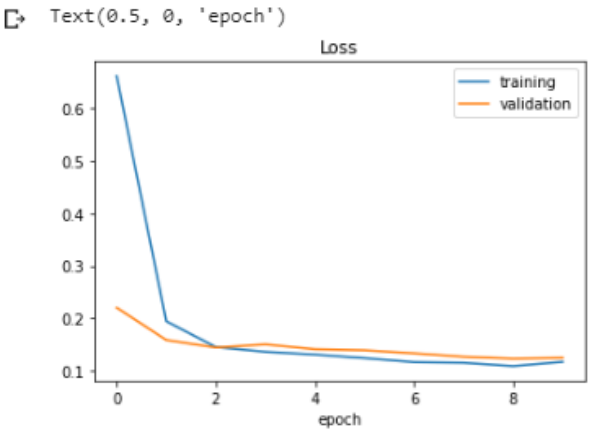

FIGURE 11*:* TRAINING AND VALIDATION

After applying the above method here are the results, the car is running on its track and the result which can observe that in the right-hand side corner there is a speedometer. The trained model can successfully control the car in different, unknown tracks. The model can do laps consistently without failing. With a bigger training dataset comprising of various scenarios, the models' capacity to remain in autonomous mode will increase. Test runs can be found at https://youtu.be/H50cCrUiOqc .

FIGURE 12: DRIVING IN AUTONOMOUS MODE

CONCLUSION

This paper gives one methodology under the stimulated condition for self-governing driving the methodologies use deep learning strategies and end to end figuring out how to accomplish copying of vehicles. The Nvidia neural network is the main frame of the driver cloning algorithm. It is consisting of five convolutional layers one leveling layer and four fully connected layers. The output we receive is the steering angle. The result its use of autonomous mode is successful autonomous driving along a predefined stimulated path through which the model was trained using smaller data sets. It ought to be stressed that all the data expected to train the system are independently created in manual mode, thus creating their own databases. We can improve our method by using a better stimulus generalization. Deficient generalizability happens as an outcome of little database in

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

which controls its application to a natural situation. As though now the car in autonomous mode is running really good along a predefined stimulator route.

REFERENCES

[1] Nicolas Gallardo, "Autonomous Decision Making for a Driver-less Car", 2017.
[2] S. Liu et al., Creating Autonomous Vehicle Systems, Morgan Claypool Publishers, 2019
[3] D. Bemstcin, and A. Kamhauser, (2012) "An Introduction to MapMatching far Personal Navigation Assistants", Princeton University, Princeton, New Jersey, August 2012
[4] Joshi and M. R. James, "Generation of accurate lane-level maps from coarse prior maps and lidar," 2014
[5] Qudsia Memon , Shahzeb Ali, Wajiha Shah, "Self-Driving and Driver-Relaxing Vehicle", 2016.
[6] Naveen S Yeshodara, 2Nikhitha Kishore, "Cloud Based Self Driving Cars", 2014
[7] https://medium.com/swlh/behavioural-cloning-end-to-end-learning-for-self-driving-cars-50b959708e59