IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Autonomous Vehicle System using CNN

[1]Kavita Jain, [2]Shivang Medhekar, [3]Sohum Khot, [4]Shreyas Bhide

[1]Assistant Professor, Dept of Computer Engineering, Xavier Institute of Engineering, Mumbai, India
[2,3,4]Students of Xavier Institute of Engineering, Mumbai, India

Abstract: Our proposed model provides an end-to-end autonomous vehicle simulation. We trained a Convolutional Neural Network (CNN) to directly map raw pixels to the steering commands from a single front-facing camera. This end-to-end approach proved to be surprisingly strong. The machine learns to drive in traffic on local roads with or without lane markings and highways with minimal training data from humans. It also works in places where the visual direction is not obvious, such as parking lots and on unpaved roads. The model learns internal representations of the required processing steps automatically, such as detecting useful road features with only the human steering angle as the training signal. We have never specifically conditioned it to detect the outline of paths, for example. Our end-to-end method optimizes all processing steps simultaneously as compared to the explicit decomposition of the problem, such as lane marking detection, route planning, and control. We claim that in the end, this will lead to better results and simpler systems. It will result in better performance because the internal components are self-optimizing to maximize overall system performance, rather than optimizing intermediate criteria chosen by humans e.g. Detecting roads. Such criteria are selected understandably for ease of human interpretation which does not automatically guarantee maximum performance of the system. Smaller networks are possible because with the minimal number of processing steps the machine learns to solve the problem.

***Index Terms* - Autonomous Vehicles, Self-Driving Car, Convolutional Neural Network (CNN), Autonomous Vehicle Simulator**

I. INTRODUCTION

There are many incidents occurring around us every day due to a lot of breaches of the traffic rules. Most of them occur because of driver's carelessness. Many public transportation services, too, lack qualified drivers. That poses a threat to the lives of millions of worldwide travellers. A lot of other problems arise, including traffic jams, because of breaches of these traffic rules. That wastes the driver's time as well as efforts. There's no doubt that self-driving cars will be the standard way of transport in the future. Major companies from Uber and Google to Toyota and General Motors are willing to spend millions of dollars to make them a reality, as trillions are expected to be worth of the futures market. We've seen a massive change in the area over the past few years with cars coming from Uber, Tesla, Waymo to have a total of 8 million miles in their history. Self-driving cars are now a reality due to many different advances in both hardware and software technologies. LIDAR sensors, cameras, GPS and ultrasonic sensors work together to obtain the data from any source. Those data are analysed using advanced algorithms in real-time, allowing the autopilot feature. A hotbed of research is currently the application of Artificial Intelligence (AI) and Machine Learning (ML) techniques to the development of autonomous driving systems.

The primary objective of this project is to make cars unmanned and to automate their functioning with the help of simulations. This will lead to synchronization as well as reduce time and efforts of the travellers. The required solution is aimed to not only facilitate improvement in the services, but should also be a driving factor for a revolution in the automobile industry. We are developing a system that won't require any driver or controlling authority for the functioning of a car. The outcome of this project is a totally deployable model for research as well as application purpose. The project is a research-based simulation project which can be further applied to real-time models and scenarios. The simulations can help us advance further into the improvement of automated self-driving automobiles systems for fuel efficiency and increased reliability.

II. LITERATURE SURVEY

Most research studies have generally focused on road lane detection over the past decades, as they contain simple geometric features and can be easily extracted from background as per resource [1].

Following the revolutionized CNNs pattern recognition of resource [2], a substantial amount of self-driving car research is done using a deep learning method.

Recently, various research efforts are being made to apply deep learning techniques in all low-level / mid-level computer vision tasks of robot navigation patterns in which the low computational cost of applied CNN makes result in resource [3] easily achievable in real-time processing.

Huval et al. resource [4] proposes a method for predicting two end-points of a local lane segment by regressing in a sliding window using CNN and combining it with RNN to detect lane boundaries.

Likewise, Hadsell et al. resource [5] study, which proposes a deep hierarchical network for extracting important and meaningful features from an input image, and the features are used to train a real-time classifier to predict travers ability. We implemented a self-supervised learning system capable of reliably classifying complex terrains at distances from the platform from 5 to over 100 m away, thereby outstripping path-planning considerably.

Chen et al. resource [6] offers an approach that, instead of learning from pixel mapping, first estimates a number of human main perception indicators that are directly related to affordance measures such as distance to nearby vehicles.

Resource [7] is a paper published by Nvidia Corporation which describes its approach towards end-to-end development of a self-driving car model using CNN.

III. Proposed System

In this paper we introduced and implemented an approach based on an open-source Udacity self-driving car simulator for self-driving cars. We suggest an approach based on the CNN to build a concept of self-driving cars. This paper illustrates the end-to-end process from the development of a dataset, using the simulator to pre-process the end results. The block diagram is mentioned in Fig. 1.
Following steps are involved in the proposed system:

1) Dataset Creation
2) Splitting the Dataset
3) Image Augmentation
4) Data Pre-processing
5) Lane Detection
6) Implementing the CNN Architecture
7) Testing the model on the simulator

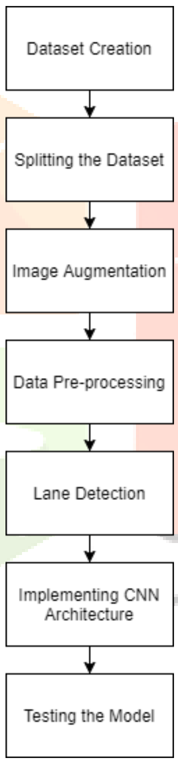


Fig. 1 Block Diagram for Proposed System

IV. Implementation Methodology

The different implemented modules of our system are as follows.

4.1 Dataset Creation

We used the Training Mode provided in the Udacity Simulator which lets the user drive the vehicle model according to their own control and captures images of the driver's point of view at frequent intervals of time along with the steering angles. We continued this until the whole lap was covered in order to gain more accuracy.

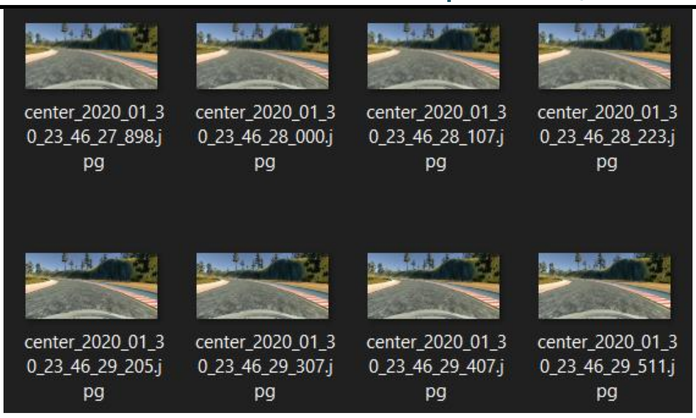

center_2020_01_30_23_46_27_898.jpg
center_2020_01_30_23_46_28_000.jpg
center_2020_01_30_23_46_28_107.jpg
center_2020_01_30_23_46_28_223.jpg
center_2020_01_30_23_46_29_205.jpg
center_2020_01_30_23_46_29_307.jpg
center_2020_01_30_23_46_29_407.jpg
center_2020_01_30_23_46_29_511.jpg

Fig. 2 Dataset Creation

4.2 Splitting the Dataset

In this step, we split the dataset into Training Set and Validation Set. This is observed with respect to the steering angles of the data. The histograms for the same are mentioned in Fig. 3.

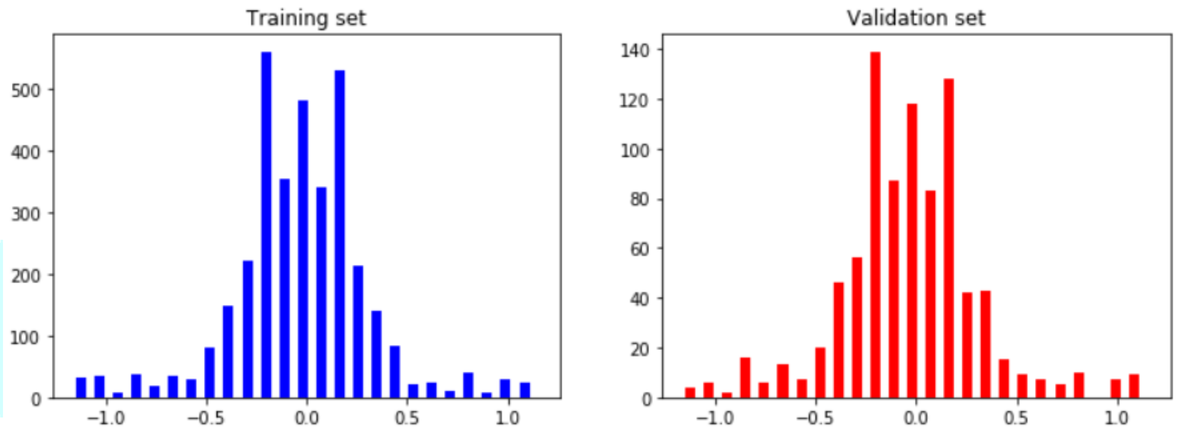


Fig. 3 Histograms of Training Set and Validation Set w.r.t Steering Angles

4.3 Image Augmentation

Image Augmentation produces training images by various processing methods or combinations of multiple processing such as zooming, rotating, rotation, flipping and other operations. In the Image Augmentation are involved the following steps.

4.3.1 Zooming

In this operation, we oversample the original image to remove any unwanted pixels and focus only on the road mapping and lane detection.

4.3.2 Panning

Panning refers to the horizontal scrolling of an image wider than the display i.e moving parallel to the current view plane.

4.3.3 Brightness Alteration

In this step, we increase or decrease the pixel intensity of each pixel of the image in order to gain accurate results.

4.3.4 Flipping

In this step, we horizontally invert i.e. flip the image.

4.3.5 Final Augmented Image

This is the final output of an augmented image by applying the above combination of operations is in Fig. 4.

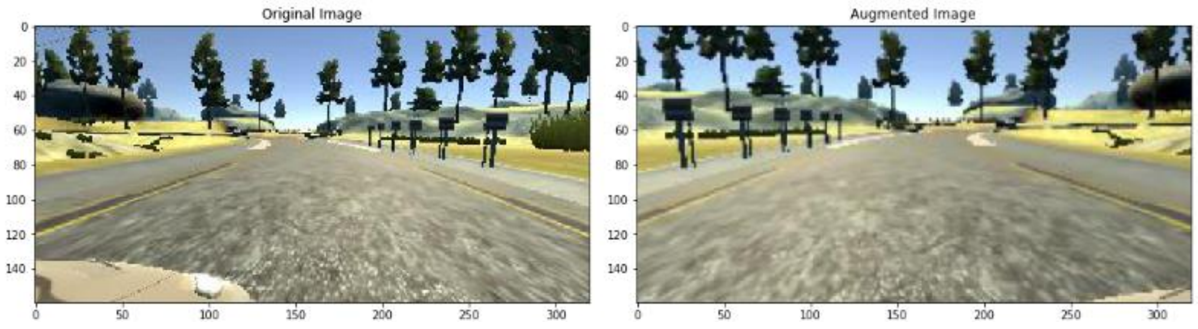


Fig. 4 Original Image and Augmented Image

4.4 Image Pre-processing

The steps involved in Image Pre-processing are as follows.

4.4.1 Gaussian Blur

The Gaussian blur feature is obtained through blurring (smoothing) an image using a Gaussian function to lower the noise level. It can be considered a non-uniform low-pass filter preserving low spatial frequency and reducing image noise and negligible details in an image. Usually, it is accomplished by transforming an image into a Gaussian kernel.

4.4.2 Converting Color Scheme from RGB to YUV

We convert the Color Scheme of the image from RGB to YUV color encoding.

4.4.3 Pre-processed image

The final pre-processed image is mentioned in Fig. 5.

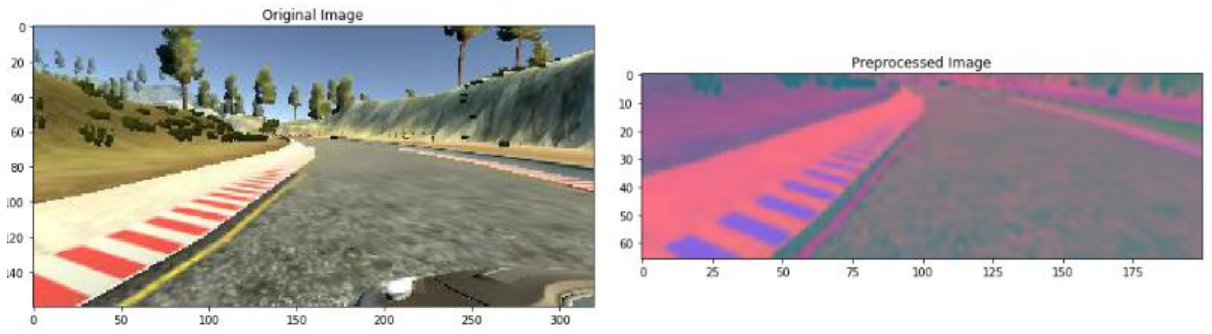


Fig. 5 Original Image and Pre-processed Image

4.5 Lane Detection

We need to identify the lanes of the road as it is one of the most common and important tasks that a human driver performs. This is crucial in order to keep the vehicle in the constraints of the lane for an autonomous vehicle. We use a Computer Vision library OpenCV in Python for detection of lanes. We use the Single Pipeline method for identifying lanes.

The steps involved in Lane Detection using Single Pipeline are as follows:

1. Convert the image from RGB to Grayscale.
2. Darken the grayscale image for reduction of contrast from discoloured road regions.
3. Conversion of image to HLS colour space.
4. Isolate yellow from HLS to get a mask for yellow lane markings.
5. Isolate white from HLS to get a mask for white lane markings.
6. Perform Bitwise OR operation on the yellow and white masks to get a common mask.
7. Perform Bitwise AND operation on the mask with the darkened image.
8. Apply a slight Gaussian Blur.
9. Apply Canny Edge Detection Algorithm to adjust the threshold to get edges.
10. Define Region of Interest in order to prevent unwanted edges from being detected.
11. Retrieval of Hough lines.
12. Consolidation and extrapolation of the Hough Lines.
13. Draw the Hough lines on the Original Image.

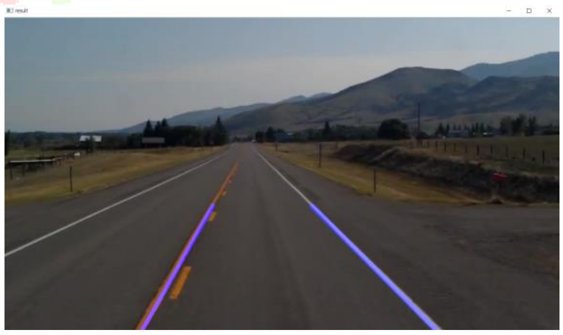
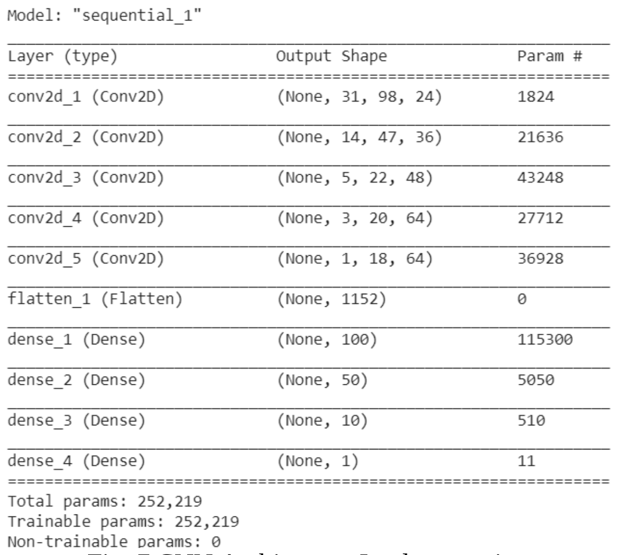


Fig. 6 Lane Detection

Fig. 6 represents the output of performing the above-mentioned series of operations in order to obtain the Hough Lines and drawing these lines on the Original Image.

4.6 Implementing the CNN Architecture

The Neural Network Architecture is referred from [7] i.e. NVIDIA paper.



```
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
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
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
```

Fig. 7 CNN Architecture Implementation

Fig. 7 represents the implemented architecture of the CNN that is used for training the model. Standardization of input is applied through a layer of Lambda which is the first layer of the model. Input is then structured in such a way that it lies within the range [-1, 1]: this works naturally as long as the frame fed to the network is within the range [0, 255]. Choosing the ELU activation function (instead of more conventional ReLU) is used for the steering regression task. The NVIDIA paper [7] does not specifically state what activation function they are using. Three fully connected layers follow conventional layers: finally, a last single neuron tries to regress the correct steering value from the features it receives from the previous layers.

V. Evaluation

After training the model, the model is tested using the Udacity Self-Driving Car Simulator. Fig.8 and Fig. 9 represent the testing of the trained model on the above-mentioned simulator. The simulator provides us an environment to try and check the reliability of the model along with the accuracy of the change steering angles with respect to the curvature of the road present in the track.

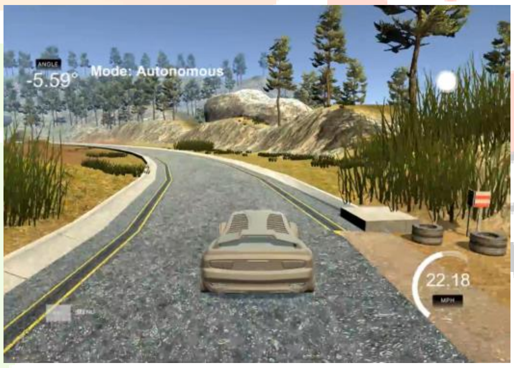


Fig. 8 Testing the model(a)

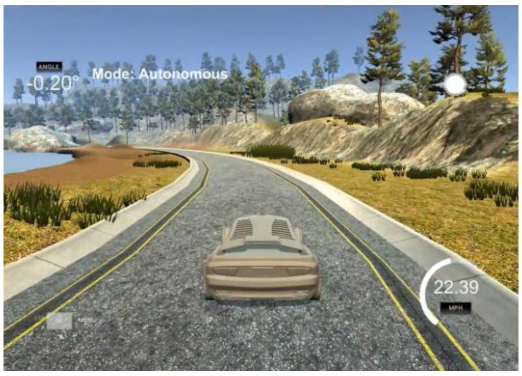


Fig. 9 Testing the model(b)

As for training, this approach uses mean squared error for the loss function. It is implemented to measure the accuracy of the model's prediction with respect to the given steering angle for each image. Fig. 10 gives an insight about this loss.

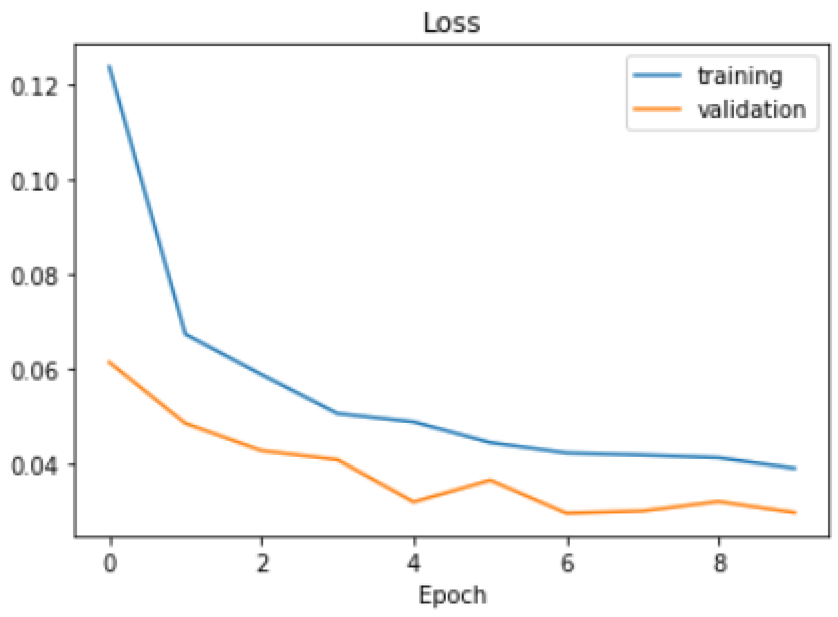


Fig. 10 Training Loss and Validation Loss Graph

VI. Conclusion

The objective of our project is to introduce an approach for autonomous vehicles. In this paper, we proposed an approach for self-driving cars based on open-source Udacity Self-Driving Car Simulator. This paper proposes a Convolutional Neural Network (CNN) based approach for the development for a self-driving car model. This paper demonstrates all the steps involved in building the model. We captured the dataset using the simulator's Training Mode. Then, this dataset was augmented with operations like zooming, panning and brightness altercation. This augmented dataset is then pre-processed using Gaussian Blur and converting the color scheme of the images in the dataset from RGB to YUV.

The accuracy of our model is 90-95% depending on the quality and accuracy of the provided dataset. The CNN architecture is inspired from resource [7] i.e. the paper presented by Nvidia Corporation which itself is conducting a research in the field of autonomous vehicles. During the development of this model, we determined that the provided CNN Architecture gives optimum results for the accuracy of the model. However, modifying the layers of the provided CNN architecture will provide varying results and may increase the model's accuracy. Thus, this paper provides an approach which can be further developed for a fully functional and real-time self-driving car.

References

[1] J. McCall, M. Trivedi, "Video based lane estimation and tracking for driver assistance: Survey system and evaluation.", IEEE Transaction on Intelligent Transportation Systems, 2006.

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks.", Neural Information Processing System Conference, 2012.

[3] L. Ran and Y. Zhang, "Convolutional Neural Network-Based Robot Navigation Using Uncalibrated Spherical Images.", MDPI, Volume 17 Issue 6, 2017.

[4] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, A. Y. Ng, "An Empirical Evaluation of Deep Learning on Highway Driving.", Computer Vision and Pattern Recognition, arxiv, 2012.

[5] R. Hadsell, P. Sermanet,J. Ben, A. Erkan, M. Scoffier, K. Kaukcuoglu, U. Muller, Y. LeCun, "Learning long-range vision for autonomous off-road driving." in Journal of Field Robot, 2009.

[6] C. Chen, A. Seff, A. Kornhauser, J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving.", IEEE Internation Conference of Computer Vision, 2015.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, "End to End Learning for Self-Driving Cars", Nvidia, April 2016.